

# Bewertung von Internet-Routing-Strategien

PROF. AXEL LEHMANN

ROBERT BRENDEL

(HRSG.)

INSTITUT FÜR TECHNISCHE INFORMATIK

Bericht Nr. 2002-07

Dezember 2002

Universität der Bundeswehr München

Fakultät für

**INFORMATIK**

Werner-Heisenberg-Weg 39 • D-85577 Neubiberg



---

A. Lehmann, R. Brendel (Hrsg.): Bewertung von Internet Routing Strategien  
Universität der Bundeswehr München  
Institut für Technische Informatik  
Fakultät für Informatik  
Werner-Heisenberg-Weg 39, D-85577 Neubiberg

---

## Vorwort

Dieser Berichtsband fasst die schriftlichen Ausarbeitungen des Seminars “*Bewertung von Internet-Routing-Strategien*” zusammen, die im Frühjahrstrimester 2002 am Institut für Technische Informatik der UniBwM im Rahmen eines Studienprojektes zum Thema “*Performability-Analyse von Internet-Routing-Strategien*” erarbeitet wurden. Ziel des Studienprojektes war es eine komplexe Aufgabenstellung, wie die der Performability-Bewertung aktueller Internet-Routing-Strategien, durch eine Gruppe von Studenten, in deren Hauptstudium durch gezielte Teilnahme an ausgewählten Vorlesungen, Seminaren und durch Anfertigung von Studienarbeiten erarbeiten zu lassen. Im einzelnen waren im Verlauf des Studienprojektes die folgenden Lehrveranstaltungen zu absolvieren:

- im Wintertrimester 2002:
  - Vorlesung “Methoden und Werkzeuge zur Leistungsanalyse”
- im Frühjahrstrimester 2002:
  - Wahlpflichtvorlesung “Simulation”
  - Praktikum “Modellbildung und Simulation”
  - Seminar “Bewertung von Internet-Routing-Strategien”
- im Herbsttrimester 2002:
  - Anfertigung einer Studienarbeit im o.g. Themenbereich.

Vor diesem Hintergrund sind Themen und Ausgestaltung des hier vorliegenden Seminars zu sehen, bei dem verschiedene Aspekte des Internets vorgestellt, analysiert und bewertet werden sollten wie beispielsweise:

- Architekturen,
- Routing-Protokolle,
- Multicasting,
- Aspekte der Leistungs-, Zuverlässigkeits-, Performability-Analyse und -Modellierung sowie
- Lastcharakterisierungen.

Durch die im Rahmen des Seminars zu erarbeitenden Themen sollten die notwendigen Grundlagen bzw. das Wissen erarbeitet werden, das die Studierenden zur Durchführung ihrer Studienarbeiten hinsichtlich einer Performability-Bewertung einzelner Internet-Routing-Strategien benötigen, bzw. das sie in die Lage versetzt, problemorientierte Modellierungs- und Bewertungsmethoden dazu einzusetzen.



# Inhaltsverzeichnis

<b>1</b>	<b>Internet Architektur und Protokolle</b>	<b>7</b>
	<i>Tom Morawitz</i>	
<b>2</b>	<b>Interne Routing-Protokolle</b>	<b>41</b>
	<i>Markus Körner</i>	
<b>3</b>	<b>Externe Routing-Protokolle</b>	<b>57</b>
	<i>Marco Schuler</i>	
<b>4</b>	<b>Routing mit Multicasting</b>	<b>75</b>
	<i>Daniel Scheppelmann</i>	
<b>5</b>	<b>AntNetz</b>	<b>97</b>
	<i>Klaus Bernhard Riepel</i>	
<b>6</b>	<b>Leistungsmodellierung</b>	<b>113</b>
	<i>Andreas Schäfer</i>	
<b>7</b>	<b>Zuverlässigkeitsmodellierung</b>	<b>147</b>
	<i>Tilo Schröder</i>	
<b>8</b>	<b>Performability Modellierung</b>	<b>177</b>
	<i>Robert Brendel</i>	



# 1 Internet Architektur und Protokolle

*Tom Morawitz*

## Inhaltsverzeichnis

---

<b>1.1</b>	<b>Einleitung</b> . . . . .	<b>9</b>
<b>1.2</b>	<b>Die Geschichte und Organisationen des Internet</b> . . . . .	<b>9</b>
1.2.1	Geschichte des Internet . . . . .	9
1.2.2	Organisation . . . . .	10
<b>1.3</b>	<b>Die Architektur des Internet</b> . . . . .	<b>11</b>
<b>1.4</b>	<b>Das ISO/OSI Referenzmodell</b> . . . . .	<b>13</b>
<b>1.5</b>	<b>Das TCP/IP Protokoll</b> . . . . .	<b>15</b>
1.5.1	Das Referenzmodell . . . . .	15
1.5.2	Internet Protokoll . . . . .	17
1.5.3	IP-Adressierung . . . . .	19
1.5.4	ICMP . . . . .	21
1.5.5	TCP . . . . .	22
1.5.6	UDP . . . . .	23
<b>1.6</b>	<b>Anwendungen im Internet</b> . . . . .	<b>24</b>
1.6.1	FTP . . . . .	24
1.6.2	POP3 . . . . .	24
1.6.3	SMTP . . . . .	25
1.6.4	HTTP . . . . .	25
1.6.5	Telnet . . . . .	25
1.6.6	SNMP . . . . .	26
<b>1.7</b>	<b>Die Zukunft des Internet</b> . . . . .	<b>26</b>
1.7.1	CIDR . . . . .	26
1.7.2	Internet Protokoll Version 6 - IPv6 (IP Next Generation) . . . . .	27
1.7.3	QoS . . . . .	29
<b>1.8</b>	<b>Zusammenfassung</b> . . . . .	<b>29</b>
	<b>Literaturverzeichnis</b> . . . . .	<b>30</b>

---

## **Kurzbeschreibung**

Das Internet ist ein Verbund von kleinen Netzen zu einem speziellen Leitungsnetz, mit dem man Computer auf der ganzen Welt verbinden kann. Es ist in erster Linie die technische Möglichkeit, Informationen weltweit auszutauschen. Welche Architektur dem zugrunde liegt und was technisch dahinter steckt, ist Inhalt dieser Seminararbeit.

Seit seiner "Erfindung" Mitte der siebziger Jahre erlebte das Internet vor allem in der jüngsten Zeit eine geradezu explosionsartige Verbreitung rund um den Globus. Die Hauptprinzipien der Architektur sind zum einen das Ende-zu-Ende-Prinzip, bei dem die Funktionalität in den Endgeräten gewährleistet wird und zum anderen das Internetprotokoll, welches sich als der Standard durchgesetzt hat. Dies soll in der Arbeit näher erläutert werden.

Die Informationen werden paketweise übertragen. Die Pakete lassen sich, Dank der Schichtenstruktur der Protokolle, flexibel aufsetzen. Wegen seiner enormen Bedeutung wird in der Arbeit auch besonders auf die TCP/IP Protokollfamilie eingegangen.

Die rasante Entwicklung wird aufgrund der wachsenden Anreize nicht gebremst werden. Denn je mehr Menschen vernetzt sind, desto mehr Menschen wollen von dieser Vernetzung profitieren und desto mehr Dienste werden auch angeboten.

Die Internet-Protokolle sind jedoch in die Jahre gekommen. Die Tatsache, dass eine Technologie ohne strukturelle Änderungen ein Wachstum von mehreren Größenordnungen verkraftet, dokumentiert sicher die Weitsicht ihrer ursprünglichen Entwickler. Aber sich ändernde Forderungen, vor allem in Sachen Sicherheit und QoS (Quality of Service) zwingen zur Veränderung der bestehenden Protokolle. Auf diese und andere Entwicklungen der Zukunft wird abschließend in dem letzten Kapitel der Seminararbeit eingegangen.

## **Abstract**

The Internet is a combine of smaller networks to a special net that is able to connect computers all over the world. First of all it is a technical opportunity to globally exchange information. The architecture and technical specification of the web will be the content of this seminar paper.

Since its invention in the middle of the 1970s and in recent years even more and faster than ever the internet is being spread all over the world. The main principle of its architecture is the end-to-end-principle where all the functionality is provided by the terminals on one hand and on the other hand the internet protocol, which made it to standard. This will be explained in the following passages.

Information is being transmitted in packages. Due to the layered structure of the protocols, these packages are able to be flexibly put together. This paper will also go into the particulars of the highly important TCP/IP protocol family.

The rapid development might not be slowed down because of the growing incentives. The more people are integrated in this network, the larger is the community that wants to take advantage of the net and therefore the more services will be provided.

So far the internet-protocols are growing old. The fact that the technology was able to grow in several scales without changes in structure is an evidence for the far-sightedness of the developers. But changing demands, especially concerning security and QoS (Quality of Service) force the existing protocols to be changed. The paper will deal with this and other future developments in the last paragraph.



## 1.1 Einleitung

Das Internet (Interconnected Networks) ist die weltweite Zusammenschaltung von vielen Computer-Netzwerken zum Austausch von Daten. Die weltweite Vernetzung wissenschaftlicher Einrichtungen durch das Internet hat in den letzten Jahren Fortschritte gemacht, die als rasant bezeichnet werden müssen. Im März 1989 waren 410 verschiedene Netzwerke, also wissenschaftliche und militärische Einrichtungen, über Internet miteinander verbunden, zwei Jahre später war die Zahl auf 2.501 und damit etwa das Sechsfache gestiegen. Die Zahl der angeschlossenen Rechner erhöhte sich von 535.000 im Juli 1991 auf etwa 1,3 Millionen im Januar 1993 [Sch93] und 2,5 Millionen das Jahr darauf [Hui96]. Die Anzahl der verbundenen Netzwerke beträgt heute schon über 130.000 [tel02]. Die aus der Vernetzung resultierenden Geschwindigkeiten und Wege der Informationsdistribution stellen eine neue Chance für die sich entwickelnde Informationsgesellschaft dar. Die hier vorgestellte Seminararbeit beschäftigt sich mit der Architektur und den Protokollen des Internets und geht dabei auch auf deren Entwicklung und Zukunftsgedanken ein. Weiterhin werden die verwendeten standardisierten Protokolle beschrieben, die die Grundlage der universellen Kommunikation zwischen den verschiedenen Netzwerken bilden.

## 1.2 Die Geschichte und Organisationen des Internet

### 1.2.1 Geschichte des Internet

Auf Initiative des amerikanischen Verteidigungsministeriums, entwickelte die DARPA (Defence Advanced Research Projekt Agency), eine Unterorganisation des Verteidigungsministeriums, in den 60er Jahren ein Forschungsprojekt, mit dem Ziel, im Falle eines nuklearen Angriffs die Befehls- und Kommunikationsstrukturen über vernetzte Computer aufrecht zu erhalten. Ein weiteres Ziel des Projekts war auch die Bedarfsdeckung an Netzwerktechnologien um Rechnerressourcen zu nutzen.

Das erste Testnetzwerk mit dem Namen "ARPANET" installierte ein Unternehmen aus Massachusetts, die Bolt Beranek & Newman Inc. Nur wenig später, zu Beginn der 70er Jahre umfasste dieses Netzwerk bereits über 50 Computer in den USA und Westeuropa. Das Organisationszentrum des Netzwerkes, das sogenannte NOC verwaltete die Firma. Damit war das erste paketvermittelnde Weitverkehrsnetz entstanden. Das heißt, man schickte erstmals die Information stückweise von A nach B, ohne eine Standleitung zu etablieren. Ein weiteres Ziel des Projektes war neben der Entwicklung der Paketvermittlungstechnik die Schaffung einheitlicher Protokolle zur Datenübertragung zwischen verschiedenen Computersystemen.

Schon Mitte der 70er Jahre reichte das Netzwerk nicht mehr aus, so dass die DARPA nach Möglichkeiten suchen musste, wie man weitere Netzwerke bauen und diese verbinden könnte. Dies führte zur Entwicklung von Techniken, wie Token Ring und Ethernet, sowie zur Weiterentwicklung der Satelliten- und Funkkommunikation.

Die vorwiegenden Nutzer waren die Universitäten sowie Militär und Regierungsorganisationen, die mit dem Wachstum des Netzes begannen, das Netz auch für ihre tägliche Arbeit, z. B. Austausch von nichtmilitärischer Informationen, Dateien und Dokumente, zu verwenden. Die damit wachsende Nutzung des Netzes führte zu einem erhöhten Vermittlungsaufwand, so dass das ARPANET in zwei separate Netzwerke, das MILNET für militärische Installationen und ein neues ARPANET für zivile Einrichtungen, gespalten wurde.

Im Jahr 1975 übernahm die DCA (Defense Communication Agency) die Kontrolle über das ARPANET. DARPA finanzierte die Entwicklung einer ganzen Reihe von Protokollen für die

Kommunikation im ARPANET. Das Ergebnis dieser Entwicklung war ein Protokoll aus zwei Komponenten: TCP (Transmission Control Protocol) und IP (Internet Protocol). Obwohl es erst zwei getrennt gewartete Netzwerke, MILNET und ARPANET, gab, wurden die Protokolle so entwickelt, dass mit ihnen mehrere Netzwerke verbunden werden konnten. IP z. B. war bereits so ausgelegt, dass es bereits Tausende von Netzwerken miteinander verbinden hätten können und so bis heute bestehen konnte. Mit der Aufnahme von TCP/IP in das an der University of California in Berkeley neu entwickelte UNIX-Betriebssystem, das Software-Distribution-Kit (BSD Unix), begann TCP/IP besonders in akademischen Kreisen sehr schnell zu wachsen.

Die Geburtsstunde des Internets war in den 80er Jahren, als das amerikanische Verteidigungsministerium vorschrieb, dass alle Computer, die mit ARPANET verbunden sind, TCP/IP einsetzen müssen.

Nach der Gründung der National Science Foundation 1986, entwickelte sich ein Netzwerk, das als Protokoll TCP/IP verwendet, und mit dem man auf NFS-Serverrechnern lesend und schreiben zugreifen konnte, als ob sie am eigenen Rechner wäre, d.h.: externe Dateisysteme konnten gemountet werden. In diesem Jahr wurde auch das Domain-Adressen-System (DNS) eingeführt, welches später erläutert wird. Ein weiteres einschneidendes Jahr war das Jahr 1992, unter anderem mit der Einführung des Internets am Kernforschungszentrum CERN in Genf. Die Entwicklung des World Wide Web (WWW) Dienstes und die Gründung der Internet Society (ISOC), die die dezentrale Struktur des Internets zu koordinieren versucht fanden ebenfalls 1992 statt. Mit Hilfe der ISOC war es möglich Standards über Ländergrenzen hinweg zu setzen und Weiterentwicklungen zu koordinieren.

Dies war ein kurzer Einblick in die Entwicklung des Internets auf der Welt, aber wie sah die Entwicklung in Deutschland aus? Mit der Gründung eines nationalen Forschungsnetzes durch den Verein zur Förderung eines Deutschen Forschungsnetzes (DFN) begann 1984 die Entwicklung in Deutschland. In Kooperation mit der damaligen Bundespost wurde das Wissenschaftsnetz (WIN) aufgebaut. Aber bis 1992 war der Zugang zu diesem Netz nur Universitäten und Forschungseinrichtungen vorbehalten. Im Jahr 1995 erfolgte die Privatisierung des NFSNET [Alp96].

### 1.2.2 Organisation

Für die Verwaltung und Weiterentwicklung des Internets gibt es mehrere, zumeist Non-Profit Organisationen. In Deutschland existiert, wie unter Punkt 2.1 beschrieben, seit 1984 das Deutsche Forschungsnetz (DFN), das vom Bundesministerium für Forschung und Technologie (BMFT) gegründet wurde und Betreiber des Wissenschaftsnetzes WIN ist.

Eine Organisation für akademische, forschungs- und lehrebezogene, sowie gemeinnützige Belange ist die Internet Society (ISOC; [www.isoc.org](http://www.isoc.org)), die im Juni 1992 gegründet wurde. Sie steht ebenfalls für die Kooperation und Koordination von Technologien und Anwendungen im Internet.

Das Deutsche Network Information Center (DE-NIC; [www.denic.de](http://www.denic.de)), das 1994 am Rechenzentrum der Universität Karlsruhe gegründet worden ist, ist zuständig für die eindeutige Vergabe von IP-Adressen. Übergeordnet gibt es das International Network Information Center (InterNIC; [www.internic.net](http://www.internic.net)), das die internationale Vergabestelle von Domain-Namen ist und sich in die einzelnen Länderorganisationen untergliedert.

Des Weiteren gibt es einige weitere Organisationen mit detaillierten Aufgaben im Internet. Das sind unter anderem die Internet Architecture Board (IAB; [www.isi.edu/iab/](http://www.isi.edu/iab/)), die die Dokumentation der Netzstruktur und der grundsätzlichen Abläufe im Internet zur Aufgabe hat oder

die Internet Assigned Numbers Authority (IANA; [www.isu.edu/iana/](http://www.isu.edu/iana/)), als zentrale Koordinationsstelle für Internet-Protokolle. Die Internet Engineering Task Force (IETF; [www.ietf.org](http://www.ietf.org)) ist die internationale Gemeinschaft von kommerziellen und nichtkommerziellen Aktiven im Internet, mit dem Ziel, technische Standards im Internet vorzuschlagen. Es ist eine Organisation, die speziell die Weiterentwicklung technischer Standards des World Wide WEB koordiniert, wie z.B. HTML oder das HTTP-Protokoll, ist das W3-Konsortium (W3C; [www.w3.org](http://www.w3.org)). Als letztes sei die neu gegründete internationale Verwaltungsorganisation Internet Corporation for Assigned Names and Numbers (ICANN) erwähnt, die seit September 2000 u.a. für die Vergabe von Internet-Adressen und Domain-Namen zuständig ist.

### 1.3 Die Architektur des Internet

Das Internet ist eine dezentrale Zusammenschaltung mehrerer Netzwerke. Der Datenaustausch geschieht über das paketvermittelnde Internet Protokoll (IP) der Protokoll-Familie TCP/IP. Die einzelnen Netzwerke haben eine hierarchische Gliederung (LAN, regionales Netzwerk, überregionales Netzwerk) [Abb1]. Diese Struktur hat sich entwickelt und ist nicht definiert. Wichtigstes Prinzip ist die Sicherstellung der Funktionalität durch die Endrechner und nicht durch das Netz, wie etwa beim Telefon. Damit muss man sich nur auf sich und nicht das Netzwerk verlassen und durch die einfachen und robusten Routingtabellen ist auch die Komplexität der Fehler geringer [Hui96]. Jeder der Rechner (Hosts), der über das Internet erreichbar ist, hat eine eindeutige Adresse, die MAC-Adresse (Hardware-Adresse eines Gerätes, das an ein Netzwerk angeschlossen ist). Aber auch die IP-Adresse (Internet-Adresse) muss eindeutig sein. Diese kann auch temporär vergeben werden. Die Rechner sind hierarchisch in Domänen (Domains) strukturiert. Die Hosts können zu einer gegebenen IP-Adresse auch einen lesbaren Namen haben. Das Domain Name System (DNS) erlaubt das Auffinden der IP-Adressen zu einem Namen. Anhand dieser Adresse können Datenpakete zu ihrem Bestimmungsort geleitet werden. Das Weiterleiten und Umleiten von Datenpaketen übernehmen spezielle Netzwerkknoten, die Router. Router sind Geräte, die unterschiedliche Netze auf der Ebene 3 des ISO/OSI-Modells (Kapitel 4) verbinden. Router sind nicht protokoll-transparent, sondern müssen in der Lage sein, alle verwendeten Informationsblöcke zu erkennen. Die logischen Adressen in einem Netzwerk werden vom Router ausgewertet. Damit werden Routingtabellen angelegt, um den optimalen Weg (Route) vom Sender zum Empfänger zu finden. Um die Routingtabellen auf dem Laufenden zu halten, tauschen die Router untereinander Informationen mit Hilfe von Routingprotokollen aus (z.B. OSPF, RIP). Moderne Router sind heute in der Regel Multiprotokollrouter. Das heißt, dass an sie unterschiedlichste LAN- und WAN-Netzwerke angeschlossen werden können, deren Vermittlungsprotokolle sie verstehen und ineinander umsetzen können. Neben der reinen Wegwahl haben viele Router heute auch die Fähigkeit, die PDUs (Protokolldateneinheit einer Schicht im OSI-Modell) höherer Netzwerkschichten auswerten zu können. Dadurch können Firewall-Funktionalitäten realisiert werden, indem über ACLs genau konfiguriert wird, welche Rechner mit welchen Protokollen über den Router kommunizieren dürfen. Die Hosts eines lokalen Netzwerks (Local Area Network, LAN), sind über Gateways an die Zugriffspunkte (Access-Points) regionaler Netzwerke verbunden. Die Verbindung von Gateway und Access-Point kann hier in Deutschland z.B. über Stickleitungen der Deutschen Telekom AG erfolgen. Ein Gateway ist eine Hard- und/oder Softwarelösung, die eine Verbindung von inkompatiblen Netzwerken schaffen kann. Ein Gateway konvertiert bis zu sieben Schichten des ISO/OSI Modells, sodass vollkommen verschiedene Netzwerksysteme miteinander verbunden werden können.

Die Verbindung der Knoten regionaler Netzwerke geschieht meist durch Leitungen des Netzbetreibers. Ein Beispiel für regionale Netzwerke ist das B-Win (Breitband Wissenschaftsnetz) des DFN. Die Knoten der regionalen Netzwerke bilden ein Wide Area Network (WAN). Dieses Netzwerk ist mit den Access-Points überregionaler Netzwerke verbunden, die durch internationale Datenverbindungsstrecken vernetzt sind. Beispiele sind die internationale Anbindung des DFN und die Netzwerkkarten des UUNET [Com00]. Ist ein Netz auf das Stadtgebiet oder einen Ballungsraum beschränkt und sollen hohe Übertragungsgeschwindigkeiten ermöglicht werden, wird ein Metropolitan Area Network eingesetzt. MAN sind in der Regel in der Lage, Multi-Media-Daten wie Sprache und Video in Echtzeit zu übertragen. Das Protokoll, das für Metropolitan Area Networks (MAN) standardisiert ist, ist der Distributed Queue Dual Bus. DQDB soll ausreichend Bandbreite bereitstellen, um für die Übertragung von Bildern, Daten und Sprache gerüstet zu sein. MAN basiert auf einer doppelten Bus-Topologie mit Daten- und Sprachzellen. Mögliche Geschwindigkeitsstufen sind 1,5 MBit/s, 2,048 MBit/s, 34 MBit/s und 45 MBit/s. Übertragungsgeschwindigkeiten bis über 600 MBit/s sind geplant. Der Anschluss an das MAN wird über Bridges und Router abgewickelt. Die Bridge ist ein Gerät zum Verbinden zweier gleichartiger Netze oder Netzsegmente. Sie werden eingesetzt, um große Netze physikalisch zu entkoppeln. Man unterscheidet Local-Bridges und Remote-Bridges. Remote-Bridges werden eingesetzt, wenn zwischen den zwei Netzwerken größere Strecken über Datenfernverbindungen zu überbrücken sind. Local-Bridges dienen hauptsächlich der Lasttrennung. Bridges arbeiten auf Schicht 2 des ISO/OSI-Modells der offenen Kommunikation und sind von höheren Protokollen unabhängig. Bridges übertragen Datenpakete anhand der MAC-Adressen. Über Filterfunktionen kann der Datenverkehr über das Netz eingeschränkt werden, indem Datenpakete nur in die relevanten Netzsegmente gelangen.

Die DQDB-Technologie könnte in Zukunft durch die verstärkte Installation von B-ISDN-Systemen auf Basis von Asynchronous Transfer Mode (ATM) an Bedeutung verlieren. ATM ermöglicht einer theoretisch unbegrenzten Anzahl von Netzbenutzern dedizierte Hochgeschwindigkeitsfestverbindungen sowohl untereinander als auch mit Servern. Als Vermittlungstechnik ("Cell Relay") soll sie im Breitband ISDN (B-ISDN) oder auch im Switched Multimegabit Data Service (SMDS-Netze) eingesetzt werden. Aber auch im LAN-Bereich macht ATM mit Hilfe von ATM-LAN-Emulationen mehr und mehr von sich reden. ATM basiert auf einem schnellen Zell-Switching (Pakete fester Größe: 48+5 Byte), das variable Bitraten (je nach Anforderung) ermöglicht. Im Zusammenhang mit ATM spricht man nicht von Nachrichtenpaketen, sondern von Nachrichtenblöcken oder Nachrichtenzellen.

Personen, die keinen Anschluss ans Internet besitzen, aber trotzdem als Warenanbieter an dem E-Commerce im Internet teilhaben wollen, können sich von einem Internet Service Provider (ISP) 'hosten' lassen: D.h. der ISP stellt die Rechner, Datenspeicher, Netzwerkanbindung und Software, evtl. sogar Entwicklungspersonal für die gewünschten Dienste zur Verfügung. Einzelpersonen können sich mit Ihrem Computer und über ein Modem bei einem ISP einwählen und z.B. über das Point-To-Point-Protocol (PPP) an das Internet als Client verbinden. Zusammengefasst ergibt das ein Netz von separaten Netzen verbunden an Exchangepunkten (NAP). Überregionale Provider verwalten komplette Internet Routing Tabellen (derzeit über 100.000 Adressen), während regionale und lokale Provider nur Teilmengen verwalten und mit Default-Adressen zum Uplink hin arbeiten [Rau00]. Die Technologie der Netzwerkverbindung hängt von den Umständen des Einsatzes und dem verwendeten Medium ab. Als Medium erlauben Glasfaserkabel die bisher schnellste Verbindung. Herkömmliche Kupferleitungen machen hingegen einen Großteil der Leitungen aus. Es existieren einige Transkontinental-Kabel,

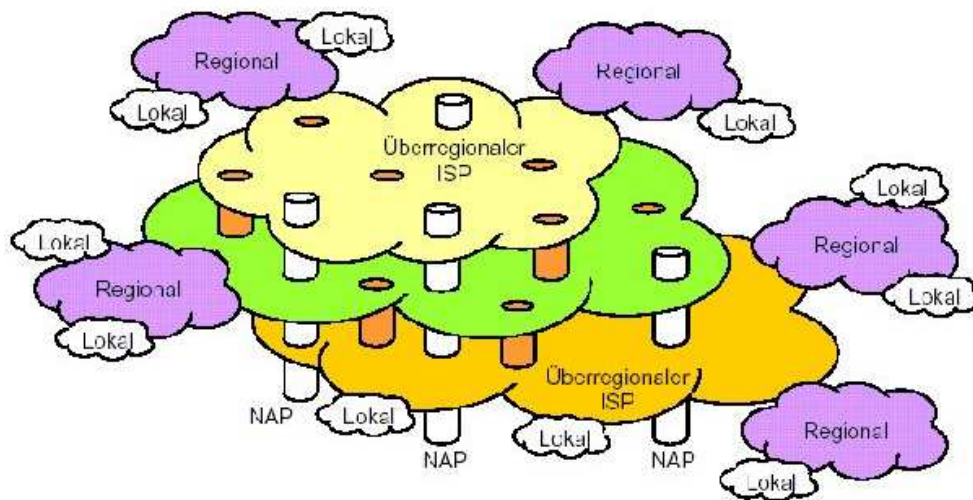


Abbildung 1.1: Internetarchitektur [Rau00], NAP: network access, ISP: internet service provider

die von speziellen Schiffen aus gewartet werden. Neben Kabelverbindungen sind für Fernverbindungen auch Satellitenverbindungen und Richtfunkstrecken im Einsatz. Auf lokaler Ebene werden neben Kupfer und Glasfaser in einigen Fällen Funkverbindungen in Form von Wave-LANs eingesetzt. Diese Form der Vernetzung eignet sich besonders für die ortsungebundenen Laptops. Die verschiedenen Netzwerkanbieter wollen ihren Nutzen maximieren, sodass keine Isolierung, sondern ein einheitliches Vernetzen auch in Zukunft der Fall sein wird, zumal die Anwendungsbreite ständig zunimmt.

## 1.4 Das ISO/OSI Referenzmodell

Nachdem unter Punkt 3 der Aufbau des Internets näher erläutert wurde, soll im Kapitel 4 der Datenaustausch zwischen Host A und B näher betrachtet werden.

Um den Aufbau von Netzwerksystemen und der darin verwendeten Netzwerk-protokollen zu beschreiben, wird oft das ISO/OSI (International Standardisation Organisation / Open System Interconnection) 7-Schichten-Modell herangezogen, weil das ISO/OSI-Modell zu einem klaren Aufbau besitzt und zum anderen als fast offizieller Standard gilt. Die Protokolle in der Praxis sind allerdings durch Erfahrungen entstanden und nicht durch Orientierung am Schichtenmodell. Das Schichtenmodell ist so aufgebaut, dass die Dienste der höheren Schichten jeweils auf die nächst-niedrigeren zurückgreifen. Dabei ist die unterste Schicht hardwareorientiert und beschäftigt sich unter anderem damit, wie die Übertragung stattfindet, wobei die oberste Schicht sich mit den Aufgaben des Users beschäftigt, wie z. B. die Übertragung von Dateien und E-Mails.

Die folgende Abbildung 2 zeigt grafisch das Referenzmodell. Im Anschluss werden die einzelnen Schichten und ihre Aufgaben näher erläutert.

*Aufgaben:*

Schicht 1: Bitübertragung

- transparente Übertragung von Bit-Sequenzen über ein entsprechendes Medium,

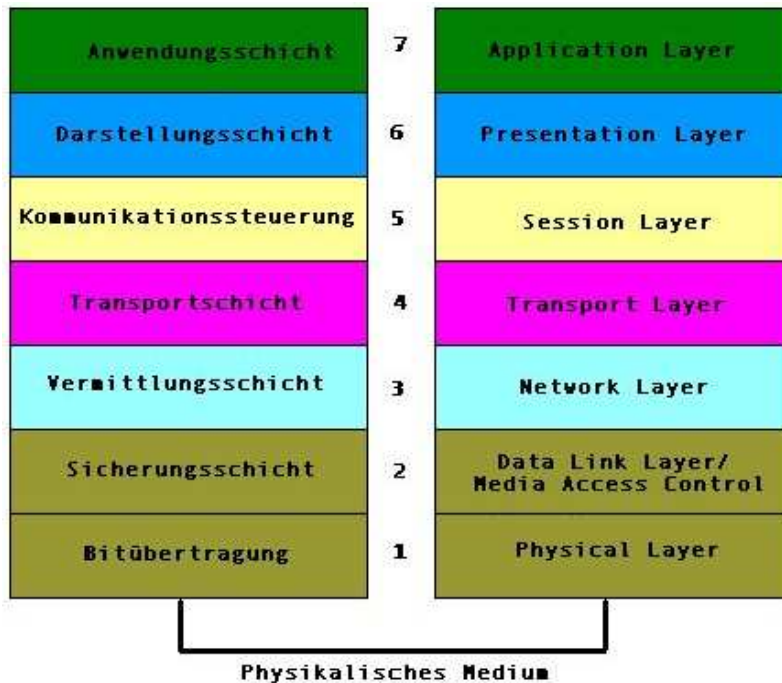


Abbildung 1.2: ISO/OSI Referenzmodell [Dan00]

- Unterstützung unterschiedlicher Übertragungsraten,
- keine explizite Fehlerbehandlung

Die Schicht 1, ist eine reine Hardwareschicht [Dan00].

Schicht 2: Sicherungsschicht

- Aufbau einer (möglichst) fehlerfreien Verbindung zwischen Endsystem und Netzzugang,
- Zusammenfassung der Daten in Blöcken
- Blocksynchronisation
- Fehlererkennung und Korrektur

Schicht 2 basiert ebenfalls auf Hardware.

Schicht 3: Vermittlungsschicht

- Auswahl und Steuerung des Transportnetzes,
- Wegewahl,
- Vermittlung,
- Kopplung unterschiedlicher Transportnetze und Multiplexing eines physikalischen Netzzugangs

Schicht 3 ist somit für das Ankommen der Daten am richtigen Ort verantwortlich.

Schicht 4: Transportschicht

- Aufbau und die Unterhaltung einer (Virtuellen) Verbindung zwischen zwei Endsystemen,

- Bereitstellung eines netzunabhängigen Transportmechanismus,
- Adressierung eines Endteilnehmers und
- Ordnung der Pakete.

Beispiele hierfür sind TCP und UDP.

Schicht 5: Kommunikationssteuerungsschicht

- Einrichtung und Steuerung von Sitzungen,
- Zugangskontrolle,
- Definition von Aufsetzpunkten

Schicht 5 erstellt sozusagen die Vorbedingungen für eine Datenübertragung.

Schicht 6: Darstellungsschicht

- globale einheitliche Informationsdarstellung und Interpretation,
- Verschlüsselung und Datenkompression

Diese Schicht arbeitet eng mit dem Betriebssystem zusammen.

Schicht 7: Anwendungsschicht

Die Anwendungsschicht ist die Ebene der spezifischen Anwendungen, wie z.B. Dateitransfer, Jobtransfer, Nachrichtensysteme oder verteilte Datenbanken.

An dem Modell sind jedoch einige Kritikpunkte anzusetzen. Die Notwendigkeit einer unter allen Umständen korrekten Datenübertragung ist nicht immer sinnvoll. Diese Überprüfung kann auch die Anwendung durchführen. Es gibt auch keine eindeutige Zuordnung der Sicherungsaufgaben. Desweiteren ist eine einheitliche Protokollwelt Utopie. Es mangelt an Flexibilität gegenüber neuen Technologien und ist zudem zu stark am Client/Server - Modell orientiert. Abschließend möchte ich noch die fehlende Festlegung von Programmierschnittstellen nennen [Rze00].

## 1.5 Das TCP/IP Protokoll

### 1.5.1 Das Referenzmodell

Im vorherigen Abschnitt wurde das OSI-Referenzmodell vorgestellt. In diesem Abschnitt soll nun das Referenzmodell für die TCP/IP-Architektur vorgestellt werden. Dieses Modell entstand durch die Weiterentwicklung von Vorschlägen die für das ARPANET gemacht wurden. Es ist nach den beiden wichtigsten Protokollen TCP und IP benannt. Da es vor der OSI-Standardisierung entstand, floss es auch in diese mit ein. Im Gegensatz zum OSI-Referenzmodell besteht das TCP/IP-Referenzmodell nur aus vier Schichten, welche sich aber nahezu in die sieben OSI-Schichten zuordnen lassen (Abbildung 3). Die Schichten heißen Application Layer, Transport Layer, Internet Layer und Network Layer. Als Ziele der Architektur wurden bei der Entwicklung die folgenden definiert: Unabhängigkeit von der verwendeten Netzwerk-Technologie, sowie von der Architektur der Hostrechner. Eine universelle Verbindungsmöglichkeit im gesamten Netzwerk sollte geschaffen und Ende-zu-Ende-Quittungen definiert werden. Standardisierte Anwendungsprotokolle waren ein weiteres Ziel.

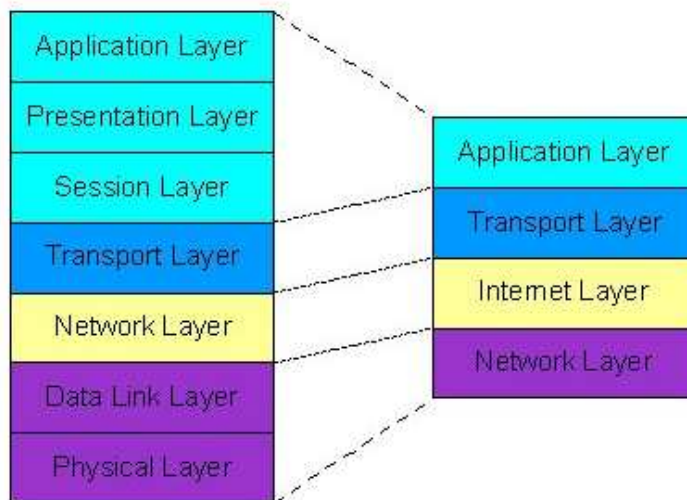


Abbildung 1.3: Vergleich des OSI-Referenzmodells mit dem TCP/IP-Referenzmodell

Verarbeitungsschicht (application layer): Die Verarbeitungsschicht enthält die höchstschichtigsten Protokolle des Referenzmodells. Zu diesen Protokollen zählen u.a., wie in Abb. 5 zu sehen: TELNET (für virtuelle Terminals), FTP (Dateitransfer) und SMTP (zur Übertragung von E-Mail). Außerdem etablierten sich noch weitere Protokolle wie z.B. HTTP (Hypertext Transfer Protocol) und DNS (Domain Name Service).

Transportschicht (transport layer): Die Transportschicht entspricht wie in Abb. 3 ersichtlich, der des OSI-Modells. Die Kommunikation zwischen den Quell- und Zielhosts soll im TCP/IP-Referenzmodell auf dieser Schicht durch zwei Ende-zu-Ende-Protokolle realisiert werden: das Transmission Control Protocol (TCP) und das User Datagram Protocol (UDP) (siehe dazu auch Abbildung 4. TCP ist ein zuverlässiges verbindungsorientiertes Protokoll, durch das ein Bytestrom fehlerfrei einen anderen Rechner im Internet übermittelt werden kann. Verbindungsorientiert heißt hier, dass bei der Übertragung von Nutzdaten eine Verbindung zwischen Sender und Empfänger hergestellt werden muss. Dies geschieht durch Aufbau eines logischen Kanals. UDP ist ein unzuverlässiges verbindungsloses Protokoll, das vorwiegend für Abfragen und Anwendungen in Client/Server-Umgebungen verwendet wird, in denen es in erster Linie nicht um eine sehr genaue, sondern schnelle Datenübermittlung geht (z.B. Übertragung von Sprache und Bildsignalen)[Hui96].

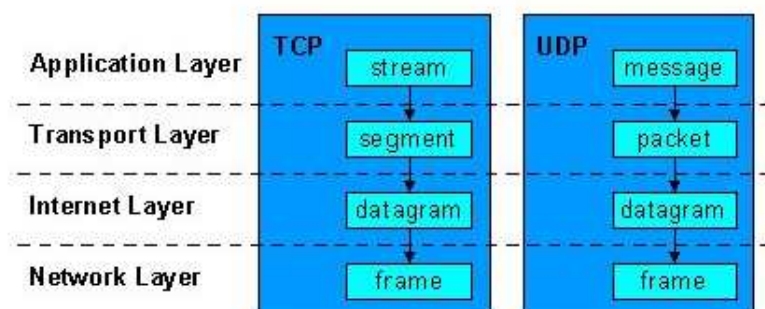


Abbildung 1.4: Vergleich TCP und UDP



Internetschicht (internet layer): Die Internetschicht hat die Aufgabe Pakete richtig zuzustellen. Die Internetschicht im TCP/IP-Modell definiert nur ein Protokoll, das IP (Internet Protocol). Derzeit wird es von allen Rechnern verwendet. Dabei spielt das Routing der Pakete eine wichtige Rolle. Das Internet Control Message Protocol (ICMP) ist als fester Bestandteil jeder IP-Implementierung auch hier aufgehängt und dient zur Übertragung von Diagnose- und Fehlerinformationen für das Internet Protocol.

Netzwerkschicht (network layer): Die Netzwerkschicht wird in der Definition nicht weiter erläutert. Fest steht eigentlich nur, dass zur Übermittlung von IP-Paketen ein Host über ein bestimmtes Protokoll an ein Netz angeschlossen werden muss. Dieses Protokoll ist im TCP/IP-Modell nicht weiter definiert und weicht von Netz zu Netz und Host zu Host ab. Da es aber den unteren OSI-Schichten entspricht, kann man auch die Hardware mit zum Verbindungsaufbau zählen.

In Abbildung 5 sind die grundlegenden TCP/IP-Protokolle für alle Schichten grafisch dargestellt.

OSI	TCP/IP					
Anwendung	RIP	TELNET	RLOGIN	NFS, NIS	Application	
Darstellung	HELLO	FTP	RSHELL	NICS TIME		
Sitzung	DNS	TFTP	REXEC	XDR		
Transport	TCP		UDP		Transmission	
Vermittlung	IP	ICMP	ARP	RARP	sonstige	Internet
Sicherung	SLIP, X.25, IEEE 802.x, usw.				Netzwerk	
Bitübertragung						

Abbildung 1.5: Grundlegende TCP/IP-Protokolle

## 1.5.2 Internet Protokoll

Das Internet Protokoll (IP) ist für Anwahl und Steuerung des Internet zuständig. IP stellt die Basisdienste für die Übermittlung von Daten in TCP/IP-Netzen bereit. Hauptaufgaben des Internet Protokolls sind die Adressierung von Hosts und das Fragmentieren von Paketen. Dabei versucht das Protokoll die Pakete bestmöglich ("best effort") von der Quelle zum Ziel zu befördern, unabhängig davon, wie viel verschiedene Netze dabei durchquert werden müssen. Eine Garantie für diese Zustellung kann jedoch nicht gegeben werden. Das Internet Protokoll enthält keine Funktionen für die Ende-zu-Ende-Sicherung oder für die Flusskontrolle. Die Funktionen von IP umfassen im wesentlichen die Definition des Adressierungsschemas, die Definition von Datengrammen, welche die Basis-einheiten für die Übermittlung von Daten im Internet bilden, sowie das Routing von Datagrammen durch das Netz. Wie oben erwähnt, zählen auch die Fragmentierung und das Zusammensetzen von Datengrammen, sowie die Übermittlung der Daten von der Transportebene zu den Aufgaben der Netzwerkschicht. IP ist ein verbindungsloses Protokoll, d.h. zur Datenübertragung wird keine Ende-zu-Ende-Verbindung der Kommunikationspartner etabliert. Ferner ist IP ein unzuverlässiges Protokoll, da es von sich aus über keine

Mechanismen zur Fehlererkennung und -behebung verfügt. Das Übermitteln von IP Paketen geschieht wie folgt [Hui96]: Der Host will ein Paket abschicken und muss deshalb den “next hop” bestimmen. Dies ist der nächste Rechner, zu dem das Paket soll. Dazu prüft er, ob die Adresse im Subnetz ist, d.h. die maskierten Bits übereinstimmen. Wenn dies der Fall ist, kann er das Paket sofort dorthin schicken, wenn nicht muss es zu einem Router. Wenn die IP-Adresse bekannt ist, muss die dazugehörige Netzwerkadresse ermittelt werden. Dies übernimmt ARP, das Adress Resolution Protocol. Dies ist ein Protokoll der TCP/IP-Protokollfamilie. Es implementiert eine Methode, mit der für einen Host, dessen IP-Adresse bekannt ist, die zugehörige physikalische Adresse (Ethernet-Adresse) ermittelt wird. Der Sender verschickt ein ARP-Paket per Broadcast und wartet darauf, dass ihm die physikalische Adresse zurückgeschickt wird. Jeder Host führt einen Cache mit bekannten Übersetzungen, um die Umsetzzeit und die notwendigen Broadcasts zu minimieren. Der Cache muss allerdings zyklisch aktualisiert werden. Es gibt auch ARP-Server die das Antworten übernehmen können. In einem solchen Fall wird von einem Proxy ARP gesprochen. Ist die Adresse des Ziels nicht lokal, wird der Router, der dem Ziel am nächsten ist, angerufen. Dieser wird über ICMP Nachrichten ausfindig gemacht. Es kann eine Host “Anfrage” gestartet werde, oder ein Router hat sich durch ein “Angebot” - Broadcast schon angeboten. Wenn der Standardrouter merkt, dass andere Router besser sind kann er auch eine ICMP-Umleitungsnachricht mit der Routeradresse an den Host schicken. Der Grund für die Entwicklung von TCP/IP-Protokollen war die Übermittlung von Daten über ein paketvermittelndes Netz (wie dem ARPANET). Ein Paket ist ein Block vom Daten und die Informationen, die notwendig sind, um sie dem Empfänger zuzustellen. Dies ist nichts anderes als ein Paket im herkömmliche Sinn bei der Post, das Paket enthält die Daten, auf dem Paket ist die Adresse des Empfängers notiert. Das Datagramm ist das Paketformat, das vom Internet Protokoll definiert ist. Ein IP-Datagramm besteht aus einem Header (siehe Abbildung 6) und den zu übertragenden Daten. Der Header hat einen festen 20 Byte großen Teil, gefolgt von einem optionalen Teil variabler Länge. Der Header umfasst alle Informationen, die notwendig sind, um das Datagramm dem Empfänger zuzustellen. Ein Datagramm kann theoretisch maximal 64 KByte groß sein, in der Praxis liegt die Größe ungefähr bei 1500 Byte (das hängt mit der maximalen Rahmengröße des Ethernet-Protokolls zusammen). Eine genaue Beschreibung liefert Anhang A.

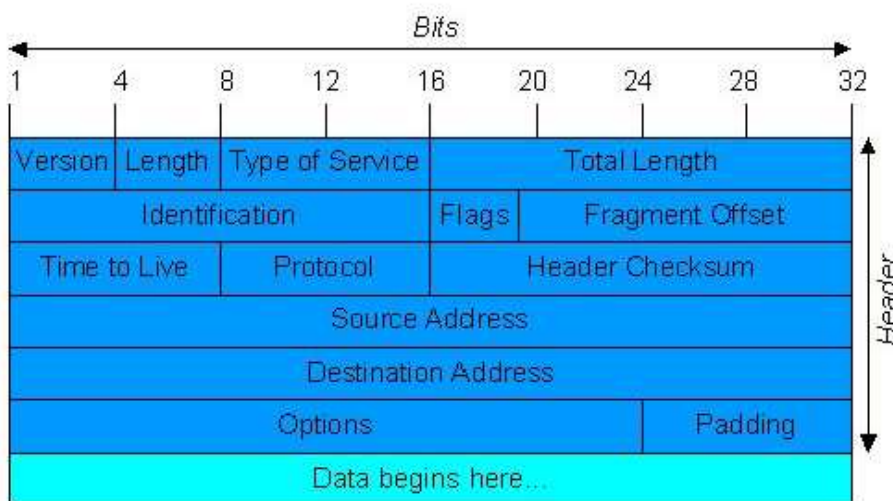


Abbildung 1.6: Der IP-Header.

Das Internet Protokoll muss dazu in der Lage sein, die Größe der Datengramme dem jeweiligen Netz anzupassen. Dies liegt darin begründet, dass Datengramme über jede Art von Netzwerk verschickt werden sollen. Jedes Netzwerk besitzt eine sogenannte maximale Paketgröße (Maximum Transfer Unit - MTU). Diese bezeichnet die maximale Paketgröße, mit der über das Netz verschickt werden kann. So dürfen z.B. Pakete, die über ein X.25-Netz verschickt werden nicht größer als 128 Byte sein. Ein Ethernet-Paket darf die Größe von 1500 Byte nicht überschreiten. Falls die MTU eines Übertragungsmediums kleiner ist als die Größe eines versendeten Pakets, so muss dieses Paket in kleinere Pakete aufgeteilt werden [Hui96].

Ein Paket kann auf dem Weg vom Quell- zum Zielhost mehrere unterschiedliche Netzwerke mit unterschiedlichen MTUs durchlaufen. Aus diesem Grund genügt es nicht, dass die Protokolle der Transportschicht nun von sich aus einfach kleinere Pakete versenden. Es muss ein flexibles Verfahren angewendet werden, dass bereits auf der Internet-Schicht kleiner Pakete erzeugen kann. Dieses Verfahren wird Fragmentierung genannt.

Fragmentierung ist das Zerteilen von Paketen durch das IP-Protokoll. Dazu muss jeder Netzwerkknosens (sei es ein Router, ein Host o.ä.) in der Lage sein. Jedes empfangende Internetprotokoll muss diese Fragmente auch wieder zum ursprünglichen Paket zusammensetzen können.

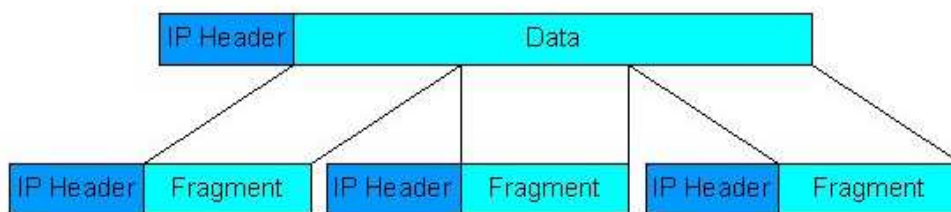


Abbildung 1.7: Fragmentierung eines IP-Pakets.

Jedes Fragment eines zerteilten Pakets erhält einen eigenen, vollständigen IP-Header. Über das Identifikationsfeld im Header können alle Fragmente eines Pakets wiedererkannt werden. Die einzelnen Fragmente eines Pakets können durchaus unterschiedliche Wege auf dem Weg zum Zielhost nehmen. Die Lage der Daten eines Fragments innerhalb der Gesamtnachricht wird mit Hilfe des Fragment Offset-Feldes ermittelt.

### 1.5.3 IP-Adressierung

Zur Adressierung eines Kommunikationspartners in Form eines Applikations-programms müssen beim Durchlaufen der vier TCP/IP-Schichten auch vier verschiedene Adressen angegeben werden. D.h. eine Netzwerkadresse (z.B. eine Ethernet-Adresse) mit einer Internet-Adresse, die Nummer des Transportprotokolls und eine Portnummer. Drei dieser Adressen finden sich als Felder im IP-Header: die Netzwerkadresse mit Internet-Adresse und die Transportprotokoll-Nummer.

Jeder Host und Router im Internet hat eine 32-Bit lange eindeutige IP-Adresse. Computer die an mehreren Netzen angeschlossen sind, haben in jedem Netz eine andere IP-Adresse. Die Netzwerkadressen werden zentral vergeben, um Adresskonflikte zu vermeiden. Dies hat wie in Kap.2.2 erwähnt das NIC bzw. später die IANA übernommen. Die Adressen werden nicht einzeln zugeordnet, sondern nach Netzklassen vergeben. Beantragt man IP-Adressen für ein

Netz, so erhält man nicht für jeden Rechner eine Adresse zugeteilt, sondern einen Bereich von Adressen, der selbst zu verwalten ist.

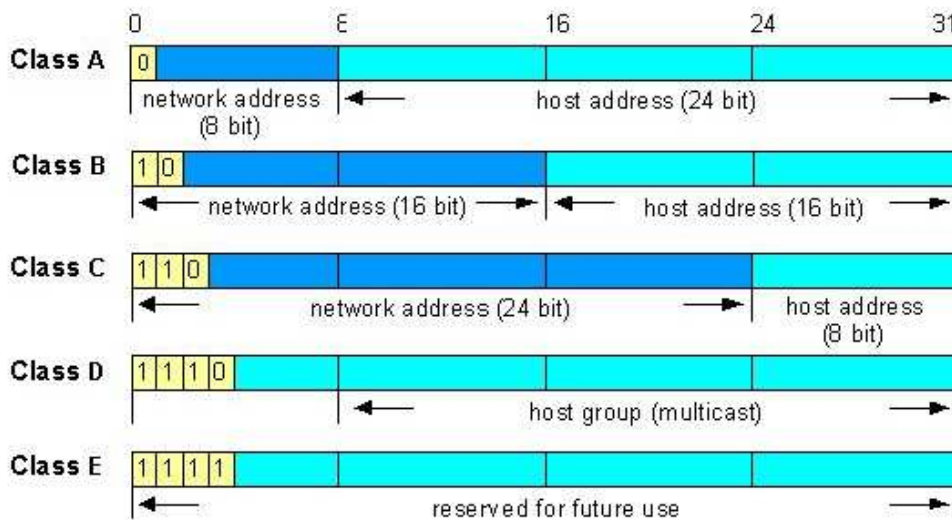


Abbildung 1.8: IP-Adressformate.

Wie die Abbildung 8 zeigt, sind IP-Adressen in verschiedene Klassen, mit unterschiedlich langer Netzwerk- und Hostadresse, eingeteilt. Die Netzwerkadresse definiert das Netzwerk, in dem sich ein Host befindet: alle Hosts eines Netzes haben die gleiche Netzwerkadresse. Die Hostadresse identifiziert einen bestimmten Rechner innerhalb eines Netzes. Eine IP-Adresse identifiziert keinen bestimmten Computer, sondern eine Verbindung zwischen einem Host und einem Netz. Einem Computer mit mehreren Netzanschlüssen (z.B. ein Router) muss für jeden Anschluss eine IP-Adresse zugewiesen werden [Com00]. IP-Adressen werden normalerweise nicht als Binärzahl, sondern in gepunkteten Dezimalzahlen geschrieben. In diesem Format wird die 32-Bit große Zahl in 4 Byte unterteilt, die mit Punkten voneinander getrennt sind. Wie zuvor beschrieben sind IP-Adressen in Klassen unterteilt. Der Wert der ersten Bits gibt die Adressklasse an. Näheres dazu finden Sie im Anhang B.

Für einige Anwendungen, z. B. häufiger verwendete Fehlerbehandlungen, werden einige Netzadressen reserviert. Die Adressen mit der Netznummer 0 zum Beispiel, beziehen sich auf das aktuelle Netz. Mit einer solchen Adresse können sich Hosts auf ihr eigenes Netz beziehen, ohne die Netzadresse zu kennen zu müssen (allerdings muss bekannt sein, um welche Netzklasse es sich handelt, damit die passende Anzahl Null-Bytes gesetzt wird). Die 127 steht für das Loopback Device eines Hosts. Pakete, die an eine Adresse der Form 127.x.y.z gesendet werden, werden nicht auf einer Leitung ausgegeben, sondern lokal verarbeitet. Dieses Merkmal wird häufig zur Fehlerbehandlung benutzt. Neben einigen Netzadressen sind auch bestimmte Hostadressen für spezielle Zwecke reserviert. Bei allen Netzwerkklassen sind die Werte 0 und 255 bei den Hostadressen reserviert. Eine IP-Adresse, bei der alle Hostbits auf Null gesetzt sind, identifiziert das Netz selbst. Die Adresse 80.0.0.0 bezieht sich so z.B. auf das Klasse A Netz 80, die Adresse 128.66.0.0 bezieht sich auf das Klasse B Netz 128.66. Eine IP-Adresse, bei der alle Host-Bytes den Wert 255 haben, ist eine Broadcast-Adresse. Eine Broadcast-Adresse wird benutzt, um alle Hosts in einem Netzwerk zu adressieren. Desweiteren werden oft Subnetze eingesetzt.

Subnetze zerlegen ein Netzwerk in mehrere kleinere Netzwerke. Die separaten Netzwerke sind meistens durch Router, verbunden. Wenn sich der Administrator einige Bits vom Host-Abschnitt der Adresse des Hauptnetzwerkes sozusagen “borgt”, so muss er TCP/IP mitteilen, welche Bits des Host-Abschnitts “geborgt” wurden, um als Netzwerkadresse zu dienen. Hier kommt die Subnetmask zum Einsatz. Eine Subnetmask besteht genauso wie die IP-Adresse aus 32 Bits. Die Bits der Netzwerk-adresse sind auf den Wert 1, die Bits der Hosts-Adresse sind auf 0 gesetzt. Der Netzwerknummernteil einer IP-Adresse wird nun mit Hilfe einer Subnetmask isoliert.

### 1.5.4 ICMP

Das Internet Control Message Protocol (ICMP) ist Bestandteil jeder IP-Implementierung und hat vor allem die Aufgabe Fehler- und Diagnoseinformationen für IP zu transportieren.

Oft wird ein ICMP auch für Testzwecke verwendet, etwa um zu ermitteln ob ein Host derzeit empfangsbereit ist.

Da ICMP sehr unterschiedliche Informationen zu transportieren hat, ist nur der Grundaufbau des ICMP-Headers gleich, siehe Abbildung 9. Die Bedeutung der einzelnen Felder im Protokollkopf wechselt jedoch.

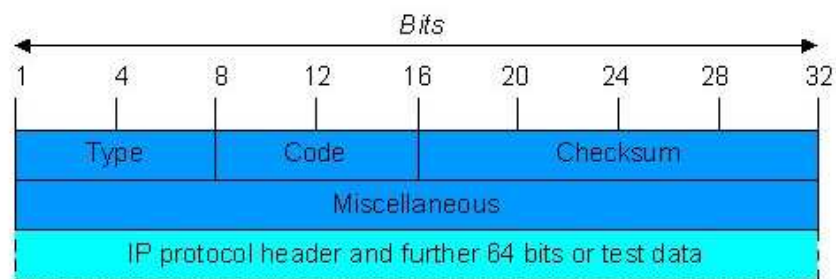


Abbildung 1.9: Der ICMP-Header (allgemeiner Aufbau)

Jeder ICMP-Nachrichtentyp wird in einem IP-Datengramm eingekapselt. Die derzeit wichtigsten ICMP-Nachrichtentypen sind im Anhang C aufgeführt.

IP verwendet ICMP, wie schon erwähnt, zum Versenden von Fehler- und Diagnosemeldungen, während ICMP zur Übertragung seiner Nachrichten IP benutzt. Das bedeutet, in der Praxis, wenn eine ICMP-Nachricht verschickt werden muss, wird ein IP-Datengramm erzeugt und die ICMP-Meldung in den Datenbereich des IP-Datengramms eingekapselt (s. Abb. 10).

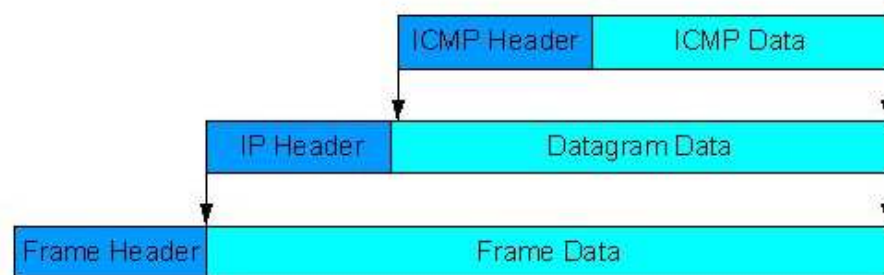


Abbildung 1.10: ICMP-Nachrichten-Einkapselung

Das Datagramm wird dann wie üblich versendet. Eine ICMP-Nachricht wird immer als Antwort auf ein Datagramm verschickt. Entweder ist ein Datagramm auf ein Problem gestoßen oder das Datagramm enthält eine ICMP-Anfrage, auf die eine Antwort versendet werden muss. In beiden Fällen sendet ein Host oder Router eine ICMP-Nachricht an die Quelle des Datagramms zurück. Näheres zu den ICMP Nachrichtentypen ist in Anhang C nachzulesen.

### 1.5.5 TCP

Die beiden wichtigsten Protokolle der Transportschicht sind TCP und UDP. Die Aufgabe von TCP besteht in der Bereitstellung eines sicheren Ende-zu-Ende Transports von Daten im Netzwerk, u.a. wird dabei die Reihenfolge der Pakete wiederhergestellt. Das Transmission Control Protocol stellt die Zuverlässigkeit der Datenübertragung mit einem Mechanismus, der als Positive Acknowledgement with Re-Transmission (PAR) bezeichnet wird, bereit. Dies bedeutet, dass das System, welches Daten sendet, die Übertragung der Daten solange wiederholt, bis vom Empfänger der Erhalt der Daten quittiert bzw. positiv bestätigt wird. Die Dateneinheiten, die zwischen den sendenden und empfangenden TCP-Einheiten ausgetauscht werden, heißen Segmente. Ein TCP-Segment besteht aus einem mindestens 20 Byte großen Protokollkopf und den zu übertragenden Daten. In jedem dieser Segmente ist eine Prüfsumme enthalten, anhand derer der Empfänger prüfen kann, ob die Daten fehlerfrei sind. Im Falle einer fehlerfreien Übertragung sendet der Empfänger eine Empfangsbestätigung an den Sender. Andernfalls wird das Datagramm verworfen und keine Empfangsbestätigung verschickt. Ist nach einer bestimmten Zeitperiode (timeout-period) beim Sender keine Empfangsbestätigung eingetroffen, verschickt der Sender das betreffende Segment erneut. Näheres zur Zeitüberwachung siehe [San98]. Kommen IP-Datengramme mit TCP-Daten bei einer Maschine an, werden diese an TCP weitergeleitet und wieder zu den ursprünglichen Byteströmen zusammengesetzt. Die IP-Schicht gibt allerdings keine Gewähr dafür, dass die Datagramme richtig zugestellt werden. Es ist deshalb, wie oben bereits gesagt, die Aufgabe von TCP für eine erneute Übertragung der Daten zu sorgen. Es ist aber auch möglich, dass die IP-Datengramme zwar korrekt ankommen, aber in der falschen Reihenfolge sind. In diesem Fall muss TCP dafür sorgen, dass die Daten wieder in die richtige Reihenfolge gebracht werden. Hierfür verwendet TCP eine Sequenznummer und eine Bestätigungsnummer [Hui96]. Die Segmentgröße wird durch zwei Faktoren begrenzt. Erstens muss jedes Segment, einschließlich des TCP-Headers, in das Nutzdatenfeld des IP-Protokolls passen (65.535 Byte); zweitens hat jedes Netz eine maximale Transfereinheit (MTU - Maximum Transfer Unit), in die das Segment passen muss. Läuft ein Segment durch eine Anzahl von Netzen und trifft dabei auf ein Netz mit einer kleineren MTU, so muss das Segment vom Router, wie unter 5.2 beschrieben, in kleinere Segmente aufgeteilt (fragmentiert) werden. TCP-Segmente ohne Daten sind zulässig und dienen der Übermittlung von Bestätigungen und Steuernachrichten.

Das TCP ist außerdem dafür verantwortlich, die empfangenen Daten an die korrekte Applikation weiterzuleiten. Zur Adressierung der Anwendungen werden auf der Transportebene deshalb sogenannte Portnummern (Kanalnummern) verwendet. Portnummern sind 16 Bit groß; theoretisch kann ein Host somit bis zu 65.535 verschiedene TCP-Verbindungen aufbauen. Eine IP-Adresse zusammen mit der Portnummer spezifiziert einen Kommunikationsendpunkt, einen sogenannten Socket. Die Socketnummern von Quelle und Ziel identifizieren die Verbindung (socket1, socket2). Eine Verbindung ist durch die Angabe dieses Paares eindeutig identifiziert. Möchte z.B. ein Host A eine Verbindung zu einem entfernten Host B aufnehmen, z.B. um den Inhalt einer Webseite anzuzeigen, so wird auf der TCP-Schicht als Zielport die Portnummer 80

für das Hypertext Transfer Protocol (http) angegeben. Die Verwaltung der Portnummern ist von der Internet Assigned Numbers Authority (IANA) [<http://www.iana.org>] übernommen worden. Portnummern sind zum Beispiel: FTP (20), Telnet (23), SMTP (25), DNS (53) und wie bereits erwähnt HTTP (80).

Die folgende Abbildung zeigt den Aufbau des TCP-Protokollkopfs.

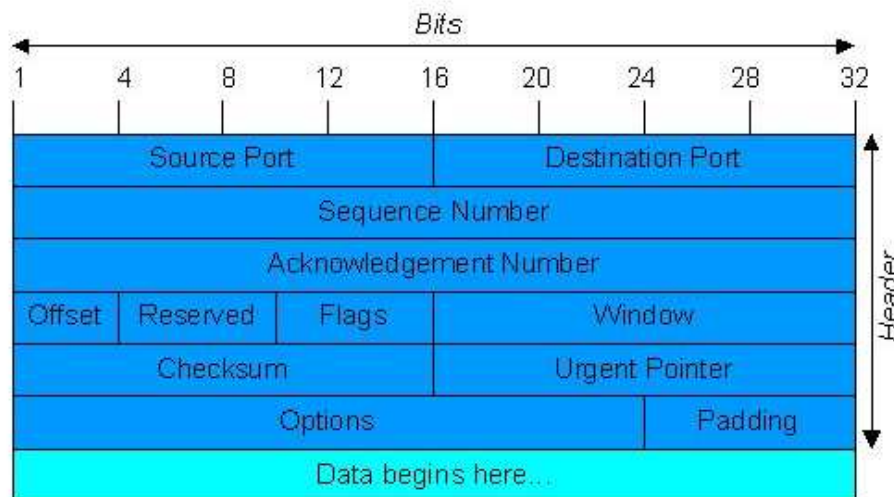


Abbildung 1.11: Der TCP-Header.

Die Bedeutung der Felder des TCP-Headers sind Inhalt des Anhang D.

### 1.5.6 UDP

Das User Datagram Protocol (UDP) ist ein unzuverlässiges, verbindungsloses Protokoll. Wie zuvor schon gesagt, bedeutet unzuverlässig in diesem Zusammenhang nicht, dass die Daten evtl. fehlerhaft beim Zielrechner ankommen, sondern, dass das Protokoll keinerlei Mechanismen zur Verfügung stellt, die sichern, dass die Daten auch tatsächlich beim Zielrechner ankommen. UDP bietet gegenüber TCP den Vorteil eines geringen Protokoll-Overheads. Viele Anwendungen, bei denen nur eine geringen Anzahl von Daten übertragen wird (z.B. Client/Server-Anwendungen, die auf der Grundlage einer Anfrage und einer Antwort laufen), verwenden UDP als Transportprotokoll, da unter Umständen der Aufwand zur Herstellung einer Verbindung und einer zuverlässigen Datenübermittlung größer ist als die wiederholte Übertragung der Daten.

Ein UDP-Segment besteht aus einem Header von 8 Byte, gefolgt von den Daten. Der Header ist in der folgenden Abbildung dargestellt:

Die Sender- und Empfänger-Portnummern erfüllen den gleichen Zweck wie beim Transmission Control Protocol. Sie identifizieren die Endpunkte der Quell- und Zielmaschine. Das Feld für die Länge enthält die Länge des gesamten Datagramms, inklusive der Länge des Protokollkopfes. Die Prüfsumme enthält die Internet-Prüfsumme der UDP-Daten, des Protokollkopfes und des Pseudo-Headers. Das Prüfsummenfeld ist optional. Enthält das Feld eine 0, wurde vom Absender keine Prüfsumme eingetragen und somit findet beim Empfänger keine Überprüfung statt.

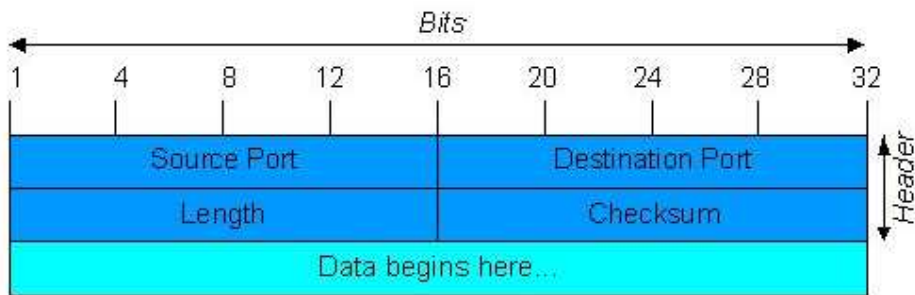


Abbildung 1.12: Der UDP-Header.

Das User Datagram Protocol liefert über die Leistungen des Internet Protokolls hinaus nur Portnummern für die Adressierung der Kommunikationsendpunkte und eine optionale Prüfsumme. Das Protokoll beinhaltet keine Transportquittungen oder andere Mechanismen für die Bereitstellung einer zuverlässigen Ende-zu-Ende-Verbindung. Hierdurch wird UDP allerdings sehr effizient und eignet sich somit besonders für Anwendungen, bei denen es in erster Linie auf die Geschwindigkeit der Datenübertragung ankommt (z.B. verteilte Dateisysteme wie NFS).

## 1.6 Anwendungen im Internet

Interessant wird das Internet durch seine Dienste. Jeder der einen Rechner ans Internet angeschlossen hat, kann im Netzwerk Dienste zur Verfügung stellen und Dienste in Anspruch nehmen. Programme, die Dienste zur Verfügung stellen, heißen Server; Programme, die Dienste in Anspruch nehmen heißen Clients. Das Zusammenspiel von Client und Server nennt sich Client/Server Architektur und wird von entsprechenden Protokollen der TCP/IP Familie definiert. Durch die Vernetzung der Hosts auf denen die Server laufen, stehen die Dienste weltweit zur Verfügung. Einige Protokolle die auf TCP oder UDP aufsetzen, sollen im folgenden vorgestellt werden.

### 1.6.1 FTP

Eine der häufigsten Anwendungen ist der Dateiaustausch. Das File Transfer Protokoll steuert den vom Benutzer angestrebten Datenaustausch zwischen zwei Rechnern. Basierend auf einer TCP-Endsystemverbindung arbeitet es nach dem Client/Server Prinzip, dessen Hauptaufgabe die Trennung von Kontroll- und Datenverbindung ist. Über die Kontrollverbindung können Kommandos gesendet werden, die die Parameter für die Übertragung (Datentyp, Übertragungsmodus, Datenstruktur, ...) und die Operation selber (z.B. Speicher, Löschen, Abholen von Dateien) festlegen. Man kann den Datentyp vorgeben, z.B. ASCII oder Binär aber auch die Übertragungsart z.B. als Block (Daten werden blockweise übertragen), als Stream (Datenstrom ohne Struktur) oder auch als Compressed (einfache Datenkompression bei Übertragung berücksichtigen). Über die Datenverbindung können nur Daten übertragen werden.

### 1.6.2 POP3

Das Post Office Protokoll Version 3 ist ein Protokoll zum Übertragen von E-Mails. Über POP3 können nur Daten empfangen werden. Es wird meist von Systemen (Private PC's) benutzt,



die nicht ständig in Betrieb sind. Diese Systeme deklarieren einen Mail-Relay Host auf einem Rechner, der ständig empfangsbereit ist. Um nun auf die eigenen E-Mails zugreifen zu können, muss der Mail-User Agent eine Verbindung zum Mail-Relay Host herstellen, um von dort die Nachrichten auf das lokale System zu kopieren. Für diese Aufgabe wurde das Post Office Protocol geschaffen. Nach dem gleichen Prinzip arbeitet auch das Interactive Mail Access Protocol (IMAP), jedoch erlaubt es neben dem Laden von Nachrichten auch noch komfortables Verwalten von Nachrichten, die auf dem Mail-Relay Host liegen.

### **1.6.3 SMTP**

Die Aufgabe von SMTP (Simple Mail Transfer Protocol) ist der zuverlässige und effiziente Transport von Mail (verschicken und empfangen). SMTP ist unabhängig von dem unterliegenden Netzprotokoll, normalerweise wird jedoch TCP verwendet und die Kommunikation geht über den TCP Port 25. Die Funktionsweise ist folgende. Der Sender erhält vom Benutzer den Auftrag, eine Mail zuzustellen. Dieser baut nun eine bidirektionale Verbindung zum Empfänger auf, der entweder das direkte Ziel der Nachricht oder ein Nachrichtenübermittler auf der Strecke zum Ziel ist. Im letzteren Falle sorgt dieser für die weitere Verbindung zum Zielrechner. Der Sender schickt Kommandos, die vom Empfänger akzeptiert oder abgelehnt werden können. Dieser ist nun für die Zustellung beim Benutzer verantwortlich.

### **1.6.4 HTTP**

Die Aufgabe des Hypertext Transfer Protocol ist die Regelung der Übertragung von Dokumenten zwischen WWW-Servern und WWW-Clients in einem verteilten Hypermedia Raum über den Well-Known Port 80. Es ist ein sogenanntes zustandloses, objektorientiertes Protokoll, das in der aktuellen Version 4.0 verwendet wird. Auf älteren Clients und Servern findet sich auch noch die Vorgängerversionen, zu welchen die neuere Version meist kompatibel ist, die jedoch wesentlich erweitert wurde. Die vereinfachte Darstellung der Arbeitsweise des Protokolls: Zu Beginn initiiert der WWW-Client eine Verbindung (Connect) zum WWW-Server. Wenn der Client die Verbindung bestätigt, wird normalerweise über den TCP-Port 80, es kann aber von dem Protokoll auch ein nicht reservierter Port angegeben werden, eine Verbindung aufgebaut. Nun kann der WWW-Client mittels eines „request“ Daten anfordern, die der WWW-Server mittels „response message“ wenn möglich liefert. Dieser Vorgang kann fortgesetzt werden, bis mittels dem Befehl „close“ der Dialog (von welcher Seite auch immer) beendet wird.

### **1.6.5 Telnet**

Schon immer war es möglich, über das Internet entfernte Rechner zu bedienen. Zunächst über eine einfache Textoberfläche: Das immer noch verwendete Telnet, später über die grafische Benutzungsoberfläche X-Window. In der Zeit des Webs, gibt es spezielle Applikationsserver, die die Oberfläche von Programmen über einen Web-Browser zugänglich machen. Allerdings: Je komplexer diese Anwendungen werden, desto fehleranfälliger sind Sie auch. Das ursprüngliche Remote Terminal Login (Telnet) ist das Protokoll für virtuelle Terminals. Es erlaubt den Zugriff auf einen am Netz angeschlossenen Rechner in Form von Terminalsitzungen (remote login). Auch dieses Protokoll basiert auf dem Client/Server Prinzip: Auf dem Server arbeitet ein Telnet-Dämon, der auf TCP-Port 23 Anfragen von der Client-Seite mittels dem Kommando telnet erwartet.

### 1.6.6 SNMP

Das Simple Network Management Protocol ist zum Transport von Management-Informationen. Zu den Bestandteilen eines SNMP-basierenden Managementsystems zählen Agents oder Proxy-Agents (meist in den zu verwaltenden Geräten selbst), ein Manager (Programm mit dem der Netzwerkverwalter arbeitet) sowie eine MIB (Management Information Base). Eine MIB ist eine Art Datenbank, welche die Beschreibung der in einem Netzwerk angeschlossenen Objekte und Funktionen, die gemanagt werden können, enthält. SNMP hat ein einfaches Kommando-Set, um Informationen zwischen Manager und Agent auszutauschen, Veränderungen an einem Netzwerkgerät vorzunehmen und Ereignisse zu melden.

## 1.7 Die Zukunft des Internet

### 1.7.1 CIDR

Im Allgemeinen zeichnen sich folgende Trends ab [vgl. Rau00]. Die Service Provider werden ein Portfolio von IP basierten Anwendungen und Diensten anbieten. Es wird flächendeckend mit breitbandigem Zugang im Access Bereich (xDSL) gerechnet. Außerdem wird die Infrastruktur aus Terabit Routers und aus Hybridsystemen (Switch und Router) bestehen. Zudem wird sich Wireless Access durchsetzen. Dienste wie E-Commerce und Telefon über IP-Netze werden alltäglich sein.

Das rasche, exponentielle Wachstum des Internet stellt auch die Protokolle vor neue Anforderungen. Das Internet Protokoll der Version 4 (IPv4) z.B., soll nun durch ein Nachfolgeprotokoll ersetzt werden.

Das Internet wandelt sich zu einem weltweiten Informations- und Unterhaltungssystem. Noch vor einiger Zeit wurde das Internet größtenteils nur von Universitäten, Regierungsbehörden (dies aber auch fast nur in den USA und hier vor allem vom Verteidigungsministerium) und einigen Firmen aus der Industrie genutzt. Seit der Einführung des World Wide Web (WWW) ist das Internet aber auch zunehmend für Privatpersonen, kleinere Firmen etc. interessant. Mit der ständig steigenden Anzahl von Benutzern des Internet werden sich auch die Anforderungen an das Netz ändern bzw. haben sich bereits geändert. Genannt sei hier nur als Beispiel das angestrebte Zusammenwachsen der Computer-, Unterhaltungs- und Telekommunikationsbranchen. Den Anforderungen, die z.B. Video-on-demand stellt, ist das Internet bzw. das Internet Protokoll in der Version 4 nicht gewachsen.

Vinton Cerf (der 'Vater' des Internet) bezeichnet in einem Interview mit der Zeitschrift c't [Kre98] das Internet "[...] als die wichtigste Infrastruktur für alle Arten von Kommunikation." Dies beinhaltet auch den Aspekt Haushaltsgeräte ans Netz anzuschließen. Die neuen Anwendungen stellen aber auch neue Anforderungen an die Sicherheit derartiger Systeme. Auf die Internet Protokolle kommen in der nächsten Zeit also völlig neue Anforderungen zu. Wie versucht wird, diese Anforderungen zu erfüllen, wird in den nächsten Abschnitten beschrieben.

Der Verknappung der Internet-Adressen durch die ständig steigende Benutzerzahl wird zunächst versucht, mit dem Classless InterDomain Routing (CIDR) entgegen zu wirken. Durch die Vergabe von Internet-Adressen in Klassen (Netze der Klassen A,B,C,...) wird eine große Anzahl von Adressen verschwendet. Hierbei stellt sich vor allem die Klasse B als Problem dar. Viele Firmen nehmen ein Netz der Klasse B für sich in Anspruch, da ein Klasse A Netz mit bis zu 16 Mio. Hosts selbst für eine sehr große Firma überdimensioniert scheint. Tatsächlich ist aber oft auch ein Klasse B Netz zu groß. Für viele Firmen würde ein Netz der Klasse C

ausreichen. Dies wurde aber nicht verlangt, da die Unternehmen die Befürchtung hatten, daß ein Klasse C Netz mit seinen bis zu 254 möglichen Hosts nicht ausreichen würde. Ein größeres Hostfeld für Netze der Klasse C (z.B. 10 Bit, das entspricht 1022 Host pro Netz) hätte das Problem der knapper werdenden IP-Adressen vermutlich gemildert. Ein anderes Problem wäre dadurch allerdings entstanden: die Einträge der Routingtabellen wären explodiert. Ein anderes Konzept verfolgt nun das Classless InterDomain Routing: die verbleibenden Netze der Klasse C werden in Blöcken variabler Größe zugewiesen. Werden beispielsweise 2000 Adressen benötigt, so können einfach acht aufeinanderfolgende Netze der Klasse C vergeben werden; das entspricht einem Block von 2048 Adressen. Zusätzlich werden die verbliebenen Klasse C Adressen restriktiver und strukturierter vergeben. Die Welt ist dabei in vier Zonen, von denen jede einen Teil des verbliebenen Klasse C Adressraums erhält, aufgeteilt.

194.0.0.0 - 195.255.255.255	Europa
198.0.0.0 - 199.255.255.255	Nordamerika
200.0.0.0 - 201.255.255.255	Mittel- und Südamerika
202.0.0.0 - 203.255.255.255	Asien und pazifischer Raum
204.0.0.0 - 223.255.255.255	Reserviert für zukünftige

Tabelle 1.1: Aufteilung des Adressraums

Jede der Zonen erhält dadurch in etwa 32 Millionen Adressen zugewiesen. Vorteil bei diesem Vorgehen ist, daß die 32 Millionen Adressen einer Region im Prinzip zu einem Eintrag in den Routingtabellen komprimiert worden sind.

### 1.7.2 Internet Protokoll Version 6 - IPv6 (IP Next Generation)

Der begrenzte Adressraum ist wohl der vorrangige Grund für eine Änderung des IP-Protokolls. Etwas Luft schafft hier CIDR, dennoch ist klar absehbar, daß auch diese Maßnahmen nicht ausreichen, um die Verknappung der Adressen für eine längere Zeit in den Griff zu bekommen. Weitere Gründe für eine Änderung des IP-Protokolls sind die oben schon erwähnten neuen Anforderungen an das Internet. Diesen Anforderungen ist IP in der Version 4 nicht gewachsen. Die IETF (Internet Engineering Task Force) begann deshalb 1990 mit der Arbeit an einer neuen Version von IP. Die wesentlichen Ziele des Projekts sind [Tan97]:

- Höhere Sicherheit (Authentifikation und Datenschutz) als das heutige IP
- Mehr Gewicht auf Dienstarten, insbesondere für Echtzeitanwendungen
- Unterstützung von Multicasting durch die Möglichkeit, den Umfang zu definieren
- Unterstützung von Milliarden von Hosts, auch bei ineffizienter Nutzung des Adressraums
- Reduzierung des Umfangs der Routingtabellen
- Vereinfachung des Protokolls, damit Router Pakete schneller abwickeln können
- Möglichkeit für Hosts, ohne Adressänderung auf Reise zu gehen
- Möglichkeit für das Protokoll, sich zukünftig weiterzuentwickeln
- Unterstützung der alten und neuen Protokolle in Koexistenz für Jahre

Die Entwicklung eines neuen Internet Protokolls wurde 1993 durch den Ruf nach neuen Vorschlägen vorangetrieben. Auf diese Anfrage wurde eine Vielzahl von Vorschlägen eingereicht. Diese reichten von nur geringfügigen Änderungen am bestehenden IPv4 bis zur vollständigen Ablösung durch ein neues Protokoll. Die drei besten Vorschläge wurden im IEEE Network Magazine veröffentlicht ([Dee93], [Fra93]). Aus diesen Vorschlägen wurde von der IETF das Simple Internet Protocol Plus (SIPP) als Grundlage für die neue IP-Version ausgewählt. SIPP ist eine Kombination aus den Vorschlägen von Deering [Dee93] und Francis [Fra93].

Natürlich wurde, als die Entwickler mit den Arbeiten an der neuen Version des Internet Protokolls begannen, auch ein Name für das Projekt bzw. das neue Protokoll benötigt. Wohl angeregt durch eine gleichnamige Fernsehsendung, wurde als Arbeitsname IP - Next Generation (IPnG) gewählt. Schließlich bekam das neue IP eine offizielle Versionsnummer zugewiesen: IP Version 6 oder kurz IPv6. Die Protokollnummer 5 (IPv5) wurde bereits für ein experimentelles Protokoll verwendet.

Die folgende Beschreibung von IPv6 orientiert sich an RFC 2460 [Internet Protocol, Version 6 (IPv6) Specification, Dec. 1998]. Viele der als erfolgreich betrachteten Merkmale von IPv4 bleiben in IPv6 voll erhalten. Trotzdem ist IPv6 im allgemeinen nicht mit IPv4 kompatibel, wohl aber zu den weiteren Internet-Protokollen, insbesondere den Protokollen der Transportschicht (TCP, UDP); eventuell nach geringfügigen Modifikationen. Die Modifikationen betreffen im wesentlichen die erweiterte Adressgröße (bisher 32 Bit auf nun 128 Bit).

Die wesentlichen Merkmale von IPv6 sind [Win01]:

- die Vergrößerung der Adressen von bisher 32 Bit auf nun 128 Bit, um einer Adressknappheit vorzubeugen. Theoretisch lassen sich damit  $2^{128} = 3,4 \cdot 10^{38}$  Adressen vergeben.
- die Flexibilisierung des Header-Formats: Der IPv6 (Basis)Header wurde vollständig geändert. Der Header enthält nur 7 statt bisher 13 Felder. Diese Änderung ermöglicht Routern, Pakete schneller zu verarbeiten. Im Gegensatz zu IPv4 gibt es bei IPv6 nicht mehr nur einen Header, sondern mehrere Header. Ein Datagramm besteht aus einem Basis-Header, sowie einem oder mehreren Zusatz-Headern, gefolgt von den Nutzdaten.
- erweiterte Unterstützung von Optionen und Erweiterungen: Die Erweiterung der Optionen ist notwendig geworden, da einige, bei IPv4 notwendige Felder nun optional sind. Darüber hinaus unterscheidet sich auch die Art, wie die Optionen dargestellt werden. Für Router wird es damit einfacher, Optionen, die nicht für sie bestimmt sind, zu überspringen. Dies ermöglicht ebenfalls eine schnellere Verarbeitung von Paketen.
- andere Dienstarten: IPv6 legt mehr Gewicht auf die Unterstützung von Dienstarten. Damit kommt IPv6 den Forderungen nach einer verbesserten Unterstützung der Übertragung von Video- und Audiodaten entgegen und unterstützt die QoS Aspekte. IPv6 bietet hierzu eine Option zur Echtzeit-übertragung.
- mehr Sicherheit: IPv6 beinhaltet nun im Protokoll selbst Mechanismen zur sicheren Datenübertragung. Wichtige neue Merkmale von IPv6 sind hier Authentifikation (authentication), Datenintegrität (data integrity) und Datenverlässlichkeit (data confidentiality).
- die Erweiterbarkeit: IPv6 ist ein erweiterbares Protokoll. Bei der Spezifikation des Protokolls wurde nicht versucht alle potentiell möglichen Einsatzfelder für das Protokoll in die Spezifikation zu integrieren. Vielmehr bietet IPv6 die Möglichkeit über Erweiterungs-Header das Protokoll zu erweitern. Damit ist das Protokoll offen für zukünftige Verbesserungen.

Ein IPv6-Datengramm besteht aus dem Basis-Header, gefolgt von den optionalen Zusatz-Headern und den Nutzdaten.

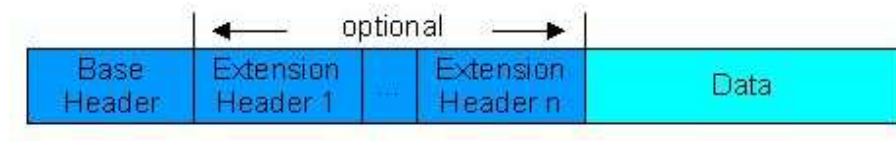


Abbildung 1.13: Allgemeine Form eines IPv6-Datengramms.

### 1.7.3 QoS

Entscheidender Faktor der Zukunft wird auch der Quality of Service, die Dienstgüte, werden. Innerhalb von ATM sind auf der Zellebene Dienstgüten definiert. Beim Verbindungsaufbau vereinbart das Endsystem mit dem Dienstanbieter unter anderem die Dienstklasse und Bandbreite. Diesen Vorgang nennt man Signalisierung. Die Dienstqualitäten sind durch die Eigenschaften Übertragungsverzögerung (Cell Delay), Verzögerungsvarianz (Cell Delay Variation), Zellverlust (Cell Loss) und Fehlerrate (Cell Error Rate) beschrieben [Rau00]. Zur Zeit sind vier QoS-Klassen (CBR, rt-VBR, nrt-VBR, UBR/ABR) entsprechend den ATM-Dienstklassen definiert. Auch im Header von IPv6 ist, wie oben beschrieben, ein QoS-Feld vorhanden. Darin gibt die absendende Applikation ihre Anforderungen an die Qualität der Übertragung an. Es sind 16 Stufen (0-15) mit aufsteigender Priorität vorgesehen. Datenverkehr der Stufen 0-7 muss bei Überlastung eines Routers warten können. Pakete der Stufen 8-15 sollten dagegen mit einer konstanten Datenrate weiterkommen, da sie z.B. zu Echtzeitanwendungen gehören können. Die Priorität entscheidet daher darüber, ob ein Paket entweder mit der gewünschten Geschwindigkeit transportiert oder verworfen wird. So wird sichergestellt, dass ein Paket den schnellsten Weg nimmt, da überlastete Strecken als undurchlässig erscheinen. Das Ziel sog. Integrated Services ist Erweiterung der existierenden Internet Architektur, um Realtime- und Non-Realtime Anwendungen zu unterstützen. Die Arbeiten an Integrated Services wurden 1997 abgeschlossen. Verschiedene "QoS Control Services" wurden dabei erarbeitet, z.B. Guaranteed Service oder Controlled Load Service. Diese sind für Realtime-Verkehr, für den das Netz Garantien übernimmt, mit folgender Charakteristik vorgesehen:

- Begrenzte Queuing Verzögerung
- Geringer Durchsatz
- Sehr kleiner Paketverlust

Der Sender schlägt die Traffic Parameter vor und der Empfänger stimmt dem Verkehrskontrakt mit dem Netz zu. Um Mechanismen wie Authentisierung, Verschlüsselung und Datenintegrität zu gewährleisten, müssen Techniken wie IPSec oder Transport Layer Security (TLS) eingesetzt werden, welche hier nicht weiter erläutert werden sollen.

## 1.8 Zusammenfassung

Das Internet hat mit seinen weltweiten Zusammenschaltungen und auch wegen seiner rasanten Entwicklung viele Gesichter. Die Hauptprinzipien der Architektur sind zum einen das Ende-zu-

## *Internet Architektur und Protokolle*

Ende-Prinzip, bei dem die Funktionalität in den Terminal, d.h. den Endgeräten steckt und nicht durch das Netz bewältigt wird, wie etwa beim Telefon. Und zum anderen das Internetprotokoll, welches sich als Standard durchgesetzt hat. Es werden Pakete verschickt, welche sich Dank der Schichtenstruktur der Protokolle, flexibel aufsetzen lassen. Durch die eindeutige Adressstruktur ist die Verbindung bijektiv. Die rasante Entwicklung wird auch aufgrund der wachsenden Anreize nicht gebremst werden. Denn je mehr Menschen vernetzt sind, desto mehr Menschen wollen von dieser Vernetzung profitieren und desto mehr Dienste werden auch angeboten.

# Literaturverzeichnis

- [Alp96] ALPAR, P.: *Kommerzielle Nutzung des Internet*. Springer, Berlin (1996)
- [Com00] COMER, DOUGLAS: *Computernetzwerke und Internet. Markt und Technik, München(2000)*
- [Dan00] DANDIK, DAMIAN: *Protokolle <http://www.damian-dandik.de>* (2000)
- [Dee93] DEERING S.E.: *SIP- Simple Internet Protocol. IEEE Network Magazin, Band 7, 1993*
- [Fra93] FRANCIS P.: *A Near Term in Architecture for Deploying Pip. IEEE Network, Magazin, Band 7, 1993*
- [Hin02] HINDEN R.: *IP Next Generation (IPng)*  
*<http://playground.sun.com/pub/ipng/html/ipng-main.html>* (2002)
- [HLW97] HARTJES K., LÖFFLER H., WESSENDORF G.: *Fortsetzung folgt: Aktuelles zur IPv6-Einführung. Ix 4/97, S. 97ff.*
- [Hol01] HOLTkamp H.: *Einführung in TCP/IP. 2001*
- [Hui96] CHRISTIAN HUITEMA: *Routing im Internet. Prentice Hall, München 1996*
- [Kre98] KREMPL S.: *Das Internet bleibt spannend! Gespräch mit V.G. Cerf, c't 3/98, S. 44ff.*
- [Rau00] RAUTENBERG, MATHIAS: *IP Weitverkehrsnetze Siemens AG (2000)*
- [Rze00] RZEHAK H.: *Kommunikationsarchitekturen, V-Skript 2000*
- [San98] SANTIFALLER M.: *TCP/IP und ONC/NFS - Internetworking mit UNIX. Addison Wesley, Bonn, Reding Massachusetts, 1998, 4. Auflage*
- [Sch93] SCHACHTNER, ANDREAS: *23. Internet Engineering Taskforce. In: iX (1993), Heft 6*
- [Tan97] TANENBAUM A.S.: *Computernetzwerke. Prentice Hall, München, 1997*
- [tel02] *<http://www.telstra.net>* (2002)
- [Win01] WiN/DFN: *IP Version 6 im WiN - Ein DFN Projekt.*  
*<http://www.join.uni-muenster.de>*

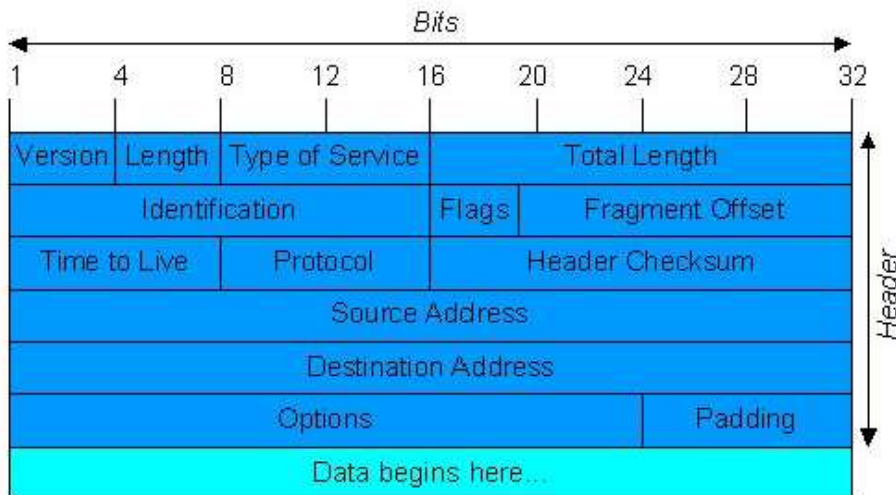


Abbildung 1.14: Der IP-Header und die Bedeutung seiner Felder

## Anhang A

### Der IP-Header

Die Felder des in der Abbildung dargestellten Protokollkopfes haben die folgende Bedeutung [Hol01]:

#### Version:

Dieses Feld beinhaltet die Versionsnummer des IP-Protokolls. Dadurch besteht die Möglichkeit auch über eine längere Zeit mit verschiedenen Versionen des IP Protokolls zu arbeiten. Während einige Hosts mit der alten Version arbeiten, können dies andere mit der Neuen. Die derzeitige Versionsnummer ist 4, aber die Version 6 des IP Protokolls befindet sich bereits in der Erprobung (siehe Kapitel 7.2)

#### Length:

Das Feld Length (Internet Header Length - IHL) enthält die Länge des Protokollkopfes, da diese nicht konstant ist. Die Länge wird in 32-Bit-Worten angegeben. Der kleinste zulässige Wert ist 5, das entspricht 20 Byte und hat zur Folge, dass im Header keine Optionen gesetzt sind. Die Headerlänge kann sich durch Anfügen von Optionen aber bis auf 60 Byte erhöhen.

#### Type of Service:

Über das Feld Type of Service kann IP angewiesen werden Nachrichten nach bestimmten Kriterien zu behandeln. Als Dienste sind hier verschiedene Kombinationen aus Zuverlässigkeit und Geschwindigkeit möglich. Das Feld selbst hat den folgenden Aufbau:

Precedence (Bits 0-2) gibt die Priorität von 0 (normal) bis 7 (Steuerungspaket) an. Die vier Flags (D,T,R,C) ermöglichen es dem Host anzugeben, worauf er bei der Datenübertragung am meisten Wert legt: Verzögerung (D), Durchsatz (T), Zuverlässigkeit (R), oder Kosten (C). Das andere Bit-Feld ist reserviert.



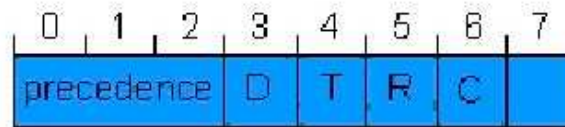


Abbildung 1.15: Aufbau ToS

**Total Length:**

Dieses Feld enthält die gesamte Paketlänge, d.h. Header und Daten. Da es sich hierbei um ein 16-Bit-Feld handelt ist die Maximallänge eines Datengramms auf 65.535 Byte begrenzt. In der Spezifikation von IP ist festgelegt, dass jeder Host in der Lage sein muss, Pakete bis zu einer Länge von 576 Bytes zu verarbeiten. In der Regel können von den Host aber Pakete größerer Länge verarbeitet werden.

**Identification:**

Über das Identifikationsfeld kann der Zielhost feststellen, zu welchem Datengramm ein neu angekommenes Fragment gehört. Alle Fragmente eines Datengramms enthalten die gleiche Identifikationsnummer, die vom Absender vergeben wird.

**Flags:**

Das Flags-Feld ist drei Bit lang. Die Flags bestehen aus zwei Bits namens DF (Don't Fragment) und MF (More Fragments). Das erste Bit des Flags-Feldes ist ungenutzt. Die beiden Bits DF und MF steuern die Behandlung eines Pakets im Falle einer Fragmentierung. Mit dem DF-Bit wird signalisiert, dass das Datengramm nicht fragmentiert werden darf. Auch dann nicht, wenn das Paket dann evtl. nicht mehr weiter transportiert werden kann und verworfen werden muss. Alle Hosts müssen, Fragmente bzw. Datengramme mit einer Größe von 576 Bytes oder weniger verarbeiten können. Mit dem MF-Bit wird angezeigt, ob einem IP-Paket weitere Teilpakete nachfolgen. Diese Bit ist bei allen Fragmenten außer dem letzten gesetzt.

**Fragment Offset:**

Der Fragmentabstand bezeichnet, an welcher Stelle relativ zum Beginn des gesamten Datengramms ein Fragment gehört. Mit Hilfe dieser Angabe kann der Zielhost das Originalpaket wieder aus den Fragmenten zusammensetzen. Da dieses Feld nur 13 Bit groß ist, können maximal 8192 Fragmente pro Datengramm erstellt werden. Alle Fragmente, außer dem letzten, müssen ein Vielfaches von 8 Byte sein. Dies ist die elementare Fragmenteinheit.

**Protocol:**

Enthält die Nummer des Transportprotokolls, an das das Paket weitergeleitet werden muss. Die Nummerierung von Protokollen ist im gesamten Internet einheitlich. Bisher wurden die Protokollnummern im RFC 1700 definiert. Diese Aufgabe ist nun von der Internet Assigned Numbers Authority (IANA) übernommen worden.

**Time to Live:**

Das Feld Time to Live ist ein Zähler, mit dem die Lebensdauer von IP-Paketen begrenzt wird. Die Einheit ist Sekunden. Zulässig ist eine maximale Lebensdauer von 255 Sekunden (8 Bit).

Der Zähler muss von jedem Netzknoten, der durchlaufen wird um mindestens 1 verringert werden. Bei einer längeren Zwischenspeicherung in einem Router muss der Inhalt sogar mehrmals verringert werden. Enthält das Feld den Wert 0, muss das Paket verworfen werden. Damit wird verhindert, dass ein Paket endlos in einem Netz umherwandert. Der Absender wird in einem solchen Fall durch eine Warnmeldung in Form einer ICMP-Nachricht informiert.

### Header Checksum:

Dieses Feld enthält die Prüfsumme der Felder im IP-Header. Die Nutzdaten des IP-Datengramms werden aus Effizienzgründen nicht mit geprüft. Diese Prüfung findet beim Empfänger innerhalb des Transportprotokolls statt. Die Prüfsumme muss von jedem Netzknoten, der durchlaufen wird, neu berechnet werden, da der IP-Header durch das Feld Time-to-Live sich bei jeder Teilstrecke verändert. Aus diesem Grund ist auch eine sehr effiziente Bildung der Prüfsumme wichtig. Als Prüfsumme wird das 1er-Komplement der Summe aller 16-Bit-Halb Wörter der zu überprüfenden Daten verwendet. Zum Zweck dieses Algorithmus wird angenommen, dass die Prüfsumme zu Beginn der Berechnung Null ist.

### Source Address, Destination Address:

In diese Felder werden die 32-Bit langen Internet-Adressen eingetragen.

### Options und Padding:

Um die Möglichkeit zu haben, dass IP-Protokoll um weitere Informationen zu ergänzen, die im ursprünglichen Design nicht berücksichtigt wurden, ist das Feld Options im Protokollkopf eingegliedert worden. Das Optionsfeld hat eine variable Länge. Jede Option beginnt mit einem Code von einem Byte, über den die Option identifiziert wird. Manchen Optionen folgt ein weiteres Optionsfeld von 1 Byte und dann ein oder mehrere Datenbytes für die Option. Das Feld Options wird über das Padding auf ein Vielfaches von 4 Byte aufgefüllt. Derzeit sind die folgenden Optionen bekannt. Die End of Option List, welche das Ende der Optionsliste kennzeichnet, No Option, diese kann zum Auffüllen von Bits zwischen Optionen verwendet werden und Security, bezeichnet, wie geheim ein Datengramm ist. In der Praxis wird diese Option jedoch fast immer ignoriert. Desweiteren Loose Source-Routing und Strict Source-Routing. Diese Option enthält eine Liste von Internet-Adressen, die das Datagramm durchlaufen soll. Auf diese Weise kann dem Datenpaket vorgeschrieben werden eine bestimmte Route durch das Internet zu nehmen. Beim Source-Routing wird zwischen Strict Source and Record Route und Loose Source and Record Route unterschieden. Im ersten Fall wird verlangt, dass das Paket diese Route genau einhalten muss. Des weiteren wird die genommene Route aufgezeichnet. Die zweite Variante schreibt vor, dass die angegebenen Router nicht umgangen werden dürfen. Auf dem Weg können aber auch andere Router besucht werden. Record Route: Die Knoten, die dieses Datagramm durchläuft, werden angewiesen ihre IP-Adresse an das Optionsfeld anzuhängen. Damit lässt sich ermitteln, welche Route ein Datengramm genommen hat. Wie anfangs schon gesagt, ist die Größe für das Optionsfeld auf 40 Byte beschränkt. Deshalb kommt es heute auch oftmals zu Problemen mit dieser Option, da weit mehr Router durchlaufen werden, als dies zu Beginn des ARPANET der Fall war. Und letztlich Time Stamp.

Diese Option ist mit der Option Record Route vergleichbar. Zusätzlich zur IP-Adresse wird bei dieser Option die Uhrzeit des Durchlaufs durch den Knoten vermerkt. Auch diese Option dient hauptsächlich zur Fehlerbehandlung, wobei zusätzlich z.B. Verzögerungen auf den Netzstrecken erfasst werden können.

## Anhang B

### IP-Adressformate und die Adressklassen

Klasse A: Die Kennzeichnung erfolgt durch das erste Byte, mit einen Wert kleiner als 128, d.h. das erste Bit der Adresse ist 0. Nur das erste Byte ist die Netzwerknummer, die letzten drei Bytes identifizieren eines Host im Netz. Es kann also nur 126 Klasse A Netze geben, die dann bis zu 16 Millionen Host in einem Netz haben können.

Klasse B: Mit einem Wert von 128 bis 191 für das erste Byte (das erste Bit ist gleich 1, Bit 2 gleich 0) wird eine Klasse B Adresse identifiziert. Die ersten beiden Bytes identifizieren das Netzwerk, die letzten beiden Bytes einen Host. Das ergibt 16.382 Klasse B Netze mit bis zu 65534 Hosts in einem Netz.

Klasse C: Klasse C Netze werden über Werte von 192 bis 223 für das erste Byte identifiziert. Es gibt ca. 2 Millionen Netze der Klasse C, d.h. die ersten drei Bytes werden für die Netzwerkadresse verwendet. Ein Klasse C Netz kann nur bis zu 254 Host beinhalten.

Klasse D: Klasse D Adressen sind Multicast-Adressen. Sie werden dazu verwendet ein Datagramm an mehrere Hostadressen gleichzeitig zu versenden. Zur Kennzeichnung hat das erste Byte einer Multicast-Adresse den Wertebereich von 224 bis 239. Wird eine Nachricht an eine Adresse der Klasse D gesendet, wird diese an alle Mitglieder der adressierten Gruppe verschickt. Bei der Übermittlung der Nachricht kann jedoch keine Garantie gegeben werden, ob die Daten tatsächlich alle Mitglieder eine Gruppe erreichen.

Der weitere Bereich der IP-Adressen von 240 bis 254 im ersten Byte ist für zukünftige Nutzungen reserviert. In der Literatur wird dieser Bereich oft auch als Klasse E bezeichnet (vgl. [Com00]).

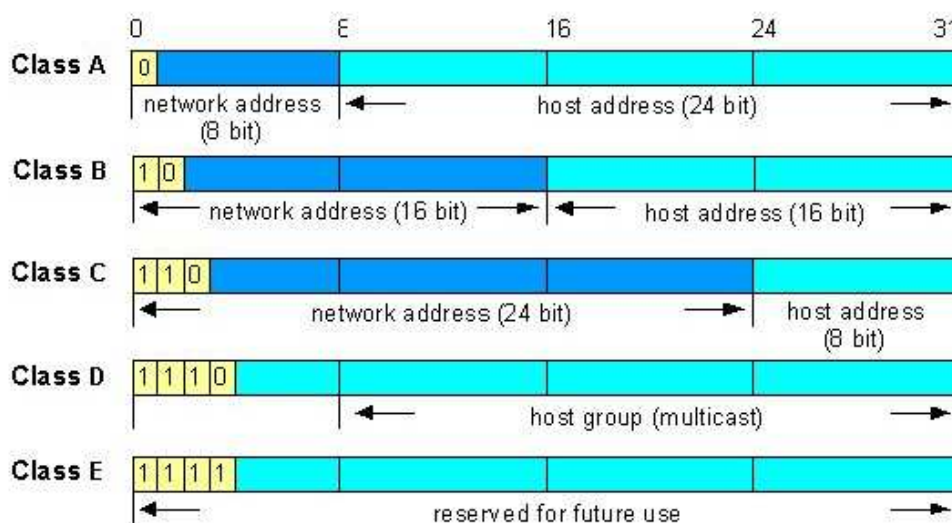


Abbildung 1.16: IP-Adressformate

## Anhang C

### ICMP-Nachrichtentypen

Die derzeit wichtigsten Nachrichtentypen sind (vgl. [Hol01]):

**Destination Unreachable (Ziel nicht erreichbar):**

Wenn ein Netzwerk, Host, Protokoll oder Port nicht erreichbar ist, oder die Source Route Option nicht erfolgreich ist, oder aber ein Paket nicht fragmentiert werden kann, weil das DF-Bit gesetzt ist, wird diese Nachricht versendet.

**Source Quench (Quelle löschen):**

Diese Nachricht wird gesendet, wenn ein Host zu viele Pakete verschickt, die aus Kapazitätsmangel nicht mehr verarbeitet werden können. Der sendende Host muss dann die Rate zum Aussenden von Nachrichten verringern.

**Parameter Problem:**

Der Absender eines Datengramms wird benachrichtigt, dass das Paket aufgrund einer fehlerhaften Angabe im IP-Header verworfen werden musste.

**Redirect:**

Wenn ein Router feststellt, dass ein Paket falsch weitergeleitet wurde, wird ein rredirectentsandt. Der sendende Host wird damit aufgefordert, die angegebene Route zu ändern.

**Time Exceeded (Zeit verstrichen):**

Hat die Lebensdauer eines Datengramms den Wert 0 erreicht, wird es verworfen und der Absender benachrichtigt. Diese Nachricht bedeutet oft, dass Pakete in einem Zyklus wandern, dass Netz überlastet ist oder die Lebensdauer für das Paket zu gering eingestellt wurde.

**Echo Reply, Echo Request:**

Mit diesen Nachrichten kann festgestellt werden, ob ein bestimmtes Ziel erreichbar ist. Ein Echo Request wird an einen Host gesendet und hat, falls der Host erreicht wird, einen Echo Reply zur Folge.

**Timestamp Request, Timestamp Reply:**

Werden die Ankunftszeit der Nachricht und die Sendezeit der Antwort mit erfasst, wird dies dann Timestamp Request bzw. Timestamp Reply genannt. Mit diesen Nachrichtentypen kann die Netzleistung gemessen werden.

## Anhang D

### Bedeutung der TCP-Header-Felder

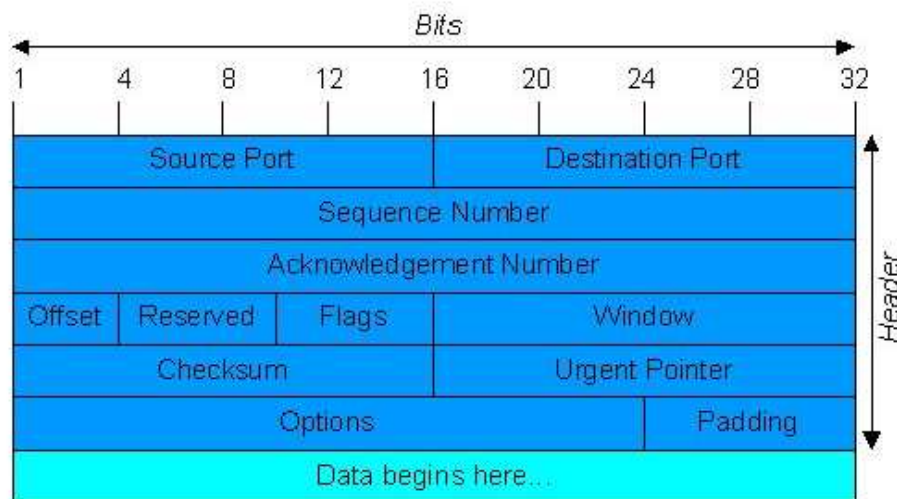


Abbildung 1.17: Der TCP Header

Die Felder des TCP-Headers haben die folgende Bedeutung (aus [Hol01]):

Source-/Destination-Port:

Die Felder Source Port (Quellport) und Destination Port (Zielport) adressieren die Endpunkte der Verbindung. Die Größe für die beiden Felder beträgt 16 Bit.

Sequence Number, Acknowledgement Number:

Die Sequenznummer und die Bestätigungsnummer sind jeweils 32-Bit-Zahlen. Die Nummern geben die Stellung der Daten des Segments innerhalb des in der Verbindung ausgetauschten Datenstroms an. Die Sequenznummer gilt in Senderichtung, die Bestätigungsnummer für Empfangsquittungen. Jeder der beiden TCP-Verbindungspartner generiert beim Verbindungsaufbau eine Sequenznummer, die sich während des Zeitraums der Verbindung nicht wiederholen darf. Dies ist allerdings durch den großen Zahlenraum von  $2^{32}$  wohl ausreichend gesichert. Diese Nummern werden beim Verbindungsaufbau ausgetauscht und gegenseitig quittiert. Bei der Datenübertragung wird die Sequenznummer vom Absender jeweils um die Anzahl der bereits gesendeten Bytes erhöht. Mit der Quittungsnummer gibt der Empfänger an, bis zu welchem Byte er die Daten bereits korrekt empfangen hat. Die Nummer gibt allerdings nicht an, welches Byte zuletzt korrekt empfangen wurde, sondern welches Byte als nächstes zu erwarten ist.

Offset:

Das Feld Offset (oder auch Header Length) gibt die Länge des TCP-Headers in 32-Bit Worten an. Dies entspricht dem Anfang der Daten im TCP-Segment. Das Feld ist notwendig, da der Header durch das Optionsfeld eine variable Länge hat.

Flags:

Mit den sechs 1-Bit-Flags im Flags-Feld werden bestimmte Aktionen im TCP-Protokoll aktiviert:

- **URG:**  
Wird das Flag URG auf 1 gesetzt, so bedeutet dies, dass der Urgent Pointer (Dringendzeiger) verwendet wird.
- **ACK:**  
Das ACK-Bit wird gesetzt, um anzugeben, dass die Bestätigungsnummer im Feld Acknowledgement Number gültig ist. Ist das Bit auf 0 gesetzt, enthält das TCP-Segment keine Bestätigung, das Feld Acknowledgement Number wird ignoriert.
- **PSH:**  
Ist das PSH-Bit gesetzt, so werden die Daten in dem entsprechenden Segment sofort bei Ankunft der adressierten Anwendung bereitgestellt ohne sie zu puffern.
- **RST:**  
Das RST-Bit dient dazu eine Verbindung zurückzusetzen, falls ein Fehler bei Übertragung aufgetreten ist. Dies kann sowohl der Fall sein, wenn ein ungültiges Segment übertragen wurde, ein Host abgestürzt ist oder der Versuch eines Verbindungsaufbaus abgewiesen werden soll.
- **SYN:**  
Das SYN-Flag (Synchronize Sequence Numbers) wird verwendet, um Verbindungen aufzubauen. Zusammen mit der Acknowledgement Number und dem ACK-Bit wird die Verbindung im Form eines Dreiwege-Handshake aufgebaut (siehe oben).
- **FIN:**  
Das FIN-Bit dient zum Beenden einer Verbindung. Ist das Bit gesetzt, gibt dies an, dass der Sender keine weiteren Daten zu Übertragen hat. Das Segment mit gesetztem FIN-Bit muss quittiert werden.
- **Window:**  
Das Feld Fenstergröße enthält die Anzahl Bytes, die der Empfänger ab dem bereits bestätigten Byte empfangen kann. Mit der Angabe der Fenstergröße erfolgt in TCP die Flußsteuerung. Das TCP-Protokoll arbeitet nach dem Prinzip eines Schiebefensters mit variabler Größe (Sliding Window). Jede Seite einer Verbindung darf die Anzahl Bytes senden, die im Feld für die Fenstergröße angegeben ist, ohne auf eine Quittung von der Empfängerseite zu warten. Während des Sendens können gleichzeitig Quittungen für die von der anderen Seite empfangenen Daten eintreffen (diese Quittungen können wiederum neue Fenstergrößen einstellen). Eine Fenstergröße von 0 besagt, dass die Bytes bis einschließlich der Acknowledgement Number minus Eins empfangen wurden, der Empfänger momentan aber keine weiteren Daten empfangen kann. Die Erlaubnis zum weiteren Senden von Daten erfolgt durch das versenden eines Segments mit gleicher Bestätigungsnummer und einer Fenstergröße ungleich Null.
- **Checksum:**  
Die Prüfsumme prüft den Protokollkopf, die Daten und den Pseudo-Header (siehe Abbildung).

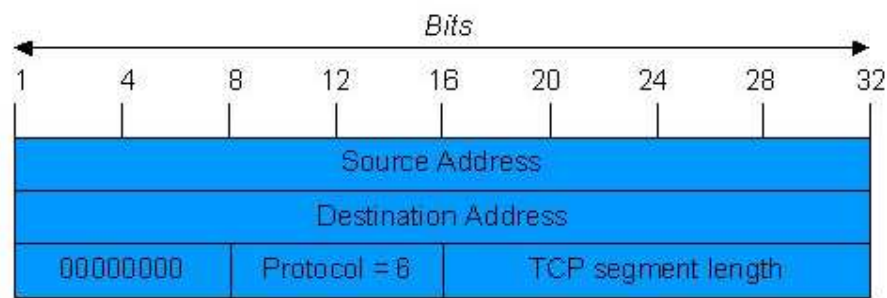


Abbildung 1.18: Der Pseudo-Header in der Prüfsumme.

Der Algorithmus für die Bildung der Prüfsumme ist einfach: alle 16-Bit Wörter werden im 1er-Komplement addiert und die Summe ermittelt. Bei der Berechnung ist das Feld Checksum auf Null gesetzt und das Datenfeld wird bei ungerader Länge um ein Nullbyte aufgefüllt. Führt der Empfänger des Segments die Berechnung auf das gesamte Segment aus - inklusive des Feldes für die Prüfsumme - sollte das Ergebnis 0 sein [Tan97]. Der Pseudo-Header enthält die 32-bit großen IP-Adressen der Quell- und Zielmaschine sowie die Protokollnummer (für TCP 6) und die Länge des TCP-Segments. Die Einbeziehung der Felder des Pseudo-Headers in die Prüfsummenberechnung hilft, durch IP falsch zugeteilte Pakete zu erkennen. Die Verwendung von IP-Adressen auf der Transportebene stellt allerdings eine Verletzung der Protokollhierarchie dar.

- **Urgent Pointer:**  
Der Urgent-Zeiger ergibt zusammen mit der Sequenznummer einen Zeiger auf ein Datenbyte. Dies entspricht einem Byteversatz zu einer Stelle, an der dringende Daten vorgefunden werden. TCP signalisiert damit, dass sich an einer bestimmten Stelle im Datenstrom wichtige Daten befinden, die sofort gelesen werden sollten. Das Feld wird nur gelesen, wenn auch das Urgent-Flag (s.o.) gesetzt ist.
- **Options:**  
Das Options-Feld soll eine Möglichkeit bieten Funktionen bereitzustellen, die im normalen TCP-Protokollkopf nicht vorgesehen sind. In TCP sind drei Optionen definiert: End of Option List, No-Operation und Maximum Segment Size. Die wichtigste dieser drei Optionen ist die Maximale Segmentgröße. Mit dieser Option kann ein Host die maximale Anzahl Nutzdaten übermitteln, die er annehmen will bzw. annehmen kann. Während eines Verbindungsaufbaus kann jede Seite ihr Maximum an Nutzdaten übermitteln, die kleinere der beiden Zahlen wird als maximale Nutzdatengröße für die Übertragung übernommen. Wird diese Option von einem Host nicht unterstützt wird als default die Vorgabe von 536 Byte verwendet.
- **Padding:**  
Das Feld Padding wird verwendet, um sicherzustellen, dass der Header an einer 32-Bit Grenze endet und die Daten an einer 32-Bit Grenze beginnen. Das Füllfeld wird mit Nullen gefüllt.





# 2 Interne Routing-Protokolle

Markus Körner

## Inhaltsverzeichnis

---

<b>2.1 Einführung</b> . . . . .	<b>41</b>
2.1.1 Was ist Routing? . . . . .	41
2.1.2 Funktionen eines Routers . . . . .	43
2.1.3 Interne vs. Externe Routing-Protokolle . . . . .	43
<b>2.2 Distanzvektor-Algorithmen</b> . . . . .	<b>44</b>
2.2.1 Einführung . . . . .	44
2.2.2 Beispiel: Routing Information Protocol (RIP) . . . . .	48
<b>2.3 Link-State-Routing-Algorithmen</b> . . . . .	<b>49</b>
2.3.1 Einführung . . . . .	49
2.3.2 Beispiel: Open Shortest Path First (OSPF) . . . . .	52
<b>2.4 RIP <math>\longleftrightarrow</math> OSPF : Vergleich und Ausblick</b> . . . . .	<b>53</b>

---

## Abstract

As part of the Seminar „Routing Strategies in the Internet“ this work lights up the topic „Internal Routing Protocols“. After a short introduction to Routing in general and its main issues it introduces the two most common Internal Routing Protocol families. It starts with the so called „Distance Vector Protocols“ before introducing the „Link State Protocols“. Both protocols are introduced by giving concrete examples to evince the fortes and foibles of each protocol family in respect to performance, security, scalability, user-friendliness and so on. After that for each protocol family a concrete implementation is discussed (RIP and OSPF). Finally this work compares the two protocol families and points out when to choose either the one or the other.

## 2.1 Einführung

### 2.1.1 Was ist Routing?

Unter dem Begriff Routing versteht man „die systematische Ermittlung, wie Daten anhand einer Adresse in Richtung Zielknoten zu befördern sind“[1] .

## Interne Routing-Protokolle

Was bedeutet dies nun konkret? Angenommen, wir haben folgendes Netz von Rechnernetzen: In diesem Beispiel ist jedes Netzwerk (ab jetzt **Knoten** genannt) mit jedem anderen verbunden.

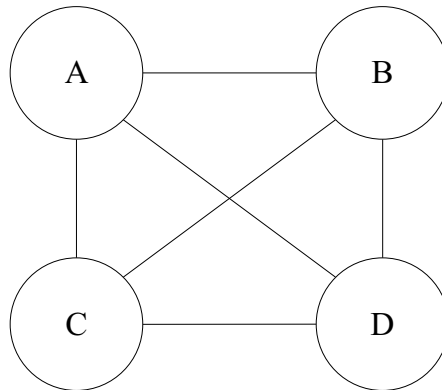


Abbildung 2.1: Ein einfacher Verbund von Netzwerken

Will nun z.B. ein Knoten A ein Datenpaket an Knoten C schicken, so sendet A dieses direkt über die Verbindung zu C an C. Ändert sich die Netztopologie allerdings nur leicht, so ist die Entscheidung wie ein Datenpaket weitergeleitet werden soll schon nicht mehr so trivial: Knoten A kann nun nicht mehr direkt die Daten an den Knoten C senden. Wie kann Knoten A

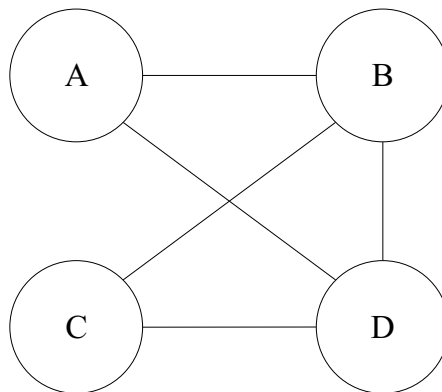


Abbildung 2.2: Die Verbindung zwischen Knoten A und C ist ausgefallen

nun wissen, wie es die Daten weiterleiten soll, oder ob der Knoten C vielleicht überhaupt nicht mehr zu erreichen ist?

Prinzipiell gibt es hier zwei Lösungsmöglichkeiten:

1. eine zentrale Stelle (ein Rechner), die die Netztopologie verwaltet und Anfragen der Knoten beantwortet (in unserem Beispiel würde A fragen, an wen es das Paket weiterleiten soll, um eine ordnungsgemäße Auslieferung sicherzustellen);
2. jeder Knoten kennt die Netztopologie und kann selber entscheiden, an wen es Daten weiterleiten muss, um eine ordnungsgemäße Auslieferung sicherzustellen.

Die erste Lösung bringt sehr viele Nachteile und Risiken mit sich. So ist z.B. ein komplettes Netz auf einen Rechner angewiesen. Fällt dieser aus, ist keine Kommunikation mehr zwischen

den Netzen möglich. Zudem widerspricht der Gedanke eines zentralen Dienstes dem Grundgedanken eines dezentralen Internet. Aus diesem Grunde wurde die zweite Lösungsmöglichkeit weiterverfolgt.

Die Entscheidung wie ein Datenpaket konkret weitergeleitet wird findet in einem so genannten *Router* statt.

## 2.1.2 Funktionen eines Routers

Ein Router hat 2 Funktionen :

1. Routing
2. Switching

Unter *Routing* versteht man die Ermittlung der Topologie des Netzes und daraus resultierend die Ermittlung der optimalen Wege zu allen erreichbaren Knoten. Dabei erstellt der Router unter Anwendung von *verteilten Algorithmen* (die in den Folgenden Kapiteln näher erläutert werden) eine Routing- sowie eine Weiterleitungstabelle. Diese Tabellen unterscheiden sich in der Information, die sie enthalten. So wird die Weiterleitungstabelle benutzt, wenn ein Paket weitergeleitet wird und muss daher genügend Informationen enthalten, damit die Weiterleitungsfunktion ausgeführt werden kann. Dies kann z.B. die Abbildung einer Netzwerknummer auf ein abgehendes Interface sein.

Die Routingtabelle wird dazu benutzt, die Weiterleitungstabelle zu erstellen. Sie enthält im Allgemeinen Abbildungen von Netzwerknummern auf nächste Hops. Darüberhinaus kann sie noch Informationen enthalten, wie Einträge erlernt wurden, so dass entschieden werden kann, wann diese entfernt werden können.

*Switching* ist der Prozess, bei dem Pakete anhand ihrer Zieladresse von dem Router weitergeleitet werden. Dazu ermittelt der Router anhand der Routingtabelle an welches Netz das jeweilige Datenpaket weitergeleitet werden muss und schaut dann in der Weiterleitungstabelle nach, an welches Ausgangsinterface dieses weitergegeben wird.

## 2.1.3 Interne vs. Externe Routing-Protokolle

Wie die Man unterscheidet *interne* (Interior Gateway Protocols [IGP]) von *externen* (Exterior Gateway Protocols [EGP]) Routing-Protokollen.

Interne Routing-Protokolle unterscheiden sich im Grunde genommen von externen Routing-Protokollen durch ihren Einsatzort. So regeln interne Protokolle die Weiterleitung von Daten innerhalb eines Autonomen Systems (AS), während externe Protokolle für eben diese Weiterleitung zwischen Autonomen Systemen zuständig sind. Ein Autonomes System ist so definiert, dass sich alle Router eines AS unter einer administrativen Kontrolle (z.B. Firmennetzwerk oder Netz einer Universität) befinden. Abbildung 1.3 verdeutlicht den Einsatzort der unterschiedlichen Protokolle. Diese Arbeit stellt im folgenden die Familie der internen Routing-Protokolle vor, erläutert deren Funktionsweisen und vergleicht deren Leistungsfähigkeit.

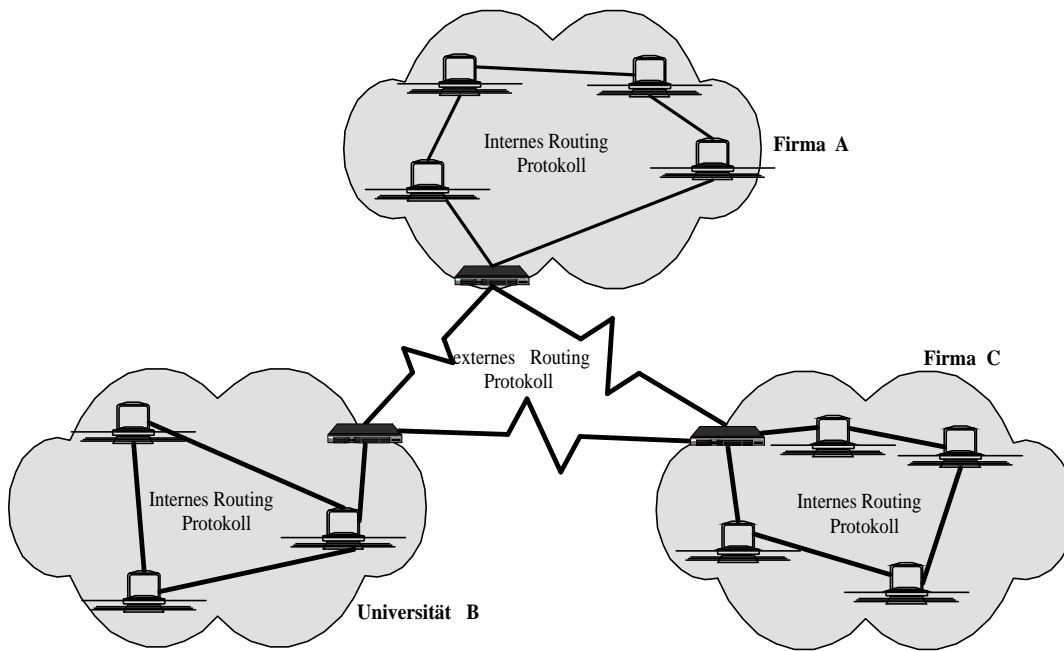


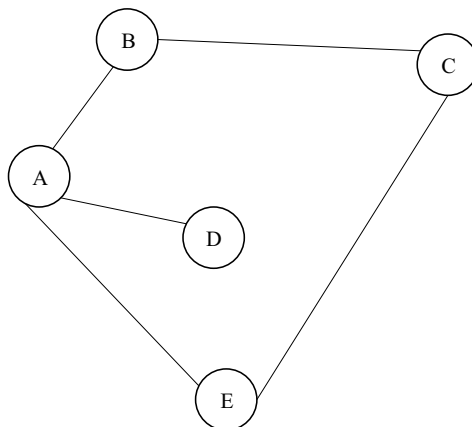
Abbildung 2.3: Einsatzort von Internen und Externen Routingprotokollen

## 2.2 Distanzvektor-Algorithmen

### 2.2.1 Einführung

Das Distanzvektor-Routing-Protokoll basiert auf dem Konzept, daß jeder Knoten seine direkten Nachbarn und die Kosten (z.B. Bandbreite der Verbindung, Kosten pro MB Datenvolumen) zu diesen kennt. Diese Informationen verwaltet jeder Knoten in einem eindimensionalen Array (Vektor) und propagiert diesen an alle direkten Nachbarn. Empfängt ein Knoten einen solchen Vektor, überprüft dieser, ob der eigene Vektor (mit den darin enthaltenen Informationen) evtl. durch die Empfangenen Informationen aktualisiert werden muss.

Dies wirkt im Moment noch sehr abstrakt. Betrachten wir daher folgendes Beispielnetzwerk: Bei diesem Beispiel sind jeder Verbindungsleitung Kosten von 1 zugeordnet. Daraus folgt, dass



einfach der Weg mit der geringsten Anzahl an Hops der günstigste ist. Die folgende Tabelle

stellt die Information da, die am Anfang im Netzwerk vorhanden ist. Zu beachten ist, dass jeder Knoten nur die Information seiner jeweiligen Zeile besitzt! Jede Zeile dieser Tabelle stellt die

	A	B	C	D	E
A	0	1	2	1	1
B	1	0	1	2	2
C	2	1	0	3	1
D	1	2	3	0	2
E	1	2	1	2	0

jeweiligen Entfernungen zu den anderen Knoten da. In diesem Beispiel besteht „die Welt“ für Knoten B folglich nur aus sich selbst und den Knoten A und C. Diese kann er mit Kosten von 1 erreichen; die Knoten D und E sind für ihn unerreichbar.

Er hätte folglich diese Routingtabelle: Diese propagiert Knoten B nun an seine direkten Nach-

Ziel	Kosten	Nächster Hop
A	1	A
C	1	C
D	$\infty$	-
E	$\infty$	-

barn.

Knoten A tut dies ebenfalls, und so empfängt Knoten B folgende Routing-Information:

Ziel	Kosten	Nächster Hop
B	1	B
C	$\infty$	-
D	1	D
E	1	E

Wenn Knoten B diese auswertet, stellt er fest, daß er mit Kosten von 2 die Knoten D und E über A erreichen kann (die Kosten, die er von A mitgeteilt bekommen hat + Kosten um A zu erreichen). Dies ist weniger als die zur Zeit vorhandenen Kosten, um diese Knoten zu erreichen ( $\infty$ ), deshalb aktualisiert der Knoten B seine Routingtabelle (siehe unten) und schickt sie wiederum an alle direkten Nachbarn.

Ziel	Kosten	Nächster Hop
A	1	A
C	1	C
D	2	A
E	2	A

Dies wird solange fortgesetzt, bis alle Knoten im Netz ein komplettes Abbild der Netztopologie besitzen. Dieses Verbreiten konsistenter Routing-Informationen über alle Knoten im Netz wird als **Konvergenz** bezeichnet. Die folgende Tabelle zeigt die in jedem Knoten letztendlich gespeicherte Entfernung. Dabei ist wiederum zu beachten, daß kein Knoten die komplette Information dieser Tabelle besitzt.

	A	B	C	D	E
A	0	1	2	1	1
B	1	0	1	2	2
C	2	1	0	3	1
D	1	2	3	0	2
E	1	2	1	2	0

**Routing-Aktualisierungen** Wie wir schon gesehen haben, sendet ein Knoten eine Routing-Aktualisierung immer dann, wenn sich die eigene Routing-Tabelle geändert hat. Man spricht in diesem Zusammenhang von einer **angestoßenen<sup>1</sup> Aktualisierung**. Ein Knoten sendet aber auch hin und wieder automatisch eine Routing-Aktualisierung, selbst wenn sich nichts geändert hat. Diese Mitteilung hat den Charakter eines Lebenszeichens. Bei dieser Form der Aktualisierung spricht man von einer **periodischen Aktualisierung**. Die Häufigkeit ist hierbei nicht zwingend vorgeschrieben. Sie schwankt zwischen mehreren Sekunden und mehreren Minuten.

**Ausfall einer Netzwerkverbindung** Was passiert nun, wenn eine Verbindungsleitung zwischen 2 Knoten ausfällt?

Diejenigen Knoten, die den Ausfall der Verbindung zuerst bemerken (z.B. dadurch, daß keine periodische Aktualisierung mehr ankommen), senden ihren Nachbarn neue Entfernungslisten. Normalerweise stabilisiert sich das System dann recht schnell. Es gibt allerdings auch Ausnahmen. Betrachten wir folgendes eingeschwungenes Netzwerk:

Nun falle die Verbindung zwischen Knoten A und D aus: Bei der nächsten Aktualisierung

---

<sup>1</sup>triggered

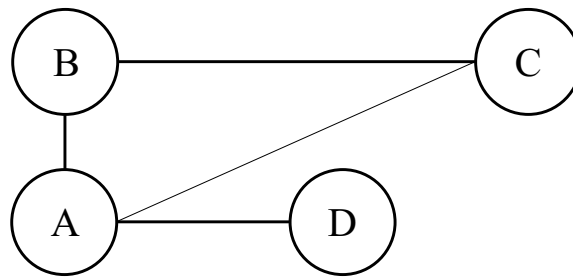


Abbildung 2.4: Beispielnetzwerk im eingeschwungenen Zustand

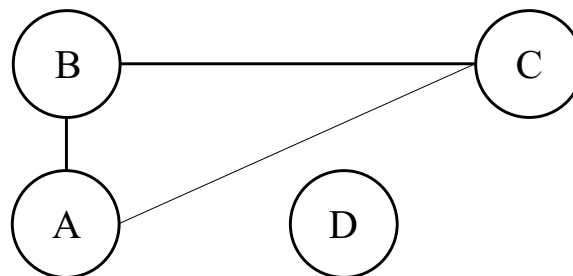


Abbildung 2.5: Netzwerk nach Ausfall der Verbindung zwischen A und D

gibt A nun unendliche Entfernung zum Knoten D an. Nun kann es aber passieren, dass genau in dem Moment, in dem Knoten A den Ausfall bemerkt hat, Knoten C eine Routing-Aktualisierung an A schickt. In dieser ist noch die Information enthalten, dass der Knoten C den Knoten D mit Kosten von 2 erreichen kann. A addiert die Kosten um C zu erreichen auf, und stellt fest, dass die Kosten um D über C zu erreichen 3 betragen. Dies ist geringer als  $\infty$  und infolgedessen wird der Eintrag in der eigenen Routingtabelle aktualisiert und wiederum eine Routing-Aktualisierung versandt. Diese erreicht auch den Knoten C, welcher feststellt, dass die Kosten um D über A zu erreichen jetzt nicht mehr 2, sondern 4 betragen. Dieser Zyklus setzt sich solange fort, bis die Entfernung einen Wert angenommen hat, der als unendlich betrachtet wird. Bis zu diesem Zeitpunkt weiß keiner der Knoten, dass D überhaupt nicht mehr zu erreichen ist. Dies wird als *Count-to-Infinity-Problem* bezeichnet.

**Lösungsmöglichkeiten für das Count-to-Infinity-Problems** Eine mögliche Lösung dieses Problems wäre, dass man eine relativ kleine Zahl als Annäherung an Unendlich annimmt. Dies wird in der Praxis auch so gehandhabt. In gängigen Protokollen wird 16 als Unendlich angesehen, was die Zeit begrenzt, die es dauert, bis ins Unendliche gezählt wird. Dies ist allerdings problematisch in Bezug auf die Netzwerkgröße. Zwei Knoten dürfen nur maximal 15 Hops voneinander entfernt sein.

Eine weitere Technik die bei Distanzvektor-Protokollen häufig verwendet wird, um solche Routing-Schleifen aufzulösen, ist das so genannte *Split-Horizon* Verfahren. Hierbei sendet ein Knoten *nur* die Routing Information an einen Nachbar-Knoten, die er **nicht** von diesem erhalten hat. So wird zum Beispiel der Eintrag  $\langle F, 5, C \rangle$  (Knoten F ist über 5 Hops erreichbar; Weitergeleitet wird an Knoten C) **nicht** an Knoten C (von dem die Information ja erhalten worden

## Interne Routing-Protokolle

sein muss!) weitergeleitet.

Eine stärkere Variante des Split-Horizon ist das *Split-Horizon with Poison Reverse*. Dabei wird die Route zwar zurückgesandt, jedoch mit negativen Informationen behaftet. In unserem Beispiel würde der Knoten also  $\langle F, \infty \rangle$  an C senden. So wird sichergestellt, dass nicht doch eine Schleife entsteht.

Diese beiden Verfahren sind jedoch nur dazu geeignet, Schleifen zwischen *genau 2* Knoten aufzulösen. Für die Auflösung größerer Routingschleifen wäre ein größerer Aufwand notwendig, welcher die schnelle Konvergenz der Algorithmen nicht mehr gewährleisten könnte. Da jedoch gerade dies ein wichtiger Aspekt der Algorithmen ist, wird in der Praxis auf die Auflösung größerer Routingschleifen verzichtet.

### 2.2.2 Beispiel: Routing Information Protocol (RIP)

Das Routing Information Protocol wurde ursprünglich als Komponente des Programms „*routed*“ der BSD Unix-Distribution entwickelt. Es wurde daraufhin in vielen Programmen verwendet und im Juni 1988 in der RFC 1058 dokumentiert. Mittlerweile existiert RIP in der Version 2 (RFC 1388), welche weitere Verbesserungen mit sich gebracht hat.

RIP ist ein sehr einfaches, leicht zu konfigurierendes Protokoll. Dies ist schon am RIP - Paketformat zu sehen: Der Großteil des Pakets besteht aus  $\langle$ Netzwerk/Adresse, Entfernung $\rangle$  -

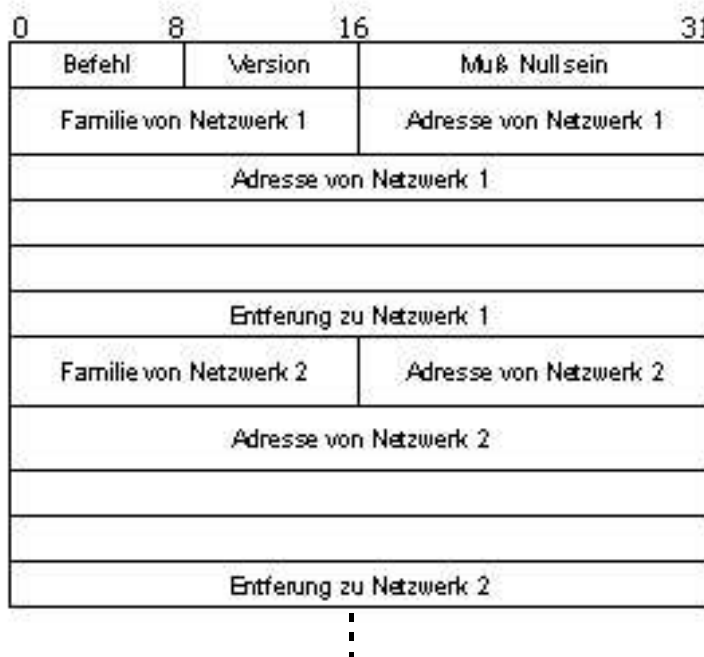


Abbildung 2.6: Ein RIP-Datenpaket

Paaren. Auffällig ist dabei, dass nur die Entfernung zum Ziel angegeben wird. Im RIP ist nur die Benutzung einer Metrik vorgesehen, nämlich die Anzahl der Hops. Weitere denkbare Metriken wie z.B. Bandbreite oder Latenzzeit einer Verbindung werden nicht unterstützt. Daraus resultiert, dass RIP immer die Verbindung mit den wenigstens Hops sucht. Die Anzahl der Hops



kann zwischen 1 und 15 liegen, während 16 als unendlich definiert ist.

Router, die RIP nutzen, senden ihre Routing-Informationen nach jeder Änderung der eigenen Routingtabelle (angestoßene Aktualisierung) sowie alle 30 Sekunden an ihre Nachbarn (periodische Aktualisierung). Dabei ist zu beachten, dass Split-Horizon implementiert ist. Wird über 6 Aktualisierungsintervalle (also 180 Sekunden) keine Information von einem Nachbarn empfangen, so wird die von ihm empfangene Information als „nicht verfügbar“ markiert. In diesem Fall gilt also ein nicht erreichbares Netzwerk noch 3 Minuten als erreichbar. Nach weiteren 4 Intervallen (also insgesamt 300 Sekunden) wird die Information endgültig aus der Routing-Tabelle entfernt. Während dieser Zeit, die als *Garbage Collection Time* bezeichnet wird, wird die Information propagiert, dass das Netzwerk nicht zu erreichen ist (Entfernung:  $\infty$ ).

## 2.3 Link-State-Routing-Algorithmen

### 2.3.1 Einführung

Die Link-State-Routing-Algorithmen stellen neben den Distanz-Vektor-Routing Algorithmen die zweite wichtige Klasse von Internen Routing-Protokollen dar. Die Grundannahmen dieser Algorithmen sind mit denen der Distanz-Vektor-Algorithmen identisch:

- Jeder Knoten kennt seine Nachbarn,
- den Zustand der Verbindungsleitungen zu diesen sowie
- die Kosten dieser Leitungen.

Dies ist die so genannte Link-State-Information. Link-State-Algorithmen gehen nun von dem Grundsatz aus, dass jeder Knoten eines Netzes eine komplette Netztopologie erstellen kann, wenn er über eben diese Information von **allen** anderen Knoten des Netzes verfügt. Daraus resultierend beruhen Link-State-Routing-Algorithmen auf zwei Mechanismen:

1. zuverlässige Verbreitung von Link-State-Informationen (Flooding);
2. Berechnung von Routen aus der Summe dieser Daten.

**Zuverlässiges Fluten (Flooding)** Unter dem Begriff *Zuverlässiges Fluten* versteht man den Prozess, der sicherstellt, dass alle Knoten eines Netzwerkes eine Kopie der Link-State-Informationen von allen anderen Knoten erhalten.

Dazu sendet ein Knoten seine Link-State-Informationen an alle direkten Nachbarn, welche diese Information wiederum an alle anderen Nachbarn weiterleiten. Dieser Prozess wird solange fortgesetzt, bis jeder Knoten über diese Information verfügt.

Die verschickte Link-State-Information, die als „**Link-State-Paket**“ (LSP) bezeichnet wird, besteht aus folgenden Komponenten:

- Eine ID des Knotens, der das LSP erzeugt hat;
- Eine Liste der Nachbarn des Knotens (inklusive Kosten um diese zu erreichen);

## Interne Routing-Protokolle

- Eine Sequenznummer;
- Eine Lebensdauer.

Die ersten beiden Komponenten dienen der Routenberechnung, während die beiden letzten das zuverlässige Fluten sicherstellen. Die Sequenznummer dient in diesem Zusammenhang dazu, alte Informationen (die sich unter Umständen noch im Netz befinden) von neuen zu unterscheiden, während die Lebensdauer dazu benutzt wird „zu alten Informationen auszusondern.

Nehmen wir an, ein Knoten A empfängt vom Knoten B ein LSP. Als erstes prüft A, ob die Lebensdauer abgelaufen ist. Sollte dies der Fall sein, wird das LSP verworfen. Als nächstes schaut A, ob in den eigenen Routing-Informationen schon eine Kopie des LSP von B vorhanden ist. Ist dies der Fall, vergleicht A die Sequenznummer des empfangenen LSPs mit dem des abgespeicherten und verwirft das empfangene LSP, falls das bereits gespeicherte LSP eine höhere Sequenznummer besitzt. In jedem anderen Fall speichert der Knoten A das LSP von B und sendet wiederum ein eigenes LSP an alle benachbarten Knoten. Dieser soeben beschriebene Mechanismus wird im folgendem Flussdiagramm noch einmal dargestellt: Analog zu den

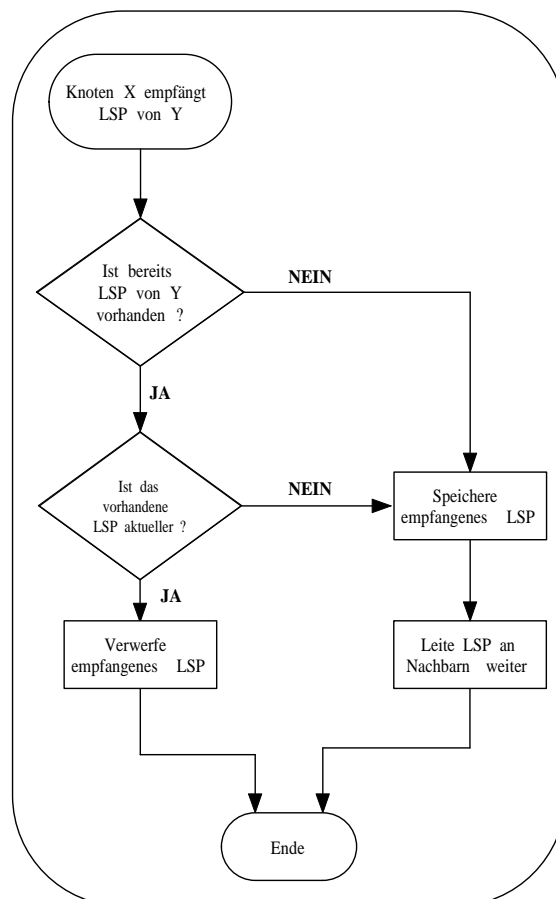


Abbildung 2.7: Vorgehen eines Knoten beim Empfang eines LSP

Distanz-Vektor-Routing Algorithmen erzeugt ein Knoten ein LSP genau in 2 Fällen:

1. in periodischen Abständen;
2. bei Änderung der Verbindung zu den direkten Nachbarn.

Dass die Verbindung zu den direkten Nachbarn noch besteht, wird durch periodisch verschickte „Hello“- Pakete sichergestellt, die quasi ein Lebenszeichen darstellen. Wird längere Zeit kein solches Paket von einem Nachbarn empfangen, geht der Knoten davon aus, dass die Verbindung zu diesem ausgefallen ist.

**Berechnung der Routen** Besitzt ein Knoten die LSPs aller anderen Knoten des Netzwerks, so berechnet er mit dem aus der Graphentheorie bekannten „Dijkstra-Algorithmus der kürzesten Pfade“ die Topologie des Netzes. Dazu verwendet er eine als **Forward-Search-Algorithmus** bezeichnete Realisierung des Dijkstra-Algorithmus. Konkret bedeutet dies, daß der Knoten zwei Listen anlegt:

1. eine sogenannte *Tentative Liste* (T-Liste) (tentativ  $\equiv$  angenommen);
2. eine *Bestätigte Liste* (B-Liste).

Beide Listen enthalten Einträge der Form  $\langle \text{Zielknoten, Kosten, Nächster Hop} \rangle$ . Der Algorithmus geht nun folgendermaßen vor:

1. Initialisiere die B-Liste mit einem Eintrag für mich selber (mit Kosten von 0).
2. Wähle den im letzten Schritt in die B-Liste eingefügten Knoten (ab sofort als Nächster bezeichnet) und betrachte sein LSP.
3. Berechne für jeden Nachbarn (Nachbar) von Nächster die Kosten um diesen zu erreichen (Kosten um Nächster zu erreichen + Kosten von Nächster zu Nachbar).
4. Steht Nachbar weder in der T- noch in der B-Liste so füge  $\langle \text{Nachbar, Kosten, Nächster Hop} \rangle$  in die T-Liste ein. Nächster Hop ist dabei der Knoten, den ich benutze um zu Nächster zu kommen. Gehe zu Schritt 6.
5. Steht Nachbar momentan in der T-Liste, überprüfe, ob Kosten geringer ist als die für Nachbar momentan in der Liste stehenden Kosten sind. Ersetze in diesem Fall den vorhandenen Eintrag durch  $\langle \text{Nachbar, Kosten, Nächster Hop} \rangle$ . Nächster Hop ist dabei der Knoten, den ich benutze um zu Nächster zu kommen.
6. Ist die T-Liste leer, so beende den Algorithmus. Die B-Liste ist die fertige Routing-Tabelle. Anderenfalls verschiebe den Eintrag mit den geringsten Kosten von der T- auf die B-Liste und fahre mit Schritt 2 fort.

Zum besseren Verständnis dieses Algorithmus schauen wir uns folgendes Beispielnetzwerk an: Die Kanten sind in diesem Beispiel mit Kosten versehen (dies könnten u.a. Latenzzeiten oder Kosten pro MB Datenvolumen sein). Gehen wir davon aus, dass der Knoten B über alle LSPs der anderen Knoten verfüge. Die Ausführung des Algorithmus ist in der folgenden Tabelle dargestellt:

Dieser Algorithmus ist zwar sehr rechenintensiv und stellt hohe Anforderungen an die Speicherkapazität des einzelnen Knotens (ein LSP für jeden Knoten im Netzwerk), es lässt sich allerdings nachweisen, dass er sich schnell stabilisiert, nicht viel Netzlast erzeugt und rasch auf Topologieänderungen sowie Knotenausfälle reagiert.

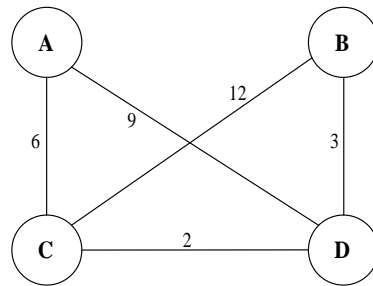


Abbildung 2.8: Beispielnetzwerk für Link-State-Routing

Schritt	Bestätigt	Tentativ	Kommentar
1	(B,0,-)		B ist das einzige neue Mitglied der B-Liste → lies sein LSP
2	(B,0,-)	(C,12,C) (D,3,D)	LSP von B besagt das wir C mit Kosten von 12 erreichen können! Dies ist besser als alles andere in beiden Listen → auf die T-Liste. Dasselbe passiert mit D!
3	(B,0,-) (D,3,D)	(C,12,C)	Setze das Mitglied mit den geringsten Kosten von der T- auf die B-Liste (hier: D). Prüfe dann das LSP des neu bestätigten Mitglieds
4	(B,0,-) (D,3,D)	(C,5,D) (A,12,D)	Die Kosten um C über D zu erreichen betragen 5, ersetze also (C,12,C) durch (C,5,D).
5	(B,0,-) (D,3,D) (C,5,D)	(A,12,D)	Verschiebe Mitglied mit den geringsten Kosten von der T- auf die B-Liste; dann prüfe sein LSP
6	(B,0,-) (D,3,D) (C,5,D)	(A,11,D)	Da A mit Kosten 6 über B erreichbar ist (insgesamt also 11), ersetze den T-Eintrag.
7	(B,0,-) (D,3,D) (C,5,D) (A,11,D)		Verschiebe das Mitglied mit den geringsten Kosten von der T- auf die B-Liste

Abbildung 2.9: Berechnung der Routing-Tabelle von Knoten B

### 2.3.2 Beispiel: Open Shortest Path First (OSPF)

OSPF ist das am weitesten verbreitete Link-State-Protokoll, welches in der Praxis verwendet wird. Die Bezeichnung setzt sich zusammen aus „Open“ und „Shortest Path First“, wobei *Open* bedeutet, dass es sich um einen offenen Standard handelt (dokumentiert in der RFC 1247 sowie in der Version 2 in der RFC 2328). *Shortest Path First* ist eine andere Bezeichnung für den verwendeten Algorithmus (es wird immer zuerst der „kürzeste“ Weg aus der Tentativen Liste verwendet).

OSPF besitzt mehrere Erweiterungen zum grundlegendem Link-State-Algorithmus. Auf diese gehe ich im folgenden kurz ein:

**Authentifikation von Routing-Nachrichten** Für die Authentifikation von Routing-Nachrichten gibt es mehrere gute Gründe. So kann ein mutwillig oder fahrlässig falsch konfigurierter Knoten ein ganzes Netzwerk zum Zusammenbruch bringen. Als Beispiel könnte man sich vorstellen, dass ein Knoten angibt, er könne jeden anderen Knoten mit Kosten von 0 erreichen. Folglich würde jeder andere Knoten alle Pakete an diesen schicken. Das Netz würde in diesem Fall vollkommen zum Stillstand kommen. Aus diesem Grund wurde in OSPF eine Sicherung in Form eines Passworts eingeführt das Routing-Aktualisierungen authentifiziert. In der ersten Version von OSPF wurde noch ein einfaches 8 Byte langes Passwort benutzt, welches sich jedoch nicht als ausreichend stark gegen böswillige Nutzer erwies. Deshalb wurde in der Version 2 ein

stärkerer Schutz implementiert.

**Zusätzliche Hierarchie** OSPF ermöglicht, es eine Domain in so genannte Bereiche (Areas) zu gliedern. Dadurch muss ein Router nicht wissen, wie er jedes einzelne Netzwerk innerhalb einer Domain erreichen kann, sondern es genügt die Kenntnis, wie er den richtigen Bereich erreicht. Dies reduziert sowohl die Anzahl der LSPs die übertragen werden, als auch die im einzelnen Knoten gespeicherte Information.

**Lastausgleich** OSPF ermöglicht es, mehrere Routen, die die gleichen Kosten für ein Ziel besitzen, zu verwalten. Diese werden dann gleichmäßig genutzt und so eine bessere Ausnutzung der Routen gewährleistet.

**Unterstützung mehrerer Metriken** OSPF ist in der Lage, mehrere Metriken gleichzeitig zu verwalten und Pakete dann nach Bedarf aufgrund einer bestimmten Metrik zu verschicken. So ist zum Beispiel der Fall denkbar, dass zwischen zwei Knoten sowohl eine Satelliten- als auch eine Glasfaserverbindung besteht. Die Satellitenverbindung habe eine Kapazität von 45 MBit/s sowie eine Latenzzeit von 300 ms, während die Glasfaserverbindung eine Kapazität von 5 MBit/s und eine Latenzzeit von 20 ms zur Verfügung stellt. In diesem Fall könnte der Knoten die Route abhängig von den zu transportierenden Daten machen. Bei einer Videokonferenz ist zum Beispiel eher eine kurze Latenzzeit der Satellitenverbindung nützlich als die hohe Bandbreite der Glasfaserverbindung. Genau konträr verhält es sich bei der Übertragung einer großen Datenmasse (z.B. eine große Binärdatei).

Denkbare Metriken wären in diesem Zusammenhang zum Beispiel:

- Anzahl der Hops,
- Kosten,
- Zuverlässigkeit,
- Durchsatz (in MBit/s),
- Minimale Verzögerung.

## 2.4 RIP $\longleftrightarrow$ OSPF : Vergleich und Ausblick

Die folgende Abbildung fasst noch einmal die wesentlichen Eigenschaften von RIP und OSPF zusammen und stellt sie gegenüber: Betrachtet man die Tabelle, so kommt man schnell zu der Überzeugung, dass OSPF RIP in eigentlich allen Punkten überlegen ist. Wozu sollte man eigentlich noch RIP verwenden?

Dabei übersieht man aber leicht den wohl schwerwiegendsten Punkt, der für den Einsatz von RIP spricht: Die Einfachheit seiner Anwendung. Gerade in kleinen Netzen macht es keinen Sinn ein aufwendig zu konfigurierendes Protokoll zu fahren, welches eindeutig überdimensioniert ist. In kleinen Netzen, in denen die Wahrscheinlichkeit relativ gering ist, dass Verbindungen ausfallen, reicht RIP vollkommen aus. Dies ist der Grund, warum in der Praxis auch heute noch häufig RIP eingesetzt wird.

RIP	OSPF
Router kennen nur direkte Nachbarn	Router haben komplette Netztopologie
Einfach zu implementieren	Relativ aufwendig zu implementieren
Relativ lange Konvergenzzeiten bei Änderungen im Netz	Schnelle Konvergenzzeiten bei Änderungen im Netz
Nur eine Metrik	Unterstützung mehrerer Metriken
Nur für relativ kleine Netze geeignet	Skalierbar → auch für grössere Netze geeignet
Häufiger Datenaustausch zwischen Knoten	Geringer Datenaustausch zwischen Knoten (im eingeschwungenem Zustand werden LSPs einmal pro h ausgetauscht)

Abbildung 2.10: Vergleich RIP ↔ OSPF

Werden die Netze jedoch größer oder sind z.B. die Anforderungen bezüglich Sicherheit, Skalierbarkeit o.ä. groß, so geht kein Weg an einem Link-State-Protokoll wie OSPF vorbei. Natürlich ist dabei auch zu bedenken, daß diese Protokolle höhere Anforderungen an die Hardware stellen.

Ein weiterer Aspekt, der bei der Auswahl eines Routingprotokolls berücksichtigt werden sollte, ist die Anpassung an zukünftige Internetstandards. So ist beispielsweise RIP in der Version 1 nicht fähig, klassenlose IP-Adressen (Supernetting) zu verarbeiten. Setzt man hier auf „das falsche Pferd“, so kann dies im Fall der Fälle zu einem erheblichen Umstellungsaufwand führen. Als ein in Zukunft auf diese Protokolle zukommendes Problem kann dabei z.B. die Einführung des IPv6-Protokolls genannt werden. Hierbei ist eine Änderung der Protokoll-Semantik wegen des vergrößerten Adressraums nicht vermeidbar<sup>2</sup>.

<sup>2</sup>RFC 2740 - OSPF for IPv6 (<http://www.ietf.org/rfc/rfc2740.txt>)

# Literaturverzeichnis

- [1] PETERSON, LARRY: „*Computernetze*“, dpunkt.verlag, 1999
- [2] HUITEMA, CHRISTIAN: „*Routing in the Internet*“, Prentice Hall, 2000
- [3] RAUTENBERG, MATHIAS: „*IP Weitverkehrsnetze*“, Siemens AG, 2000
- [4] MATHIAS HEIN: „*TCP/IP*“, 5. aktualisierte und erweiterte Ausgabe, MITP-Verlag, 2000
- [5] Kevin Washburn und Jim Evans, „*TCP/IP*“, 2. aktualisierte und erweiterte Ausgabe, Addison-Wesley, 1997
- [6] *RFC 1058, RIP*  
<http://www.ietf.org/rfc/rfc1058.txt>
- [7] *RFC 1723, RIP v2*  
<http://www.ietf.org/rfc/rfc1723.txt>
- [8] *RFC 2453, RIP v2*  
<http://www.ietf.org/rfc/rfc2453.txt>
- [9] *RFC 1247, OSPF*  
<http://www.ietf.org/rfc/rfc1247.txt>
- [10] *RFC 2328, OSPF*  
<http://www.ietf.org/rfc/rfc2328.txt>





# 3 Externe Routing-Protokolle

Marco Schuler

## Inhaltsverzeichnis

---

<b>3.1</b>	<b>Einleitung</b> . . . . .	<b>58</b>
<b>3.2</b>	<b>Wozu Externe Routing Protokolle?</b> . . . . .	<b>58</b>
3.2.1	Die Routing-Explosion . . . . .	58
3.2.2	Autonome Systeme . . . . .	59
3.2.3	EGP $\neq$ EGP . . . . .	60
<b>3.3</b>	<b>EGP</b> . . . . .	<b>60</b>
3.3.1	Die Idee von EGP . . . . .	60
3.3.2	Routing-Taktiken mit EGP . . . . .	63
3.3.3	Probleme mit EGP . . . . .	63
<b>3.4</b>	<b>BGP</b> . . . . .	<b>64</b>
3.4.1	Die Idee von BGP . . . . .	64
3.4.2	Routing-Taktiken mit BGP . . . . .	68
3.4.3	Probleme mit BGP . . . . .	68
<b>3.5</b>	<b>CIDR und BGP-4</b> . . . . .	<b>69</b>
3.5.1	B-Netz-Problem . . . . .	69
3.5.2	Supernetting . . . . .	70
3.5.3	Routing-Taktiken mit CIDR und darüber hinaus . . . . .	71
<b>3.6</b>	<b>Ausblick</b> . . . . .	<b>72</b>
	<b>Literaturverzeichnis</b> . . . . .	<b>72</b>

---

## Abstract

This work introduces the protocols used for external routing. It describes why external is differentiated from internal Routing. Subsequently, the external routing protocols in their temporal development are shown. Thereby an operation breakdown of the protocols is given. This work also describes, the problems of the protocols. Finally a short view is given on future developments.

## Kurzbeschreibung

In dieser Arbeit werden die Protokolle für das externe Routing vorgestellt. Es wird erläutert, wieso externes von internem Routing unterschieden wird. Anschließend werden die externen Routing-Protokolle in ihrer zeitlichen Entwicklung dargestellt. Dabei wird auf die Arbeitsweise der einzelnen Protokolle eingegangen. Ebenso wird aufgezeigt, wo die Grenzen des jeweiligen Protokolls liegen. Zum Abschluss wird ein kurzer Ausblick auf zukünftige Entwicklungen gegeben.

## 3.1 Einleitung

Das Internet ist ein globales Netz, in dem tausende von Netzwerken und Computern miteinander kommunizieren. Diese Kommunikation erfolgt über spezielle Rechner, so genannte Router. Router sind die Verbindungsstellen im Internet. Sie übertragen die Daten im Netz und wissen, auf welchem Weg diese Daten ihr Ziel erreichen können. Um dieses Wissen zu erhalten, kommunizieren die Router untereinander mit Hilfe der Routing-Protokolle. Diese definieren verschiedene Verfahren mit deren Hilfe die Router Informationen über die Erreichbarkeit verschiedener Ziele austauschen.

Die Routing-Protokolle lassen sich in interne Routing-Protokolle („interior gateway protocols“ - IGP) und externe Routing-Protokolle („exterior gateway protocols“ - EGP) unterteilen. Diese Ausarbeitung befasst sich mit den EGP. Es wird erläutert, wieso die EGP von den IGP unterschieden werden, welche EGP es gibt, warum und in welcher Abfolge sie entwickelt wurden und welche Routing-Taktiken mit den verschiedenen EGP möglich sind.

Diese Ausarbeitung baut auf den Seminararbeiten „Internet - Architektur und Protokolle“ von T. Morawitz sowie „Interne Routing Protokolle“ von M. Körner auf.

## 3.2 Wozu Externe Routing Protokolle?

### 3.2.1 Die Routing-Explosion

Anfang der 80er Jahre bestand das Internet aus dem ARPA-Net und einigen weiteren daran angeschlossenen Netzwerken [1]. Diese Netzwerke wurden wie ein einzelnes großes Netz verwaltet. Auf allen Routern lief das gleiche Protokoll und jeder Router kannte jedes angeschlossene Netzwerk. Anfänglich war diese Struktur ausreichend. Doch die immer stärker wachsende Anzahl von Rechnern und Netzen führte zu einer Reihe von Problemen:

- Für jedes Netzwerk muss ein Router einen Eintrag in seinen Routing-Tabellen vornehmen. Mit der steigenden Zahl von Netzen wächst auch die Größe der Routing-Tabellen und damit der Speicherbedarf der Router.
- Es gibt immer mehr Router. Die möglichen Wege durch das Netz werden länger, die Routing-Tabellen größer.
- Mit den größer werdenden Routing-Tabellen wächst der Zeitbedarf für die Berechnung günstiger Wege durch das Netz.
- Jedes mal, wenn sich an einem Teil der Netz-Topologie etwas ändert, muss die Veränderung im Netz propagiert werden. Dazu muss ein Router, je nach verwendetem Protokoll,

die gesamte Routing-Tabelle oder einen Teil davon an alle seine Nachbarn übertragen. Diese so genannten Updates werden mit steigender Zahl von Netzen und Routern sowohl häufiger als auch umfangreicher. Dadurch steigt die Belastung der Router. Gleichzeitig können weniger Nutzdaten übertragen werden, da die verfügbare Bandbreite durch die Updates eingeschränkt wird.

- Mit der wachsenden Zahl von Routern steigt die Zahl von Fehlern. Gleichzeitig wird die Fehlersuche immer schwieriger, weil beispielsweise unterschiedliche Router von unterschiedlichen Firmen das jeweilige Protokoll unterschiedlich implementiert haben.
- Die Einführung neuer Protokolle wird fast unmöglich. Ein Routing-Protokoll ist eine verteilte Anwendung, die auf allen Routern eines Netzes gleichzeitig läuft. Um ein neues Routing-Protokoll einzuführen, müssten alle Router gleichzeitig abgeschaltet und mit dem neuen Protokoll neu gestartet werden. Das ist ein fast unmögliches Unterfangen, wenn man bedenkt, dass die einzelnen Teile des frühen Internet unterschiedlichen Firmen und Organisationen gehörten.

Es musste also eine Lösung gefunden werden, die einerseits die ausufernden Probleme klein hält, andererseits die Verbundenheit des Netzes aufrecht erhält.

### 3.2.2 Autonome Systeme

Um die Routing-Explosion zu vermeiden wurde das „Problem Internet“ in Teilprobleme zerlegt. Man teilte also das Internet in viele kleinere Netze auf. Diese wurden Autonome Systeme (AS) genannt. Auch das ARPANET, der Urvater des Internet, wurde zu einem AS. Das ARPANET sollte dabei eine Art „Basisnetz“ („Backbone“) darstellen, mit dem jedes AS verbunden ist. Untereinander sollten die AS keine direkte Verbindung besitzen. Somit entstand eine Baumstruktur mit dem AS ARPANET als Wurzel. Ein AS zeichnet sich durch folgende Eigenschaften aus:

- Es ist in sich zusammenhängend (und bleibt es auch). Das bedeutet, dass innerhalb des AS alle Netzwerke und Rechner durch ein Routing-Protokoll verbunden sind und miteinander kommunizieren können.
- Es verfügt über eine einheitliche Administration.
- Es zeigt nach Außen ein einheitliches Verhalten. Von Außen betrachtet wirkt das AS wie eine homogene Einheit.
- Es hat eine weltweit eindeutige und unveränderliche Nummer. Diese besteht aus 16 Bit, kann also Werte von 0 bis 65535 annehmen. Über diese Nummer kann ein AS eindeutig identifiziert werden. Sie wird genau wie die Netzwerk-Nummern von der „Internet Assigned Numbers Authority“ (IANA [13]) zentral vergeben. Zur besseren Handhabbarkeit vergibt die IANA die Nummern jedoch nicht direkt. Sie teilt die freien AS-Nummern regionalen Organisationen zu. Diese vergeben dann die Nummern an neue AS. Für Europa unterhält die „Réseaux IP Européens“ (RIPE [14]) diesen Dienst.

Größe und Struktur eines AS ist dabei nicht festgelegt. Sie kann von einem einzigen kleinen Netz bis zu einem komplexen Netzwerk vieler Netze reichen.

## Externe Routing-Protokolle

Innerhalb eines AS entscheidet die Administration über Art, Umfang und Konfiguration der verwendeten Protokolle. Da die hierfür verwendeten Protokolle innerhalb der AS eingesetzt werden fasst man sie unter dem Namen „interne Routing-Protokolle“ („interior gateway protocols“ - IGP) zusammen.

Daneben müssen auch zwischen den AS Daten ausgetauscht werden können, um die Verbundenheit des gesamten Internet sicherzustellen. Die Router eines AS müssen herausfinden können, wie Rechner eines anderen AS erreicht werden können. Es mussten also ein neue Routing-Protokolle entwickelt werden, mit deren Hilfe die Router unterschiedlicher AS kommunizieren können. Diese Protokolle werden „externe Routing-Protokolle“ („exterior gateway protocols“ - EGP) genannt.

### 3.2.3 EGP $\neq$ EGP

An dieser Stelle sei noch auf einen Punkt verwiesen, der zu Missverständnissen führen kann. Die Abkürzung EGP gibt es in zwei Bedeutungen. Zum einen ist es die allgemeine Bezeichnung für alle externen Routing-Protokolle - zum anderen bezeichnet es ein spezielles externes Routing-Protokoll. In beiden Fällen heißt EGP „exterior gateway protocol“.

## 3.3 EGP

Das erste Protokoll, welches der Kommunikation zwischen verschiedenen Autonomen Systemen (AS) dienen sollte, war das „externe Routing-Protokoll“ („exterior gateway protocol“ - EGP). Es wurde von 1982 bis 1984 als Standard der „Internet Engineering Task Force“ (IETF [12]) veröffentlicht [6].

### 3.3.1 Die Idee von EGP

EGP wird nicht zur eigentlichen Datenübertragung verwendet. Es ermöglicht jedoch den Routern verschiedener AS Informationen über die gegenseitige Erreichbarkeit auszutauschen. Zu diesem Zweck sieht das Protokoll die folgenden drei Stufen der Kommunikation vor, auf die im Anschluss näher eingegangen wird:

- ermitteln, welche Nachbarn vorhanden sind
- prüfen, ob ein (vorher ermittelter) Nachbar noch erreichbar ist
- mitteilen, welche Netzwerke über den jeweiligen Nachbar erreichbar sind

EGP unterscheidet also zwischen der grundsätzlichen Nachbarschaft zweier Router und der tatsächlich bestehenden und funktionierenden Kommunikationsverbindung zwischen ihnen. EGP verwendet das „internet protocol“ (IP), um Routing-Informationen auszutauschen. Dies bringt verschiedene Vorteile mit sich. So kann EGP detailliert einzelne Paketverluste und Antwortzeiten überwachen, da auf IP-Ebene keine gesicherte Ende-zu-Ende-Verbindung besteht. Die Sicherung der Daten, also die Kontrolle, ob alle Informationen auch vollständig und in der richtigen Reihenfolge ankommen, erfolgt erst auf der Ebene des „transmission control protocol“ (TCP). Dies wird jedoch von EGP nicht verwendet. EGP kann also aus dem Ankommen und Ausbleiben der IP-Pakete Schlussfolgerungen über die Leitungsqualität ziehen, und darauf reagieren. Eine Reaktion kann zum Beispiel das Trennen einer schlechten Verbindung sein. Ein

weiterer Vorteil, der durch die Verwendung von IP entsteht ist der relativ geringe Protokoll-Overhead, der bei Verwendung von beispielsweise TCP deutlich größer gewesen wäre.

Ein IP-Paket, wie es von EGP verwendet wird, besteht nur aus dem IP-Header und den EGP-Daten (Abbildung 3.1).

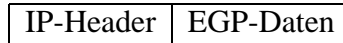


Abbildung 3.1: IP-Paket mit EGP-Daten

Im Gegensatz dazu folgt nun die Skizze eines IP-Paketes, wenn TCP verwendet werden würde. Es besteht aus dem IP-Header und dem TCP-Paket (Abbildung 3.2).

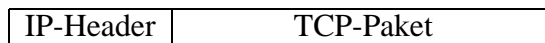


Abbildung 3.2: IP-Paket mit einem TCP-Paket im Datenteil

Das TCP-Paket selbst besteht wiederum aus dem TCP-Header und den EGP-Daten (Abbildung 3.3).

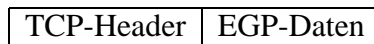


Abbildung 3.3: TCP-Paket mit EGP-Daten

Die Verwendung von IP hat jedoch auch Nachteile. Wie bereits angesprochen garantiert IP das Ankommen der Pakete nicht. EGP muss also die übermittelten Routing-Daten prüfen und gegebenenfalls korrigieren. Das wirkt sich negativ auf die Komplexität von EGP aus. Das Protokoll ist also durch die Verwendung von IP komplizierter zu implementieren, als es bei der Verwendung von TCP wäre.

EGP läuft nicht auf allen Routern eines AS. Nur spezielle, von der Administration ausgewählte Router stellen mittels EGP den Kontakt zu anderen AS her. Diese Router werden als „Grenz-Router“ („border router“) bezeichnet.

### Ermitteln der Nachbarn

Das Ermitteln der Nachbarn erfolgt in einem einfachen Handshake-Verfahren. Ein Router sendet eine Anfrage nach neuen Routern „am anderen Ende der Leitung“. Als Antwort erhält er entweder eine Bestätigung, dann gilt die Nachbarschaft als hergestellt, oder eine Ablehnung, dann gilt der ablehnende Router als nicht existent. Router können die Nachbarschaft aus verschiedenen Gründen verweigern, häufiger Grund ist zum Beispiel ein administratives Verbot.

Um eine bestehende Nachbarschaft zu beenden wird eine Beenden-Nachricht verschickt, die von der Gegenseite bestätigt wird. Damit existiert diese Nachbarschaft nicht mehr. Das Beenden einer Nachbarschaft wird beispielsweise zum kontrollierten Schließen einer Verbindung genutzt, wenn ein Router abgeschaltet werden soll.

### Erreichbarkeit der Nachbarn

EGP prüft regelmäßig, ob die Verbindung zwischen benachbarten Routern noch besteht. Dazu sendet ein Router eine Hallo-Nachricht („Hello“) an seinen Nachbarn. Dieser reagiert mit einer

„Ich habe dich gehört“-Antwort („I heard you“ - IHU). Dieser Handshake passiert typischerweise alle 30 Sekunden.

Erfolgt nun auf ein Hello kein IHU, so wird die Verbindung nicht sofort als unterbrochen gewertet. Umgekehrt gilt eine unterbrochene Verbindung auch nicht sofort als wieder hergestellt, wenn ein Handshake geklappt hat. In einer Implementierung von EGP wird ein mehrstufiges Verfahren vorgeschlagen [7]. Eine unterbrochene Verbindung wird dann als wiederhergestellt gewertet, wenn von den letzten vier Handshakes mindestens drei erfolgreich waren. Umgekehrt gilt eine bestehende Verbindung erst dann als unterbrochen, wenn von den letzten vier Handshakes kein einziger Erfolg hatte. Dieses Verfahren reduziert die negativen Auswirkungen von schlechten Verbindungen, die zu stetigen Fluktuationen der Routing-Informationen und zu verstärktem Datenaufkommen führen würden.

### Erreichen der Netzwerke

Wie bereits beschrieben (siehe 3.3.1), ist es das Ziel von EGP den Routern eines AS Informationen über die Erreichbarkeit von Netzwerken in anderen AS zu übermitteln. Dies geschieht, indem der Router, der entsprechende Informationen erhalten will, aktiv nach diesen Informationen fragt. Er verschickt eine Anfrage an alle seine (erreichbaren) Nachbarn. Von jedem erhält er eine vollständige Liste von Netzen, die über diesen Nachbarn zu erreichen sind. Dieses Nachfragen wird als Polling bezeichnet.

Nun kann der Router die empfangenen Netze in seine Routing-Tabelle aufnehmen. Dabei kann es vorkommen, dass ein Netz dort bereits vorhanden ist, beispielsweise wenn dieses Netz auch über einen anderen Router erreichbar ist. Dann verwendet EGP eine einfache Metrik, um zu entscheiden, welcher Eintrag beibehalten werden soll. Die Metrik besteht aus einer Zahl, welche 8 Bit lang ist und somit Werte von 0 bis 255 annehmen kann. Diese Zahl ist jedoch nicht als Anzahl der Hops zu verstehen, die diese Information zurückgelegt hat, wie es zum Beispiel beim „routing information protocol“ (RIP) der Fall ist. EGP schreibt keine genaue Handhabung dieser Metrik vor. Festgelegt wurde, dass der Wert 255 für unendlich steht und damit Unerreichbarkeit anzeigt. Weiterhin bedeutet eine kleine Metrik einen „besseren“ Weg zum Ziel als eine große Metrik. Die kleine Metrik ist daher vorzuziehen. Was in diesem Fall „besser“ bedeutet ist durch EGP nicht festgelegt. Bei der Metrik kann es sich um die Anzahl der Hops handeln, die Metrik kann eine Latenzzeit sein, sie kann aber auch ein willkürlicher Wert sein, den ein AS-Administrator vom Router dort eintragen lässt. Was die Metrik letztlich bedeutet, und welche Werte von den Routern in der Metrik eingetragen werden, liegt einzig und allein bei der Administration des jeweiligen Autonomen Systems und an der Implementation von EGP die auf dem entsprechenden Router läuft.

Um jedes Daten-Paket in die richtige Richtung weiterleiten zu können, muss ein Router seine Routing-Informationen ständig aktualisieren. Wie bereits beschrieben erfolgt dies über ein Polling-Verfahren. Der Router könnte nun versuchen ständig und ununterbrochen ein Polling mit seinen Nachbarn zu betreiben. Dies würde zu einer Auslastung der Verbindung führen. Es könnten keine Nutzdaten mehr übertragen werden. Aktualisiert ein Router seine Informationen dagegen zu selten, kann er nicht zeitgerecht auf Änderungen der Netz-Topologie reagieren und sendet seine Daten-Pakete möglicherweise in die falsche Richtung. Weiterhin muss EGP bei jedem Polling die gesamte Routing-Information übertragen und nicht etwa nur die Veränderungen zur vorherigen Information. Das liegt an der Verwendung von IP. Es ist nicht garantiert, dass die zuletzt erhaltenen Information vollständig ist. Die Menge der jeweils übertragenen Routing-Informationen ist also relativ groß. Es muss demnach ein Kompromiss zwischen der

verfügbaren Bandbreite für die Nutzdaten und der Aktualität der Routing-Informationen gefunden werden. Bei EGP geschieht die Aktualisierung deshalb typischerweise nur alle zwei Minuten.

### 3.3.2 Routing-Taktiken mit EGP

Wenn Router eines Autonomen Systems die Nachbarschaft mit Routern eines anderen Autonomen Systems akzeptieren, stimmen sie gleichzeitig zu, dass das andere Autonome System seine Daten durch das eigene Autonome System übermitteln darf. Dies kann aus vielen Gründen nicht erwünscht sein, beispielsweise wenn die Datenübertragung Kosten verursacht oder wenn die Bandbreite im eigenen Autonomen System durch die Daten des anderen Autonomen Systems zu stark begrenzt würde. Ein Administrator kann deshalb seinen Routern die Nachbarschaft zu bestimmten Routern verbieten. Ein gängiges Verfahren hierzu ist es, den Routern eine Liste aller potentiell möglichen Nachbarn zu geben. Router auf dieser Liste werden als Nachbarn akzeptiert, andere nicht.

Mehr Einfluss auf die Routing-Taktiken ist mit EGP kaum möglich, da bei der Verwendung von EGP mehrere Probleme auftreten.

### 3.3.3 Probleme mit EGP

EGP ist fehleranfällig. Ein einzelner, falsch konfigurierter Router kann das Routing zwischen den Autonomen Systemen erschweren oder ganz verhindern. Dazu reicht es schon, wenn er für ein Netzwerk eine zu hohe oder zu niedrige Metrik angibt. Hierfür ein Beispiel: Abbildung 3.4 zeigt drei miteinander verbundene AS. In den AS gibt es Netzwerke und am Rand der AS stellen die Grenz-Router (A bis D) die Verbindung zwischen den AS her. Router A gibt die Erreichbarkeit der Netze in AS 1 mit einer bestimmten Metrik weiter. Diese Information gelangt über Router B und C auch zu Router D. Router D könnte nun, aufgrund einer Fehlkonfiguration, für die Netze in AS 1 eine kleinere Metrik angeben als Router A. Sobald Router C seine Routing-Informationen aktualisiert, bekommt er diese zu kleine Metrik von D mitgeteilt. C wird nun seine Routing-Tabelle ändern und alle Datenpakete für Netze in AS 1 zu Router D schicken, weil dieser den kürzesten Weg dorthin anzeigt. D schickt diese Datenpakete jedoch wieder an C, weil dies der einzige Weg zu AS 1 ist. Es entsteht eine Schleife, die nicht automatisch aufgelöst werden kann. An dieser Stelle ist manuelle Konfiguration oft der einzige Lösungsweg.

Ein weiteres Problem bei EGP ist die langsame Konvergenz der Routing-Informationen. Da Änderungen an der Topologie im schlechtesten Fall nur alle zwei Minuten weitergegeben werden, können in der komplexen Topologie des Internet möglicherweise Stunden vergehen, bis alle Routing-Tabellen aktualisiert sind. Darüberhinaus beansprucht EGP relativ viel Bandbreite, da bei jeder Aktualisierung der Routing-Informationen die gesamten Routing-Tabellen übertragen werden.

Das größte Problem mit EGP aber ist das Unvermögen Schleifen sicher aufzulösen. EGP war nur für den Einsatz in der bereits erwähnten Baumstruktur ohne Schleifen gedacht (siehe 3.2.2). Entsprechend wurde EGP auch konzipiert. EGP besitzt nur eine einfache Metrik in Form einer Zahl (siehe 3.3.1). Damit ist es dem Protokoll nicht möglich, Schleifen in den Topologien sicher aufzulösen. Das obige Beispiel hat gezeigt, dass sich Datenpakete sogar in eigentlich schleifenlosen Topologien in einer Schleife verfangen können. Selbst wenn alle Schleifen manuell aufgelöst sind und die Router zu allen Zielen den kürzesten Pfad anzeigen ist EGP überfordert. Denn in vielen Fällen ist nicht der kürzeste Weg durch das Netz auch der gewünschte Weg.

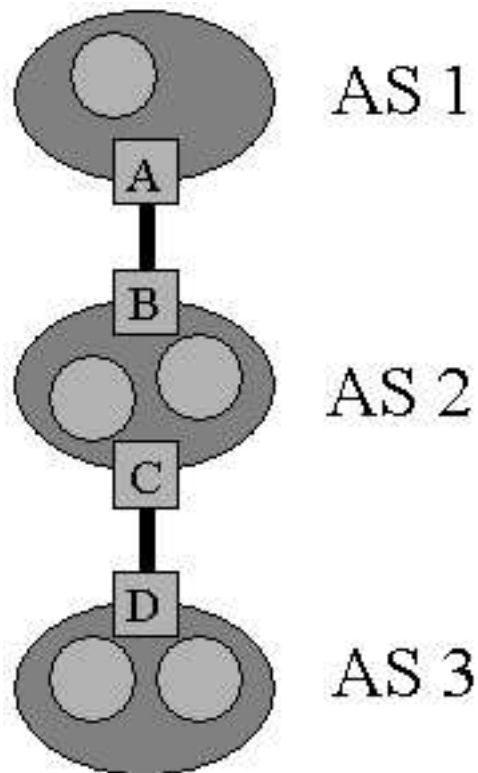


Abbildung 3.4: drei Autonome Systeme

Oft ist zum Beispiel der schnellste, sicherste, billigste oder durchsatzstärkste Weg erwünscht. Den Anforderungen des immer weiter wachsenden Internet ist EGP also nicht mehr gewachsen gewesen. Ein neues externes Routing-Protokoll musste entwickelt werden.

## 3.4 BGP

Um die Unzulänglichkeiten von EGP zu überwinden wurde das „border gateway protocol“ (BGP) entwickelt. Es wurde in den Versionen BGP-1 (1989), BGP-2 (1990) und BGP-3 (1991) als Standard von der IETF veröffentlicht [9].

### 3.4.1 Die Idee von BGP

Um Routing zu betreiben gibt es verschiedene Ansätze. Einer sind die so genannten Distanzvektoren. Dabei wird für jedes erreichbare Ziel eine Richtung und eine Distanz gespeichert. Ähnlich verfährt auch EGP und hat verdeutlicht, dass dieser Ansatz für externes Routing nicht ausreicht, da Schleifen nicht sicher aufgelöst werden können. Eine weitere Möglichkeit ist ein Linkstate-Ansatz. Dabei wird jedem Router die gesamte Netz-Topologie mitgeteilt. Aus diesen Informationen kann er selbständig den besten Weg zu einem Ziel ermitteln. Damit könnten Schleifen dann sicher aufgelöst werden. Ein Linkstate-Ansatz wäre aber für externes Routing deutlich zu komplex. So wurde beispielsweise für das interne Routing-Protokoll „open shortest path first“ (OSPF), welches einen Linkstate-Ansatz verwendet, ein Maximum von 200 Knoten empfohlen. Als BGP-1 1989 entwickelt wurde gab es aber bereits über 700 Autonome Systeme



[1]. Für BGP musste also ein völlig neuer Ansatz entwickelt werden.

Es entstand das Konzept der so genannten Pfadvektoren. Für jedes Ziel wird nicht nur die Richtung, sondern der gesamte Pfad bis zum Ziel gespeichert und an die Nachbarn weitergegeben. Das Verfahren wird in 3.4.1 genauer beschrieben.

BGP geht zur Ermittlung der Routing-Informationen ähnlich wie EGP vor. Es benutzt folgende Nachrichtentypen zur Kommunikation:

- „Open“ (Öffnen einer Verbindung)
- „Keep Alive“ (bestehende Verbindung prüfen)
- „Update“ (Änderungen der Netz-Topologie mitteilen)
- „Notification“ (Fehlermeldung oder Antwort auf eine Anfrage)

Wie man sehen kann, entsprechen die ersten drei Nachrichtentypen in etwa denen von EGP. Ein Unterschied tritt jedoch deutlich hervor. Wie aus dem Namen Update bereits ersichtlich ist, wird nicht wie bei EGP jedesmal die gesamte Liste der erreichbaren Netzwerke übertragen, sondern nur das, was sich an dieser Liste geändert hat. Das ist möglich, weil BGP als Kommunikationsprotokoll das „transmission control protocol“ (TCP) verwendet. TCP erhöht zwar den Protokoll-Overhead (siehe 3.3.1), garantiert jedoch das richtige Ankommen der Informationen, das heißt es stellt sicher, dass alle übertragenen Informationen vollständig und in der richtigen Reihenfolge ankommen. Da man also sicher sein kann, dass alle früheren Informationen sicher angekommen sind, reicht es aus nur die Veränderungen zu übertragen. Tritt keine Veränderung an der Topologie ein, müssen auch keine Routing-Daten übermittelt werden. Die für eigentliche Anwendungen zur Verfügung stehende Bandbreite wächst.

### **Open**

Da BGP auf TCP beruht, muss vor dem Herstellen einer BGP-Verbindung zwischen zwei Routern zuerst eine TCP-Verbindung hergestellt werden. Ist dies geschehen, erfolgt ein mehrfacher Handshake. In diesem werden die Protokollversion und bestimmte Mindest- und Höchstzeiten abgesprochen. Die beiden Router senken dabei langsam ihre Mindestanforderungen an ihr Gegenüber, bis sie sich auf einen gemeinsamen Standard geeinigt haben. BGP bietet die Möglichkeit in den Open-Nachrichten Beglaubigungsdaten zu übermitteln. Damit können die Router Kennungen und Passwörter übermitteln, um sich gegenseitig zu authentifizieren. Das Verschlüsseln der Routing-Informationen, die zwischen den Routern ausgetauscht werden, ist allerdings nicht sehr sinnvoll, da TCP selbst die Daten unverschlüsselt überträgt. Demnach enthält die Beschreibung von BGP auch keinen Sicherheitsalgorithmus, der die Routing-Informationen sichert [9].

### **Keep Alive**

Eine Keep-Alive-Nachricht wird verschickt, um das Weiterbestehen der Verbindung zu überprüfen. Jeder Router sendet sie typischerweise alle zwei Minuten an alle seine Nachbarn. Erhält ein Router von einem seiner Nachbarn keine Keep-Alive-Nachricht, so schickt er ihm eine entsprechende Fehlermeldung und bricht die Verbindung ab. Die Keep-Alive-Nachrichten werden nicht beantwortet. Das Ankommen der eigenen Nachrichten kann ein Router aus dem Ausbleiben der Abbruch-Fehlermeldung von seinem jeweiligen Kommunikationspartner schließen.

Marker (16 Byte)	Länge(2 Byte)	Typ (1 Byte)
------------------	---------------	--------------

Abbildung 3.5: Aufbau des BGP-Headers

Bricht beispielsweise eine Verbindung zwischen zwei Routern vollständig zusammen, dann erhalten beide vom jeweils anderen keine Keep-Alive-Nachrichten mehr. Jeder sendet (an sein nicht mehr vorhandenes Gegenüber) eine Fehlermeldung und bricht die Verbindung ab. Die physikalisch unterbrochene Verbindung wird also auch logisch getrennt.

Die Keep-Alive-Nachrichten sind nur 19 Byte groß und bestehen aus dem BGP-Header, der in Abbildung 3.5 dargestellt ist. Dabei enthält das Feld „Marker“ die in 3.4.1 beschriebenen Beglaubigungsdaten. Wie bereits erläutert wird BGP aber nicht gesichert. Um den nun ungenutzten Marker in einem definierten Zustand zu halten, sind in diesem immer alle Bits auf 1 gesetzt, das heißt der Marker besteht aus 16 Bytes mit dem Wert 255. Das Feld „Länge“ enthält die Länge der an den Header angehängten Daten. Im Falle einer Keep-Alive-Nachricht gibt es außer dem Header keine Daten - das Feld „Länge“ enthält 2 Bytes mit dem Wert 0. Das Feld Typ enthält eine Zahl von 1 bis 4 die den Typ der Nachricht beschreibt. 1 für Open, 2 für Update, 3 für Notification und 4 für Keep Alive. Eine vollständige Keep-Alive-Nachricht sieht also wie in Abbildung 3.6 dargestellt aus. Durch die geringe Größe einer Keep-Alive-Nachricht und

255 255 255 255 255 255 255 255 255 255 255 255 255 255 255	0 0	4
---	-----	---

Abbildung 3.6: eine Keep-Alive-Nachricht

die Tatsache, dass sie nicht beantwortet werden muss, wird nur wenig Bandbreite verbraucht. Es können mehr Nutzdaten übertragen werden.

### Update

Update-Nachrichten beinhalten, wie bereits angesprochen (siehe 3.4.1), nur die Veränderung zu vorher bereits übermittelten Informationen. Sobald ein Router eine Veränderung der Netz-Topologie feststellt, aktualisiert er seine Routing-Tabelle. Hat sich in der Tabelle etwas verändert, so wird diese Änderung sofort an alle Nachbarn weitergegeben. Update-Nachrichten werden also bei BGP, im Gegensatz zu EGP (siehe 3.3.1), unaufgefordert versendet. Damit konvergiert die Routing-Information deutlich schneller als bei EGP.

Eine Update-Nachricht besteht aus einem Pfad, Pfadattributen und einer Liste von Netzwerken. Der Pfad ist eine Liste von Autonomen Systemen, die Schritt für Schritt einen Weg durch das Internet beschreibt. Die Pfadattribute beschreiben verschiedene Eigenschaften die dieser Pfad hat (siehe 3.4.1). Die Liste von Netzwerken schließlich enthält alle die Netzwerke, die genau über den angegebenen Pfad erreicht werden können. Zur weiteren Erklärung sei Abbildung 3.7 angeführt. Router A kann das Netzwerk N direkt erreichen. In der Routing-Tabelle von Router A ist der Pfad zu diesem Netzwerk also leer. Meldet er die Erreichbarkeit von N an Router B, so hängt Router A die Nummer seines eigenen AS vorn an den Pfad des Netzwerkes N an. In der Routing-Tabelle von Router B erfolgt also ein Eintrag des Netzwerkes N mit dem Pfad „AS 1“. Router B gibt diese Information über interne Routing-Protokolle an Router C weiter. Gibt nun Router C die Erreichbarkeit von N an D bekannt, fügt er wieder sein eigenes AS vorn an den Pfad von N an. D trägt also das Netzwerk N mit dem Pfad „AS 2, AS 1“ in seine Routing-Tabelle ein und gibt die Information an E weiter. Dieser verfährt ebenso und

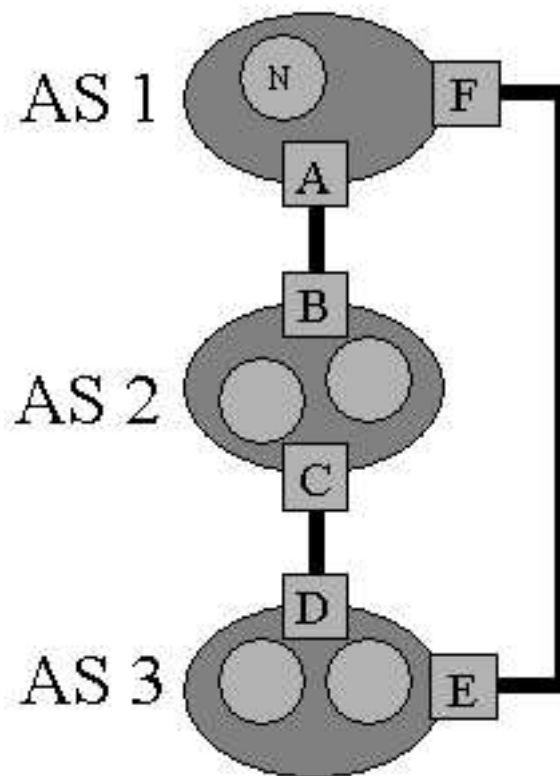


Abbildung 3.7: drei Autonome Systeme

meldet an Router F die Erreichbarkeit des Netzes N über den Pfad „AS 3, AS 2, AS 1“. Router F findet im Pfad, der ihm von E gemeldet wird, sein eigenes AS vor. Damit steht fest, dass diese Routing-Information eine Schleife zurückgelegt hat, weil nur ein Router aus dem AS 1 das AS 1 in diesen Pfad eingetragen haben kann. Die gesamte Update-Nachricht, die diesen Pfad enthält wird vom Router F sofort gelöscht. Dadurch geht die Erreichbarkeit von N für den Router F nicht verloren, da das Netz N ja schon auf dem kürzeren Pfad, ohne die Schleife, erreicht werden konnte. Ein Entstehen von Schleifen wird durch dieses Prinzip unmöglich gemacht, egal wie komplex die Struktur des Internet auch wird.

So wie eben beschrieben erhält schließlich jeder Router die Pfade zu allen Netzen. Dabei kommt es vor, dass ein Router ein Netz mit zwei verschiedenen Pfaden angezeigt bekommt. Im Beispiel bekommt Router C das Netz N von B auf dem Pfad „AS 1“ und von D auf dem Pfad „AS 3, AS 1“ angezeigt. Nun entscheidet der Router anhand der Pfadattribute, welcher Weg der aus seiner Sicht „bessere“ ist (siehe 3.4.1). Dieser wird dann in der Routing-Tabelle eingetragen.

### Pfadattribute in BGP

Pfadattribute beschreiben die Eigenschaften eines Pfades (siehe 3.4.1). Mit diesen Eigenschaften haben Router die Möglichkeit, aus mehreren verschiedenen Pfaden zu einem Ziel den für sie besten auszuwählen. Kriterien hierfür können unter anderem die Anzahl der AS durch die der Pfad läuft, die minimale Bandbreite, die Latenzzeit, die Kosten, die Stabilität, die Datensicherheit oder die Abhörsicherheit sein. Einige dieser Eigenschaften sind transitiv, zum Beispiel die minimale Bandbreite. Bei der minimalen Bandbreite muss jeder Router den bisherigen Wert für

die minimale Bandbreite aus den Pfadattributen auslesen und mit der minimalen Bandbreite im eigenen AS vergleichen. Der kleinere der beiden Werte wird als neues Pfadattribut eingetragen und mit dem nächsten Update an die Nachbarn gemeldet. Andere Attribute, wie zum Beispiel die Kosten, sind kumulativ. Sie müssen von Router zu Router aufaddiert werden. Jeder Pfad kann mehrere dieser Attribute besitzen. Zusätzlich enthält der Pfad selbst gewisse Informationen. Aus ihm ist ersichtlich wieviele und welche AS der Pfad durchläuft. Einem Router stehen damit viele Informationen zur Verfügung. Es lassen sich komplexe Auswahlfunktionen erstellen. So können beispielsweise bestimmte Attribute oder auch bestimmte AS von der Administration des Routers eine Gewichtung erhalten und so unterschiedlich stark in die Bestimmung des „besten“ Pfades einfließen.

### 3.4.2 Routing-Taktiken mit BGP

Wie bereits beschrieben, kann mit BGP detailliert der Pfad ausgesucht werden, auf dem Informationen versendet werden. Damit hat ein Router, beziehungsweise die Administration eines Routers, viel Spielraum bei der Pfadauswahl. Allerdings kann nur aus den Pfaden ausgewählt werden, die dem Router von Außen mitgeteilt werden. Einen neuen Pfad, der dem Router nicht von Außen gemeldet wurde, kann der Router nicht festlegen. Genaugenommen kann ein Router also nur bestimmen, zu welchem Router ein Datenpaket als nächstes gelangen soll.

Mit BGP kann wie bei EGP bestimmt werden, welche Router, und damit welche Autonomen Systeme, Nachbarn werden dürfen und welche nicht.

### 3.4.3 Probleme mit BGP

BGP löst alle Probleme, die bei EGP aufgetreten sind. Allerdings schafft es sich auch ein neues - die Größe der Routing-Tabellen. Durch BGP muss ein Eintrag in einer Routing-Tabelle neben den zu erreichenden Netzwerken auch den Pfad dorthin und die Pfadattribute enthalten. Weiterhin muss der Router speichern, welchen seiner Nachbarn er diese Informationen bereits übermittelt hat, da er neue Informationen unaufgefordert an alle seine Nachbarn senden muss. Auch die Anzahl der Einträge wächst rapide an. Dadurch erreichen Routing-Tabellen leicht eine Größe von einigen Megabyte [8]. Bei dieser Größenangabe sind lediglich die reinen Routing-Daten einbezogen. Zusätzlich muss ein Router, je nach Implementation des BGP, viele zusätzliche Verwaltungs-Daten speichern. Das Problem an der Größe der Routing-Tabellen ist nun nicht nur das Speichern selbst, sondern auch das Verwalten dieser Daten. In ihnen muss sehr schnell gesucht, gelesen und geschrieben werden. Mit wachsender Größe der Tabellen sinkt die Geschwindigkeit, mit der dies geschieht. Von dieser Geschwindigkeit ist aber der Paketdurchsatz eines Routers entscheidend abhängig.

Anfang der 90er Jahre kündigten sich weitere Probleme, nicht mit BGP, sondern mit dem Internet Protokoll IPv4 an. IPv4 verwendet 32 Bit um eine Adresse im Internet zu beschreiben. Von diesen 32 Bit wird ein Teil zur Adressierung des Netzwerkes, der Rest zur Adressierung des Hosts (innerhalb des Netzwerks) verwendet. Dabei treten 3 Typen von Netzen auf:

- Klasse-A-Netze: Diese verwenden 8 Bit für das Netzwerk und 24 Bit für die Hosts. Es gibt also maximal  $2^8 = 256$  Klasse-A-Netzwerke mit jeweils  $2^{24} = 16.777.216$  möglichen Hosts. Die Netzadressen der Klasse-A-Netze reichen von 0.x.x.x bis 127.x.x.x.
- Klasse-B-Netze: Klasse-B-Netze verwenden 16 Bit für die Netzwerkadresse und 16 Bit für die Hostadressen. Somit können höchstens  $2^{16} = 65.536$  Klasse-B-Netze mit jeweils

$2^{16} = 65.536$  Hosts existieren. B-Netz-Adressen liegen im Bereich von 128.0.x.x bis 191.255.x.x.

- Klasse-C-Netze: Bei C-Netzen werden 24 Bit für die Netzadresse verwendet, womit 8 Bit für die Hosts bleiben. Damit lassen sich  $2^{24} = 16.777.216$  C-Netze mit je  $2^8 = 256$  Hosts realisieren. Die Adressen der C-Netze reichen von 192.0.0.x bis 223.255.255.x.

Mit 32 Bit sind über 4 Milliarden Adressen ansprechbar. In der Praxis jedoch stehen bei Weitem nicht so viele Adressen zur Verfügung. Ein Teil des Adressraumes (von 224.0.0.0 bis 255.255.255.255) ist für Multicast und andere Anwendungen reserviert. Aber auch von den Adressen innerhalb der A-, B- oder C-Netze werden viele verschwendet, da die zugewiesenen Adressen nicht vollständig genutzt werden.

Beispiel: Ein Unternehmen will ein Netz mit 2.500 Hosts aufbauen. Da ein C-Netz hierfür zu klein wäre beantragt es ein Klasse-B-Netz. Dieses bekommt das Unternehmen von einer dazu berechtigten Organisation (IANA [13], RIPE [14], DENIC [15]) zugewiesen, beispielsweise das Netz 130.17.x.x. Das Unternehmen verwendet von den möglichen 65.536 Adressen innerhalb dieses Netzes jedoch nur 2.500. Damit bleiben 63.036 Adressen ungenutzt. Somit kann nur ein Bruchteil der verfügbaren Host-Adressen auch wirklich verwendet werden. Das Beispiel zeigt noch ein weiteres Problem auf. B-Netze sind sehr beliebt, weil die 256 Hosts eines C-Netzes vielen Unternehmen und Organisationen nicht ausreichen. Diese weichen somit auf Klasse-B-Netze aus. Doch es gibt nur 65.536 Klasse-B-Netze. Und bereits 1994 war die Hälfte der verfügbaren Klasse-B-Netze vergeben [1]. Die freien Internet-Adressen, und schlimmer noch, die freien Klasse-B-Netze wurden knapp.

Es musste also eine Lösung gefunden werden, die die großen Routing-Tabellen kleiner macht und gleichzeitig die Adressknappheit beseitigt - zumindest bis eine neue Version von IP eingeführt ist, welche deutlich mehr Adressen zulässt.

## 3.5 CIDR und BGP-4

Um die durch BGP und IPv4 verursachten Probleme zu lösen wurde das „classless inter domain routing“ (CIDR) entwickelt. Es sollte die Routing-Tabellen verkleinern und gleichzeitig die Knappheit von Internet-Adressen und Klasse-B-Netzen soweit zeitlich nach hinten schieben, bis eine neue Version von IP entwickelt und implementiert worden ist. CIDR ist dabei kein Protokoll sondern eine Idee, wie man diese Probleme beherrschen kann. Implementiert wurde CIDR in BGP-4 welches von 1992 bis 1994 entwickelt wurde und 1994 von der „Internet Engineering Task Force“ (IETF [12]) als Standard veröffentlicht wurde [10].

### 3.5.1 B-Netz-Problem

Da nur noch wenige Klasse-B-Netze zur Verfügung standen, jedoch eine große Anzahl von Klasse-C-Netzen frei war, versuchte man mehrere Klasse C-Netze zu einem Netz zusammenzufassen. Dabei versuchte man sich auch am jeweiligen Bedarf zu orientieren und gerade so viele C-Netze verteilen, wie benötigt wurden. Das hat den Vorteil, dass man keine B-Netze für kleinere Unternehmen verbraucht. Diesen werden nur so viele C-Netze zugewiesen wie sie auch benötigen. Damit werden wesentlich weniger Hosts verschwendet. Ganz so frei wie eben beschrieben ist dieses Zusammenfassen jedoch nicht. Um die zugewiesenen C-Netze einfach verwalten zu können muss ihre Anzahl 1, 2, 4, 8, 16, 32 oder 64 sein. Warum das so ist wird in

3.5.2 erläutert. Doch auch mit dieser Einschränkung ist die Verschwendung von Host-Adressen noch relativ gering. Mindestens die Hälfte des vergebenen Adressbereiches wird auch genutzt. Hat beispielsweise ein Unternehmen 8 C-Netze zugewiesen bekommen, so nutzt es von den  $8 \cdot 256 = 2.048$  möglichen Hosts mindestens 1024. Würde es weniger nutzen hätte es nur 4 Netze mit  $4 \cdot 256 = 1.024$  Hosts zugewiesen bekommen.

Das Zusammenfassen von C-Netzen löst das Problem der knappen B-Netze und beseitigt einen großen Teil der Host-Adressen-Verschwendung. Es hilft aber nicht beim Problem der wachsenden Routing-Tabellen - im Gegenteil. Die Routing-Tabellen wachsen dadurch noch stärker, da nun für ein Unternehmen nicht mehr nur ein Klasse-B-Netz-Eintrag vorhanden sein muss, sondern mehrere Klasse-C-Netz-Einträge.

### 3.5.2 Supernetting

Um nun auch die Größe der Routing-Tabellen wieder in den Griff zu bekommen, wandte man das so genannte „Supernetting“ an. Dabei geht es im wesentlichen um das geschickte Zusammenfassen von mehreren Netzadressen zu einer. Diese Prinzip greift an mehreren Stellen.

Zum einen wurden die zur Verfügung stehenden Klasse-C-Netze regional nach Kontinenten aufgeteilt, da es Sinn macht **regional zusammen** liegende Adressen auch **zusammen** zu **routen**. Das bedeutet, dass verschiedene Daten, deren Ziele regional nahe beieinander liegen auch auf den gleichen Pfaden dorthin gelangen können. Man fasst quasi mehrere beieinander liegende Ziele zu einem einzigen, größeren Ziel zusammen und benötigt für dieses Ziel nur noch einen Eintrag in den Routing-Tabellen. Europa bekam die C-Netz-Adressen 194.0.0.x bis 195.255.255.x zugewiesen. Wie man sieht, sind bei allen diesen Adressen die ersten sieben Bit gleich. Das erste Byte dieser Adressen lautet also immer 1100001\_ . Das bedeutet, dass alle Adressen, die dem Muster 1100001\_ .x.x.x entsprechen, zu einem europäischen Klasse-C-Netz gehören. Die dazugehörige Schreibweise lautet 194.0.0.0/7. Das hat zur Folge, dass jeder Router, der sich außerhalb Europas befindet, alle europäischen C-Netze (immerhin 131072 C-Netze mit jeweils 256 Hosts) unter einem einzigen Routing-Tabellen-Eintrag mit der Zieladresse 194.0.0.0/7 zusammenfassen kann. Diese Regionale Unterteilung des Adressraumes wird auch innerhalb der Kontinente fortgesetzt, dann natürlich nicht mit sieben, sondern mit mehr relevanten Bits.

Ein weiterer Punkt an dem sich Supernetting bezahlt macht, ist die in 3.5.1 beschriebene Zusammenfassung von C-Netzen um B-Netze zu sparen. Die Auswahl wieviele und welche C-Netze zusammengefasst werden ist nicht frei, sie erfolgt nach einem bestimmten Prinzip. Die zusammengefassten Adressen sind numerisch aufeinanderfolgend und müssen sich wie bei der regionalen Zusammenfassung bündeln lassen. Das führt dazu, dass nur 1, 2, 4, 8, 16, 32 oder 64 C-Netze zusammengefasst werden können. Wenn also fünf C-Netze benötigt würden um einen Bedarf zu decken, dann werden acht vergeben. Dieses Vorgehen kostet zwar ein paar Host-Adressen, vereinfacht aber den zusammengefassten Eintrag in einer Routing-Tabelle. Zur Erläuterung wird nun das Beispiel aus 3.4.3 fortgesetzt:

Das Unternehmen benötigt nach wie vor 2.500 Hosts. Nun bekommt es jedoch statt einem Klasse-B-Netz mehrere Klasse-C-Netze zugewiesen. 2.500 Hosts entsprechen 10 C-Netzen ( $10 \cdot 256 = 2.560$ ). Es bekommt also sechzehn C-Netze zugewiesen - beispielsweise von 194.123.16.x bis 194.123.31.x. In den Routing-Tabellen lassen sich diese sechzehn C-Netze zusammenfassen als 194.123.16.0/20. Man hat also wieder nur einen einzigen Eintrag für dieses Unternehmen in den Routing-Tabellen, hat aber ein Klasse B-Netz gespart. Der Adressraum, der bei Verwendung des B-Netzes verschwendet gewesen wäre entspricht 240 C-Netzen mit 61.440

Host-Adressen.

### 3.5.3 Routing-Taktiken mit CIDR und darüber hinaus

Obwohl BGP und CIDR viele Möglichkeiten bieten, das Routing zu beeinflussen, sind ihnen doch Grenzen gesteckt. Mit ihnen kann nicht explizit ein zu wählender Pfad vorgegeben werden (siehe 3.4.2).

Mit einigen Tricks kann man sich hier aber behelfen. Wenn der Pfad für ein Paket einen bestimmten Punkt durchlaufen soll, kann man wie folgt vorgehen. An das zu sendende IP-Paket, bestehend aus einem IP-Header und beispielsweise einem TCP-Paket (Abbildung 3.8), wird einfach noch ein IP-Header vorn angehängt (Abbildung 3.9).

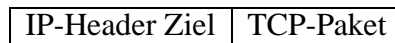


Abbildung 3.8: einfaches IP-Paket mit IP-Header und einem TCP-Paket im Datenteil

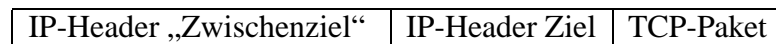


Abbildung 3.9: ein um einen weiteren IP-Header erweitertes IP-Paket

In diesem neuen Header wird das Zwischenziel als Routing-Ziel angegeben. Dadurch wird das Paket zuerst zu diesem Zwischenziel geleitet, dort „ausgepackt“, und als normales IP-Paket zum endgültigen Ziel weitergeleitet. Allerdings steigt durch diese Methode auch die Datenmenge (ein IP-Header ist 20 oder 32 Byte lang) und die in den Routern zu verrichtende Arbeit.

Eine weitere Möglichkeit gezielt Pfade für Pakete zu bestimmen ist das „source demand routing“ (SDR). Das entsprechende Protokoll heißt „source demand routing protocol“ (SDRP). Der sendende Router ermittelt dabei einen, aus seiner Sicht optimalen, Pfad zum Ziel, unabhängig davon, ob ihm andere Router diesen Pfad von Außen gemeldet haben. Den Datenpaketen wird dieser Pfad im Header explizit mitgegeben. Der Unterschied zu BGP ist der, dass hier der Router explizit einen Weg zum Ziel „zusammenbaut“ und nicht nur einen der von Außen gemeldeten Pfade auswählt. Die Router, die das Paket weiterleiten, tun dies nicht mittels ihrer eigenen Routing-Tabellen, sondern aufgrund der Pfad-Anweisungen im Header des Datenpaketes. Der sendende Router hat somit direkten Einfluss auf alle Routing-Schritte (nicht nur auf den ersten wie bei BGP).

Ein weiterer Ansatz, der parallel zu BGP entwickelt wurde heißt „inter domain policy routing“ (IDPR). Es versucht einen Linkstate-Ansatz (siehe 3.4.1 auf Ebene der Autonomen Systeme) anzuwenden. Wie bereits beschrieben ist dies ein sehr komplexes Unterfangen. Da die Anzahl der zu verwaltenden Knoten bei IDPR deutlich größer ist, als zum Beispiel bei OSPF. OSPF ist für maximal 200 Knoten ausgelegt, IDPR muss aber AS verwalten, von denen es zur Zeit über 3000 gibt. Um dem Problem der eigentlich zu großen Anzahl von AS zu begegnen, versucht IDPR mehrere der Grenz-Router der AS zu „virtuellen Gateways“ zusammenzufassen. Dadurch werden die zu verwaltenden Datenbanken etwas kleiner.

## **3.6 Ausblick**

Mit CIDR sind die wichtigsten Probleme des Routing im Internet kurzfristig gelöst. Langfristig kann auch CIDR, implementiert in BGP-4, die Adressknappheit nicht verhindern, es wird in wenigen Jahren nicht mehr ausreichen um die Adressknappheit zu umgehen. Das bereits fertig entwickelte IPv6 sollte zügig eingeführt werden um auch in Zukunft das Wachstum des Internet zu ermöglichen. IPv6 bietet mit 128-Bit-Adressen und vielen Verbesserungen im Bereich Datensicherheit ausreichen Spielraum hierfür. Als etwas überzogenes Beispiel sei angeführt, dass IPv6 genügend Adressen bereit hält um jedem Quadratmikrometer Erdoberfläche über 660 Milliarden Adressen zuzuweisen.



# Literaturverzeichnis

- [1] Huitema, Christian: „Routing im Internet“, deutsche Übersetzung der ersten Auflage von [2], Prentice Hall, 1996
- [2] Huitema, Christian: „Routing in the Internet“, 2. Auflage, Prentice Hall, 2000
- [3] Tanenbaum, Andrew S.: „Computernetzwerke“, 3. rev. Auflage, Pearson Studium, 2001
- [4] Peterson, Larry: „Computernetze“, dpunkt.verlag, 1999
- [5] Rautenberg, Mathias: „IP Weitverkehrsnetze“, Skript zur gleichnamigen Vorlesung an der UniBwM, Siemens, 2000
- [6] RFC 827, EGP, <http://www.ietf.org/rfc/827.txt>, 1982
- [7] RFC 911, EGP under Berkeley UNIX 4.2, <http://www.ietf.org/rfc/911.txt>, 1984
- [8] RFC 1265, BGP Protocol Analysis, <http://www.ietf.org/rfc/1265.txt>, 1991
- [9] RFC 1267, BGP-3, <http://www.ietf.org/rfc/1267.txt>, 1991
- [10] RFC 1519, CIDR, <http://www.ietf.org/rfc/1519.txt>, 1993
- [11] RFC 1654, BGP-4, <http://www.ietf.org/rfc/1654.txt>, 1994
- [12] Homepage der IETF, <http://www.ietf.org/>
- [13] Homepage der IANA, <http://www.iana.org/>
- [14] Homepage der RIPE, <http://www.ripe.net/>
- [15] Homepage der DENIC, <http://www.denic.de/>



# 4 Routing mit Multicasting

*Daniel Scheppelmann*

## Inhaltsverzeichnis

---

<b>4.1</b>	<b>Einleitung</b> . . . . .	<b>76</b>
<b>4.2</b>	<b>Gruppenkommunikation</b> . . . . .	<b>77</b>
4.2.1	Formen der Kommunikation . . . . .	77
4.2.2	Gruppeneigenschaften und deren Auswirkungen . . . . .	79
<b>4.3</b>	<b>Multicast-Routing</b> . . . . .	<b>82</b>
4.3.1	Grundlagen . . . . .	82
4.3.2	Multicasting in der Vermittlungsschicht . . . . .	83
4.3.3	Routingstrategien in der Vermittlungsschicht . . . . .	84
4.3.4	Multicasting in der Transportschicht . . . . .	90
<b>4.4</b>	<b>Das M-Bone</b> . . . . .	<b>92</b>
4.4.1	Grundlagen . . . . .	92
4.4.2	Adressierung . . . . .	92
<b>4.5</b>	<b>Zusammenfassung und Ausblick</b> . . . . .	<b>94</b>
4.5.1	Zusammenfassung . . . . .	94
4.5.2	Ausblick . . . . .	95
<b>4.6</b>	<b>Abkürzungen</b> . . . . .	<b>95</b>
	<b>Literaturverzeichnis</b> . . . . .	<b>95</b>

---

## Abstract

As contribution to the Seminar „Routing Strategies in the Internet“ this work lights up the topic „Multicasting“. After a short discussion of special aspects and problems connected to common group communications, it introduces different strategies for the required routing-protocols. So called source-based and core-based systems build the two main strategies to resolve the net load problem coming along with multicasting. After leaving the switching layer and a short discussion of multicasting within the transport layer, this work finally treats the Mbone. Mbone (Multicasting Backbone) as a test range for world-wide multicasting is still the one and only possibility for inter-domain-multicasting and is yet well used even for commercial applications.

## Kurzbeschreibung

Als Beitrag zum Seminar „Routingstrategien im Internet“ beschäftigt sich diese Arbeit mit dem Thema „Multicasting“. Nach einem kurzen Überblick über spezielle Aspekte und Probleme allgemeiner Gruppenkommunikationen führt sie unterschiedliche Strategien für das Design der benötigten Routingprotokolle ein. Die beiden wichtigsten Ansätze für die Lösung des beim Multicasting unvermeidlichen Netzlastproblems bilden zum einen die so genannten „quellbasierten“ Konzepte und zum anderen das Konzept der „Bäume mit Rendezvous-Stellen“. Nach der Betrachtung dieser in der Vermittlungsschicht angewendeten Verfahren und einem kurzen Überblick über das Multicasting in der Transportschicht behandelt die Arbeit abschließend das Mbone. Das Mbone (Multicasting Backbone), ursprünglich gedacht als Forschungsgrundlage für weltweites Multicasting, ist bis heute die einzige und daher auch kommerziell genutzte Möglichkeit, Multicasting über die Grenzen einzelner Internetdomänen hinweg zu betreiben.

## 4.1 Einleitung

Das Internet hat in den letzten Jahren explosionsartig an Bedeutung für das menschliche Zusammenleben gewonnen. Egal ob E-Mail, Newsticker oder das einfache “Surfen“ von Homepage zu Homepage, keine dieser neuen technischen Errungenschaften ist ohne beträchtlichen Verlust von Komfort und Lebensqualität für viele von uns wegzudenken. Viel mehr ist davon auszugehen, dass in den nächsten Jahren weitere technische Möglichkeiten in unser Leben Einzug halten werden und so manches heute als unlösbar abgetanes Alltagsproblem schon bald der Vergangenheit angehören wird.

Was aber bietet sich als zu lösendes Problem für das Internet der Zukunft an? Liegt irgendwo Potential offen, dessen Ausnutzung schon sehr schnell zu neuen Erfolgen des Internets führen könnte?

Ein einfacher aber treffsicherer Ansatz um so geartete Fragen zu beantworten ist folgender: Man analysiert, wo die Stärken der heutigen Internettechnologie liegen und wo sich uns andererseits Schwächen offenbaren. Dabei wird man relativ schnell feststellen, dass der Datentransfer im Punkt-zu-Punkt-Verkehr heutzutage exzellent funktioniert und die vielen darauf beruhenden Dienste den Hauptteil der heutigen Internetkommunikation ausmachen. Weniger weit entwickelt dagegen sind Techniken und Dienste, die sich mit **Gruppenkommunikation** beschäftigen, die also in erster Linie den Datentransfer von einem Sender an mehrere Empfänger gewährleisten.

Hier setzt nun diese Seminararbeit an. Nach einem kurzen Überblick über allgemeine Grundsätze der **Gruppenkommunikation** in Kapitel 2 soll sie in Kapitel 3 den aktuellen Stand der **Multicasting**-Technik zeigen. Also jener Technik, die ohne Rückgriff auf den Aufbau vieler Einzelverbindungen Daten eines einzelnen Senders an mehrere Empfänger im Internet überträgt.

Beim **Multicasting** sind, wie soll es auch anders sein, mehrere Schichten des Internetschichtenmodells beteiligt. Wegen des Seminarthemas – „Routingstrategien im Internet“– liegt der Schwerpunkt dieser Arbeit beim **Multicasting in der Vermittlungsschicht**. Hier nämlich findet das eigentliche Routing statt, also die zielorientierte Weiterleitung von IP-Paketen von Router zu Router, während beispielsweise die **Transportschicht** unter anderem für das Nachfordern nicht erhaltener Pakete verantwortlich ist. Völlig ausgeblendet werden die anderen Schichten nicht; der **Transportschicht** ist ein eigener, wenn auch etwas kürzer Abschnitt gewidmet.

Ein kurzer Überblick wird abschließend im vierten Kapitel über das so genannte **M-Bone** geboten. Dieses **M-Bone** – ausgeschrieben **Multicasting-Backbone** – bildet die derzeitige technische Grundlage für weltweites **Multicasting**. **M-Bone-Anwendungen** wie Internet-Videokonferenzen oder die immer häufiger angesprochene „Virtuelle Hochschule“ sind nur zwei Beispiele für erfolgreiches **Multicasting** in der heutigen Zeit. Sie sind allerdings, zumindestens was ihre Einrichtung angeht, noch immer Experten vorbehalten.

„**Multicasting für Jedermann**“, also einfache Anwendungen für den Alltag und alle damit verbundenen Möglichkeiten, sind im Internet nach wie vor wenig vertreten.

## 4.2 Gruppenkommunikation

Wie aus dem Einleitungsteil bereits hervorgeht, ist das Multicasting eine von mehreren Kommunikationsformen. In diesem Kapitel wird das Multicasting zunächst von anderen Kommunikationsformen abgegrenzt.

Als weiteres wird zu typischen Gruppeneigenschaften wie Gruppengröße und Gruppentopologie Stellung genommen - Eigenschaften die gerade im Hinblick auf das Multicasting und dessen technische Umsetzung von Bedeutung sind. Aus diesen Eigenschaften und auch aus anderen allgemeinen Betrachtungen lassen sich einige Grundprobleme für das Multicasting ableiten, deren Darstellungen ebenfalls im zweiten Abschnitt eingearbeitet sind.

### 4.2.1 Formen der Kommunikation

Mit „Formen der Kommunikation“ ist in erster Linie gemeint, wie viele Sender jeweils mit wie vielen Empfängern sprechen. Jede Kommunikationsform lässt sich dabei mit unterschiedlich großem Aufwand technisch realisieren. Teilweise ist es sogar möglich, komplexere Kommunikationsformen durch einfachere zu emulieren. Mehr dazu jedoch im folgenden Abschnitt:

- **Unicast**



Abbildung 4.1: Unicast

Beim Unicasting (Abbildung 2.1) spricht genau ein Sender mit genau einem Empfänger. [1] Dies ist die sicherlich einfachste Kommunikationsform. Im Internet bauen fast alle Dienst auf dem Unicast-Prinzip auf, demzufolge ist dessen technische Umsetzung sehr weit fortgeschritten. Unicasting hat aber auch Bedeutung für die Umsetzung anderer Kommunikationsformen.

Im folgenden Kapitel wird deutlich werden, dass auch auch im Bereich des Multicasting auf Unicast-Techniken zurückgegriffen wird.

- **Multicast**

Beim Multicast (Abbildung 2.2) spricht, wie eingangs beschrieben, ein einzelner Sender zu mehreren Empfängern (vgl. [1]).

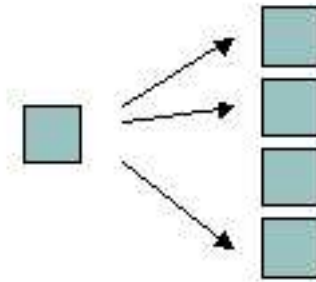


Abbildung 4.2: Multicast

Die technische Umsetzung von Multicasting ist gerade bei einem paketorientierten Medium wie dem Internet komplizierter als die des Unicasting. Multicasting kann über ein mehrfaches Unicasting emuliert werden (Ein Unicast pro Empfänger), jedoch liegt es auf der Hand, dass mit diesem Verfahren der Sender und auch das verbindende Medium beliebig stark belastet werden. Man denke einfach mal an die Übertragung eines wichtigen Fußballspiels im Internet. Ein millionenfaches breitbandiges Unicasting stellt jeden Server und jede Datenanbindung vor heutzutage unlösbare Probleme.

Viel effizienter ist das Kopieren von Datenpaketen bei deren Transport. Und zwar genau dort, wo eine Verzweigung vom Sender zu den Empfängern im Netz vorliegt.

- **Broadcast**

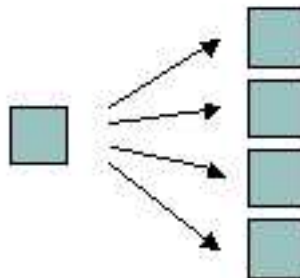


Abbildung 4.3: Broadcast

Broadcasting (Abbildung 2.3) ist ebenfalls eine Kommunikation von einem Sender zu mehreren Empfängern (vgl. [1]). Die Abgrenzung zum Multicasting liegt darin, dass Broadcasting ungezielt verläuft. So ist die Ausstrahlung von Fernseh- oder Hörfunkprogrammen dem Broadcasting zuzuordnen, während ein Internet-Videostream mit ausgewählt versorgten Empfängern zum Multicasting gehört.

Ähnlich wie das Unicasting hat auch das Internet-Broadcasting eine nicht zu vernachlässigende Bedeutung im Bereich der Multicast-Technologie. Auch hierzu mehr im nächsten Kapitel.

- **Netcast**

Netcasting ist ein weiterer Begriff für den Versand von Daten eines einzelnen Senders an mehrere Empfänger.[1] Er wurde jedoch als Übersetzung des ursprüngliche Broadcasting-

Begriffes (also Broadcasting im Sinne des Fernseh- oder Hörfunks) in die Welt des Internets geschaffen. Seine technische Umsetzung wird im Endeffekt auf Internet-Multicasting oder Internet-Broadcasting hinauslaufen.

- **Anycast**

Als letzte Kommunikationsform mit einem Sender und mehreren Empfängern sei das Anycasting genannt.[1] Anycasting bezeichnet den Umstand, dass ein einzelner Sender mehrere Empfänger anspricht, aber nur von einem oder sehr wenigen eine Antwort erwartet. Ein Beispiel für Anycasting wäre der Zugriff auf eine verteilte Datenbank: Nur derjenige Datenbankserver soll antworten, der meine Anfrage bearbeiten kann.

Anycasting ist eine Sonderform unter den Kommunikationsformen und beruht bezüglich der Anfragen entweder auf Broadcasting oder auf Multicasting. Antworten werden in den meisten Fällen dem Unicasting überlassen bleiben.

- **Concast**

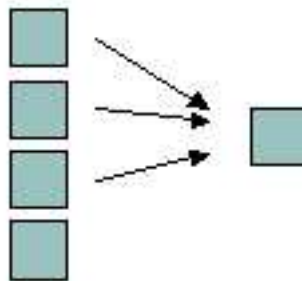


Abbildung 4.4: Concast

Concast (Abbildung 2.4) ist die Kommunikation mehrerer Sender mit einem einzelnen Empfänger (vgl. [1]).

Für das Concasting ist keine separate technische Umsetzung für das Internet vonnöten, da weder die Last im Netz noch die Belastung des Empfängers geringer würde als bei einer Emulation über Unicasts. Grund hierfür ist, dass jedes zu empfangende Datenpaket einzigartig ist und keinerlei Möglichkeit besteht, durch Einsatz von Kopien Teilnetze zu entlasten.

- **Multipeer**

Beim Multipeering (Abbildung 2.5) senden mehrere Sender an mehrere Empfänger.[1] Als Beispiel hierfür sei eine Diskussionsrunde genannt. Hier kann jeder Empfänger bei Bedarf auch in die Rolle eines Senders wechseln. Außerdem kommt es häufig vor, dass mehrere Sender gleichzeitig sprechen und dabei die komplette Zuhörerschaft adressieren.

Multipeering benötigt ebenfalls keine separate technische Umsetzung im Internet. Es lässt sich ideal durch mehrere Multicasts emulieren.

## 4.2.2 Gruppeneigenschaften und deren Auswirkungen

Dieser Abschnitt soll, wie Eingangs erläutert, verschiedene Eigenschaften von Gruppen darstellen und deren Auswirkungen auf die technische Realisierung des Multicastings verdeutlichen.

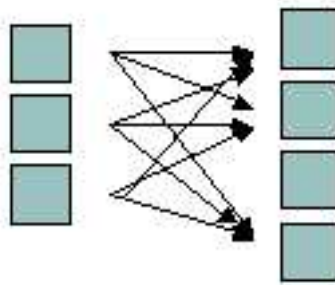


Abbildung 4.5: Multipeer

Unter dem Begriff „Gruppe“ sind dabei mehrere im Internet verteilte Rechner zu verstehen, die gemeinsam Multicasting-Dienste nutzen wollen; also zum Beispiel den Sender und die Zuschauer einer Internet-gestützten TV-Übertragung oder die Teilnehmer einer Videokonferenz.

Wesentliche Gruppeneigenschaften wären:

- **Größe der Gruppe**

Die Größe einer Gruppe hat großen Einfluss auf das Multicasting.[2] So lassen sich kleine Gruppen problemlos über mehrfaches Unicasting versorgen, während sehr große Gruppen in ihren Anforderungen schnell an das technische Limit des Internets kommen. Erinnerung sei hier an mein zuvor genanntes Beispiel des wichtigen Fußballspiels, dass nur behelfsmäßig und sehr eingeschränkt durch einen breitbandigen Massen-Unicast übertragen werden kann.

Auch der Verwaltungsaufwand eines echten Multicastings ist in großen Gruppen naturgemäß höher als in kleinen Gruppen.

- **Dynamik der Gruppe**

Ein weiterer wichtiger Punkt für das Multicasting ist die Dynamik der Gruppe. So macht es einen nicht unwesentlichen Unterschied aus, ob eine Multicast-Übertragung für ein konstantes Publikum durchgeführt wird - so zum Beispiel für eine Videokonferenz mit wenigen wichtigen Teilnehmern - oder ob im Sinne des Netcasting ein Multimediaangebot verbreitet wird, in das sich ständig neue Zuschauer einloggen und bisherige Zuschauer ausloggen.[1]

- **Angewiesenheit auf Zuverlässigkeit**

Eine wesentliche Frage bei allen Datentransporten im Internet ist, ob verlorene Pakete nachgeordnet werden müssen und ob die Reihenfolge der Pakete in jedem Fall wiederhergestellt werden muss, so sie zum Beispiel durch unterschiedliche Wegewahl beim Routing in ihrer Ankunft durcheinander gewürfelt wurden.[3]

Während man diese Frage beim Unicasting ohne viel nachzudenken durch die Wahl des passenden Transportprotokolls (im wesentlichen TCP oder UDP) löst, muss man im Hinblick auf das Multicasting ein neues Problem betrachten: **Quittungsimplosionen**. Diese entstehen immer dann, wenn ein Multicasting-Transportprotokoll in großen Gruppen Quittungen für erfolgreich versendete Pakete implementiert (ACK). Es entsteht unter



Umständen aber auch dann Probleme, wenn Nachbestellungen nicht erhaltener Pakete versendet werden (NACK).[4]

Auf die Sendung eines einzelnen Paketes erfolgt bei sehr großen Gruppen unter Verwendung von Quittungen eine hohe Belastungen des Senders und seiner unmittelbaren Netzanbindung.

- **Bekanntheit untereinander**

In manchen Gruppen ist es sehr wichtig, dass alle Mitglieder untereinander bekannt sind. So zum Beispiel bei einer überschaubaren Videokonferenz oder einem über das Internet realisierten Whiteboard. Ausgenommen von dieser Bekanntheit sind natürlich etwaige passive Zuschauer, die aber nicht zwingend notwendig zugelassen sein müssen.[1]

In anderen Fällen ist eine Bekanntheit untereinander unnötig. So ist es wenig nützlich, dass unterschiedliche Nutzer einer Internet-Übertragung - zum Beispiel der Übertragung eines Fußballspiels - sich gegenseitig kennen. Eine solche gegenseitige Bekanntheit kann sogar ausdrücklich unerwünscht sein. Man bedenke nur mal die Interessen des Datenschutzes.[1]

- **Topologie der Gruppe**

Ganz unabhängig von der Größe einer Gruppe aber auch von den anderen bisher genannten Punkten ist die Topologie einer Gruppe für die technische Umsetzung eines Multicastings von enormer Wichtigkeit.[2]

Die Frage, wie die einzelnen Gruppenmitglieder mit dem Internet verbunden sind - vom Modem bis hin zur Breitbandstandleitung - oder wie eine solche Gruppe geographisch verteilt ist, spielt für die ausschöpfbaren technischen Möglichkeiten eine wichtige Rolle. So ist es sicher unmöglich, qualitativ ansprechende Audio- und Videoelemente in eine Internet-Konferenz zu integrieren, deren Teilnehmer an einem 56k-Modem hängen.

Ferner ist es deutlich einfacher, Multicastings in einzelnen autonomen Systemen, also zum Beispiel innerhalb einer Firmendomäne durchzuführen, als vergleichbare Vorhaben weltweit umzusetzen. Dies werden die folgenden Kapitel aber noch verdeutlichen.

- **Sicherheitsbedürfnis der Gruppe**

In Zeiten eines steigenden Sicherheitsbedürfnisses, sei es durch zunehmende Internet-Kriminalität oder durch die immer weiter schreitende Integration des Internets in immer neue, nun auch sicherheitsrelevante Lebensbereiche, bleibt es zu klären, ob Multicasting-Dienste sorgenfrei einsetzbar sind. Wie lassen sich zum Beispiel Schlüssel in einer Gruppe austauschen? Wie kann man Datenverkehr regional eingengen?

- **Senderechte innerhalb der Gruppe**

Eine eher technische Frage bei der Gruppenkommunikation ist die Frage nach der Vergabe von Senderechten. So kann es gewünscht sein, dass zum Beispiel auf einer Videokonferenz nur einem ausgewähltem Personenkreis das Rederecht vorbehalten bleibt.[1]

Ferner kann sich jeder vorstellen was passiert, wenn bei einer Netcasting-Übertragung andere als der dafür vorgesehene Rechner Sendeversuche unternehmen.

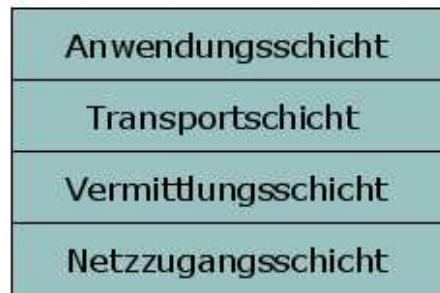


Abbildung 4.6: Die Schichten des Internetschichtenmodells [1]

## 4.3 Multicast-Routing

### 4.3.1 Grundlagen

Multicasting spielt sich in mehreren Schichten des Internetschichtenmodells (Abbildung 3.1) ab.[1] Dabei übernimmt jede ins Multicasting eingebundene Schicht gewisse Kernaufgaben.

- **Netzzugangsschicht**

Von unten beginnend haben wir zunächst die Netzzugangsschicht. Dies ist jene Schicht, die die Anbindung eines Rechners in das Internet ausmacht. Ethernet, Token Ring oder ATM (Asynchronous Transfer Mode) sind einige der hier anzusiedeln Technologien.[3]

Abgesehen von der Gruppenadressierung auf tiefster Ebene, die notwendiger Weise aus den Gruppenadressen der höheren Vermittlungsschicht abgeleitet werden muss, ist die Betrachtung der Netzzugangsschicht für das Multicasting uninteressant.

Ein Beispiel für die Adressierung auf tiefster Ebene sind die so genannten MAC-Adressen (Media-Access-Control-Adressen). Hierbei handelt es sich um weltweit eindeutige, hardwaregebundene Adressen die u.a. auf für jede Ethernet-Karte vorhanden sind.[4]

- **Vermittlungsschicht**

Die Vermittlungsschicht ist im Internet für die Durchleitung der einzelnen Datenpakete durch die verschiedenen Router und Teilnetze verantwortlich. Für diese Vermittlungsschicht hat sich als Protokoll über die Jahre das so genannte **IP-Protokoll** durchgesetzt.[3] (IP steht für **I**nternet **P**rotocol). Das IP-Protokoll ist, wie bereits angedeutet, ein paketorientiertes Protokoll. Für die Weiterleitung der Pakete werden eigens vergebene IP-Adresse verwendet. Jeder Router entscheidet paketweise, an welchen seiner Nachbarn er die einlaufenden Daten weiterleitet. Nach außen wirkt dieses Verfahren so, als ob sich die Datenpakete wie von selbst ihren Weg durch das Internet bahnen. Sie sind nicht, wie es beispielsweise bei einem Telefongespräch der Fall ist, auf eine fest eingerichtete Verbindung angewiesen. Das IP-Protokoll realisiert folglich eine so genannte „Ende-zu-Ende“-Kommunikation [5].

Um nun Multicasting über IP zu ermöglichen, müssen - neben der Einführung von Multicasting-Adressen im IP-Protokoll - auf den Routern des Internets entsprechende Routingprotokolle laufen. Diese müssen mit Multicastadressen umgehen können und bei der Weiterleitung von Datenpaketen die gewünschten Verteilbäume aufbauen.[2]

- **Transportschicht und Anwendungsschicht**

Während die Vermittlungsschicht lediglich den Fluß der Datenpakete durch das Internet ermöglicht, ist die Transportschicht gemeinsam mit der Anwendungsschicht für alle weiteren Punkte verantwortlich, die sich aus den Grundlagen zur Gruppenkommunikation ergeben. Fragen des Umgangs mit Quittungen, Verschlüsselung oder eine lastabhängigen Fluß- bzw. Staukontrolle fallen in diese Schichten.[1] Im Abschnitt 3.3 werden die zwei am weitesten verbreiteten Transportprotokolle (TCP und UDP) vorgestellt.

Im Hinblick auf das Thema des Seminars, „Routingstrategien im Internet“ ist der Schwerpunkt der Arbeit in der Vermittlungsschicht angesiedelt. Trotzdem befindet sich, wie eben angekündigt, weiter hinten ein kurzer Abschnitt über Multicasting-Anteile in der Transportschicht.

### 4.3.2 Multicasting in der Vermittlungsschicht

#### IP-Adressierung von Multicast-Gruppen

Um Multicasting betreiben zu können, muss man zunächst die Möglichkeit bekommen, Gruppen zu adressieren. Ohne diese Möglichkeit ist man auf die bereits angesprochene, für viele Fälle ungeeignete Multicasting-Emulation über Unicasting angewiesen. Alternativ könnte man nur, sollte sich ein Multicasting auf ein einzelnes Netzwerk beschränken, auf einfaches Netzwerkbroadcasting zurückgreifen (alle Bits nach der Netzwerkadresse gleich eins; in Klasse-C-Netzen[4] beispielsweise hat das letzte Byte den Wert 255).

Da dies sicherlich nicht zufrieden stellend ist, hat man neben den Adressklassen A, B und C die Adressklasse D für das Multicasting eingeführt.[6] Ein Wert größer oder gleich 224 im ersten Byte der IP-Adresse weist auf eine solche Multicast-Adresse hin.

220.193.211.255	Reines Netzwerkbroadcasting (Klasse C)
225.199.100.101	Netzwerkübergreifende Multicast-Adresse (Klasse D)

#### Beitritt zu Gruppen - IGMP

Aufgabe des **IGMP- Internet Group Management Protocol** ist es, in Endsystemen, die an das Internet angeschlossen sind, also zum Beispiel einem LAN, Gruppenmitglieder einer Multicastgruppe festzustellen. Verfügt ein Router über die Informationen, dass hinter ihm ein Mitglied einer Multicastgruppe sitzt, kann er je nach Routingprotokoll am Multicasting teilnehmen, und sein Endsystem mit den Daten der jeweiligen Gruppe versorgen.[1]

IGMP ist integraler Bestandteil des IP-Protokolls. Es funktioniert innerhalb eines Endsystems durch Versand von **IGMP-Paketen**, wobei es sich um spezielle IP-Pakete mit der Lebenszeit (TTL) eins handelt.[3] Diese minimale TTL führt dazu, dass IGMP-Pakete ihr Endsystem nicht über den bzw. die angeschlossenen Router verlassen können.

IGMP-fähige Router senden nach einer festgelegten Zeitspanne - empfohlen werden laut RFC (Request for Comments - Spezifikationsdokument) 125 Sekunden - allgemeine Anfragen über Gruppenmitgliedschaften in ihr(e) Endsystem(e).[7] Die angeschlossene Rechner ermitteln dann, so sie Mitglied einer Gruppe sind, eine zufällige Antwortzeit. Nach Verstreichen dieser Antwortzeit, die im übrigen unterhalb der eben genannten festgelegten Zeitspanne für allgemeine Anfragen liegen muss, melden sie beim Router ihre Mitgliedschaften an. Sollte bis dahin

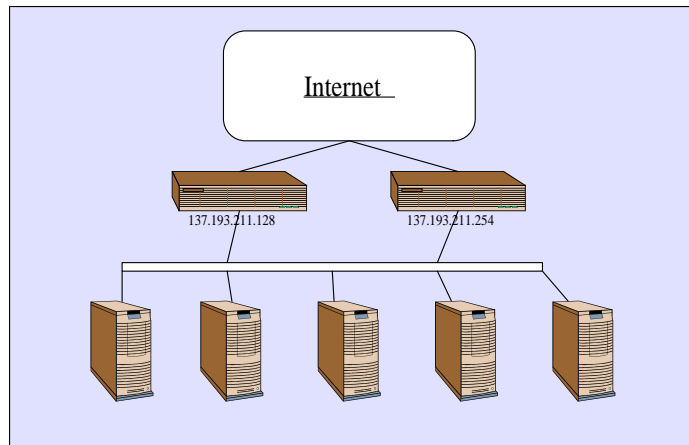


Abbildung 4.7: LAN mit zwei Routern ins Internet

schon ein anderer Rechner des betreffenden Endsystems eine Mitgliedschaft zu einer identischen Gruppe gemeldet haben, wird eine zusätzliche Anmeldung unterlassen. Das betroffene Endsystem wird dann bereits mit den Daten der Gruppe versorgt.

Unterbleibt nach einer neuerlichen allgemeinen Anfrage die Meldung für eine derzeit unterstützte Gruppe, so wird die Versorgung des Endsystems mit deren Daten eingestellt.

Neben diesem Mechanismus besteht für die angeschlossenen Rechner außerdem die Möglichkeit, sich ohne vorherige allgemeine Anfragen von sich aus beim Router für eine Gruppenmitgliedschaft anzumelden oder abzumelden.

Sind, wie es in Abbildung 3.1 ja der Fall ist, mehrere IGMP-Router für ein Endsystem vorhanden, so stellen alle diejenigen Router ihre IGMP-Arbeit ein, die eine allgemeine Mitgliedschaftsanfrage von einem Router mit niedriger IP erhalten. In der Abbildung 3.1 ist der Router mit der 128 folglich fürs Multicasting verantwortlich und wird deshalb **Querier** genannt.

Die Aufgabe der Vermittlungsschicht ist bekanntlich die Versorgung von interessierten Rechnern mit Datenpaketen. Die Genauigkeit, mit der einzelne Rechner versorgt oder nicht versorgt werden, beschränkt sich jedoch auf ganze Endsysteme. Entweder ein Endsystem wird mit den Multicast-Daten einer Gruppe versorgt oder es wird nicht versorgt. Interessierte Rechner können also entweder die notwendigen Pakete aus ihrem Endsystem entnehmen (Netzwerk-Port belauschen), oder sie müssen die entsprechende Versorgung des Endsystems, wie oben geschildert, zunächst beantragen.

Die exakte Auflösung von Gruppenmitgliedschaften ist wegen der eben beschriebenen Unschärfe innerhalb der Vermittlungsschicht also nicht möglich. Sie und weitere Aspekte wie Sicherheit und Zuverlässigkeit bleiben der Transportschicht oder der Anwendungsschicht überlassen.[2]

### 4.3.3 Routingstrategien in der Vermittlungsschicht

Nachdem zuletzt dargestellt wurde, wie Endsysteme ihren Routern mitteilen, dass sie Mitglieder einer Multicastgruppe enthalten, wird die Arbeit sich im folgenden den unterschiedlichen Routingstrategien widmen. Denjenigen Strategien also, die für das Weiterleiten von Multicastingdaten zwischen den verschiedenen Routern im Internet verantwortlich sind.

Vorweg sei gesagt, dass Multicasting bisher nur bis zur Ebene der autonomen Systeme (Teilnetze, die unter einheitlicher Administration stehen[8]) abschließend implementiert ist.[1] Alles

was darüber hinaus geht, also insbesondere weltweites Multicasting, ist nach wie vor im Versuchsstadium oder beruht allenfalls auf Quasi-Standards. Im letzten Kapitel jedoch mehr dazu.

Wenn im folgenden also den Begriff „Netz“ verwendet wird, dann ist weniger das komplette Internet gemeint (es wird auch „Netz der Netze“ genannt), als vielmehr ein räumlich begrenztes Netz. Vereinfacht könnte man sagen, dass solche Netze gemeint sind, die von Internen Routing-Protokollen[9] verwaltet werden.

Über die Jahre haben sich drei grundlegende Konzepte für das Routing im Multicast-Betrieb durchgesetzt:

- Fluten
- Quellbasiertes Multicasting
- Bäume mit Rendezvous-Stellen

In den folgenden Abschnitten werden diese drei Konzepte vorgestellt und anschließend auf zwei real existierende Protokolle eingegangen.

## Fluten

Das **Fluten (engl. Flooding)** ist der einfachste Ansatz für das Multicast-Routing.[2] Die Datenpakete werden vom Sender an alle angeschlossenen Router gesendet. Diese machen anschließend nichts anderes, als das Datenpaket wiederum an alle angeschlossenen Router weiterzuleiten.

Wie man sicher leicht einsieht, führt das Fluten sehr schnell zu einer hohen Netzlast. Wie in einer Kettenreaktion vermehrt sich ein einzelnes Datenpaket und belastet einen und den selben Router möglicherweise mehrfach. (Ich gehe davon aus, dass IP-basierte Netze so gut wie immer Schleifen enthalten). Letztlich kann nur die TTL des Pakets ein Zusammenbrechen des Netzes verhindern. Diese TTL sollte, und das liegt auf der Hand, nicht größer sein als der Weg vom Sender zum am weitesten entfernten Router des Netzes.

Fluten ist von seinem Wesen her dem Broadcasting zuzuordnen. Als Strategie für den Transport von Nutzlasten eignet es sich aus oben genannten Gründen nicht oder nur sehr eingeschränkt. Zwar gibt es Ansätze, wie man das Fluten effizienter gestalten könnte, zum Beispiel durch Ausschluss der Neuversendung eines Pakets mittels Einsatz einer History-Datenbank auf jedem Router. Aber diese Verbesserungen sind mit meist deutlichen Leistungseinbußen der Router verbunden, da beispielsweise die Datenbankabgleiche aus unserem Beispiel nicht annähernd so schnell implementiert werden können, wie das herkömmliche Weiterleiten von Paketen mittels einer Routingtabelle. Moderne Router-Systeme nutzen hierfür teilweise Hardwarefunktionen ihrer Netzschnittstellen, was um Größenordnungen schneller ist.

Trotz aller Einschränkungen ist das Fluten keineswegs zu vernachlässigen. Für die Administration eines Netzes, ich denke da beispielsweise an den Protokolltransfer von Routingprotokollen, kommt es regelmäßig zum Einsatz. Immer dann, wenn es darum geht, wenige und meist auch kleine Datenpakete an alle Router eines Netzes zuzusenden, ist das Fluten erste Wahl.

## Quellbasiertes Multicasting

Beim eben vorgestellten Fluten haben die beteiligten Router keinerlei Informationen über die adressierte Gruppe benötigt. Zumindestens nicht für die Entscheidung über das Weiterleiten der

Pakete an andere Router, sondern nur für die Erkennung, dass es sich überhaupt um Multicasting-Pakete handelt. Beim nun vorzustellenden quellbasierten Multicasting verhält es sich genau so; daher stammt auch der Begriff „Quellbasiert“.[2]

Quellbasiertes Multicasting ist vereinfacht gesagt eine Fortentwicklung des Flutens. Der folgende Abschnitt zeigt vier quellbasierte Strategien, die man als sukzessive Weiterentwicklungen des Flutens betrachten kann.

**Quellbasiertes Multicasting - RPF** Beim RPF, oder ausgeschrieben **Reverse Path Forwarding** werden wie beim Fluten einlaufende Datenpakete an alle angeschlossenen Schnittstellen, sprich an alle benachbarten Router weitergeleitet. Ausgenommen davon sind allerdings solche Pakete, die den weitersendenden Router nicht auf dem kürzesten Weg erreicht haben [1] .

IP-Pakete enthalten neben der Empfängeradresse auch immer die Adresse des Absenders [6]. Sollte in den Routingtabellen des weitersendenden Routers eine andere Schnittstelle für die Sendung von Daten an eben diese Absenderadresse eingetragen sein als die, auf der das Multicasting-Paket eingetroffen ist, dann wird es verworfen. Wie im Abschnitt über das Fluten bereits angesprochen, sind Adressabgleiche mit Routingtabellen sehr schnell. Die sich aus dem Reverse Path Forwarding ergebenden Verbesserungen sind also denkbaren Erweiterungen des Flutens (hier sind die eben angesprochenen Datenbankeinträge gemeint) vorzuziehen.

Das Reverse Path Forwarding reduziert den von einem Multicast hervorgerufenen Datenverkehr ganz erheblich. Die Ausbreitung der Datenpakete reduziert sich von einer kettenreaktionsartigen Explosion zu einer strahlenförmigen Verteilung radial zum Sender, bei der Schleifenbildungen grundsätzlich ausgeschlossen sind.

Auch das Reverse Path Forwarding ist kein Multicasting. Die Tatsache, dass nach wie vor alle Router eines Netzes von den Datenpaketen erreicht werden, zeigt, dass es sich beim RPF wie zuvor beim Fluten um ein Broadcasting-Verfahren handelt.

**Quellbasiertes Multicasting - RPB** Eine Weiterentwicklung des Reverse Path Forwarding ist das **Reverse Path Broadcasting**. Es unterscheidet sich von seinem Vorgänger darin, dass Pakete zusätzlich zu den eben erläuterten Maßnahmen nicht über Schnittstellen weitergeleitet werden, über die sie auf einem längeren als dem kürzest möglichen Weg den Empfänger erreichen würden.[1]

Für das Reverse Path Broadcasting benötigen die Router folglich die Sende-Routingtabellen ihrer Nachbarrouter. Dies sollte aber weder bei Distant-Vector-Protokollen noch bei Link-State-Protokollen ein Problem darstellen. Bei ersteren werden nämlich genau diese Tabellen für den Aufbau der Routingtabellen in regelmäßigen Abständen verteilt und bei letzteren kennen alle Router die komplette Netztopologie und können die gewünschte Tabelle selbst erstellen.

Eine alternative Möglichkeit für den Aufbau einer Routingtabelle für das Reverse Path Broadcasting ist das Versenden so genannter **Poison Reverse**-Pakete. Hierbei antworten die nachgeordneten Router auf den Erhalt unerwünschter Pakete mit einer Art Abmeldung beim zuvor weitersendenden Router. Daraufhin werden Multicasting-Pakete gleichen Absenders nicht mehr an die sich „beschwerenden“ Router weitergesendet.

Während - bildlich gesprochen - beim RPF die nachgeordneten Router alle Versuche der Datenpakete unterbinden, von der strahlenförmigen Ausbreitung im Netz abzuweichen, gewährleistet das Reverse Path Broadcasting nun, dass diese Versuche erst gar nicht unternommen werden. Der aufkommende Traffic bei einer Multicast-Übertragung wird also nochmals verringert.

Ein richtiges Multicasting bietet auch das Reverse Path Broadcasting nicht. Auch hier werden alle Router eines Netzes von den Datenpaketen erreicht. Wie der Name es also anzeigt, ist RPB ein Broadcasting-Verfahren.

**Quellbasiertes Multicasting - TRPB** Während alle zuvor genannten Verfahren eine schrittweise Verringerung der Netzlast beim Multicasting ermöglichten, kümmert sich das so genannte **Truncated Reverse Path Broadcasting** um die Entlastung der angeschlossenen Endsysteme, also zum Beispiel Ethernet- bzw. Token-Ring-LAN's. Bisher wurden - ohne dass ich es explizit erwähnt habe - alle auf den Routern einlaufenden Multicast-Pakete in die angeschlossenen Endsysteme weitergeleitet.[1]

Sollten also beispielsweise in einem Uni-Netz eine oder mehrere breitbandige Multicast-Übertragungen stattfinden, dann wurden bisher alle angeschlossenen LAN's damit belastet. Und zwar unabhängig davon, ob es interessierte angeschlossene Rechner in ihnen gab oder nicht.

Der Zusatz „Truncated“ bedeutet nun nichts anderes, als dass man unter Nutzung von beispielsweise IGMP uninteressierte LAN's von der Besendung mit Multicast-Paketen bestimmter Gruppen ausnimmt. Sicherlich wird hiermit viel Nutzen unter wenig Aufwand gewonnen.

**Quellbasiertes Multicasting - RPM** Einen nicht unwesentlichen Schritt macht die Entwicklung des **RPM - Reverse Path Multicasting** aus. Nachdem alle zuvor genannten Verfahren dafür sorgten, dass ein Multicasting-Paket durch das ganze Netz geleitet wird - mehr oder weniger effizient versteht sich - und damit für sich gesehen Broadcasting-Verfahren waren, bietet das RPM eine neue wesentliche Eigenschaft: Der Datenverkehr wird auf interessierte Teilnetze beschränkt.[1]

Nachdem die Datenpakete einer Gruppe zunächst gemäß des TRPB in alle Bereiche eines Netzes vorgestoßen sind, melden sich uninteressierte Router bei vorgeordneten Routern ab. Unter uninteressierten Routern verstehe ich dabei solche Router, die kein interessiertes Endsystem versorgen müssen und von denen auch keine nachgeordneten Router abhängen (z.B. klassische Gateway-Router). Ferner gelten solche Router als uninteressiert, bei denen sich neben etwaiger eigener Endsysteme alle nachgeordneten Router für die Daten der entsprechenden Gruppe abgemeldet haben.

Dieses Abmelden oder genauer gesagt **Pruning** muss jedoch im Hinblick auf die mögliche Dynamik der jeweiligen Gruppe wieder rückgängig zu machen sein. Entweder man setzt auf ein (nicht zu kurzes) Zeitlimit für Pruning-Informationen und fällt regelmäßig in das TRPB zurück, oder man realisiert ein aktives Wiederanmelden, auch **Grafting** genannt.

Reverse Path Multicasting reduziert den Datentransfer für eine Multicast-Sendung nochmals ganz erheblich. Ganze Teilnetze in Netzen mit einer Baumstruktur werden entlastet. Ferner wird erstmals wirklich ein Multicasting implementiert, nachdem alle zuvor genannten Verfahren eher dem Broadcasting zuzuordnen waren.

**Quellbasiertes Multicasting - Übersicht** Bevor nun eine ganz neue Kategorie der Multicasting-Verfahren vorgestellt wird, die der Bäume mit Rendezvous-Stellen, nochmal ein kurzer Überblick über das Fluten und die quellbasierten Verfahren:

Tabelle 1 zeigt, wie die Verbesserungen Schritt für Schritt mit jedem neuen Konzept eingeführt wurden. Auch ist gut zu erkennen, dass erst das RPM ein richtiges Multicastingverfahren ist:

	<b>Paketannahme</b>	<b>Paketweiterleitung</b>	<b>Netzbereich</b>	<b>Art</b>
<b>Fluten</b>	alle	alle	ganzes Netz	Broadcasting
<b>RPF</b>	von kürzestem Pfad	alle	ganzes Netz	Broadcasting
<b>RPB</b>	von kürzestem Pfad	auf kürzestem Pfad	ganzes Netz	Broadcasting
<b>TRPB</b>	von kürzestem Pfad	auf kürzestem Pfad	Endsysteme nach Gruppen	Broadcasting
<b>RPM</b>	von kürzestem Pfad	auf kürzestem Pfad	Teilnetze nach Gruppen	Multicasting

Tabelle 4.1: Übersicht über das Fluten und die quellbasierten Konzepte

### Bäume mit Rendezvous-Stellen

Nachdem die quellbasierten Routing-Verfahren zur Realisierung eines Multicastings schlicht und einfach das ganze Netz mehr oder weniger effektiv mit den Daten einer Gruppe versorgt haben, geht die Strategie der **Bäume mit Rendezvous-Stellen** andere Wege.

Hier werden für jede Gruppe spezielle Router auserkoren, die die Verteilung der Daten an alle Router mit Gruppenmitgliedern durchzuführen haben. Diese auserkorenen Router werden **Rendezvous-Stellen** genannt.[2]

Router, die aufgrund angeschlossener Endsysteme Interesse an den Datenpaketen einer Gruppe haben, melden sich per Unicast bei der Rendezvous-Stelle. Beim Transport der entsprechenden Anmeldepakete tragen die zur Rendezvous-Stelle weiterleitenden Router die Information, dass sie nun Teil eines Multicast-Baumes werden zusammen mit der zukünftigen Ausgangsschnittstelle in ihre Multicast-Routing-Tabellen ein.

Will ein Router nun Datenpakete für eine Multicasting-Gruppe versenden, so richtet er sie zunächst als Unicast an die zugehörige Rendezvous-Stelle. Von hier aus existiert dann der entsprechende Verteilbaum zu allen interessierten Routern.

Eine andere Möglichkeit wäre es, die Routingtabellen bidirektional einzurichten. Also für den Fall, dass eine Anmeldung zur Gruppe  $g$  von Schnittstelle  $A$  zur Schnittstelle  $B$  eines zwischenlagerten Routers läuft, werden dann zwei statt einem Eintrag in die Tabelle vorgenommen. Nämlich wie bisher der Eintrag, dass Pakete der Gruppe  $g$ , die auf Schnittstelle  $B$  einlaufen auf Schnittstelle  $A$  weiterzuleiten sind und zusätzlich nun auch ein Eintrag für die Weiterleitung der Pakete der Gruppe  $g$  von Schnittstelle  $A$  zu Schnittstelle  $B$ .

Mit diesem Verfahren kann man für die Versendung von Datenpaketen auf Unicast-Anteile verzichten, da alle versendeten Pakete den entsprechenden Ast des Baumes zunächst rückwärts zur Rendezvous-Stelle durchlaufen würden, um dann von der Rendezvous-Stelle aus über den ganzen Baum zu allen Empfängern zu gelangen.

### Vergleich der Strategien

Auf den ersten Blick erscheint einem das zuletzt vorgestellte Konzept der Bäume mit Rendezvous-Stellen als das fortschrittlichste. Jedoch gibt es auch Anwendungsbereiche in denen die anderen Konzepte klare Vorteile aufweisen.

Im Einzelnen:

Bäume mit Rendezvous-Stellen verwenden von vorn herein gezielte Verteilbäume und stützen sich nicht auf das recht primitive Konzept des **Flutens** ab. Jedoch sind sie aufgeschmissen, wenn die Rendezvous-Stelle als so genannter **Single Point of Failure** einmal vorübergehend oder komplett ausfällt.

Ein weiteres Problem ist die hohe **Lastkonzentration** in der direkten Umgebung der Rendezvous-



Stelle. Während direkt benachbarte Router der Rendezvous-Stelle mit hoher Wahrscheinlichkeit alle Pakete einer Gruppe zu transportieren haben, werden weiter entfernt liegende Router mit deutlich geringerer Wahrscheinlichkeit bemüht.

Nicht zuletzt spricht dann noch der höhere administrative **Aufwand**, der sich in der Findung und Einrichtung der Rendezvous-Stellen begründet eher für den Einsatz quellbasierter Konzepte als für den Einsatz von Bäumen mit Rendezvous-Stellen.

Alles in allem wird in der Praxis nur dann auf Bäume mit Rendezvous-Stellen zurückgegriffen, wenn in sehr großen Netzen nur wenige Gruppenmitglieder anzufinden sind, also von einer geringen **Gruppendichte** auszugehen ist. In diesem Falle würden quellbasierte Konzepte eine unnötige hohe Netzlast auslösen und fallen deshalb in die zweite Wahl.[1]

Die abschließende Tabelle 2 gewährt noch einmal Überblick über die genannten Punkte:

Kriterium	Quellbasiert	Rendezvous-Stellen
Grundkonzept Fluten	ja	nein
Single Point of Failure	nein	ja
Lastkonzentration	nein	ja
Aufwand	gering	hoch
Gruppendichte	hoch	niedrig

Tabelle 4.2: Vergleich Quellbasiert - Rendezvous-Stellen

## Praktische Beispiele

**DVMRP** Als eines von zwei praktischen Beispielen für Multicasting-Protokolle soll das **DVMRP** (Distant Vector Multicast Routing Protocol) dienen. Hierbei handelt es sich um ein quellbasiertes Protokoll, das in Netzen eingesetzt wird, die für den Unicast-Betrieb auf RIP (Routing Information Protocol; Internes Unicast-Routing-Protokoll; [8]) zurückgreifen. DVMRP läuft dabei mit eigenen Routingtabellen parallel zu RIP und basiert seit Version 3 auf RPM, nachdem es zuvor TRPB zur Grundlage hatte.[10]

Sollten einzelne Router in einem RIP-Netz nicht die Unterstützung von DVMRP bieten, dann werden diese über RIP-Unicasts getunnelt. Damit sich jedoch alle DVMRP-Router in einem Netz kennenlernen, werden so genannte **Neighbor-Probe-Pakete** eingesetzt. Diese werden ebenfalls per RIP-Unicast gesendet und bei fehlender Unterstützung von DVMRP an die nächstliegenden Router weitergeleitet. Der für Distant-Vector-Protokolle übliche Austausch von Routingtabellen erfolgt anschließend ausschließlich mit den festgestellten DVMRP-Nachbarn.

**PIM** Das zweite Beispiel für Multicasting-Protokolle ist das **PIM (Protocol Independent Multicasting)**. Wie der Name es schon andeutet, handelt es sich hierbei um ein Protokoll, das unabhängig vom installierten Unicast-Protokoll arbeitet.[11]

Genauer gesagt wird der **Sparse Mode** dieses Protokolls vorgestellt, da dieser für weit verteilte Gruppen gedacht ist und daher auf dem Konzept der Bäume mit Rendezvous-Stellen beruht. Der andere Modus - **PIM Dense Mode** - ist für dichte Gruppen gedacht und beruht folglich auf einem der quellbasierenden Konzepte.[1]

In einem Netz, dessen Multicasting auf PIM Sparse beruht, wird ein bestimmter Router als so genannter **Bootstrap-Router** vorkonfiguriert. Dieser Bootstrap-Router ist verantwortlich für

die Einteilung der Rendezvous-Stellen. Die Kandidaten für solche Rendezvous-Stellen bewerben sich per Unicast beim ihm, was ihn aber nicht davon abhält, sich für die eine oder andere Gruppe selbst als Rendezvous-Stelle einzuteilen.

Die Anmeldepakete, die weiter oben bei der Einführung in das Konzept der Bäume mit Rendezvous-Stellen beschrieben wurden und die für den Aufbau des Multicasting-Baumes verantwortlich sind, nennen sich bei PIM Sparse **Join-Pakete**. Diese Join-Pakete werden periodisch gesendet um den Verteilbaum aufrecht zu erhalten.

Die Abmeldung aus dem Multicasting-Baum kann bei PIM Sparse nun entweder durch Unterlassen des periodischen Sendens der Join-Pakete oder über gezielte Abmeldung geschehen. Letztere wird wie beim RPM **Pruning** genannt.

Eine besondere Fähigkeit des PIM Sparse ist es, **senderspezifische Bäume** aufzubauen.[1] Sollte also ein Router einen besonders hohen Sendeanteil innerhalb einer Gruppe haben, so kann die Rendezvous-Stelle gemeinsam mit den anderen beteiligten Routern den Aufbau eines separaten Verteilbaumes für diesen Sender veranlassen. Ist dieser Baum für einen der beteiligten Router aufgebaut, so meldet sich dieser für den Ast zur Rendezvous-Stelle ab und der senderspezifische Baum nimmt Gestalt an.

PIM Sparse basiert bezüglich des Versendens von administrativen Paketen auf periodischen Wiederholungen. Daher kann auf das Quittieren dieser Pakete verzichtet werden. Ferner ist es weniger wahrscheinlich, dass getroffene Entscheidungen über den Aufbau von Bäumen oder senderspezifischen Bäumen unumkehrlich werden bzw. zu dauerhaften Fehlentwicklungen führen können. Dieses Prinzip wird **Soft-State** genannt.

### 4.3.4 Multicasting in der Transportschicht

Wie weiter oben bereits angesprochen ist die Vermittlungsschicht des Internetschichtenmodells lediglich für die Durchleitung der Datenpakete durch die Netze zuständig.[3] Ob im Bereich des Multicastings mehr Systeme bzw. Router von den Datenpaketen erreicht werden, als eigentlich notwendig, spielt für die Vermittlungsschicht nur bezüglich einer eventuell reduzierbaren Netzlast eine Rolle. Hauptaugenmerk der Vermittlungsschicht ist und bleibt, dass zumindest alle diejenigen Systeme die Daten bekommen, die sie auch benötigen. Das Erreichen uninteressierter Systeme oder Router schadet im Zweifelsfall nicht.

Die Realisierung genauerer Gruppenzugehörigkeiten und weiterer Aspekte der Gruppenkommunikation (also u.a. Sicherheit, Zuverlässigkeit oder Senderechte), müssen von der Transportschicht und der Anwendungsschicht realisiert werden. Ich werde deshalb kurz auf die beiden am meisten im Internet verbreiteten Transportprotokolle TCP und UDP eingehen und ihre Bedeutung für das Multicasting darstellen. Auf die Beschreibung weiterer Protokolle oder Anwendungen werde ich verzichten; sie sind nicht Bestandteil des Seminarthemas.

#### TCP

Das **Transport Control Protocol (TCP)** ist ein zuverlässiges und verbindungsorientiertes Transportprotokoll.[4] Das bedeutet, dass gesendete Pakete mittels Quittungen auf ihre Ankunft überprüft und gegebenenfalls neu gesendet werden. Außerdem wird die Reihenfolge der Pakete bei Bedarf wiederhergestellt, was insbesondere zur Wiederherstellung zerstückelt übertragener Dateien notwendig ist.

TCP unterstützt keinerlei Funktionen für den Datentransfer in Gruppen und ist als reines Unicasting-Protokoll anzusehen. Nicht zuletzt auch deswegen, weil das Problem der Quittungs-

implosionen beim Nutzdatentransfer mit TCP unausweichlich wäre.

Trotzdem ist TCP in der Welt des Multicastings alles andere als bedeutungslos. Im Kapitel über die Vermittlungsschicht wurde das eine oder andere mal auf Unicasts verwiesen. Die Grafting-Pakete des RPM zum Beispiel werden in der praktischen Umsetzung mittels TCP realisiert.

Weitere Anwendungsbeispiele für TCP lassen sich im Bereich der Anwendungssoftware für das Multicasting finden. Will man beispielsweise den Mitgliedern einer Gruppenkommunikation Schlüssel für die Sicherung des Datenverkehrs übermitteln, so bietet sich in nicht wenigen Szenarios (ich denke da u.a. an eine gesicherte Videokonferenz) TCP als Transportprotokoll an.

## UDP

Ein anderes wichtiges Protokoll in der Welt des Internets ist das **User Datagram Protocol (UDP)**.<sup>[4]</sup> Im Unterschied zu TCP arbeitet es ohne Quittungen, ist im Gegenzug dafür jedoch nicht zuverlässig. Sollten also UDP-Pakete verloren gehen, dann werden sie nicht nachgesendet. Auch die Reihenfolge der Pakete wird nicht wiederhergestellt. Sollte ein später versendetes Paket also früher eintreffen, als ein nach ihm versendetes, dann wird diese Vertauschung von UDP nicht beseitigt.

UDP ist in folge dessen für den integren Transport von Dateien ungeeignet, ein einzelner Paketfehler (also Verlust oder Vertauschung), führt zu einem Totalverlust der ganzen Übertragung.

Ganz anders sieht es aus bei der Realisierung von Audio- oder Videoübertragungen. Sollten hier einzelne Pakete verloren gehen oder verspätet eintreffen, wirkt sich dies allenfalls in einem kurzen Knistern oder einer minimalen Bildstörung aus, was in den wenigsten Fällen tragisch wäre.

UDP ist für das Multicasting geeignet. Ohne das Problem der Quittungsimplosionen ist es im Gegensatz zu TCP auch für den Nutzdatenverkehr zu gebrauchen. Sein hoher Entwicklungsstand, nicht zuletzt wegen seiner großen Bedeutung im Bereich des Unicastings, führt dazu, dass es im Bereich des Multicastings eine zentrale Stellung einnimmt.

Will ein Rechner einer Videokonferenz zuschauen, muss er sich zunächst über IGMP für die entsprechende Multicasting-Gruppe anmelden. Gerade Multimediaelemente werden im Internet häufig mittels UDP übertragen. Sollte dies also auch bei der angesprochenen Videokonferenz der Fall sein, so bleibt dem Rechner nur noch überlassen, sein nun versorgtes LAN auf dem UDP-Port zu belauschen und alle eintreffenden Gruppenpakete aufzunehmen.

## Andere Protokolle

Neben den beiden weitläufig bekannten Protokollen TCP und UDP gibt es auch eine ganze Reihe expliziter Multicasting-Transportprotokolle. Um nur mal einen Eindruck von der Vielfalt zu vermitteln, seien einige Beispiele genannt<sup>[1]</sup>:

- XTP (Xpress Transport Protocol)
- MTP (Multicast Transport Protocol)
- RMP (Reliable Multicast Protocol)
- LBRM (Log-Based Receiver-Reliable Multicast)

- SRM (Scalable Reliable Multicast)
- RMTP (Reliable Multicast Transport Protocol)

Alle diese Protokolle versuchen auf unterschiedliche Art und Weise die verschiedenen Aspekte der Gruppenkommunikation zu realisieren und setzen dabei verschiedene Schwerpunkte. Die einen versuchen die Verlässlichkeit in der Multicastingwelt zu implementieren, während andere in erster Linie große Gruppendynamiken zu handeln bemüht sind. Wie es auch nicht anders zu erwarten ist, setzen sie sich denn auch mit unterschiedlichem Erfolg in der Internetwelt durch.

## 4.4 Das M-Bone

### 4.4.1 Grundlagen

Wie bereits erwähnt, ist das Multicasting nur bis zur Größenordnung der autonomen Systeme umfassend realisiert. Will man nun aber weltweites Multicasting durchführen, dann reichen die Grenzen der größten autonomen Systeme (ein Beispiel wäre das Backbone der Deutschen Telekom) nicht mehr aus.

Da die aktuellen Exterior-Gateway-Protokolle, die ja für das Inter-Domain-Routing verantwortlich sind, noch immer keine ausreichenden Multicast-Fähigkeiten besitzen, greift man für weltweites Multicasting auf das so genannte **MBone (Multicast Backbone)** zurück.[2]

Das MBone war zunächst als Forschungsgrundlage für weltweites Multicasting gedacht. Doch wie es mit allen Dingen läuft, die sich am Markt einmal durchgesetzt haben, wird wohl auch das MBone nicht so schnell von der Bildfläche verschwinden. Von 1992, dem Gründungsjahr des MBones, bis 1998 haben sich in einem exponentiellen Wachstum 12000 Einzelnetze in das MBone integriert, bzw. bieten 12000 Einzelnetze MBone-Dienste an.

Das MBone basiert technisch gesehen auf vielen einzelnen Multicast-fähigen Netzen, die über **Tunnel** verbunden sind. Es ist also notwendig, dass die Grenzrouter der MBone-Einzelnetze Unicast-Verbindungen durch nicht Multicast-fähige Netze beherrschen (Abbildung 4.1). Dies lässt sich zum einen durch **IP-Einkapselungen** realisieren, wobei schlicht und einfach die Unicast-Adresse des zu erreichenden Grenzrouters angegeben wird. Eine andere Möglichkeit ist das **Loose-Source-Record-Routing**[3], ein in IP implementiertes Verfahren, bei dem die Router mittels des IP-Headers genaue Anweisungen bekommen über welche Router sie die Pakete jeweils weiterzuleiten haben. Letzteres Verfahren weist jedoch deutliche Nachteile für den Fall auf, dass einzelne Router unzuverlässig arbeiten bzw. sich die Topologie der zu tunnelnden Netze verändert.

### 4.4.2 Adressierung

Ein wesentliches Problem beim weltweiten Multicasting ist die eindeutige Adressierung. Für das Multicasting gab es bisher keine zentrale Vergabestelle für Adressbereiche, wie man es aus der Unicasting-Welt kennt. Es war bzw. ist Praxis, einfach per Zufall eine derzeit unbenutzte Klasse-D-Adresse zu nehmen.

Für Multicasting, das auf begrenzte Netze beschränkt bleibt, stellt diese Vorgehensweise aufgrund der Größe des Adressraums sicherlich kaum ein Problem dar. Möchte man sich aber weltweit mitteilen, können vermehrt Kollisionen auftreten.

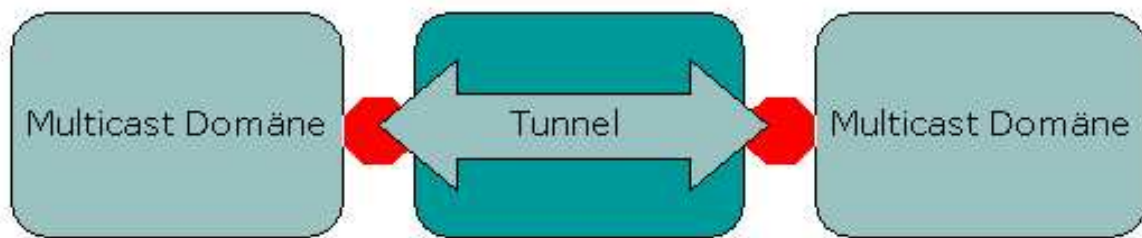


Abbildung 4.8: Tunneln von nicht Multicast-fähigen autonomen Systemen

Ein Lösungsansatz ist das Projekt **MASC (Multicast-Address-Set-Claim)**. [12] Hierbei sollen - hierarchisch angeordnet - dedizierte Router aufgestellt werden, bei denen Gruppen ihre Adressen anfordern können. Diese Router wiederum erhalten ihre zu verteilenden Adressen bzw. Adressbereiche von den ihnen vorgeordneten Stellen.

Auch nach der endgültigen Umsetzung des MASC-Projektes wird man versuchen, Adressen in ihrer regionalen Gültigkeit zu begrenzen. Dies kann mehrere Gründe haben: [1]

- Zum einen vergrößert man die Zahl der verfügbaren Gruppenadressen. Angenommen, eine Gruppe innerhalb des Deutschen Forschungsnetzes (DFN) möchte eine Videokonferenz veranstalten. Es wäre glatte Verschwendung, nicht dieselbe Adresse für eine Gruppenkommunikation zu verwenden, die sich beispielsweise auf Nordamerika beschränkt.
- Ein weiterer Punkt wäre die Einschränkung gefluteter Bereiche. Wieso soll ein quellbasiertes Netz ohne Gruppenmitglieder, das zufällig an das MBone angeschlossen ist, mit unnötigem Datentransfer belastet werden?
- Ein letzter Punkt, den ich bei dieser Gelegenheit ansprechen möchte, ist die Sicherheit von Daten. Über die Begrenzung der Adressgültigkeit kann ich zum Beispiel verhindern, dass eine Videokonferenz mein Netz oder mein Subnetz verlässt.

Für die Adressraumbegrenzung gibt es zwei wesentliche Ansätze. Zum einen das **TTL-Scoping**, zum anderen die Einrichtung von **administrativen Bereichen** anhand der Klasse-D-Adressen. [1]

**TTL-Scoping** Beim TTL-Scoping bekommen die Grenzrouter der einzelnen MBone-Domänen den Auftrag, bei der Weiterleitung der Datenpakete nicht den Wert eins von der TTL des jeweiligen Pakets abzuziehen, sondern vordefinierte größere Werte, die in Tabelle 3 einzusehen sind. Möchte ich nun, dass mein Datenpakete regionale Grenzen überschreiten, dann muss ich zur gewöhnlichen TTL diesen Wert hinzu addieren.

**Administrative Bereiche** Bei der Einteilung der Adressbereiche über die IP-Adressen benutzt man die letzten drei Byte gemäß Tabelle 4. Das erste Byte wird nicht für die Einteilung verwendet, sondern dient lediglich der Erkennung als Multicast-Adresse (Klasse D). Der Wert 240 ist zufälliger Natur.

Wie die entsprechenden Netzbetreiber die vier Bereiche innerhalb ihrer eigenen Infrastruktur einteilen bleibt ihnen überlassen. Abgesehen sicherlich von der höchsten Kategorie „erweiterter organisatorischer Bereich“, die möglichst für weltweites Multicasting reserviert bleiben sollte.

TTL-Wert	Größe des Bereiches
0	Knoten
1	Subnetz
32	Domäne
64	Region
128	Kontinent
255	unbegrenzt

Tabelle 4.3: Übersicht über die Werte des TTL-Scopings

Adressbytes	Größe des Bereiches
ab 240.255.0.0	lokaler Bereich
ab 240.253.0.0	erweiterter lokaler Bereich
ab 240.192.0.0	organisatorischer Bereich
ab 240.0.0.0	erweiterter organisatorischer Bereich

Tabelle 4.4: Übersicht über die Werte der administrativen Einteilung

## 4.5 Zusammenfassung und Ausblick

### 4.5.1 Zusammenfassung

Neben einer einleitenden Passage zu den Grundlagen der Gruppenkommunikation und den sich daraus ergebenden technischen Problemstellungen, habe ich als Kernbestandteil der Arbeit verschiedene Routingstrategien für das Multicasting dargestellt.

Nach der Vorstellung des relativ primitiven Flutens bin ich weiter gegangen zu den so genannten quellbasierten Verfahren. Diese Verfahren - allesamt betrachten für die Weiterleitung von Datenpaketen lediglich deren Absenderadresse - habe ich als sukzessive Weiterentwicklung des Flutens eingeordnet. Angefangen vom RPF (Remote Path Forwarding) bis hin zum RPM (Remote Path Multicasting) wird mit jeder Entwicklungsstufe die bei einer Multicasting-Übertragung auftretende Netzlast reduziert.

Als Gegenkonzept zu den quellbasierten Verfahren habe ich dann das Konzept der Bäume mit Rendezvous-Stellen eingeführt. Hier werden für den Aufbau von einzelnen Gruppenkommunikationen dedizierte Router ermittelt (Rendezvous-Stellen), von denen aus sich Verteilbäume für die Daten aufbauen lassen. Die Vorteile dieses Konzeptes, nämlich Multicasting für stark verstreute Gruppen in sehr großen Netzen zu ermöglichen, habe ich gegen unabweisliche Nachteile abgegrenzt. Höherer administrativer Aufwand und die Abhängigkeit von der jeweiligen Rendezvous-Stelle inklusive der sie umgebenden Infrastruktur favorisieren den weitstmöglichen Einsatz der quellbasierten Konzepte.

Nach einer kurzen Vorstellung zweier realer Protokolle (DVMRP und PIM Sparse) habe ich eine Abgrenzung der Aufgabenstellung für das Multicast-Routing geschaffen. Multicast-Routing in der dafür vorgesehenen Vermittlungsschicht (siehe Internetschichtenmodell) beschränkt sich einzig und allein auf die zur Verfügungstellung von Datenpaketen in mindestens allen daran interessierten Systemen. Genauere Auflösungen von Gruppenzugehörigkeiten und andere Aspekte der Gruppenkommunikation fallen in die Zuständigkeit der Transportschicht.

Kurze Übersichten über TCP und UDP sowie die Darstellung der Vielfalt expliziter Multicasting-Transportprotokolle runden Kapitel 3 ab.

Als letztes habe ich mich dem so genannten MBone zugewendet. Dieses „Multicasting Backbone“ basiert auf der Tunnelung nicht Multicasting-fähiger Netze und stellt die Möglichkeit zu weltweitem Multicasting dar.

Die Darstellung der Notwendigkeit einer weltweit einheitlichen Adressvergabe und der Notwendigkeit zur Beschränkung von Adressgültigkeitsbereichen (beispielsweise für deren Mehrfachnutzung) bilden den Abschluss der Arbeit.

#### 4.5.2 Ausblick

Multicasting ist nach wie vor sehr wenig im alltäglichen Internetgebrauch vertreten. Veranstaltungen wie Videokonferenzen oder die Übertragung einer virtuellen Vorlesung bleiben nicht zuletzt wegen des erhöhten Administrationsaufwandes wenigen Internetusern vorbehalten.

Erst eine Integration des Multicastings in die Standard-Exterior-Gateway-Protokolle und die Bereitstellung der entsprechenden Dienste durch die Internet Service Provider könnten zum „Multicasting für Jedermann“ führen.

Mit einem steigenden Marktanteil von Multicasting-Angeboten würde dann auch die Entwicklung der Multicasting-Protokolle forciert werden, so dass diese möglicherweise eines Tages aus ihrem Dasein als Forschungsobjekte bzw. als Arbeitsmittel einer ganz exklusiven Gesellschaft von Internet-Experten heraustreten könnten.

## 4.6 Abkürzungen

<b>ACK</b>	Acknowledge
<b>ATM</b>	Asynchronous Transfer Mode
<b>DFN</b>	Deutsches Forschungsnetz
<b>DVMRP</b>	Distant Vector Multicast Routing Protocol
<b>IGMP</b>	Internet Group Management Protocol
<b>IP</b>	Internet Protocol
<b>LAN</b>	Local Area Network
<b>MAC</b>	Media Access Control
<b>MASC</b>	Multicast Address Set Claim
<b>MBone</b>	Multicast Backbone
<b>NACK</b>	Not Acknowledge
<b>PIM</b>	Protocol Independent Multicasting
<b>RFC</b>	Request for Comments (Spezifikationsdokument)
<b>RPB</b>	Reverse Path Broadcasting
<b>RPM</b>	Reverse Path Multicasting
<b>RPF</b>	Reverse Path Forwarding
<b>TCP</b>	Transport Control Protocol
<b>TRPB</b>	Truncated Reverse Path Broadcasting
<b>TTL</b>	Time To Live

# Literaturverzeichnis

- [1] RALPH WITTMANN UND MARTINA ZITTERBART, *“Multicast“*, dpunkt.verlag 1999, ISBN 3920993403
- [2] CHRISTIAN HUITEMA, *“Routing im Internet“*, Prentice Hall 1996, ISBN 3827295262
- [3] MATHIAS HEIN, *“TCP/IP“*, 5. aktualisierte und erweiterte Ausgabe, MITP-Verlag 2000, ISBN 3826640799
- [4] KEVIN WASHBURN UND JIM EVANS, *“TCP/IP, 2. aktualisierte und erweiterte Ausgabe“*, Addison-Wesley 1997, ISBN 3827311454
- [5] DR. MATHIAS RAUTENBERG, Vorlesung *“IP-Weitverkehrsnetze“*, Universität der Bundeswehr München, Wintersemester 2002
- [6] TOM MORAWITZ, *“Internet - Architektur und Protokolle“*, Vortrag und Ausarbeitung zum Seminar *“Routingstrategien im Internet“*, Universität der Bundeswehr München, Frühjahrstrimester 2002
- [7] RFC 2236, *“IGMP Version 2“*,  
<http://www.ietf.org/rfc/rfc2236.txt?number=2236>
- [8] MARCO SCHULER, *“Externe Routing-Protokolle“*, Vortrag und Ausarbeitung zum Seminar *“Routingstrategien im Internet“*, Universität der Bundeswehr München, Frühjahrstrimester 2002
- [9] MARCUS KÖRNER, *“Interne Routing-Protokolle“*, Vortrag und Ausarbeitung zum Seminar *“Routingstrategien im Internet“*, Universität der Bundeswehr München, Frühjahrstrimester 2002
- [10] RFC 1075, *“DVMRP“*,  
<http://www.ietf.org/rfc/rfc1075.txt?number=1075>
- [11] RFC 2362, *“PIM Sparse Mode“*,  
<http://www.ietf.org/rfc/rfc2362.txt?number=2362>
- [12] RFC 2909, *“DVMRP“*,  
<http://www.ietf.org/rfc/rfc2909.txt?number=2909>



# 5 AntNetz

*Klaus Bernhard Riepel*

## Inhaltsverzeichnis

---

<b>5.1</b>	<b>Einleitung</b> . . . . .	<b>98</b>
<b>5.2</b>	<b>Ameisen als Vorbild</b> . . . . .	<b>99</b>
<b>5.3</b>	<b>Travelling Salesmann Problem</b> . . . . .	<b>100</b>
5.3.1	Problembeschreibung . . . . .	100
5.3.2	Agentenbasiertes Lösungsverfahren . . . . .	100
5.3.3	Vergleich mit Greedy Algorithmus . . . . .	101
<b>5.4</b>	<b>AntNet</b> . . . . .	<b>101</b>
5.4.1	Routingtabelle des AntNet . . . . .	101
5.4.2	Routing im AntNet . . . . .	102
5.4.3	Algorithmus des AntNet . . . . .	102
5.4.4	Bewertung gefundener Wege . . . . .	105
<b>5.5</b>	<b>Vergleich zwischen AntNet und anderen Routingmethoden</b> . . . . .	<b>106</b>
5.5.1	Vergleichsalgorithmen . . . . .	106
5.5.2	Simulierte Netzwerke . . . . .	107
5.5.3	Simulierte Netzlasten . . . . .	107
5.5.4	Ergebnisse . . . . .	108
5.5.5	Zusammenfassung . . . . .	110
<b>5.6</b>	<b>Schluss</b> . . . . .	<b>110</b>
5.6.1	Einordnung in den Kontext des Seminars . . . . .	110
5.6.2	Ergebnisse und Ausblicke . . . . .	111
	<b>Literaturverzeichnis</b> . . . . .	<b>111</b>

---

## Kurzbeschreibung

Diese Seminararbeit zeigt die Funktionsweise von agenten-basiertem Internetrouting am Beispiel des sogenannten AntNets. Das AntNet beruht auf einem System unabhängiger Agenten, die, jeder für sich, Informationen über die Topologie des Netzwerkes sammeln. Die Agenten kommunizieren nicht direkt, sondern geben ihre Informationen an einen Router weiter, der sich dann aus allen Einzelinformationen ein Gesamtbild macht. Dieses Verhalten ähnelt dem staatenbildender Insekten. Der Algorithmus des AntNet wird im Folgenden auch mit einigen weiteren, aktuellen Routing-Algorithmus verglichen, um dessen Leistungsfähigkeit beurteilen zu können.

## Abstract

This paper explains the work of agent-based internet-routing. Therefore I explain the so called AntNet. This is a system of independent, mobile agents collecting information about the topology of a network. Their communication is indirect and is mediated by a router, building a routing table from all information he gets from his agents. This form of communication is very similar to the communication of social insects. The algorithm of AntNet will be compared to some other, actual routing methods to show his effectiveness.

## 5.1 Einleitung

Auf der Suche nach neuen Technologien stößt man irgendwann an Grenzen, bedingt durch mangelnde Fantasie, mangelnde Rechenleistung oder die Grenzen des Machbaren. Viele Erfindungen, die für die Menschheit bahnbrechend waren, wurden in ähnlicher Weise schon vor langer Zeit in der Natur benützt. Hierfür gibt es viele Beispiele. Neues erfinden und Bestehendes verbessern, das geht am leichtesten durch anschauen, denn warum nicht den Wissensvorsprung nutzen, den die Natur in Millionen von Jahren aufgebaut hat.

Flugzeuge sind das beste Beispiel, wie wir lernen konnten und immer noch können. Durch einfaches kopieren der Flügelform eines Vogels haben wir die nötigen Dinge über Auftrieb gelernt, um ein unförmiges Objekt einem Vogel gleich in die Lüfte zu bekommen. Aber damit war die Entwicklung noch nicht zu Ende. Weitere Erforschung der Flügel brachte weitere Erkenntnisse. Sogenannte Winglets, die an den Enden der normalen Flugzeugflügel angebracht werden, helfen Treibstoff sparen. Vorbild war der Adler, an dessen Flügelspitzen einige Federn abstehen. Aber auch in Bereichen, in denen man für den Flugzeugbau keine Hilfen erwartet, hat die Natur noch einen technologischen Vorsprung. Die Haie haben eine sehr raue Haut, die den Strömungswiderstand vermindert. Derselbe Effekt funktioniert auch ausserhalb des Wassers und wird inzwischen auch für Flugzeuge verwendet. Diese Beispiele zeigen uns, dass wir noch viel von der Natur lernen können und müssen, wenn wir Neues erfinden und Bestehendes verbessern wollen. Abstraktes denken erleichtert manchmal Zusammenhänge zwischen der Tier- oder Pflanzenwelt zu sehen und auf unsere Probleme zu übertragen.

Das Internet, von Menschen geschaffen, ohne ein Vorbild in der Natur, lässt vermuten, dass wir es ohne Hilfe von aussen weiterentwickeln müssen. Doch auch hier gibt es Möglichkeiten zu lernen. Ameisen, die keine direkte Kommunikation untereinander haben, finden trotzdem als Kollektiv schnell den besten Weg zu einer Futterquelle.

Computer, die zwar miteinander kommunizieren können, aber aufgrund der Masse niemals alle

auf einmal erreichen können, müssen sich mit ähnlichen Bedingungen wie Ameisen auseinandersetzen. Sie suchen den kürzesten Weg zu einer Quelle (Futter- oder Datenquelle) und müssen sich dabei auf ständig wechselnde Voraussetzungen einstellen. Bereits bekannte Wege können unterbrochen werden. Andere Individuen (Ameisen oder Computer) könnten bereits bessere Wege kennen. Zu viele Ameisen oder Computer auf einer Strecke können zu Verstopfungen führen. Das sogenannte AntNet, ein funktionierendes Modell aufbauend auf einem agentenbasierten System, schickt seine Agenten wie Ameisen aus, um so schnellere Wege durch das Internet zu finden. Dies funktioniert, indem jeder Agent ein Teilproblem löst. Durch indirekte Kommunikation zwischen den Agenten wird so aus vielen Teillösungen eine Gesamtlösung. Dabei gibt es am Ende keinen richtigen und keinen falschen Weg. So, wie es viele Wege von München nach Hamburg gibt, gibt es viele Wege durch das Internet. Die direkte Verbindung ist meistens die schnellste, aber wenn alle diese Verbindung nutzen, leidet darunter die Geschwindigkeit. Durch geschicktes Ausweichen auf Nebenstraßen kann man jedoch starken Verkehr ausweichen und so im Endeffekt schneller als auf der Hauptstraße sein. Das AntNet nutzt Autobahnen, Haupt- und Nebenstraßen, indem es die Datenpakete entsprechend der Leistungsfähigkeit der Datenkanäle verschickt. Wie das genau funktioniert, wird im Folgenden erklärt.

## **5.2 Ameisen als Vorbild**

Um aus der Natur neue Technologien zu bekommen, muss man genau beobachten. Ameisen laufen ohne eine für uns sichtbare Ordnung kreuz und quer durch die Gegend. Und doch, wenn erstmal eine Futterquelle gefunden ist, bildet sich sehr schnell eine Ameisenstraße, die sich dann auch nichtmehr durch Hindernisse aufhalten lässt. Eine Unterbrechung wird schnell umgangen und die Ameisen finden trotz ihrer winzigen Gehirne schnell den kürzesten Weg. Doch wie machen die Ameisen das?

Eine Ameise hinterlässt auf ihrem Weg eine Pheromonspur. Damit kann sie den Rückweg finden, gibt aber auch anderen Ameisen den Weg vor. Eine Ameise wird, mit einer gewissen Wahrscheinlichkeit, dieser Pheromonspur folgen und dabei diese Spur durch eigene Duftstoffe verstärken. Damit werden immer mehr Ameisen auf diese Spur gelenkt, denn die Wahrscheinlichkeit, einen gewissen Weg zu nehmen, wird von der Stärke der Pheromonspur vorgegeben. So bilden schnell sogenannte Ameisenstraßen, wie man sie oft an Bäumen, im Wald oder in der Küche beobachten kann.

Wenn ein Hindernis die Ameisenstraße blockiert, zum Beispiel ein heruntergefallener Ast, finden die Ameisen nach kurzer Zeit einen Weg um das Hindernis herum. Das erstaunliche dabei ist, dass es sich dabei meist um den kürzesten Weg handelt. Dieses Phänomen wird mit folgenden Beispiel anschaulich erklärt:

Ein Hindernis bietet meist (mindestens) zwei Wege, um es zu umgehen: einen kurzen Weg und einen (oder mehrere) langen Weg. Etwa die Hälfte der Ameisen werden den kurzen Weg wählen, die andere Hälfte den langen Weg, da beide Wege noch nicht durch Pheromone verstärkt sind. Da die Ameisen auf dem kurzen Weg jedoch schneller ihr Ziel erreichen und somit auch wieder schneller auf dem Heimweg sind, wird der kurze Weg in der gleichen Zeit durch mehr Pheromone markiert als der längere Weg. Das bewirkt eine erhöhte Wahrscheinlichkeit für eine Wahl des kürzeren Weges, das wiederum zu einer Verstärkung führt. Da auf beiden Wegen die Pheromonspur der Ameisen langsam verdunstet, diese auf dem kurzen Weg jedoch ständig erneuert wird, wird dieser bald zur neuen Ameisenstraße und der lange Weg wird kaum noch

beachtet.

Das Internet, mit seinen vielen möglichen Wegen stellt den Datenpaketen ein ähnliches Problem wie den Ameisen: Wie finde ich den kürzesten Weg und was tun wenn der Weg unterbrochen ist?

## 5.3 Travelling Salesmann Problem

Das Verhalten der Ameisen stellt eine interessante Variante dar, um möglichst kurze Wege zwischen zwei Punkten zu finden. Es lassen sich viele Probleme finden, die sich durch Ameisen lösen ließen. Dazu muss man nur das Verhalten der Ameisen in einen Algorithmus umsetzen und dann auf das Problem anpassen. Eines dieser möglichen Probleme ist das Travelling Salesman Problem (TSP).

### 5.3.1 Problembeschreibung

Das Travelling Salesman Problem (reisender Handelsverteter) lässt sich einfach erklären. Ein Handelsvertreter will eine bestimmte Anzahl an Städten besuchen und dabei so wenig wie möglich wenig autofahren. Dazu braucht er die kürzeste Verbindung zwischen allen Städten.

Schematisch dargestellt sucht man einen möglichst kurzen geschlossenen Kantenzug, in dem jeder Punkt  $P$  des Graphen  $G$  genau einmal vorkommt (Hamiltonscher Graph).

Der Aufwand, dieses Problem exakt zu lösen, ist hängt von der Anzahl der zu besuchenden Städte ab. Der Aufwand beträgt  $O(n!)$ , wenn man stur jede mögliche Kombination der Wege ausprobiert.

Mit deutlich weniger Aufwand kommen agentenbasierte System zu guten Lösungen, jedoch nicht zur Ideallösung. Je mehr Durchläufe die agentenbasierten Systeme machen, desto besser werden ihre Ergebnisse und nähern sich immer weiter an die Ideallösung an.

### 5.3.2 Agentenbasiertes Lösungsverfahren

Dieses System setzt Agenten (künstliche Ameisen) ein, um eine Lösung für das TSP zu finden. Zu Beginn werden  $m$  Agenten auf zufällig ausgewählte Städte gesetzt. In jedem Durchgang bewegen sich die Agenten eine Stadt weiter. Die Auswahl der Stadt hängt von der Entfernung und von der Konzentration künstlicher Pheromone ab. Kurze Verbindungen haben eine höhere Wahrscheinlichkeit als lange. Nachdem sich jeder Agent einmal bewegt hat, markieren diese ihre gewählten Wege. Die Menge an hinterlassenen Pheromonen ist indirekt proportional zur Weglänge. Zusätzlich wird nach jedem Durchgang der kürzeste Weg von allen Ameisen noch einmal mit künstlichen Pheromonen verstärkt.

Man sieht an diesem Algorithmus Ähnlichkeiten aber auch Unterschiede zu den realen Ameisen. Die Agenten kommunizieren indirekt, nur über die Pheromonstärke der Wege, und sie wählen bevorzugt Wege mit hoher Konzentration. Allerdings kennen die Agenten im voraus die Weglänge zwischen zwei Städten und nutzen dies als weiteres Kriterium zur Wegewahl und die Agenten haben ein Gedächtnis, das bereits besuchte Städte speichert.

Die Lösung setzt sich aus den Einzellösungen der Ameisen zusammen. Man folgt immer den

Weg mit der höchsten Pheromonkonzentration zu einer noch nicht besuchten Stadt.

### 5.3.3 Vergleich mit Greedy Algorithmus

Ein Greedy-Algorithmus (greedy bedeutet gierig) 'frisst' sich durch den Graphen, indem es an jeder Abzweigung den kürzesten zur Verfügung stehenden Weg wählt.

Auf dem ersten Blick funktioniert der agentenbasierte Algorithmus genau wie das Greedy-Verfahren. Der Unterschied liegt darin, dass es mehrere Agenten gibt, und nicht nur einen Ablauf des Algorithmus. Ausserdem wählen die Agenten nicht zwangsweise den kürzesten Weg zwischen zwei Punkten, im Gegensatz zum Greedy Algorithmus. Abhängig von der Anzahl und der Agenten und der Laufzeit des Algorithmus, sind die agentenbasierte Ergebnisse deutlich besser als die des recht einfachen Greedy-Algorithmus.

## 5.4 AntNet

Effektives Routing wird durch verteilte Router, die eine durch eine hohe Redundanz gesichert werden und eine gute Fehlertoleranz haben, erreicht. Das AntNet, basierend auf einem Multi-Agenten System, bietet genau diese Möglichkeiten. Die Idee zum AntNet entstand aus weiteren Problemen, die durch ein Multi-Agenten System gelöst wurden. Der zugrundeliegende Gedanke ist ein System, das immer wieder Minimallösungen von Teilproblemen sucht und diese zu einer Gesamtlösung zusammensetzt.

Das AntNet sendet dazu seine Agenten zu ihren Zielorten aus. Auf dem Weg sammeln diese Informationen über den gewählten Weg und aktualisieren auf dem Rückweg die Routingtabellen der dazwischenliegenden Router.

### 5.4.1 Routingtabelle des AntNet

Im Gegensatz zu herkömmlichen Routern besitzen Router des AntNet eine veränderte Routingtabelle und zusätzlich ein Array.

1. Die Routing Tabelle  $T$  ist wie eine normale Distanzvektortabelle der herkömmlichen Verfahren aufgebaut, mit dem Unterschied, dass in der Tabelle Wahrscheinlichkeitseinträge sind. Für jeden Zielknoten  $d$  wird in der Tabelle für jeden möglichen Weg  $n$  (also alle Nachbarn des Routers) die Wahrscheinlichkeit  $P_n$  gespeichert, mit der dieser Weg gewählt werden soll. Je schneller dieser Weg an das gewünschte Ziel führt, desto höher ist die Wahrscheinlichkeit dass dieser Weg gewählt wird.

$$\sum_{n \in N_k} P_{nd} = 1, d \in [1, N], N_k = \{Nachbar(k)\}, N = \{Netzwerkknoten\} \quad (5.1)$$

Die Summe der Wahrscheinlichkeiten um zu einem Punkt  $d$  über alle möglichen Wege  $N_k$  ist (natürlich) 1.

2. Die Liste enthält für jeden Knoten ein Array  $M_k(\mu_d, \sigma_d^2, W_d)$ , das einfache statistische Daten über den Datenverkehr des Netzwerks aus Sicht des lokalen Knotens  $k$  enthält. Der Mittelwert und die Varianz werden durch die Laufzeiten der Agenten beeinflusst.  $W_d$  speichert die beste Laufzeit innerhalb eines bestimmten Beobachtungsfensters (zum

Beispiel immer nur die fünf letzten Agenten). Das Beobachtungsfenster wird deswegen beschränkt, damit nicht alte und nichtmehr aktuelle Daten Einfluss auf Routingentscheidungen haben. Der Mittelwert  $\mu_d$  gibt die erwartete Laufzeit zu dem Zielknotenpunkt an und die Varianz  $\sigma_d^2$  gibt Hinweise über die Stabilität der Verbindung. Je höher die Varianz desto unterschiedlicher waren die Laufzeiten zum Zielknoten. Ein Grund hierfür können Leitungs- und Serverausfälle sein.

Während der Entwicklung des AntNet hat sich folgendes Modell am besten bewährt:

$$\mu_d \leftarrow \mu_d + \eta(o_{k \rightarrow d} - \mu_d) \quad (5.2)$$

$$\sigma_d^2 \leftarrow \sigma_d^2 + \eta((o_{k \rightarrow d} - \mu_d)^2 - \sigma_d^2) \quad (5.3)$$

wobei  $o_{k \rightarrow d}$  die neue Laufzeit zwischen dem Knoten  $k$  und seinem Ziel  $d$  ist. Der Faktor  $\eta$  dient zur Relativierung der neuen Messwerte zu den zu den bisherigen Werten und ist vom Beobachtungsfenster abhängig.

Die Tabelle  $T$  und die Datenstruktur  $M$  speichern verschiedene Ansichten des Netzwerks. In  $M$  wird die Entfernung/Zeit zum Ziel gespeichert, in  $T$  werden die verschiedenen Wege zum Ziel anhand der Wahrscheinlichkeit bewertet.

## 5.4.2 Routing im AntNet

Herkömmliche Routingmethoden sind vorhersehbar, sofern sich am Netzstatus nichts ändert. Die Routingentscheidungen werden Anhand der Tabelleneinträge festgelegt. Zwar werden bei adaptiven Routingmethoden, die ihre Entscheidungen auch von der Netzlast abhängig machen, Datenpakete auf verschiedene Wege geschickt, jedoch auch diese sind vorhersehbar.

Das AntNet unterscheidet sich in dieser Hinsicht sehr von anderen Routing Algorithmen. An jedem Zwischenknoten wird eine neue Zufallsentscheidung anhand der Routingtabelle getroffen. Kein Weg der Datenpakete ist vorhersehbar. Zwar ist es am wahrscheinlichsten, dass nur die besten Wege gewählt werden, allerdings wird dies in nicht garantiert.

Ein einfaches Beispiel zeigt dies deutlich: In einem Beispielnetzwerk sind zwischen dem Anfangs- und Endknoten vier Zwischenknoten, es müssen also fünf Routing entscheidungen getroffen werden. Wenn die direkten Wege (also die kürzesten Wege) jeweils eine Wahrscheinlichkeit von 90% haben, so reduziert sich die Wahrscheinlichkeit, dass ein Datenpaket den schnellsten Weg nimmt auf etwa 60%.

$$0.9^5 = 0.59 \quad (5.4)$$

Das zufallsgesteuerte Verhalten der Router lässt den Fall zu, dass Datenpakete im Kreis geroutet werden, ohne dass sie jemals am Ziel ankommen. Dieser, sehr unwahrscheinliche, Fall wird dadurch reduziert, dass die Agenten, welche Daten für die Routingtabellen sammeln, ebenfalls in solche Schleifen geraten können, diese Situation erkennen und die Wahrscheinlichkeiten für diesen Fall senken.

## 5.4.3 Algorithmus des AntNet

Das AntNet ist abhängig von seinen Agenten. Diese wandern durch das Netz und sammeln Daten über die Netzstruktur, Auslastung und Datenverkehr. Mit diesen Daten werden die Rou-

tingtabellen aktualisiert. Im Folgenden ist der der Lebenszyklus eines einzelnen Agenten kurz beschrieben:

- In kurzen Zeitabständen werden von jedem Router mobile Agenten zu einem zufällig ausgewählten Ziel losgeschickt.
- Die Agenten verhalten sich konkurrierend und unabhängig. Sie kommunizieren nur indirekt durch die Informationen, mit denen sie die Routing Tabellen aktualisieren.
- Jeder einzelne Agent sucht einen Minimalpfad zwischen seiner Quelle und seinem Ziel.
- Jeder Agent bewegt sich Schritt für Schritt zu seinem Zielpunkt. In jedem Zwischenknoten entscheidet eine einfache stochastische Entscheidung über den nächsten Schritt. Für diese Entscheidung wird auf (i) lokale Informationen (ii) private Informationen des Agenten zurückgegriffen.
- Auf seinem Weg sammelt der Agent über die Dauer, die Datenansammlung am Knotenpunkt und die Kennung des Routers.
- Sobald der Agent den Zielpunkt erreicht hat, kehrt dieser um und nutzt denselben Weg zurück zum Startserver.
- Auf dem Rückweg werden die lokalen Routingtabellen jedes Knotenpunkts mit den gesammelten Daten aktualisiert.
- Am Ende seines Weges stirbt der Agent.

Im Folgenden werden die einzelnen Punkte näher erläutert.

1. In regelmäßigen Zeitabständen  $\Delta t$  werden von jedem Netzwerkknoten  $s$  einen vorwärtsgerichteten Agenten  $F_{s \rightarrow d}$  zu einem Zielknoten  $d$  ausgeschiedt um einen einfachen und schnellen Weg zwischen den beiden zu finden. Die vorwärtsgerichteten Agenten haben die gleiche Priorität wie normale Datenpakete, um denselben Verzögerungsbedingungen in Warteschlangen ausgesetzt zu sein. Die Wahrscheinlichkeit, dass ein Agent  $F_{s \rightarrow d}$  zu einem bestimmten Zielpunkt  $d$  losgeschickt wird, hängt vom lokalen Datenverkehr ab. Je mehr Datenpakete zu dem Knoten  $d$  losgeschickt werden, desto höher ist die Wahrscheinlichkeit, dass ein Agent zu diesem Knoten ausgesendet wird. Wenn  $f_{sd}$  die Anzahl der Pakete des Datenstroms  $s \rightarrow d$ , dann ist die Wahrscheinlichkeit für einen vorwärtsgerichteten Agenten nach  $d$ :

$$p_d = \frac{f_{sd}}{\sum_{d'=1}^N f_{sd'}} \quad (5.5)$$

2. Auf ihrem Weg speichern die Agenten den genauen Pfad mit jedem Zwischenknoten, der aktuellen Datenverkehrslast und der bereits vergangenen Reisezeit. Diese Informationen werden auf einem Stack  $S_{s \rightarrow d}(k)$  gespeichert.
3. An jedem Knoten wählt der Agent zufällig den nächsten Schritt aus. Dabei werden nur Knoten gewählt, die noch nicht besucht wurden. Wenn dies nicht möglich ist, wählt der Agent seinen nächsten Schritt aus den bereits besuchten Nachbarn seinen nächsten Schritt aus.
4. Wenn der Agent eine Schleife bemerkt, also er einen Knotenpunkt zweimal besucht, wird die gesamte Schleife vom Stack des Agenten gelöscht. Wenn die Schleife größer als die halbe Lebenszeit des Agenten ist, wird die Ameise zerstört, da der Weg anscheinend aufgrund schlechter (Zufalls-) Entscheidungen falsch war. Damit wird verhindert, dass die Routingtabellen mit falschen Informationen erneuert werden.
5. An ihrem Zielort wird ein neuer, rückwärtsgerichteter Agent erzeugt  $B_{d \rightarrow s}$  erzeugt. Der ankommende Agent überträgt dem neuen Agenten den kompletten Stack  $S_{s \rightarrow d}(k)$  und stirbt dann.
6. Der rückwärtsgerichtete Agent  $B_{d \rightarrow s}$  nimmt denselben Weg zurück, jedoch mit maximaler Priorität, um nicht durch Datenpakete aufgehalten zu werden. Seine Aufgabe ist es, seine Informationen so schnell wie möglich an alle Knotenpunkte zu liefern.
7. Bei jedem Knoten auf dem Rückweg aktualisiert der Agent die Routingtabelle  $Z_k$  und das Array  $M_k$ . Alle Einträge mit Bezug auf den ursprünglichen Zielort werden neu berechnet und, mit gewissen Vorsichtsmaßnahmen, auch alle Einträge zu den Knoten zwischen den den Zielknoten und dem momentanen Aufenthaltsort. Da die Ergebnisse für die Zwischenknoten nur ein Seiteneffekt sind und damit das eigentliche Ziel, den kürzesten Pfad zu finden, nicht perfekt erfüllen, werden nur die besten Ergebnisse verwertet. Schlechte Wegzeiten werden ignoriert und nur am Startpunkt des Agenten in die Tabellen einberechnet.
  - a) Die Liste  $M_k$  wird mit den Werten des Stack  $S_{s \rightarrow d}(k)$  aktualisiert.  $\mu_d$  und  $\sigma_d^2$  werden neu berechnet (siehe Formeln 4.2 und 4.3) und  $W_d$  wird neu bestimmt.
  - b) In der Routing Tabelle  $T_k$  wird die Wahrscheinlichkeit  $P_{fd'}$  erhöht (das ist die Wahrscheinlichkeit um den Nachbar  $f$  zu wählen wenn der Zielort  $d'$  ist) und gleichzeitig werden alle anderen Wahrscheinlichkeiten  $P_{nd'}$  gesenkt. Der Betrag dieser Änderung hängt von der Qualität des gefundenen Weges ab.

$$P_{fd'} \leftarrow P_{fd'} + r(1 - P_{fd'}) \quad (5.6)$$

$$P_{nd'} \leftarrow P_{nd'} - rP_{nd'}, n \in N_k, n \neq f \quad (5.7)$$

Formel 4.6 gibt die Erhöhung der Wahrscheinlichkeit für den gefundenen Weg nach  $d'$  über den Nachbarknoten  $f$  an. Je geringer die Ausgangswahrscheinlichkeit, desto größer ist die Verstärkung. Dadurch werden neue Wege schnell in das System integriert.

Formel 4.7 gibt die Senkung aller anderen Wahrscheinlichkeitseinträge der Tabelle



$T_k$  zu dem Zielknoten  $d'$  an.

Die Summe der Wahrscheinlichkeiten bleibt weiterhin 1 (siehe Formel 4.1).

Die Ankunftsrate hat natürlich ebenso Einfluss auf die Wahrscheinlichkeitsverteilung. Agenten, die schlechte Wege wählen oder sogar in Schleifen hängen bleiben, werden nach Ablauf ihres TTL (time to live) sterben und so den gefundenen Weg nicht verstärken. So werden schlechte Wege ausgesondert, indem sie im Gegensatz zu mittelmäßigen und guten Wegen nicht verstärkt werden.

Die Erhöhung/Senkung der Wahrscheinlichkeit hängt schließlich noch von dem Faktor  $r$  ab. Dieser wird von der Qualität des gefundenen Weges vorgegeben. Das bewirkt, dass gute Wege einen größeren Bonus erhalten als mittelmäßige Wege.

#### 5.4.4 Bewertung gefundener Wege

Um die Wahrscheinlichkeiten neu zu berechnen, muss man die neuen Messwerte der Agenten irgendwie mit den Werten anderer Wege vergleichen.

- Die einfachste Methode ist es, den Faktor  $r$  konstant zu lassen. Dadurch wird keinerlei Unterscheidung zwischen den Wegen getroffen, alle erhalten die gleiche Verstärkung. Trotzdem stellt sich eine Differenzierung zwischen guten und schlechten Wegen ein. Das liegt im Folgenden begründet:
  - Agenten auf guten Wegen kehren schneller zurück als andere. Damit erhöhen sie diese Wahrscheinlichkeit geringfügig früher als andere Agenten. Wenn nun während dieser Zeitspanne neue Agenten ausgesickt werden, wählen sie mit einer leicht erhöhten Wahrscheinlichkeit den besseren Weg und verstärken diesen damit weiter. Dieser Effekt wirkt sich nur sehr gering auf das System aus.
  - Agenten auf schlechten Wegen haben eine höhere Wahrscheinlichkeit, ihr Ziel nicht oder zu spät zu erreichen. In diesem Fall wird ihr Weg nicht positiv verstärkt, im Gegensatz zu den Wegen mit besseren Ankunftsdaten. Dieser Effekt ist deutlicher zu spüren.

Diese Methode liefert im Vergleich zu anderen Routing Methoden nur durchschnittliche Ergebnisse und ist nicht zu empfehlen.

- Der Faktor  $r$  wird von der Laufzeit  $T$  des Agenten und den stochastischen Daten  $M_k$  abhängig gemacht. Dabei hat sich folgende Variante am effektivsten herausgestellt:

$$r = c_1 \left( \frac{W_{best}}{T} \right) + c_2 \left( \frac{I_{sup} - I_{inf}}{(I_{sup} - I_{inf}) + (T - I_{inf})} \right) \quad (5.8)$$

$W_{best}$  ist dabei die beste Wegzeit innerhalb des Beobachtungsfensters (der letzten  $n$  Versuche).

$I_{sup}$  und  $I_{inf}$  sind die Grenzen eines Konfidenzintervalls um den Mittelwert  $\mu$ .  $I_{inf}$  wird

auf den Wert von  $W_{best}$  gesetzt.

$I_{sup} = \mu + z(\sigma/\sqrt{|W|})$  mit  $z = 1/\sqrt{(1-\gamma)}$  und  $\gamma$  dem gewählten Konvidenzintervall.

Die Formel 4.8 besteht aus 2 Summanden. Der erste Summand gibt das Verhältnis von bester Laufzeit zur gemessenen Laufzeit wieder. Der zweite Summand die gemessene Laufzeit in das richtige Verhältnis zum Konvidenzintervall setzt. Dabei wird die Varianz der Laufzeiten beachtet, da dies ein Indikator für die Stabilität der gefundenen Wege ist.

Die Korrekturfaktoren  $c_1$  und  $c_2$  setzen die beiden Summanden ins richtige verhältnis zueinander. Da der erste Summand eine größere Rolle spielt, wird dieser etwa höher bewertet. In den Testfällen wurde für das AntNet  $c_1 = 0.7$  und  $c_2 = 0.3$  verwendet.

Die besten Ergebnisse für das Antnet werden erzielt, wenn das Konvidenzintervall  $\gamma$  zwischen 75% und 80% und  $c_2$  zwischen 0.15 und 0.35 liegt.

## 5.5 Vergleich zwischen AntNet und anderen Routingmethoden

Der Algorithmus des AntNet zeigt in der Theorie viele Vorteile. Diese müssen jedoch erst durch Simulationen verifiziert werden. Marco Dorigo und Gianni Di Caro, die beiden Erfinder und Entwickler der Antnets haben dazu eine Netzwerksimulation in C++ geschrieben und darin das AntNet sowie sechs weitere Routing Algorithmen nachprogrammiert, um diese zu testen.

### 5.5.1 Vergleichsalgorithmen

Um ein möglichst breites Spektrum an Vergleichsdaten zu erhalten, wurden folgende Routingalgorithmen in der Netzwerksimulation integriert:

- **OSPF** (open shortest path first, statisches link-state Protokoll)
- **SPF** (shortest path first, adaptives link-state Protokoll)
- **BF** (Bellmann-Ford, adaptives distanz-vektor Protokoll)
- **Q-R** (Q-Routing, adaptives distanz-vektor Protokoll)
- **PQ-R** (Predictive Q-Routing, adaptives distanz-vektor Protokoll)
- **Daemon** (adaptiv, optimales Routing. Künstlicher Algorithmus, der zu jeder Zeit den kompletten Netzwerkzustand kennt und deswegen immer den idealen Weg findet. Dient nur zum Vergleich, um zu sehen wie nah die restlichen Algorithmen an den Idealzustand kommen.)

### 5.5.2 Simulierte Netzwerke

Da Vergleichsdaten erst aussagekräftig werden wenn die Algorithmen an verschiedenen Netzwerken mit verschiedensten Netzlasten getestet wurden, wurden drei verschiedene Netze für die Simulation nachgebildet.

1. **SimpleNet** ist ein kleines Netzwerk mit nur acht Knoten und neun Verbindungen. Es wird mit einer Bandbreite von 10MBit/s simuliert.
2. **NFSNET** ist das US Backbone von 1987. Es besteht aus 14 Knoten mit 21 Verbindungen und hat eine Bandbreite von 1.5MBit/s.
3. **NTTnet** ist das Haupt-Backbone Japans aus 57 Knoten, die über 162 Kanten verbunden sind. Es hat eine Bandbreite von 6MBit/s

### 5.5.3 Simulierte Netzlasten

Die vorgestellten Netzwerke werden durch verschiedene, zufällig generierte Daten belastet. Die Datenerzeugung ist von zwei Faktoren abhängig: Ort und Zeit.

Örtliche Verteilungen:

- Gleichverteilt (U): bei allen Netzwerkknoten werden gleichmäßig viele Sitzungen angefordert.
- Zufallsverteilt (R): Alle Anforderungen für eine Sitzung sind an zufällig ausgewählte Knoten gerichtet.

Zeitliche Verteilungen:

- Poisson (P): für jeden Knoten gibt eine Poissonverteilung die Ankunft neuer Sitzungen vor.
- Fix (F): Zu Beginn der Simulation wird für jeden Knoten ein fester Wert festgelegt, der die Anzahl der Sitzungen bestimmt.
- Kurzfristig (TMPHS): eine kurzfristige Spitzenbelastung des Netzwerks wird an einigen Knoten (Hot Spots) erzeugt).

Die Datenmengen, die in einer Sitzung versendet werden, können konstant und variabel sein.

## AntNet

- Konstante Bitrate (CBR): es werden damit Übertragungen wie Videostreams und Telefongespräche simuliert.
- Variable Bitrate (GVBR): der erzeugte Datenstrom schwankt um einen bekannten Erwartungswert.

Diese unterschiedlichen Faktoren lassen sich beliebig kombinieren und decken insgesamt ein möglichst weites Spektrum an möglichen Situationen ab. Jede gewöhnliche Netzbelastung sollte damit in der Simulation getestet werden können.

### 5.5.4 Ergebnisse

Es werden zwei Metriken zur Bestimmung der Leistungsfähigkeit definiert: Durchsatz und Verzögerung. Der Durchsatz gibt die korrekt zugestellte Datenmenge pro Sekunde an (Bit/s), während die Verzögerung den zeitlichen Abstand von 90% der Datenpakete angibt.

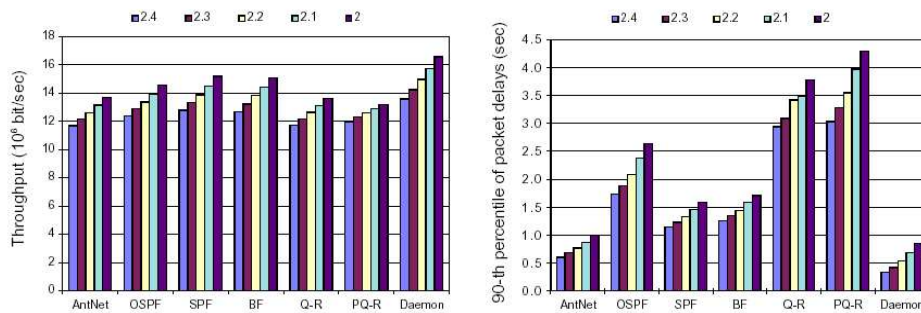
Die Ergebnisse der Simulationen sind stark abhängig von der allgemeinen Netzlast. Bei geringer Datenlast zeigen alle Algorithmen gleich gute Ergebnisse. Erst bei einer hohen Datenlast werden die Unterschiede der Algorithmen deutlich. Der Durchsatz bleibt bei allen Algorithmen ähnlich, jedoch zeigt die Verzögerung der Pakete deutliche Unterschiede.

### SimpleNet

Die Experimente mit dem SimpleNet wurden gemacht um die genaue Lastverteilung der einzelnen Verfahren genauer zu analysieren. Es wurde hierfür eine fixe Verteilung mit konstanter Bitrate gewählt. Die erzeugte Datenlast wurde so hoch angesetzt, dass ein effektives Routing über nur einer Leitung nichtmehr möglich war. Dabei zeigte das AntNet Ergebnisse im Durchsatz, die sehr nahe am Daemon Algorithmus waren. Zweitbesten Algorithmus war PQ-R, alle anderen waren durchschnittlich 30% schlechter als das AntNet. Das liegt daran, dass Aufgrund der geringen Größe des Netzes die Datenlast vom AntNet gleichmäßig verteilt werden kann, ohne dass dabei schlechte Wege gewählt werden können.

### NFSNET

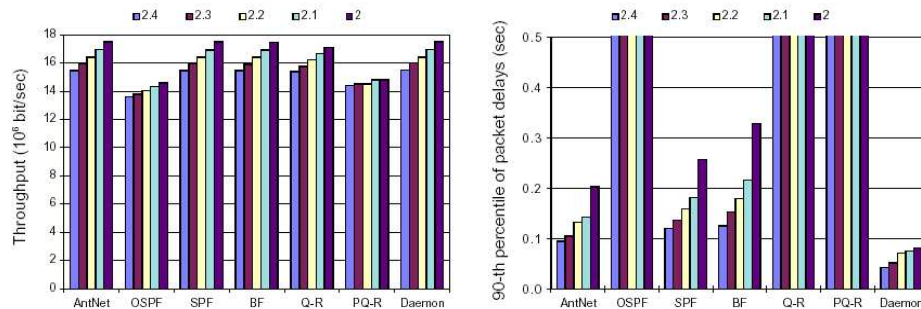
Mit dem NFSNET wurden mehr Testdurchläufe als mit dem SimpleNet gemacht. UP, RP, UP-HS und TMPHS-UP Verteilungen wurden mit unterschiedlichen Netzlasten kombiniert. Bei der UP Verteilung (lokal gleichverteilt und zeitlich poissonverteilt) haben BF und SPF einen besseren Durchsatz als die restlichen Varianten. Dieses Ergebnis ändert sich jedoch wenn man die Verzögerung miteinbezieht. Dort zeigt das AntNet die besten Werte, deutlich besser als alle anderen Ergebnisse.



Die linke Tabelle zeigt den Datendurchsatz bei UP-Datenverkehr, die rechte die Verzögerung der Pakete an. Für jeden Algorithmus sind vier Werte von 2.4s bis 2.0s angegeben. Diese stehen für die mittleren Abstände zwischen zwei Sitzungen (bei einem gleichzeitigen mittleren Abstand von 0.005s zwischen zwei Datenpaketen)

Bei einer RP Verteilung ähneln die Ergebnisse des Datendurchsatzes von AntNet, SPF und BF. Jedoch bei der Verzögerung zeigt das AntNet (wieder) die besten Ergebnisse aller Algorithmen. Allerdings ist selbst das AntNet deutlich schlechter als der Daemon Algorithmus, bedingt durch die schweren Testbedingungen.

Im letzten, den UP-HS Fall zeigen OSPF und PQ-R die schlechtesten Ergebnisse im Durchsatz. Extrem werden die Ergebnisse der Verzögerungen. Keiner der Algorithmen schneidet (im Vergleich zum Daemon) gut ab. Trotzdem sind die Unterschiede zwischen den Algorithmen gravierend. Wie erwartet ist das AntNet weiterhin besser als restlichen Routingmethoden.

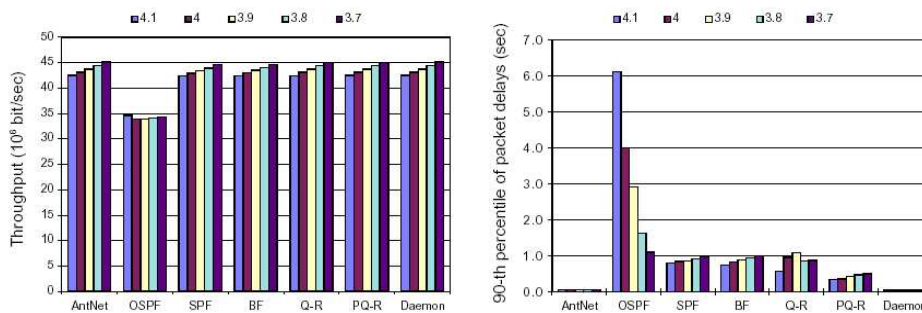


UP-HS (gleichmäßige Poissonverteilung mit Hot-Spots)

## NTTnet

Die selben Experimente wie im NFSNET wurden im NTTnet wiederholt. Diese zeigten noch deutlicher, dass das AntNet am leistungsfähigsten von allen Algorithmen ist.

In allen Fällen wird deutlich, dass die Unterschiede im Datendurchsatz gering sind und nur OSPF mit leicht schlechterer Leistung auffällt.



*NTTnet mit UP-HS Datenverkehr (gleichmäßige Poissonverteilung mit Hot-Spots)*

Als Zusammenfassung kann man sagen, dass AntNet unter fast allen Bedingungen die beste Leistung gezeigt hat. Dabei waren die Ergebnisse sehr dicht am Daemon Algorithmus, also fast perfektes Routing.

### 5.5.5 Zusammenfassung

Das AntNet bietet mehrere Vorteile gegenüber seinen Mitstreitern. Aufgrund seiner Arbeitsweise sammelt es mehr Informationen im Netz als andere Algorithmen und kann dementsprechend seine Entscheidungen aufgrund eines genaueren Lagebildes besser treffen. Weiterhin ist das AntNet weniger fehleranfällig, da es sich nicht auf Informationen, die es von Nachbarknoten bekommt und die potentiell falsch sein können, verlassen muss, sondern wird durch Agenten, die auf dem neuesten Stand sind, aktualisiert. So können sich keine Fehler ausbreiten, denn falsche Annahmen an einem Router werden dadurch eliminiert, dass Agenten die aufgrund dieser falschen Annahmen geroutet werden ihr Ziel nicht erreichen und damit die falschen Informationen auch nicht verbreiten.

Ein weiterer Vorteil liegt in der Wahrscheinlichkeitsverteilung in der Routing Tabelle. Dies wirkt sich bei geringer Datenlast kaum aus, bei einem starken Netzwerkverkehr jedoch werden die Leitungen gemäß ihrer Qualität benützt. Das bewirkt eine gleichmäßige Lastverteilung, die bei anderen Routingmethoden mit deterministischer Tabelle nicht möglich ist.

## 5.6 Schluss

Das AntNet bietet eine ungewohnte Sicht auf das Routing. Bisher war man gewohnt, dass der Router alle Pakete einer Sitzung über einen Weg schickt (sofern dieser Weg nicht unterbrochen wird). Irritierend ist zunächst dass jedes Paket einen zufälligen Weg nimmt. Man ist verwundert, dass die Pakete trotzdem, zum Teil deutlich schneller, an ihr Ziel finden, obwohl sie vom direkten Weg abweichen.

### 5.6.1 Einordnung in den Kontext des Seminars

Das agentenbasierte Routing ist ein neuer Aspekt des Internet Routings. Es trifft genau auf die Fragestellung des Seminars „Bewertung von Internet Routing Strategien“, da es als neuartiger Algorithmus in seiner Leistungsfähigkeit mit anderen Verfahren verglichen werden muss. Dorigo und Di Caro haben für ihre Experimente verschiedene Normen festgelegt, um Leistungsfähigkeit und Zuverlässigkeit zu messen. Ihr Hauptaugenmerk lag zwar im Bereich der

Performance, jedoch zeigten sie nebenbei auch die Zuverlässigkeit des AntNet Algorithmus. Ihre Ergebnisse sprechen für das AntNet. Obwohl andere Routing Algorithmen leicht bessere Ergebnisse bei geringem Datenverkehr haben, spielt das AntNet als bekanntester Vertreter für agentenbasierte Systeme bei hohem Datenaufkommen seine Stärken aus.

### **5.6.2 Ergebnisse und Ausblicke**

Der Nachteil des AntNet ist sein Vorteil. Es verteilt die Datenpakete auf verschiedene Wege. Bei geringer Netzlast werden dann „gute“ Wege nicht voll ausgenutzt, was zu Einbußen führt. Aber bei starker Netzlast werden eben diese guten Wege nicht überladen, was einen Geschwindigkeitsvorteil bringt.

Noch hat sich das AntNet nicht im realen Einsatz bewährt, die Ergebnisse wurden bisher nur durch Experimenten ermittelt. Ob es sich durchsetzen wird ist weiterhin eine offene Frage. Bisher sind die Netze nicht so überlastet, dass man zu Veränderungen gezwungen ist. Weiterhin ist es für Netzadministratoren wichtig, ihr Subnetz am laufen zu halten. Der Umstieg auf ein ihnen unbekanntes, kaum getestetes Routingverfahren ist vielen zu unsicher und mit zuviel Arbeit verbunden.

Das AntNet wurde von Dorigo und Di Caro zwar sehr weit entwickelt und verfeinert, jedoch ist es sicher noch nicht ausgereizt. Eine Idee wäre es zum Beispiel dass vorwärtsgerichtete Agenten wie echte Ameisen Datenpakete mitnehmen. Damit würde man sich mehr an das Vorbild annähern. Gedanklich besser wäre es, wenn Datenpakete die vorwärtsgerichteten Agenten huckepack mitnehmen. Das würde besser zur bisherigen Erzeugungsfunktion der Agenten passen (je mehr Datenpakete zu einem Ziel geschickt werden desto mehr Agenten werden in diese Richtung geschickt).

Meiner Meinung nach bietet das AntNet sehr interessante Ansätze, Probleme leicht und effektiv zu lösen. Auch wenn es wahrscheinlich nie in den realen Einsatz kommt, sollte man sich zumindest mit der allgemeinen Funktionsweise vertraut machen.

# Literaturverzeichnis

- [1] [CaDo1] G. DI CARO, M. DORIGO, „*Ant Colonies for Adaptive Routing in Packet-switched Communication Networks*“ in *Parallel Problem Solving from Nature - PPSN V - 5th International Conference, Amsterdam, September 1998, Springer Verlag, Lecture Notes in Computer Science, Vol 1498*
- [2] [CaDo2] G. DI CARO, M. DORIGO, „*Antnet: Distributed Stigmergetic Control for Communications Networks*“ in *Journal of Artificial Intelligence Research 9, 1998*, <http://iridia.ulb.ac.be/gdicaro/papers/jair.ps.gz> (Link vom 09.07.02)
- [3] [Unb1] UNBEKANNT, „*Ant Colony Optimisation*“ Artikel von erstellt von Phil Grant am 25.04.02, <http://www.wins.uva.nl/arnoud/OOAS/fwi/Chap12.ps.gz> (Link vom 09.07.02)
- [4] [ABe1] AXEL BECKERT, „*Staatenbildende Insekten als Vorbilder für Software-Agenten*“ Seminarvortrag Bionik Sommersemester 2000, <http://fsinfo.cs.uni-sb.de/abe/w5/bionik/ACO-Seminar-Vortrag.pdf> (Link vom 09.07.02)
- [5] [Unb2] UNBEKANNT, „*Open Shortest Path First*“ 2002, [http://www.cisco.com/univercd/cc/td/doc/cisintwk/ito\\_doc/ospf.htm](http://www.cisco.com/univercd/cc/td/doc/cisintwk/ito_doc/ospf.htm) (Link vom 09.07.02), Seite vom 20.02.2002
- [6] [ABo1] ANDREAS BORCHERT, „*Rechenaufwand beim Problem des Travelling Salesman*“ 1999, <http://www.mathematik.uni-ulm.de/sai/ss99/prog/skript/backtracking-19.html> (Link vom 09.07.02), Seite vom 29.06.1999
- [7] [JFj1] JOHANN BRAGI FJALLDAL, „*An introduction to Ant Colony Algorithms*“ 1999, <http://www.cogs.susx.ac.uk/lab/nlp/gazdar/teach/atc/1999/web/johannf/tsp.html> (Link vom 09.07.02), Seite vom 15.02.1999



# 6 Leistungsmodellierung

*Andreas Schäfer*

## Inhaltsverzeichnis

---

<b>6.1 Zielsetzung dieses Beitrags</b>	<b>114</b>
6.1.1 Leistungsmodellierung	114
6.1.2 Zuverlässigkeitsmodellierung	115
6.1.3 Performability-Modellierung	115
6.1.4 Kapitelübersicht	115
<b>6.2 Leistung und Leistungsmodellierung</b>	<b>116</b>
6.2.1 Leistung: Metriken und deren Analyse	116
6.2.2 Techniken der Leistungsanalyse	121
<b>6.3 Modelle und Modellbildung</b>	<b>123</b>
6.3.1 Modelle	123
6.3.2 Modellbildung und Modellbildungsprozess	123
<b>6.4 Stochastische Prozesse - Markov-Ketten</b>	<b>128</b>
6.4.1 Stochastische Prozesse	128
6.4.2 Markov-Prozess und Markov-Ketten	129
<b>6.5 Einführung in die Warteschlangentheorie (Queueing Theory)</b>	<b>131</b>
6.5.1 Einbediener-Warteschlangenmodell (Single-server queueing models)	131
6.5.2 Warteschlangennetzwerke (Queueing network models)	134
<b>6.6 Petri-Netze</b>	<b>136</b>
6.6.1 Petri-Netze – Entstehung und Ausprägung	136
6.6.2 Deterministische und stochastische Petri-Netze (DSPN)	141
<b>6.7 Zusammenfassung und Einordnung</b>	<b>143</b>
<b>Literaturverzeichnis</b>	<b>143</b>

---

## Abstract

Performance modelling is one of the means which is most frequently used to analyse the dynamic behaviour of complex systems in order to save money in developing new systems, or to evaluate and to improve existent systems such as multi processor-systems or routing systems. Furthermore it is meant to support computer designers in modelling and developing new systems, and to evaluate new parameters which have not already be considered at in the past.

This work introduces and presents the process of modelling and some possibilities of modelling to conduct performance analysis of computer systems.

## Kurzbeschreibung

Leistungsmodellierung ist eines der Mittel, das am häufigsten gebraucht wird, um komplexe Systeme auf deren Leistungsfähigkeit hin zu analysieren, um Geld bei der Entwicklung neuer Systeme zu sparen, oder Bewertung und Verbesserung schon bestehender Computer-Systeme wie Multi-Prozessor-Anlagen oder Routing-Systeme durchzuführen. Ferner wird sie benutzt, um Entwickler dabei zu unterstützen, neue Systeme zu entwickeln, und um neue Parameter in die Entwicklung zu berücksichtigen.

Diese Seminararbeit beschäftigt sich mit dem Modellbildungsprozess, sowie den Möglichkeiten der Modellierung, um Leistungsanalyse an Computer-Systemen durchzuführen.

## 6.1 Zielsetzung dieses Beitrags

Im Rahmen des Seminars “**Bewertung von Internet-Routing Strategien**” werden verschiedene Aspekte des Internet-Routings betrachtet, sowie Methoden und Mittel zu dessen Analyse. Einige dieser Aspekte werden in verschiedenen Seminararbeiten behandelt. Zu diesen Aspekten gehört unter anderem auch die Bewertung der **Leistungsfähigkeit**, engl.: *performance*, anhand von Modellen, welche der Inhalt dieser Seminararbeit ist. Weitere Aspekte sind die Modellierung von **Zuverlässigkeit**, engl.: *dependability*, und die der **Performability**.

Um dem Leser mit dem Ziel dieses Beitrages vertraut zu machen, seien kurz die zuvor genannten Begriffe kurz erläutert.

### 6.1.1 Leistungsmodellierung

In der Informatik beschäftigt man sich generell mit informationsverarbeitenden Systemen, d.h. mit Systemen zur Transformation, Speicherung und Übertragung von Daten bzw. von Informationen. Für Nutzer wie Betreiber solcher Systeme steht die Leistungsfähigkeit im Vordergrund. Leistungsmodellierung versucht mit den Mitteln der Modellbildung, die im Kapitel 4 beschrieben werden, bei der Analyse eines existierenden Systems, Lösungen zur Verbesserung der Leistung zu finden. Auch versucht sie zu ergründen, ob die Analyse eines Systems überhaupt möglich ist. Im speziellen ist gemeint, dass man bei einer bekannten Last, die als Menge der geforderten Dienste an ein System angesehen werden kann, die auf das System wirkt, durch

Änderung bekannter oder Einfügung neuer Komponenten eine “maximale” Leistung in diesem System erreichen möchte.

### 6.1.2 Zuverlässigkeitsmodellierung

Zuverlässigkeitsmodellierung beschäftigt sich, wie der Begriff schon aussagt, damit, wie zuverlässig ein informationsverarbeitendes System ist. Dies bedeutet, man betrachtet ein vorliegendes System hinsichtlich auftretender Fehler und deren Auftretshäufigkeit. Dabei ist man auch an den Folgen unter anderem für die Verfügbarkeit des Systems interessiert. Hierbei läßt man im allgemeinen die Leistungsbetrachtungen außer acht, weil diese andere Zeitintervalle und Kenngrößen berücksichtigt. So kann es sein, dass man für ein System, das man schon durch Leistungsmodellierung in seiner Leistung “optimiert” hat, einen anderen Aufbau und eine andere Anordnung der Komponenten ermittelt, um es zuverlässiger zu machen.

Informationen zur Zuverlässigkeitsmodellierung kann man bei [Sch02] erhalten.

### 6.1.3 Performability-Modellierung

Aus den Gründen, dass man Zuverlässigkeit und Leistungsfähigkeit in einem System haben möchte, kommt man verstärkt zur parallelen Betrachtung von Leistung und Zuverlässigkeit in der Entwicklung von informationsverarbeitenden Systemen. Dies geschieht in der Performability-Modellierung, die die zuerst genannten Bewertungsaspekte in etwa vereint. Performability ist hierbei ein künstlicher Begriff. Er vereint die englischen Begriffe *Performance*, was soviel wie Leistung bedeutet, und *dependability*, was mit Zuverlässigkeit übersetzt wird. Bei Fragen der Performability-Modellierung wird auf [Bre02] verwiesen.

### 6.1.4 Kapitelübersicht

Zur Einführung wird in diesem Abschnitt einen kurzer Abriß über die Inhalte der nachfolgenden Kapitel geben, um dem Leser einen bessere Übersicht zu ermöglichen.

Das Kapitel “**Leistung und Leistungsmodellierung**” versucht, den Begriff der “**Leistung**”, der als Übersetzung des englischen Wortes **performance** benutzt wird, an einem einfachen Modell darzustellen und zu erklären. Im weiteren wird auf **Metriken** zur Bestimmung von Leistung eingegangen (s. Kapitel 6.2).

Danach werden **Techniken der Leistungsanalyse** vorgestellt, also Möglichkeiten, die eine Analyse der Leistungsfähigkeit eines Systems zulassen (s. Kapitel 6.2.2).

Im Abschnitt “**Modelle und Modellbildung**” wird, nach Definition des System-Begriffs, die Erstellung von Modellen zur Leistungsanalyse von Systemen behandelt, sowie der Prozess zur Modellbildung (s. Kapitel 6.3)

Kapitel 6.4 beschäftigt sich mit **stochastischen Prozessen**. Nach Definition und Vorstellung verschiedener Arten von stochastischen Prozessen wird auf Markov-Prozesse, sowie die damit eng verbundenen Markov-Ketten eingegangen.

In den nachfolgenden zwei Abschnitten werden Modellierungsmethoden betrachtet, die bei der Leistungsanalyse bzw. Leistungsmodellierung am häufigsten herangezogen werden, und deren Lösung oft durch Markov-Ketten beschrieben wird. Es handelt sich hierbei erstens um **Warteschlangen-Systeme**, die in Kapitel 6.5 eingeführt und beschrieben werden. Hierbei werden zunächst einfache Einbediener-Warteschlangen-Systeme betrachtet, deren Eigenschaften beschrieben und die verschiedenen Möglichkeiten deren Auswertung dargestellt. Desweiteren werden Netze von Warteschlangen-Systemen vorgestellt, anschließend deren Bedeutung innerhalb der Modellierung von Internet-Routing Systemen herausgestellt, sowie deren Stärken und Schwächen angesprochen.

Zum Zweiten werden **Petri-Netze** in Kapitel 6.6 betrachtet. Diese werden in deren verschiedenen Ausführungsformen dargestellt, die für eine Modellierung von Internet-Routing-Strategien relevanten Erweiterungen näher betrachtet. Zum Abschluß werden wiederum Stärken und Schwächen dargestellt.

Im Kapitel 6.7 wird diese Ausarbeitung in den Kontext des Seminars eingeordnet und deren Bedeutung herausgestellt.

## 6.2 Leistung und Leistungsmodellierung

Rechner- und Kommunikationssysteme werden von verschiedenen Interessensgruppen nach unterschiedlichen Kriterien beurteilt. Diese Kriterien sind zum Beispiel neben der Benutzerfreundlichkeit und Verfügbarkeit, Ausfallsicherheit und Kosten, Leistung bzw. Leistungsfähigkeit. Hierbei sind diejenigen Beurteilungsmerkmale von besonderer Bedeutung, die quantitativ meßbar sind. Verfügbarkeit, Ausfallsicherheit und Leistung besitzen diese Eigenschaft, wohingegen Benutzerfreundlichkeit beispielsweise ein subjektives Kriterium ist. Zur Bewertung der eben vorgestellten quantifizierbaren Merkmale von Computersystemen werden unter anderem **Leistungsanalyse**, **Zuverlässigkeitsanalyse** sowie **Performability-Analyse** betrieben. Performability-Analyse verbindet Leistungsanalyse und Zuverlässigkeitsanalysen. Um weiteren Ausführungen zur Performability-Modellierung und Zuverlässigkeitsmodellierung zu erhalten wird auf [Bre02] und [Sch02] hingewiesen.

Unterschiede zwischen Leistungs- und Zuverlässigkeitsanalyse werden im Abschnitt Metriken (Kap. 6.2) herausgestellt, da diese dort am besten verdeutlicht werden können.

### 6.2.1 Leistung: Metriken und deren Analyse

In diesem Unterkapitel wird auf den Leistungsbegriff, engl.: "*performance*", eingegangen und darlegt, wie die "*Performance*" eines Rechensystems, Kommunikationsnetzen etc. durch Leistungsanalysen analysiert werden kann.

Bei der Bewertung der Leistung in einem System versucht man den nicht nur Leistung zu beurteilen, sondern auch Entwurfsalternativen zu bestimmen, zwei oder mehrere Systeme zu vergleichen, den "bottleneck", also den Engpaß innerhalb der Hardware oder Software, zu finden, die Lastcharakterisierung richtig aufzustellen, Anzahl und Größe einzelner Komponenten festlegen und die Leistung bei Lasten, die in Zukunft auftreten, vorhersagen zu können. Ein System könnte hierbei als Ansammlung verschiedenster Hardware oder Software angesehen werden. Bei Hardware-Komponenten kann es sich um einen Prozessor handeln, bei Software beispielsweise

se um eine Datenbank, oder um ein Netzwerk von Rechnern, wie ein Internet-Routing-System ([Jai91], Seite 3/4).

Also kann man die Aufgabe der Leistungsanalyse als “Untersuchung und Optimierung des **dy-namischen Ablaufgeschehens innerhalb und zwischen** den Komponenten eines Rechensystems” formulieren. [Leh00]

### System - Last - Leistung

An einem einfachen Modell wird nun gezeigt, wie sich ein System, das Gegenstand der Leistungsanalyse ist, darstellen läßt.

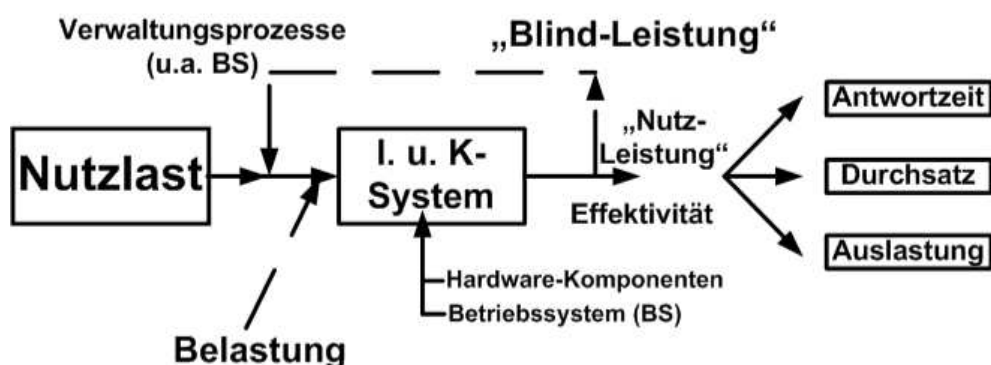


Abbildung 6.1: Modell eines Systems mit Einflußgrößen

Abbildung 6.1 zeigt ein Informationsverarbeitungs- und Kommunikationssystem (**I. u. K-System**), auf das eine **Nutzlast** von außen wirkt. Bei der Last könnte es sich um Anfragen an eine Datenbank oder um Befehle, wie zum Beispiel zur Berechnung eines numerischen Problems, handeln.

Welche Charakteristiken diese Lasten aufweisen, ist in [Bch02] näher beschrieben.

Diese Last entspricht nicht der **Gesamtlast** des Systems. Diese Gesamtlast, hier als **Belastung** am Dienstzugangspunkt zum System dargestellt, hat noch weitere Anteile. Diese sind **Verwaltungsprozesse** und entspricht der Leistung, die benötigt wird, um die in einem System vorhandene Hardware, wie Prozessor, Speicher, usw., und Software, beispielsweise Betriebssystem, zu betreiben.

Letztendlich betrachtet man bei der Leistungsanalyse das, was an **Effektivität**, nach Abzug der **Blind-Leistung**, zur Bewältigung der in der Last gestellten Anforderung übrigbleibt. Dies kann man als Nutz-Leistung identifizieren. An dieser Stelle sei eine kurze Definition der Leistung angeführt:

“Leistung: **Geschwindigkeit** und **Qualität**, mit der ein Auftrag oder eine Menge von Aufträgen von einer Datenverarbeitungsanlage verarbeitet wird.” (Quelle: [Mey93])

In dieser kurzen Beschreibung werden schon Kenngrößen genannt, mit denen man Leistung quantitativ messen kann. Als Beispiel für solche Kenngrößen, oder auch **Metriken** genannt, sind hier **Antwortzeit**, **Durchsatz** und **Auslastung** aufgeführt. Es gibt auch weitere Metriken, die bei der Leistungsanalyse Berücksichtigung finden. Diese sind beispielsweise Kapazität, Bedienzeit, Wartezeit, Verweilzeit, Reaktionszeit oder Rechenzeit. Einen Überblick über Leistungsmaße bietet [Lan92] auf den Seiten 18 ff.

Die in Abbildung 6.1 dargestellten Kenngrößen sind diejenigen, die am häufigsten in Leistungsanalysen betrachtet werden, und werden im folgenden Abschnitt näher erläutert.

### Metriken

Analysiert man ein System, so gibt es verschiedene Metrik-Klassen, nach denen man es betrachten kann. In Abbildung 6.2 sind die Metrik-Klassen der Leistungsanalyse, die der Zuverlässigkeitsanalyse, sowie die der Performability-Analyse dargestellt.

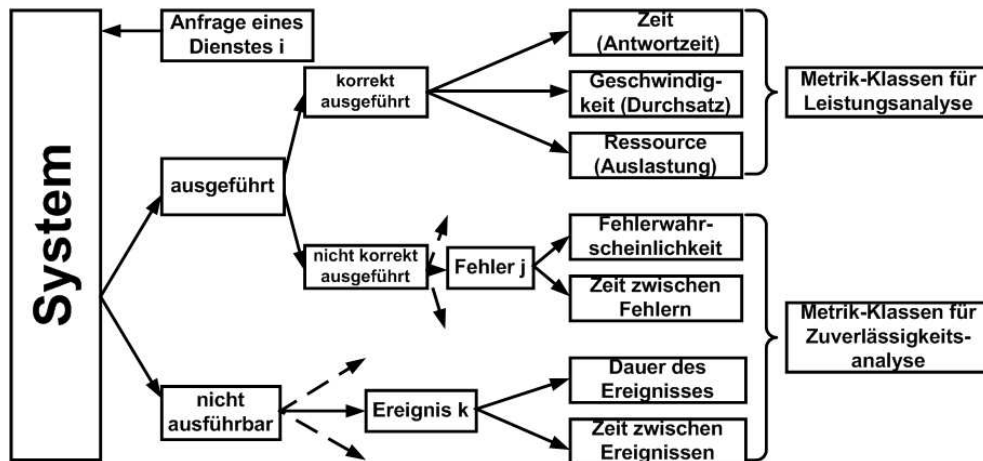


Abbildung 6.2: Mögliche Ausgänge bei einer Anfrage an ein System (entnommen: [Jai91], Seite 33)

Das in Abbildung 6.1 dargestellte Modell, lässt sich in der Abbildung 6.2 wiederfinden. Es wird eine Anfrage eines Dienstes an ein System gestellt. Es gibt unterschiedliche Ausgangsmöglichkeiten für diese Dienstanfrage. Im Folgenden werden die Ausgänge nach der Art des Analyseverfahrens dargestellt:

### Leistungsanalyse

Bei Leistungsanalysen werden der durch Anwender angefragte Dienste ausgeführt. Diese **müssen** bei **korrekt** ausgeführt werden. Man betrachtet das System bei der Bewertung der Leistung in den Metrik-Klassen **Zeit**, engl.: **time**, **Geschwindigkeit**, engl.: **rate** und **Ressource**, engl.: **ressource**. Als eine repräsentative Leistungskenngröße der jeweiligen Klasse sind in Klammern *Antwortzeit*, *Durchsatz* und *Auslastung* dargestellt. Dies sind auch die dargestellten Leistungskenngrößen aus Abbildung 6.1.

## Zuverlässigkeitsanalyse

Bei Zuverlässigkeitsanalysen ist man an allen Ausgängen für durch Anwender erfragte Dienste interessiert. Die möglichen Ausgänge sind eine korrekte Ausführung der Dienste, eine Ausführung, die nicht korrekt ist und zu einem Fehler führt, oder eine Ausführung ist nicht möglich. Man will bei der Zuverlässigkeitsanalyse bei **Auftreten eines Fehlers** deren **Auftrittswahrscheinlichkeit und -häufigkeit**, sowie die **Zeit zwischen zwei Fehlern** ermitteln. Ferner ist man bei einer **Nicht-Ausführung** eines Dienstes an der **Dauer des Ereignisses** interessiert, wie auch an der **Zeit zwischen den Ereignissen**.

## Performability-Analyse

Performability-Analysen vereinen die Metrik-Klassen der beiden zuvor genannten Analyseverfahren. Sie versuchen in gleichem Maße, Zuverlässigkeit wie auch Leistung eines Systems zu optimieren und berücksichtigen alle zuvor genannten Kenngrößen.

Nachdem die verschiedenen Metrik-Klassen aus Abbildung 6.2 beschrieben wurden, werden die in Abbildung 6.1 schon eingeführten Metriken als Repräsentanten der jeweiligen Metrik-Klasse der Leistungsanalyse näher definiert.

**Antwortzeit**, engl.: **response time**, ist definiert als das Zeitintervall, das zwischen Anfrage eines Benutzers und der Antwort durch das System beim Benutzer liegt, wie es in Abbildung 6.3 dargestellt ist. Diese Definition jedoch ist sehr vereinfacht, da ja Anfrage und Antwort nie sofort erfolgen, sondern immer mit Zeitverzögerungen verbunden sind. Ein Nutzer benötigt aber Zeit zum Stellen seiner Anfrage, wie auch das Beantworten derselben durch das System Zeit in Anspruch nimmt. Dies stellt Abbildung 6.4 anschaulich dar. Die Antwortzeit kann auf zwei verschiedene Arten erfaßt werden. Zunächst kann sie als Zeitspanne zwischen Ende der Anfrage eines Benutzers und Beginn der Systemantwort (Definition 1) definiert werden, wie auch als Intervall zwischen Ende der Anfrage eines Benutzers und Ende der Systemantwort (Definition 2).

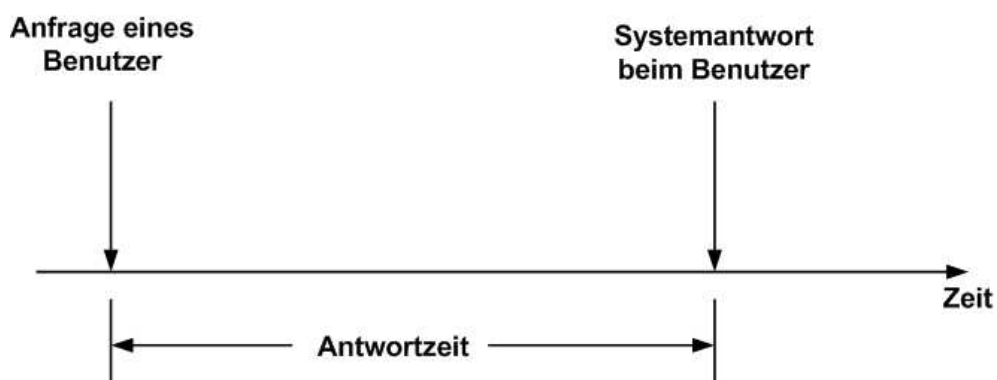


Abbildung 6.3: *Anfrage und Antwort ohne Verzögerungen* (entnommen: [Jai91], Seite 37)

**Durchsatz**, engl.: **throughput**, ist definiert als Verhältnis von Anfragen pro Zeiteinheit, die ein System bearbeiten kann. Für stapelverarbeitende Systeme mißt man den Durchsatz in Jobs, die pro Sekunde durchgeführt werden können. Durchsatz bei interaktiven Systemen wird in

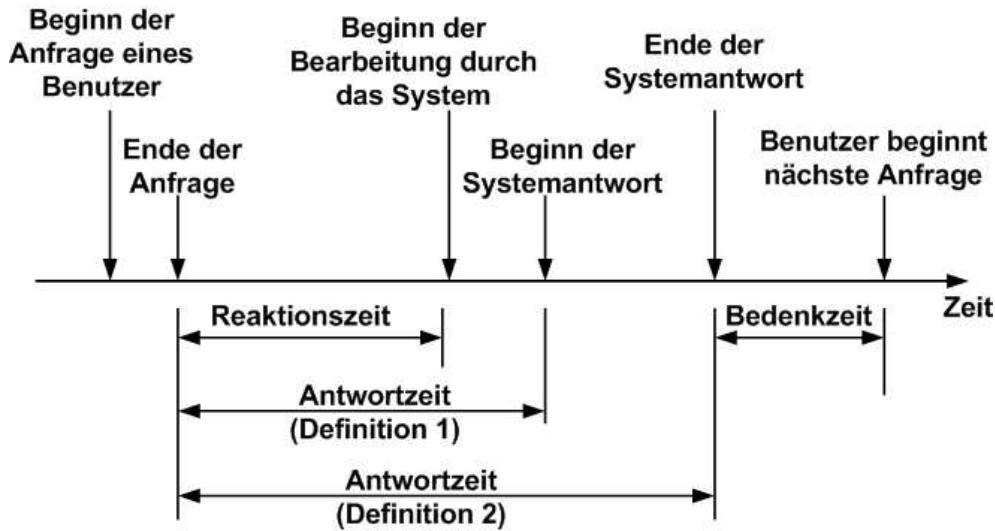


Abbildung 6.4: Anfrage und Antwort in Realität (entnommen: [Jai91], Seite 37)

Anfragen pro Sekunde bewertet, bei einem Prozessor spricht man von Millionen Befehlen pro Sekunde (MIPS) oder Millionen Gleitkommaoperationen pro Sekunde (MFLOPS), bei Netzwerken in Paketen pro Sekunde (pps) oder Bits pro Sekunde (bps), bei Abwicklungsverarbeitungssystemen in Transaktionen pro Sekunde (TPS).

Der Durchsatz eines Systems vergrößert sich anfangs bei wachsender Last auf das System. Bei einer der Grenzlaster hält dieser Anstieg inne; in den meisten Fällen nimmt dieser vielleicht sogar ab, wie Abbildung 6.5 zeigt.

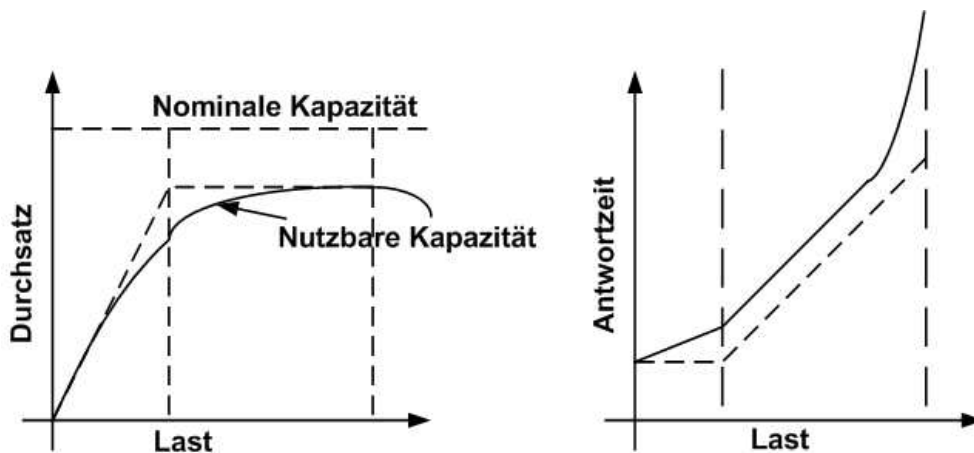


Abbildung 6.5: Kapazität eines Systems (entnommen: [Jai91], Seite 38)

**Auslastung** einer Ressource, engl.: **utilization**, ist der Zeitanteil, während dessen eine Ressource damit beschäftigt ist, den erfragten Dienst auszuführen, bezogen auf den Gesamtzeitraum einer Leistungsbewertung. Die Periode, in der eine Ressource nicht benutzt wird nennt man Leerlaufzeit, engl.: **Idle-Time**.

Zum einen gibt Ressourcen, die sich entweder in Benutzung oder im Leerlauf befinden. Deshalb wird der Leerlauf betrachtet, um die Auslastung festzustellen. Zum anderen gibt es Res-



sources wie Speicherbausteine, die nicht immer in ihrer vollen Kapazität genutzt werden; daher werden ihre Auslastung als Durchschnittsanteil, engl.: *average fraction*, der Benutzung über den Bewertungszeitraum gemessen.

Für detailliertere Beschreibungen der für die Leistungsanalyse bedeutenden Metriken sei auf [Jai91] S.37 ff. verwiesen.

In weiteren werden die verschiedenen Ansätze der Leistungsanalyse und Zuverlässigkeitsanalyse, die sich hauptsächlich mit den Fehlern, deren Häufigkeit und die Zeit zwischen zwei solchen Ereignissen beschäftigt, dargelegt.

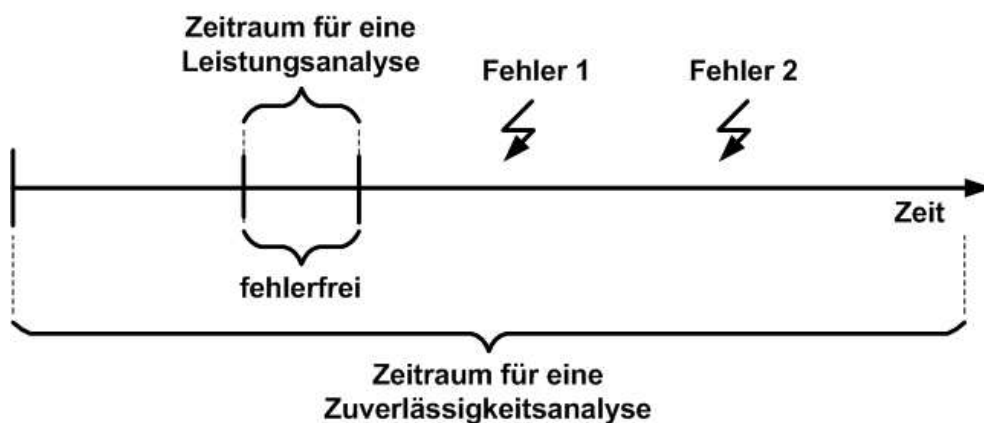


Abbildung 6.6: Zeiträume von Leistungs- bzw. Zuverlässigkeitsanalyse

Neben den schon erwähnten verschiedenen Metrik-Klassen, die die Analyseverfahren betrachten, gibt es noch einen zusätzlichen Unterschied, nämlich das Zeitintervall, während dessen dem ein System betrachtet wird, wie Abbildung 6.6 verdeutlicht. Während die Leistungsanalyse in sehr kleinen Intervallen betrieben wird, in denen meist ein fehlerfreies System vorausgesetzt wird, damit eine verwertbare Aussage aus den Ergebnissen gewonnen werden kann, muss bei der Zuverlässigkeitsanalyse ein erheblich größerer Beobachtungszeitraum gewählt werden, um Ergebnisse als auswertbar bezeichnen zu können, weil man in einem zu klein gewählten Beobachtungsintervall eventuell die Ereignisse, wie Fehler oder Nicht-Ausführung eines Dienstes, nicht eintreten.

## 6.2.2 Techniken der Leistungsanalyse

Dieser Abschnitt gibt einen kurzen Überblick über *Techniken der Leistungsanalyse*, welche Mittel man zur Leistungsmodellierung heranziehen kann.

Abbildung 6.7 stellt eine mögliche Einteilung dieser Techniken dar.

**Deterministische Bewertung** wird an realen Systemen durchgeführt. Diese werden entweder durch Messungen in Form von Benchmarks oder Testen mit einer gegenwärtigen Arbeitslast oder durch den Bau von Prototypen (engl.: *Prototyping*) durchgeführt. Beim Prototyping entwickelt man Software-Systeme, die Nachbildungen schon bestehender Systeme darstellen. Dabei erhält man entweder durch ablauforientierte oder ausführungorientierte Simulation die Ergeb-

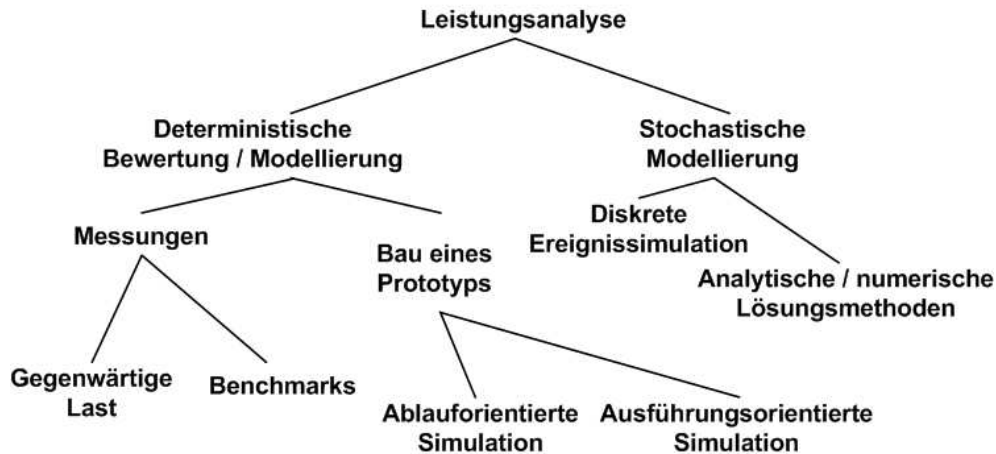


Abbildung 6.7: Techniken der Leistungsanalyse (entnommen: [Lin98], Seite 6)

nisse der Leistungsanalyse.

Man spricht von **deterministischer Modellierung**, wenn ein System bei gleichem Anfangszustand und identischer Eingabe von Parametern immer dieselbe Reaktion zeigt.

Der *Vorteil* einer deterministischen Bewertung ist, dass man einen Vergleich zweier ähnlicher Systeme durchführen und anhand der Ergebnisse das bessere und leistungsfähigere von beiden ermitteln kann.

Der wesentliche *Nachteil* der deterministischen Methode besteht darin, dass stochastische Ereignisse gar nicht oder nur sehr schwer nachstellbar sind. Ursache dafür ist, daß wahrscheinlichkeitsbedingte Einflüsse nicht statisch reproduzierbar sind. Diese Tatsache vereinfacht zwar die Berechnung bzw. Auswertung, jedoch muß man dafür einen geringeren Realitätsgrad akzeptieren.

Bei den Erscheinungsformen heutiger Computer- oder Netzwerksysteme bedarf es einer Analyse, die eher einen stochastischen als einen deterministischen Ansatz verlangt. **Stochastische Modellierung** bietet dem Systementwickler die Möglichkeit, zufällige Ereignisse in die Entwicklung eines Systems einfließen zu lassen. Im Gegensatz dazu ist bei einem deterministischen Ansatz der Ablauf von Vorgängen im System fest vorgegeben. Gerade in Bezug auf Netzwerke reicht dies jedoch nicht aus. Beispielsweise kann die Wahl des Übertragungsweges in einem Router bei Betrachtung dessen Funktionsweise auf einer logischen Ebene nicht festgelegt sein, sondern unterliegt Zufälligkeitseinflüssen. Daraus resultiert der Vorteil der stochastischen Modellierung im Vergleich zur deterministischen Methode, dass Computerentwickler bereits in einer frühen Entwicklungsphase "schlechte" Entwürfe für Systeme "auszuschalten" und dadurch viel Geld einsparen können. Ferner kann man in der Modellierung eines komplexes System unnötige Details ausklammern, die bei einer Analyse nicht berücksichtigt werden müssen.

Bei den Analysetechniken der stochastischen Modellierung unterscheidet man zwischen der diskreten Ereignissimulation und analytischer / numerischer Lösungsmethoden, wie es in Abbildung 6.7 dargestellt ist. Im weiteren Verlauf werden letztere näher betrachtet.

Zu weiteren Erläuterungen zur Leistungsmodellierung sei auf Kapitel 1.2 von [Lin98] verwiesen, wo nähere Ausführungen zu dieser zu finden sind.

## 6.3 Modelle und Modellbildung

Sofern ein Mensch einen Ausschnitt der Realität als System identifiziert, wendet er bereits Kriterien an, nach denen er seine Beobachtungen bewertet und ordnet. Somit gibt es in der Systemanalyse große Freiräume, denn die untersuchten Systeme werden problemabhängig definiert und uninteressante Erscheinungen a priori ausgeklammert ([Pag91], S.2). Auf diese Weise hat sich der Mensch schon in gewisser Weise ein Modell gebildet.

### 6.3.1 Modelle

Durch den Vorgang der Modellbildung werden Systeme in **Modelle** hinsichtlich bestimmter Fragestellungen und Zielsetzungen abgebildet. Die Modelle sind wiederum selbst Systeme, die aber die Elemente und Relationen des Ursprungssystems in veränderter Weise darstellen ([Pag91], S.4).

Schon Niemeyer beschrieb dies in ähnlicher Weise:

*Modelle sind materielle oder immaterielle Systeme, die andere Systeme so darstellen, dass eine experimentelle Manipulation der abgebildeten Strukturen und Zustände möglich ist.* ([Nie77], S. 57)

Zu einem gegebenen System können mehrere unterschiedliche Modelle erstellt werden, abhängig von der Zielsetzung der Modellstudie und der subjektiven Sichtweise des Modellbildners.

Beim Übergang vom Ursprungssystem (Original) zum Modell findet eine Abstraktion und Idealisierung statt. Ein Modell stellt das Original also vereinfacht dar; dadurch erst wird die Untersuchung komplexer Systeme handhabbar. Da die Ergebnisse, die mit dem Modell erzielt werden, auf das Original übertragen werden sollen, ist eine ausreichend genaue Abbildung der wesentlichen Eigenschaften nötig. Welche Eigenschaften als wesentlich betrachtet werden, hängt von der Fragestellung und Zielsetzung der Untersuchung ab.

Modelle können nach verschiedenen Kriterien klassifiziert werden: nach der Art der **Untersuchungsmethode**, nach der Art der **Zustandsübergänge** und schließlich nach dem intendierten **Verwendungszweck** ([Pag91], S. 4). Nähere Ausführungen zu den Klassifikationen der Modelle sind in [Pag91] Seiten 4 ff. beschrieben.

### 6.3.2 Modellbildung und Modellbildungsprozess

Bei der Modellbildung wird durch Abstraktion und Idealisierung die Komplexität eines Systems reduziert. Das nachzubildende System wird hier als Realsystem bezeichnet.

#### Schematische Darstellung des Modellbildungsprozesses

Der Prozess der Modellbildung und der Simulation, die das entworfene Modell validiert und schließlich Rückschlüsse auf das Realsystem zulässt, läßt sich in mehrere Schritte zerlegen, wie Abbildung 6.8 zeigt. Mittels des Modellentwurfs kommt man auf ein **konzeptuelles Modell**, auch **kommunikatives Modell** genannt, welches aus dem Abstraktions- und Idealisierungsprozess resultiert. Es ist ein rein gedankliches Modell, das nur in der Vorstellung existiert und dieses konzeptuelle Modell wird beispielsweise in Form von umgangssprachlichen Texten,

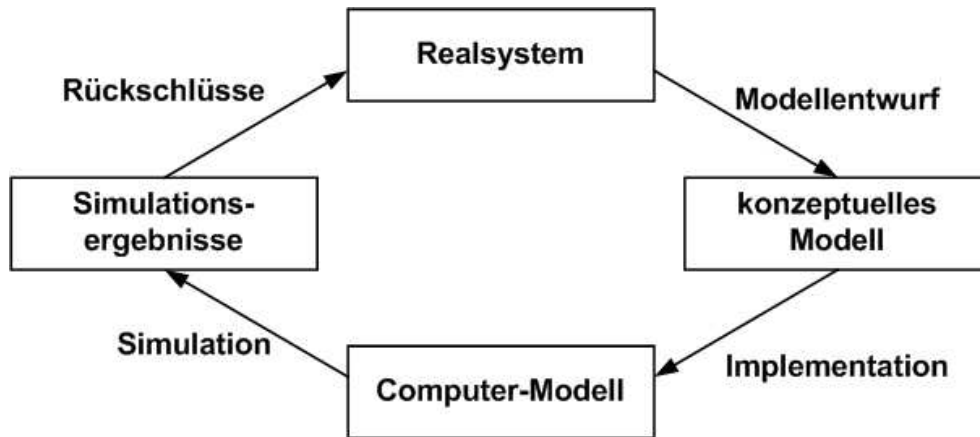


Abbildung 6.8: Schematische Darstellung der Modellbildung und Simulation (entnommen: [Pag91], S.11, abgewandelt)

Modelldiagrammen oder mathematischen Formeln beschrieben. Aus dem konzeptuellen Modell entwickelt der Modellbildner das formale Modell, das eine exakte Beschreibung des Modells ist, worin beispielsweise in graphischer oder mathematischer Form das Modell dargestellt ist; quantifizierbare Größen werden auch mit aufgeführt, die für das Modell wichtig sind.

Schließlich kommt es zum Entwurf eines **ausführbaren Modells**, beispielsweise ein Computer-Modell, das, sofern lauffähig, in der Simulation, als eine Möglichkeit zur Auswertung, ausgeführt wird, und deren Ergebnisse, wie schon oben beschrieben, in gewissen Grenzen Rückschlüsse auf das Realsystem erlauben. Simulation ist aber nur eine der möglichen Lösungstechniken. Es gibt aber auch die Möglichkeit analytischen und numerischen Lösung, die in Kapitel 5 und folgenden nähere Betrachtung finden.

Diese Darstellung der Modellbildung ist stark vereinfacht. Bei näherer Betrachtung der Prozesses der Modellbildung lassen sich die Arbeitsschritte des Modellentwicklers weiter verfeinern, was im folgenden Abschnitt beschrieben wird ([Pag91], S. 11).

### Phasen des Modellbildungsprozesses

Der gesamte Modellbildungsprozess ist in der Abbildung 6.9 in Anlehnung an [Käm90] dargestellt. Rechtecke kennzeichnen dabei Aktivitäten des Modellentwicklers und Ellipsen deren Voraussetzungen bzw. Resultate. Die Phasen werden einschließlich der Phase 3 näher ausgeführt.

#### 1. Problemdefinition

Am Anfang jeder Modellstudie sollte eine möglichst gründliche Problemdefinition stehen. Sie legt die zu untersuchenden Fragestellungen und zu erreichenden Ziele fest. Bei der genauen Problemabgrenzung muß dem Modellentwickler möglichst vollständig bewußt sein, unter welchem Blickwinkel er an die Problemstellung herangeht, da in diesem Anfangsschritt bereits eine Bandbreite von Lösungsmöglichkeiten abgesteckt wird.

Ist die Fragestellung der Untersuchung nicht eindeutig formuliert, so kann nicht entschieden werden, welche Systemkomponenten im Modell abzubilden sind und von welchen abstrahiert werden kann. Hierbei gilt es, zu erschließen, was das Ziel der Analyse ist, so dass ein Modell erstellt werden kann, das das System abbildet und Fehler, die in dieser frühen Phase gemacht

werden, sind später nur mit großem Aufwand zu korrigieren, da hierzu die Modellstudie in der Regel wieder neu begonnen werden muß.

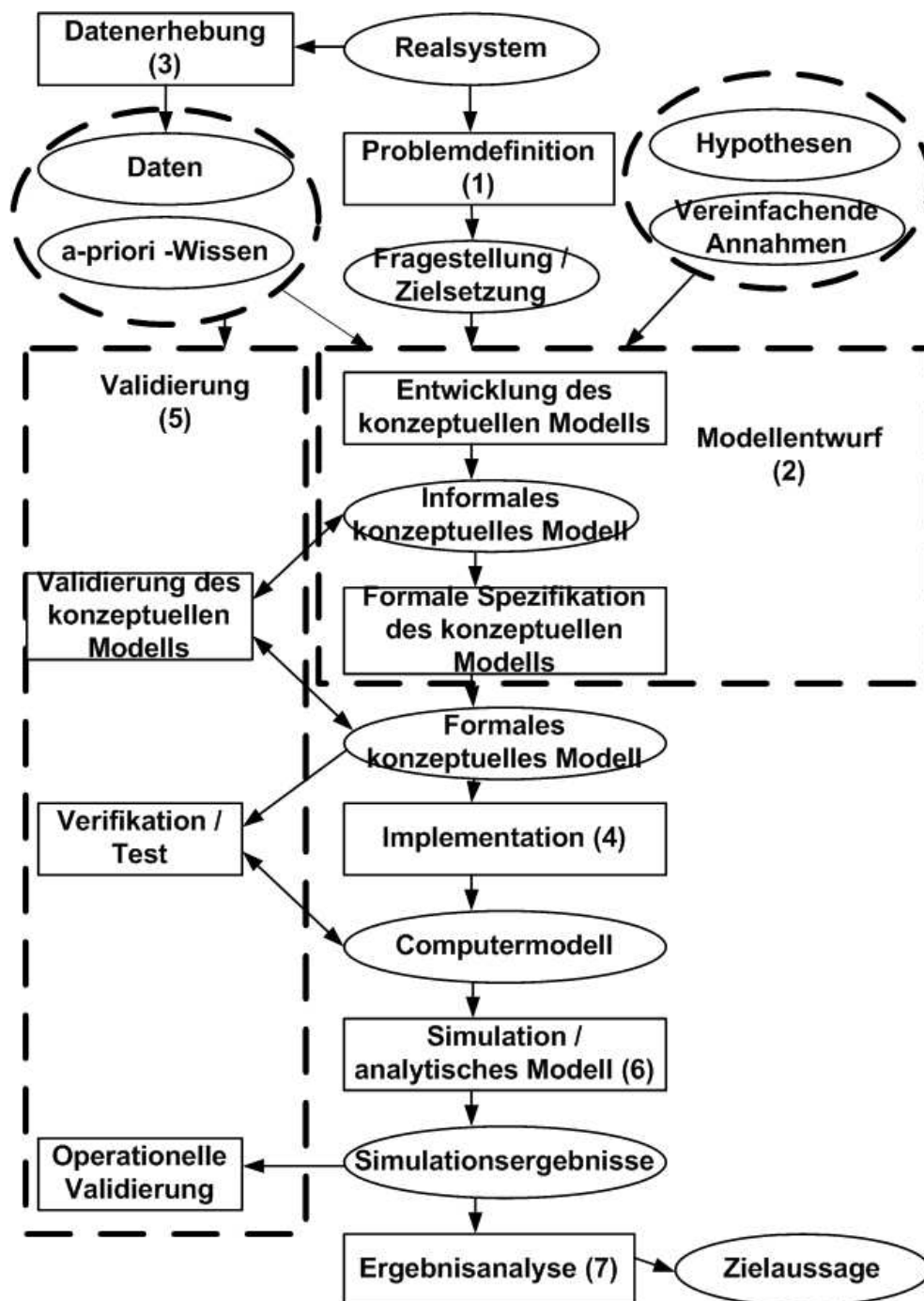


Abbildung 6.9: Die Phasen des Modellbildungsprozesses (entnommen: [Pag91] S. 12)

## 2. Modellentwurf

Der erste Schritt in der Phase des Modellentwurfs ist die **Entwicklung des konzeptuellen Modells**. Hierbei ist, ausgehend von der zu untersuchenden Fragestellung, eine Abgrenzung des zu modellierenden Systems gegenüber seiner Umgebung vorzunehmen. Auf der Basis der Definition der Systemgrenzen wird eine erste Erfassung der Systemstruktur vorgenommen. In

diesem ersten Abstraktionsschritt werden relevante Systemobjekte, ihre Attribute und Beziehungen ausgewählt und es wird entschieden, welche Systemaspekte für die zu untersuchende Fragestellung unwesentlich sind und unberücksichtigt bleiben können [Sch82].

Die relevanten Systemgrößen und ihre Beziehungen werden zunächst informal beschrieben. Hierfür wird meist ein grafisch-deskriptives Medium verwendet (Struktur-, Kausal- oder Flußdiagramme, Petri-Netz-ähnliche Darstellungen usw.).

Auf dieser Grundlage erfolgt die **Wahl eines geeigneten Modelltyps**. Ein System kann abhängig von der gewählten Fragestellung und Betrachtungsweise meistens durch Modelle verschiedenen Typs (z. B. durch ein kontinuierliches oder ein diskretes Modell) dargestellt werden. So kann man die Abfertigung der Kunden an den Kassen eines Supermarktes als diskreten Prozeß im Verlauf der Zeit ansehen. Betrachtet man aber die gesamte Masse der Kunden, die in den Supermarkt strömen, einkaufen, sich an den Kassen stauen, abgefertigt werden und den Supermarkt verlassen, so ist auch die Darstellung dieser "Masse" als "*Menschenfluß*" in einem kontinuierlichen Modell gerechtfertigt.

Ferner ist zu entscheiden, ob das System oder Teile davon durch Zufallsprozesse in einem stochastischen Modell beschrieben werden sollen, oder ob ein deterministisches Modell zu verwenden ist. In stochastischen Modellen lassen sich auch solche Komponenten des Systems durch Zufallsprozesse nachbilden, die zwar prinzipiell deterministisch beschreibbar wären, aufgrund ihrer Komplexität und Undurchschaubarkeit aber ein zufällig erscheinendes Verhalten zeigen. Die Abbildung deterministischer Systeme auf stochastische Modelle ist ein wichtiges Mittel der Komplexitätsreduktion.

Das Ergebnis der beschriebenen Arbeitsschritte ist das informale konzeptuelle Modell.

Der nächste Schritt im Rahmen des Modellentwurfs ist die **formale Spezifikation des konzeptuellen Modells**, d. h. eine exakte Beschreibung des Modells in einem geeigneten Formalismus. Voraussetzung für eine mathematische Darstellung ist die Quantifizierbarkeit der Systemgrößen, damit Gleichungen oder Entscheidungsregeln zur Beschreibung der Beziehungen dieser Größen aufgestellt werden können.

Auf welche Informationen stützt sich ein Modellentwickler in der Entwurfsphase?

Zum einen sind dies empirische Daten, anhand derer induktiv Schlußfolgerungen über bestimmte Eigenschaften des Realsystems gezogen werden. Auf der Basis von Beobachtungsdaten werden Gesetzmäßigkeiten postuliert, die sich in den oben genannten Schätzwerten (z. B. Verteilungsparametern) ausdrücken.

Zum anderen wird der Modellentwickler sein "a-priori"-Wissen einbringen, also jene Fachkenntnisse und Erfahrungen, die er vor Beginn der Modellstudie erworben hat. Aus diesem Wissen kann er deduktiv bestimmte Aussagen über das Realsystem ableiten.

Wo sicheres Wissen fehlt, werden Hypothesen aufgestellt, die in späteren Validierungsphasen abgesichert werden müssen. Im Rahmen des Abstraktions- und Idealisierungsprozesses macht der Modellentwickler schließlich bestimmte vereinfachende Annahmen, die sich ebenfalls im Rahmen der Validierung bewähren müssen. Ergebnis der gesamten Modellentwurfsphase ist ein formales konzeptuelles Modell.

### 3. Datenerhebung

Da im Rahmen des Modellentwurfs genau festgelegt wird, welche Konstanten und Variablen das Modell enthält und wie diese quantifiziert werden, ist die **Ermittlung der erforderlichen Daten** eng mit den verschiedenen Ebenen des Modellentwurfs verflochten. Daher ist dieser Schritt des Modellbildungsprozesses zeitlich parallel zur Entwurfsphase zu sehen. Es müssen Werte für Modellkonstanten, Anfangswerte oder Zeitreihen für Modellvariablen sowie Vertei-

lungstypen und -parameter für stochastische Modellgrößen ermittelt werden.

Die Datenerhebung kann einen beträchtlichen Teil des Aufwandes einer Simulationsstudie ausmachen. Beobachtungsdaten können prinzipiell als unausgewertete Zeitreihen in ein Modell einfließen (etwa zu Validierungszwecken). Häufiger werden die Daten jedoch zu statistischen Kennwerten (insbesondere Verteilungsparametern) aggregiert und in dieser Form weiterverwendet. Dies hat den Vorteil, daß eine Verallgemeinerung gegenüber den speziellen Beobachtungsdaten erreicht wird. In den meisten Modellstudien werden historische Daten vorliegen, die einen Teil des Datenbedarfs abdecken, aber durch gezielte neue Messungen am Realsystem ergänzt werden müssen. Schätzwerte können auch durch Expertenbefragungen gewonnen werden, wenn eine Neuerfassung und Auswertung der entsprechenden empirischen Daten aus Aufwandsgründen nicht in Frage kommt.

Die **Phase 4** des Modellbildungsprozesses ist die **Implementation**. Sie wird in Abhängigkeit vom verwendeten Modelltyp der zur Verfügung stehenden Hard- und Software in einer Programmiersprache realisiert, die eine höhere Programmiersprache wie C++, Java oder Smalltalk sein kann oder eine Simulationssprache, wie zum Beispiel SHARPE oder Simul8.

In der **Modellvalidierung, Phase 5** des Prozesses der Modellbildung, wird die Gültigkeit des gewählten Modells überprüft. Die Bewertung des Modells hinsichtlich seiner Validität (Gültigkeitsnachweis, Modellvalidierung) verläuft parallel zu anderen Phasen des Modellbildungsprozesses.

Ob ein Modell "richtig" ist, läßt sich jedoch nicht allgemein, d.h. unabhängig von bestimmten Fragestellungen, Zielsetzungen und Bewertungen, nachweisen. Daher kann es auch keinen allgemeingültigen, umfassenden Validitätstest geben. Vielmehr muß man während des Modellentwicklungsprozesses auf verschiedenen Ebenen vielfältige Prüfungen vornehmen, um das Modell nach außen hin transparent und seine Eignung für die jeweilige Problemstellung glaubhaft zu machen ("model credibility"). Für die Modellvalidierung bieten sich die folgenden Stufen an:

- Validierung des konzeptuellen Modells
- Modellverifikation
- Operationale Modellvalidierung

In **Phase 6** erfolgt die **Simulation**, sofern das Modell in ein ausführbares Modell in einer Programmiersprache umgesetzt wurde, oder eine **numerische Analyse**, falls das Modell mit Hilfe von Differentialgleichungen oder Matrizen berechnet werden kann.

Ein Modellexperiment oder Simulationsexperiment besteht aus einem oder mehreren Simulationsläufen. In jedem Simulationslauf wird das dynamische Modellverhalten unter vorher festgelegten Bedingungen generiert. Dabei werden die Eingabewerte bzw. experimentellen Parameter von Lauf zu Lauf variiert. Wie bei der Simulation, kommt es auch bei einem analytischen Modell zur Variierung der Eingabewerte und Parameter.

Die Ergebnisse, also die Ausgabedaten der Simulationsläufe oder der numerischen Analyse, müssen in angemessener Form präsentiert werden. Dabei spielt die grafische Aufbereitung der Daten eine wichtige Rolle.

Die **Ergebnisanalyse** ist die letzte Phase des Modellbildungsprozesses, wie er in Abbildung 4.2 zu sehen ist. Die Ergebnisse, die durch Simulation oder numerische Analyse erlangt wurden, geben auch nach einer geeigneten Aufbereitung noch keine direkte Antwort auf die Fragestellung der Modellstudie. Vielmehr ist eine umfassende und sorgfältige Ergebnisanalyse und -bewertung erforderlich, um eine Zielaussage zu gewinnen. Im Rahmen der Ergebnisanalyse werden z. B. die Simulationsdaten aus verschiedenen Läufen verglichen, um die Auswirkungen der Eingabewerte auf das Modellverhalten zu untersuchen. Bei stochastischen Modellen müssen die Wahrscheinlichkeitsverteilungen (bzw. deren Parameter) der Ausgabevariablen mit statistischen Methoden geschätzt werden. Eine Bewertung der Ergebnisse erfolgt auf der Grundlage der ursprünglichen Fragestellung und Zielsetzung der Simulationsstudie. Wenn im Verlauf des Modellbildungsprozesses Einschränkungen vorgenommen wurden, die die Anwendbarkeit des Modells betreffen, müssen diese bei der Interpretation der Ergebnisse unbedingt berücksichtigt werden. Nur auf dieser Grundlage ist eine Übertragung der aus der Analyse gewonnenen Zielaussage auf das Realsystem zulässig.

Zusätzlich sollte parallel zu diesen Phasen eine umfassende **Dokumentation** erstellt werden. Sie dient der Kommunikation aller am Modellbildungsprozess beteiligter Personen. Ferner ist die Dokumentation Arbeits- und Beurteilungsgrundlage für alle, die in Zukunft mit diesem Modell arbeiten und dieses nutzen. ([Pag91], S. 10-18)

## 6.4 Stochastische Prozesse - Markov-Ketten

Nach Einführung in die Modelle und den Modellbildungsprozess, wird im Folgenden auf Methoden zur numerischen Lösung von stochastischen Modellen eingegangen.

### 6.4.1 Stochastische Prozesse

In der analytischen Modellierung werden nicht nur Zufallsvariablen sondern auch Folgen von Zufallsvariablen, die Funktionen der Zeit sind, verwendet.

Wird beispielsweise ein Prozessor betrachtet und die Anzahl der Jobs über einen Zeitraum, so kann man diese als Funktion über die Zeit ansehen. In einem Vergleich verschiedener Systeme bezüglich der Jobs, die durch die Prozessoren bearbeitet werden, kann man aber feststellen, dass diese Anzahl der bearbeiteten Jobs eine Zufallsvariable ist. In ähnlicher Weise ist die Wartezeit in der Warteschlange nicht diskret festlegbar oder vorhersehbar. Solche Zufallsfunktionen oder Folgen derselben werden als **stochastischer Prozess** bezeichnet.

Stochastische Prozesse werden nach verschiedenen Merkmalen klassifiziert:

Klassifikationsmerkmal **Zustandsraum**:

Man bezeichnet einen stochastischen Prozess als **zustandsdiskret** (engl.: **discrete state**) sofern die möglichen erreichbaren Zustände endlich oder zumindest abzählbar sind. Am genannten Beispiel der Prozessoren würden die bearbeiteten Jobs, die sicherlich in einem betrachteten Zeitintervall abzählbar sind, als Zustandsraum erkennbar sein oder etwa könnte die Anzahl von Stellplätzen in einem Parkhaus, sofern beispielsweise eine Untersuchung bezüglich des Parkverhaltens von Nutzern des Parkhauses durchgeführt werden soll, als Zustandsraum bezeichnet



werden. Man bezeichnet im Falle der Abzählbarkeit der Zustände den stochastischen Prozess auch als **stochastische Kette**, (engl.: **stochastic chain**).

Ist der Zustandsraum überabzählbar, so wird der Prozess als **zustandskontinuierlich** (engl.: **continuous state**) bezeichnet.

Klassifikationsmerkmal **Zeitparameter**:

Wie bei den Zustandsräumen unterscheidet man beim Zeitparameter zwischen einem diskreten (engl.: **discrete time**) und einem kontinuierlichen (engl.: **continuous time**) Fall.

Ferner können auch noch nach **Art von Zuständen** unterschieden werden, ob diese transient oder absorbierend sind. Transiente Zustände besitzen die Wahrscheinlichkeit 0, absorbierende Zustände können nicht mehr verlassen werden.

Es gibt verschiedene Arten von stochastischen Prozessen. Folgende sind die in der Literatur mit am häufigsten erwähnten:

- Geburts- und Todes-Prozess
- Poisson-Prozess
- Markov-Prozess

Zur näheren Beschreibung der ersten beiden und weiterer stochastischen Prozessen wird auf Kap. 2 in [Ave98], Kap. 3 in [Hav01], Kap. 30.4 in [Jai91] und Kap. 7 in [Lan92], verwiesen.

Im folgenden Abschnitt wird der für die Analyse von stochastischen Modellen bedeutendste stochastische Prozess vorgestellt, der Markov-Prozess.

### 6.4.2 Markov-Prozess und Markov-Ketten

In diesem Unterkapitel werden die Eigenschaften von Markov-Prozessen behandelt, die Darstellung von gestellten Problemen in Form von Markov-Ketten angesprochen, sowie die Möglichkeiten der Lösungs- und Ergebnisfindung bei denselben gezeigt.

Es wird hier auf mathematische Terminologien in Form von Definitionen verzichtet und auf Kap. 4 in [Bre02] verwiesen, wo Markov-Prozesse und Markov-Ketten formal eingeführt werden.

Wenn der zukünftige Zustand eines Prozesses unabhängig von der Vergangenheit ist, sondern nur vom gegenwärtigen Zustand abhängt, so nennt man diesen Prozess **Markov-Prozess**. Diese Markov-Eigenschaft, die man als Gedächtnislosigkeit bezeichnet, ermöglicht es, einen Prozess leichter zu untersuchen, denn man muß nicht den gesamten Verlauf berücksichtigen. Diese Arten von Prozessen wurden nach A.A. Markov benannt, der diese 1907 definierte und analysierte.

Es wird nun ein Beispiel einer Markov-Kette gezeigt, das auch in den zwei Folgekapiteln eine Rolle spielen wird.

Angenommen, man betrachtet als System der Realität ein Parkhaus, bei dem das Parkverhalten der Nutzer analysiert werden soll, so kann man dies auf verschiedenste Weise tun.

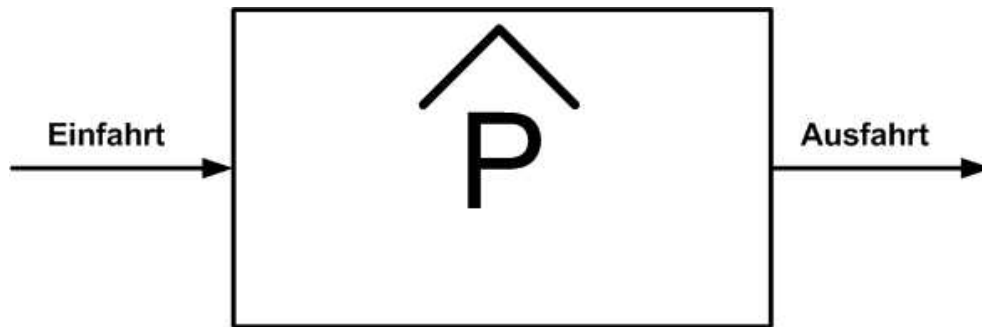


Abbildung 6.10: Vereinfachtes Modell eines Parkhauses

Man könnte sich an einem bestimmten Tag an der Ein- und Ausfahrt des Parkhauses postieren und jedes Auto bei der Ein- und Ausfahrt notieren und somit Informationen über das Verhalten der Nutzer erlangen. Diese Werte sind aber nur repräsentativ für diesen einen Tag und es können diese Ergebnisse nicht zwingend auf andere Tage übertragen werden.

Es gilt somit ein Analysemodell zu entwickeln, das es ermöglicht, aus Daten, die man als a-priori-Wissen, und Beobachtungen, die man getätigt hat, das Parkverhalten der Nutzer zu analysieren. Zur Vereinfachung der Analyse sei vorausgesetzt, es gebe nur eine Einfahrt und eine Ausfahrt, ferner kann ein Auto entweder einfahren oder ausfahren, wie in Abbildung 6.10 gezeigt.

Diese vereinfachte System aus der Realität lässt sich nun durch ein Markov-Kette darstellen.

Abbildung 6.11 zeigt nun ein Modell für das Parkhaus.

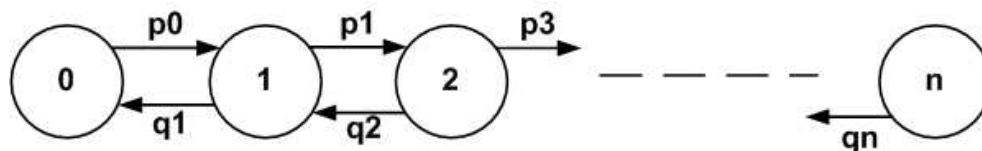


Abbildung 6.11: Vorgestelltes Parkhaus als Markov-Kette

Hier repräsentieren die Zustände **0**, **1**, **2** bis **n** die Anzahl der Stellplätze im Parkhaus. Die Pfeile mit Gewichtung repräsentieren Wahrscheinlichkeiten des Übergangs von von einem in den nächsten.  $p_x$  ist die Wahrscheinlichkeit für die Einfahrt eines Autos,  $q_y$  die, für das Verlassen des Parkhauses durch ein Auto. Da ein Auto entweder nur einfahren **oder** ausfahren kann, hängt der Folgezustand, wie schon beschrieben, nur vom gegenwärtigen ab.

Um diese Markov-Kette nun zu analysieren und ein Ergebnis zu bekommen gibt es verschiedene Möglichkeiten.

Man kann mit Hilfe der grafischen Repräsentation des Problems eine Übergangsmatrix erstellen, die dann bei Berechnung die angibt, welcher Zustand mit welcher Wahrscheinlichkeit eingenommen wird.

Diese könnte wie folgt aussehen:

$$P = \begin{pmatrix} 1 - p_0 & p_0 & 0 & 0 & \dots \\ q_1 & 1 - (q_1 + p_1) & p_1 & 0 & \dots \\ 0 & q_2 & 1 - (q_2 + p_2) & p_2 & \dots \\ 0 & 0 & q_3 & 1 - (q_3 + p_3) & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}$$

Die Übergangswahrscheinlichkeiten bzw. Verweilwahrscheinlichkeiten der einzelnen Zustände sind in der Matrix aufgeführt.  $1 - p_0$  in der Ecke oben links stellt die Wahrscheinlichkeit des Verweilens des Systems in Zustand 0 dar, wenn sich das System im Zustand 0 befindet.  $p_0$  ist die Wahrscheinlichkeit des Übergangs des Systems von Zustand 0 in Zustand 1, heißt, dass dann ein Auto in das Parkhaus einfährt und einen Parkplatz belegt.

Eine weitere Möglichkeit zur Lösung von Markov-Ketten sind Differentialgleichungen. Zur weiteren Vertiefung sei auf [Ave98], Kap. 3 in [Hav01], Kap. 31.1 in [Jai91] und Kap. 7 in [Lan92], verwiesen.

Ferner gibt es Simulations-Tools, die eine Analyse solcher Modelle zulassen. Auf diese wird in Kap. 5 [Bre02] näher eingegangen.

Markov-Ketten und Markov-Prozesse haben eine weitere wichtige Bedeutung in der Modellierung von Systemen. Sie werden zur Lösung von Warteschlangensystemen und Petri-Netzen herangezogen, da diese unter gewissen Voraussetzungen spezielle Markov-Ketten sind.

Auf diese wird in den folgenden beiden Kapiteln eingegangen.

## 6.5 Einführung in die Warteschlangentheorie (Queueing Theory)

In Computersystemen nutzen und teilen sich Software und Hardware die Systemressourcen, wie Datenbus, Prozessor oder Speicher. Da aber beispielsweise der Prozessor nur eine Anforderung bearbeiten kann, warten alle, die diesen auch benötigen, in Warteschlangen. Die Warteschlangentheorie hilft dabei, die Wartezeit, die verschiedenste Anfragen in einem System haben können, zu bestimmen. Aus diesen kann die Antwortzeit bestimmt werden, und somit auch die Zeit errechnet werden, die eine Anfrage im System verweilt. Deshalb haben Warteschlangenmodelle eine große Bedeutung in der Leistungsanalyse.

### 6.5.1 Einbediener-Warteschlangenmodell (Single-server queueing models)

In diesem Abschnitt werden zunächst Wartemodelle betrachtet, die aus genau einer “Station” bzw. “Bediener” bestehen. Ein (Ein-Stationen-)Wartemodell besteht aus einem

- Warteraum (Warteschlange, Queue) und einer
- Bedieneinheit (Prozessor, Server)

Dies ist in der Abbildung 6.12 dargestellt.

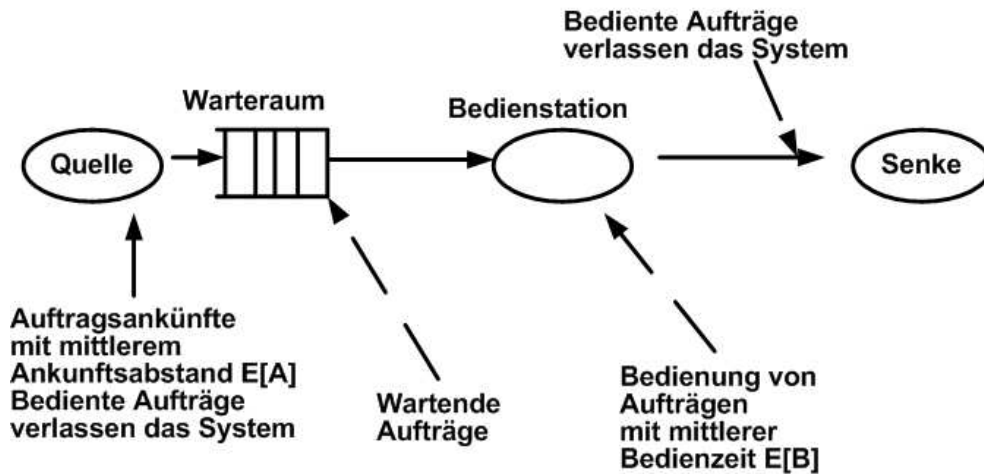


Abbildung 6.12: Einfaches Warteschlangenmodell eines Bedienersystems

Eintreffende Aufträge (oder Kunden, tasks, etc.) gelangen zunächst in den Warteraum, um evtl. nach einer Wartezeit von der Bedieneinheit abgefertigt zu werden. Fertig bediente Aufträge verlassen schließlich das System. Dieses Modell ist eine starke Abstraktion der Realität, aber gerade aus diesem Grund ist es sehr vielseitig einsetzbar. Zur Charakterisierung der Ankunftsverhaltens und der Bedienwünsche von Aufträgen verwendet man stochastische Verteilungen. Der Ankunftsabstand zweier unmittelbar aufeinander folgender Aufträge wird durch eine Zufallsvariable definiert, die einer bestimmten stochastischen Verteilung genügt. Auch die Bedienzeit wird als Zufallsvariable betrachtet. Für eine Analyse zur Ermittlung von Leistungsmaßen verwendet man oft nur die Mittelwerte der Verteilungen und außerdem begnügt man sich mit sogenannten stationären Lösungen, die das Verhalten für “lange” Beobachtungszeiträume beschreiben.

### Die Kendall-Notation

Zur Klassifikation von einfachen Wartesystemen benutzt man die sog. Kendall-Notation.

$A / B / m / K / Q / BS$

- A Verteilung der Zwischenankunftszeiten
- B Verteilung der Bedienzeiten
- m Anzahl identischer Bedienstationen ( $m = 1$ )
- K Aufnahmekapazität des Systems
- Q Anzahl der Aufträge in der Quelle ( $Q < \infty$ )
- BS Bedienstrategie (falls Bedienstrategie  $\neq$  First come first serve (FCFS))

Meist werden solche Systeme in einer Kurzform beschrieben, nämlich

$A / B / m$

wobei ein unbegrenzter Warteraum ( $K = \infty$ ) und eine unbegrenzte Quelle ( $m = \infty$ ) sowie die FIFO-Strategie vorausgesetzt wird. Für A und B werden meist die folgenden Verteilungen verwendet.

Beispiele für Verteilungen:

$G$	“Generelle” Verteilung (allgemeine Bedienzeitverteilung)
$M$	Negativ-Exponentielle Verteilung ( $M =$ Markovsch)
$E_k$	Erlang-k-Verteilung
$D$	Deterministische Verteilung (konstante Zeiten)
$H_k$	Hyper-Exponentielle Verteilung mit k Phasen
$COX_k$	Cox-Verteilung mit k Phasen

Zur näheren Beschreibung der Verteilungen seien hier auf einschlägige Werke der Wahrscheinlichkeitstheorie verwiesen.

### Beispiel Parkhaus

Das im Kapitel 5 schon vorgestellte Parkhaus-Beispiel kann auch mit Hilfe der Warteschlangenmodelle dargestellt werden, wie es in Abbildung 6.13 zu sehen ist.

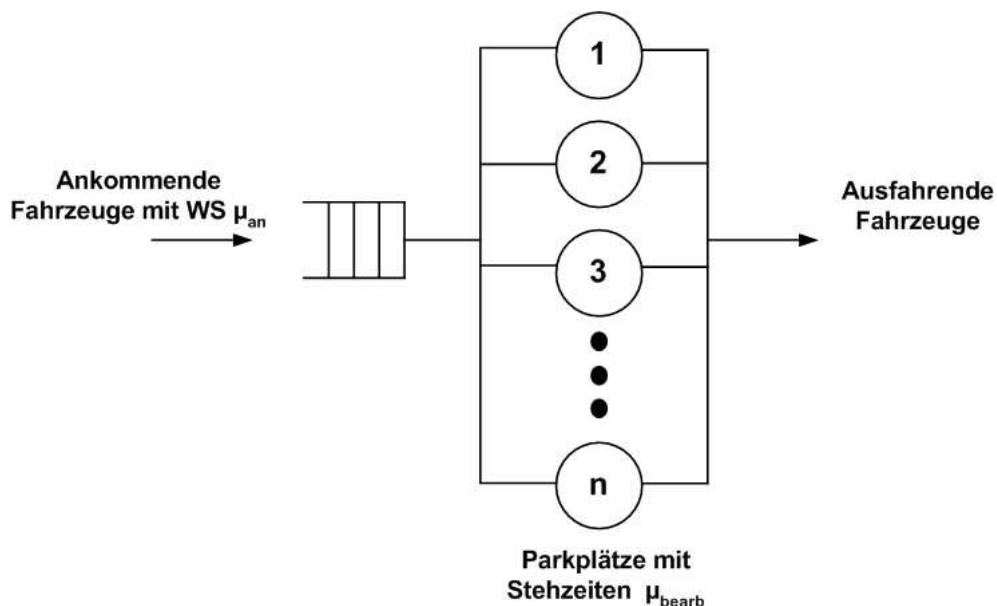


Abbildung 6.13: Darstellung des Systems Parkhaus als Warteschlangenmodell

Das Parkhaus in der Kendall’schen Notation könnte als  $\mathbf{M} / \mathbf{M} / \mathbf{n}$ -Warteschlangenmodell gesehen werden, sofern man annimmt, dass Zwischenankunftszeit  $\mu_{an}$  und Bearbeitungszeit  $\mu_{bearb}$  exponential verteilt sind, was das jeweilige  $\mathbf{M}$  in der Notation zeigt.  $\mathbf{n}$  bezeichnet die Anzahl gleicher Bedienstationen des Modells.

Bei einer Analyse dieses Systems kann man unter anderem folgende Ergebnisse erhalten:

1. Parameter:

$\mu_{an}$	=	Zwischenankunftszeit parkwilliger Nutzer
$\mu_{bearb}$	=	Bearbeitungszeit an den Bedienstationen
$n$	=	Anzahl der Stellplätze im Parkhaus

2. Durchlaufintensität:  $\rho = \mu_{an} / (n\mu_{bearb})$

3. Das System ist solange stabil, wenn die Durchlaufintensität  $\rho$  kleiner 1 ist.

4. Die Wahrscheinlichkeit, dass kein Auto im Parkhaus ist :

$$p_0 = \left[ 1 + \frac{(n\rho)^n}{n!(1-\rho)} + \sum_{i=1}^{n-1} \frac{(n\rho)^i}{i!} \right]^{-1}$$

5. Die Wahrscheinlichkeit, dass  $m$  Parkplätze belegt sind:

$$p_m = \begin{cases} p_0 \frac{(n\rho)^m}{m!} & , \quad m < n \\ , & \\ p_0 \frac{\rho^m n^n}{n!} & , \quad m \geq n \end{cases}$$

Weitere Kenngrößen, die aus den Angaben berechnet werden können, findet man bei [Jai91], S.528/529. Dort sind auch weitere Arten "Einbediener-Warteschlangenmodelle", teilweise mit Puffern bei den Warteschlangen, diskutiert.

### 6.5.2 Warteschlangennetzwerke (Queueing network models)

Im vorherigen Abschnitt wurden "Einbediener-Warteschlangenmodelle" behandelt. In der Praxis bestehen viele Systeme aus mehreren solchen Einheiten, die jeweils eigene Warteschlangen haben. Als Beispiel sei nur ein Kommunikationssystem genannt, in dem Pakete über Router und Kabelverbindungen von der Quelle zum Ziel übertragen werden.

Nach kurzer Einführung in die Terminologie der Warteschlangennetzwerke, werden drei verschiedene Netzwerktypen vorgestellt, wovon auf einen näher eingegangen wird.

**Warteschlangennetzwerke**, engl.: **queueing networks**, im folgenden **QNs** abgekürzt, bestehen aus einer Anzahl von Warteschlangenstationen  $1, \dots, M$ , die untereinander verbunden. Individuelle Warteschlangenstationen oder Knoten sind voneinander unabhängig, sind aber in der Hinsicht miteinander verbunden, dass der Eingangsstrom von "Kunden" eines Knotens sich aus dem Ausgangsstrom eines oder mehrerer Knoten zusammensetzt. Es wird hier von einem **offenen QN** ausgegangen, da ein Internet-Routing-System eher dieser Art entspricht.

- Zur Betrachtung geschlossener QNs sei hier auf [Hav01], Kap. 11 verwiesen. -

Ferner geht man von einer Quelle, von der die Kunden das System betreten, welche unerschöpflich ist, und in die sie nach erhaltener Leistung das System wieder verlassen. Eine formale Art, ein QN zu beschreiben ist ein gerichteter Graph, wobei die Knoten die Warteschlangenstationen repräsentieren und die gerichteten Kanten das festlegen, welchen Weg der Kunde durch den Graph nehmen muss, um von Knoten zu Knoten zu gelangen. Die Kanten, die auch mehrfach aus einem Knoten herausführen können, sind evtl. mit einer Wahrscheinlichkeit belegt, mit der ein Kunde den Weg der Kante entlang wählt, oder mit Zwischenankunftsrate. Eingang und Ausgang des QN sind als besondere Knoten gekennzeichnet und werden meist mit 0 bezeichnet.

Es gibt verschiedene Arten von QNs:

- **Feed-forward QNs (FFQNs)**

Sequentielle Anordnung von Warteschlangenstationen, es existieren keine Zyklen

- **Jackson QNs**  
Sequentielle Anordnung wird aufgehoben, Jobs können an schon besuchten Stationen nochmals ankommen.
- **QNA Warteschlangennetzwerk-Klasse**

Zur Vertiefung der ersten beiden Warteschlangennetzwerke sei auf [Hav01], Kap. 10 verwiesen. Auf letztere Klasse wird noch kurz eingegangen, um die Bedeutung dieser zu zeigen.

In den ersten beiden Warteschlangennetzwerken werden Ankunftszeit auf einen Poisson-Prozess und Bearbeitungszeit auf einen negativen-exponential-Prozess eingeschränkt. QNA, Queuing Network Analyzer, ist eine Methode zur Modellierung von Warteschlangennetzwerken, die eine schnelle Analyse großer offener QNs mit festen Routing-Wahrscheinlichkeiten und FCFS als Bedienstrategie erlaubt. Die wichtigste Eigenschaft, die QNA besitzt, ist, dass Zwischenankunftszeiten auch anders als Poisson-verteilt sein können, sowie die Bedienzeit nicht exponential-verteilt sein muß. Bei QNA sind alle Ankunfts-Prozesse Erneuerungsprozesse, die durch die ersten beiden Ereignisse charakterisiert sind, wobei aufeinanderfolgende Ankünfte immer noch unabhängig voneinander sind. Ähnlich verhält es sich bei der Bedienzeit; im besonderen können konstante Bedienzeiten annähernd nachgebildet werden.

Mit QNA ist die Rechen-Komplexität linear abhängig von der Anzahl der Warteschlangenstationen. Sind die Gleichungen des Routings gelöst, können alle Warteschlangenstationen separat betrachtet werden. Dieser Vorteil hat auch Nachteile, wie immer. Die Zerlegung des Gesamt-Warteschlangennetzwerkes in einzelne Warteschlangenstationen ist nur eine Annäherung. Dies gilt es immer zu berücksichtigen. Hier nun eine Zusammenfassung der Annäherungen, die bei QNA getroffen werden:

1. Leistungswerte für das Netzwerk als gesamtes erhält man unter der Annahme, dass den Knoten stochastisch unabhängige Flußparameter zugeordnet werden
2. Durchflußströme sind Erneuerungsprozesse, wie oben beschrieben
3. Es wird eine GI / G / m - Annäherung für die Leistung an einem Knoten verwendet, die aber nur für Warteschlangenstationen mit M / M / 1 oder M / G / 1 exakt ist
4. Einige Gleichungen basieren nicht auf einer exakten mathematischen Herleitung, sondern auf Näherungsergebnissen, die man manchmal erst nach intensiver Simulation erhalten hat.

Zu mathematischen Lösungsmethoden des QNA wird auf [Hav01], Kap. 10 verwiesen, der eine detaillierte Vertiefung erlaubt.

Zusammenfassend kann man sagen, dass die Modellierung mit Warteschlangennetzwerken Möglichkeiten der Analyse von Routing- und Kommunikationsnetzwerken bietet. Ein Beispiel der Modellierung eines Telekommunikationsnetzwerkes ist bei [Hav01], S.220-225 beschrieben.

## 6.6 Petri-Netze

Dieses Kapitel beschäftigt sich zunächst mit der “Theorie” von Petri-Netzen. Verschiedene Typen derselben werden hier vorgestellt und werden anschließend die für eine Modellierung von Internet-Routing-Systemen relevanten Formen näher betrachtet. Dies wird durch Vorstellung von Anwendungen dieser Art von Petri-Netzen und die Analyse letzterer etabliert. Zum Schluß werden die Vorteile zur Modellierung, wie auch die Nachteile im letzten Abschnitt gegenübergestellt.

### 6.6.1 Petri-Netze – Entstehung und Ausprägung

Im Jahre 1962 betrachtete Carl Adam Petri in seiner Dissertation [Pet62] das Problem, zusammenwirkende (co-operating), konkurrierende (concurrent) oder solche Prozesse, die in Wettbewerb zu einander stehen (competing), in einem graphischen Modellformalismus darzustellen. (Kap. 2.3, S. 31 [Lin98])

Er stellte diesen Formalismus in Form eines bipartiten Graphen dar.

**Definition 6.6.1 (Bipartiter Graph)** *Ein Graph heißt **bipartit**, wenn*

- *die Knotenmenge  $V$  aus zwei disjunkten Teilmengen  $V_1$  und  $V_2$  besteht, d.h.:*

$$V = V_1 \cup V_2 \text{ und } V_1 \cap V_2 = \emptyset$$

- *es nur Kanten  $(v_1, v_2) \in E$  bzw.  $(v_2, v_1) \in E$  gibt, mit:*

$$v_1 \in V_1 \text{ und } v_2 \in V_2, \text{ d.h.:}$$

$$E \subseteq V_1 \otimes V_2 \cup V_2 \otimes V_1$$

Folie 336, [Bor00]

Die Modelle, die Petri in seiner Dissertation zum ersten Mal beschrieb, nennt man **Petri-Netze**.

**Definition 6.6.2 (Petri-Netz)** *Ein Petri-Netz ist ein Bipartiter Graph*

$G = (S; T; E; M_0)$  *mit endlicher Knotenmenge  $V = S \cup T$  und endlicher Kantenmenge  $E \subseteq S \otimes T \cup T \otimes S$ .*

*Die Knoten in  $S$  heißen **Stellen** (auch: Plätze) und repräsentieren Zustände, Bedingungen usw. (graphisch durch Kreise dargestellt).*

*Die Knoten in  $T$  heißen **Transitionen** (auch: Übergänge oder Hürden) und repräsentieren Zustandsübergänge, Aktivitäten usw. (graphisch durch Kästen bzw. Striche dargestellt).*

*Die Kanten in  $E$  repräsentieren kausale und zeitliche Abhängigkeiten. Sie werden im allgemeinen durch Pfeile dargestellt, die auch Wertigkeiten besitzen können.*



In  $M_0$  sind die Anfangsmarkierungen aufgeführt, also in welchen Stellen sich Token befinden, die angeben, welcher Zustand aktuell bzw. aktiv ist.

Folie 337 [Bor00]

### Beispiel 6.1.1

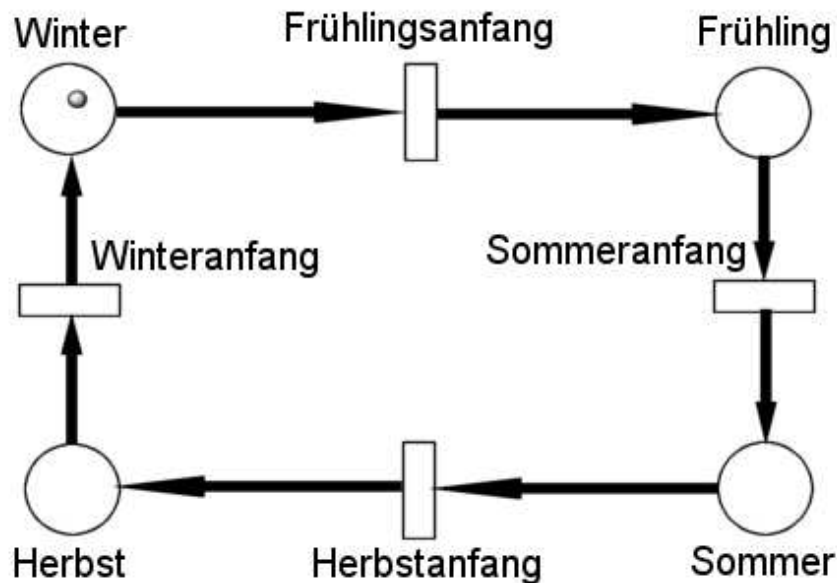


Abbildung 6.14: Modell eines zyklischen, sequentiellen Prozesses (entnommen: [Bor00] Folie 338)

Um ein einfaches Petri-Netz als Beispiel anzuführen, sei im folgenden der Wechsel der Jahreszeiten dargestellt. Dies ist in Form eines **“Stellen-Transitions-Netzes”** repräsentiert.

Wie in der Definition 6.6.2 dargestellt, sind die Jahreszeiten als Zustände während des Jahres als Kreise dargestellt. Die Tage im Jahr, die den Wechsel der Jahreszeiten festlegen sind als Übergänge in Form der Kästen dargestellt. Ein Token, der sich im Kreis der Jahreszeit “Winter” befindet, markiert den aktuellen, derzeit aktiven Zustand. Wird der Tag des Frühlingsanfangs im Kalender erreicht, so ist die Bedingung für den Jahreszeitenwechsel erfüllt, und der Token wandert in den Zustand “Frühling”, um die dann aktuelle Jahreszeit zu bezeichnen. Dies Beispiel ist ein sehr einfaches, um die Schematik von Petri-Netzen darzustellen.

Petri-Netze in Form der Stellen-Transitions-Netze, wie hier eingeführt, dienen der funktionalen Analyse, d.h. sie bieten Möglichkeiten zur Überprüfung gewisser Korrektheitseigenschaften von Modellen nebenläufiger und paralleler Systeme. Jedoch sind die Möglichkeiten eines einfachen Stellen-Transitions-Netzes beschränkt, so können beispielsweise die Token nicht unterschieden werden, sowie eine Berücksichtigung eines Zeitintervalls bei Aktivierung der Transition ist auch nicht gegeben.

Deshalb gibt es verschiedene Erweiterungen der Petri-Netze.

Im Folgenden werden nun diese Erweiterungen und Ausprägungen zu Petri-Netzen vorgestellt.

## Farbige Petri-Netze

Da in einfachen Petri-Netzen die Tokens nicht unterscheidbar sind, werden Systeme schnell komplex, wenn man ein reales System modellieren möchte. Deshalb wurden die klassischen Petri-Netze zu **farbigen Petri-Netzen**, engl.: **Colored Petri Nets**, im folgenden CPNs abgekürzt, erweitert. So kann man Token mit verschiedenen Attributen belegen. Die Transitionen sind dann auch farbig, reagieren bzw. feuern bei bestimmten Farbvorraussetzungen bei den darvorliegenden Stellen.

## Zeitbehaftete Petri-Netze

Anfang der 70er Jahre entstanden die ersten **zeitbehafteten Petri-Netz-Modelle**, engl.: **Time-augmented Petri Nets**, die weitgehend auf eine Erweiterung der Petri-Netz-Struktur verzichten und so eine funktionale Analyse prinzipiell zulassen. Ferner wird in vielen Fällen die Zeit so in die Beschreibung integriert (Markov-Modelle), daß algebraisch-numerische oder für Teilklassen algebraisch-analytische Methoden, wie sie aus dem Bereich der Warteschlangentheorie bekannt sind, eingesetzt werden können.

Allen Ansätzen für zeitbehaftete Petri-Netze ist gemeinsam, daß sie die Schaltregel insofern modifizieren, daß aktivierte Transitionen nicht mehr feuern können, sondern sofort oder nach einer festgelegten Zeitdauer feuern müssen. Im Konfliktfalle wird eine Transition i.a. zufällig ausgewählt.

Nahezu alle zeitbehafteten Petri-Netz-Modelle lassen sich in zwei große Klassen einteilen, wobei als Unterscheidungsmerkmal dient, ob die Zeitdauern mit den Transitionen oder den Stellen verknüpft werden.

Wir betrachten im Folgenden zuerst

- Zeit an den Transitionen, danach
- Zeit an den Stellen.

### Zeitbehaftete Petri-Netze mit Zeitverbrauch an den Transitionen

Ein Großteil der zeitbehafteten Petri-Netz-Modelle bindet die Zeit an die Transitionen, d.h. es muß eine gewisse Zeit zwischen Aktivierung und Schaltung einer Transition vergehen. Diese Modelle lassen sich nach folgenden zwei Kriterien klassifizieren.

- Schaltregel (pre-selection-Modelle oder Race-Modelle)
- Art des Zeitfortschritts (deterministisch, deterministische Intervalle, stochastisch)

#### 1. Klassifizierung nach der Schaltregel

- Pre-Selection-Modelle (Reservierung von Marken):

Ist eine Transition aktiviert, so werden die zur Schaltung benötigten Marken auf den Eingabestellen als reserviert gekennzeichnet. Nach einer bestimmten Zeitdauer feuert die Transition, entfernt dabei die von ihr auf den Eingabestellen reservierten Marken und belegt die Ausgabestellen mit Marken. Stehen mehrere Transitionen in Konflikt, so wird zufällig entschieden, welche Transition zuerst die Marken reservieren darf. Reservierte Marken dürfen von anderen Transitionen nicht verwendet

werden.

- Race-Modelle (Wettlauf aktivierter Transitionen):

Ist eine Transition aktiviert, so vergeht eine gewisse Zeitspanne bis die Transition feuert, sofern sie dann noch aktiviert ist. Stehen mehrere Transitionen in Konflikt, so feuert diejenige, welche am schnellsten ist, daher der Name dieser Klasse

## 2. Klassifizierung nach Zeitfortschritt

- deterministisch (konstante Schaltzeiten):

Jeder Transition wird eine Konstante ( $> 0$ ) zugeordnet, die die Zeit zwischen Aktivierung und Schaltung angibt.

- deterministische Zeitintervalle (Minimale und maximale Schaltdauer):

Jeder Transition wird ein Zeitintervall  $[a, b]$  zugeordnet. Eine Transition muß mindestens  $a$  Zeiteinheiten aktiviert bleiben, bevor sie die Marken reservieren (bei Pre-Selection- Modellen) bzw. schalten (bei Race-Modellen) darf.

- stochastisch (zufällige Schaltdauer):

Jeder Transition wird eine kontinuierliche Zufallsvariable zugeordnet. Die Realisierungen dieser Zufallsvariablen geben die Zeit zwischen Aktivierung und Feuerung an.

siehe auch [MüC00] Kap. 2

Durch Mischung oben genannter Merkmale lassen sich verschiedene Modellarten ableiten. Im folgenden werden drei näher betrachtet.

### Stochastische Petri-Netze (SPN)

SPNs wurden von Natkin und Molloy Anfang der 80er Jahre eingeführt. SPNs ordnen jeder Transition eine exponentiell verteilte Schaltdauer zu. Für jede Transition wird die Schaltdauer durch Angabe einer Rate, der sog. Schaltrate (auch: Feuerungsrate) spezifiziert.

**Definition 6.6.3 (Stochastische Petri-Netze)** *Ein stochastisches Petri-Netz basiert auf einem Stellen-Transitions-Netz, bei dem jeder der Transitionen  $t_i, i = 1, \dots, m$ , eine exponentiell verteilte Schaltrate  $\lambda_i$  zugeordnet ist. Damit kann ein Stochastisches Petri-Netz dargestellt werden als  $SPN = (S, T, C^-, C^+, m_0, \Lambda)$ , wobei  $ST = (S, T, C^-, C^+)$  ein Stellen-Transitions-Netz,  $m_0$  eine Anfangsmarkierung und  $\Lambda = (\lambda_1, \lambda_2, \dots, \lambda_m)$  ein Vektor von Schaltraten ist.*

Eine Rate kann markierungsabhängig sein, in diesem Fall schreibt man  $\lambda_i(m_i)$ , wobei  $m_i$  die aktuelle Markierung darstellt. Die mittlere Schaltdauer einer Transition  $t_i$  in Markierung  $m_i$  ist gegeben durch den Kehrwert der Schaltrate  $(\lambda_i(m_i))^{-1}$ .

Mit dieser Erweiterung läßt sich unser Beispiel des Parkhauses auch mit Hilfe von Petri-Netzen realisieren.

In Abbildung 6.15 ist das Parkhaus als Stellen-Transitions-Netz realisiert.

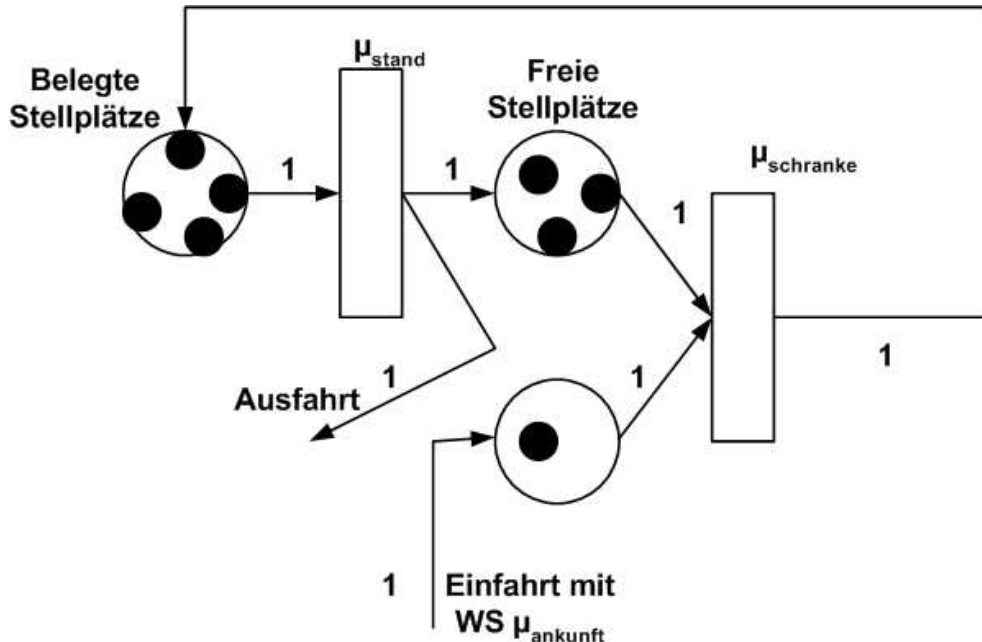


Abbildung 6.15: Darstellung des Systems Parkhaus in Form eines Stellen-Transitions-Netzes

Ein Auto fährt mit der Wahrscheinlichkeit  $\mu_{ankunft}$  in das Parkhaus ein. Sofern noch Parkplätze frei sind, und ein Auto vor der Schranke steht, schaltet die Transition mit der Feuer-Rate  $\mu_{schranke}$ , die das Öffnen der Schranke und die Einfahrzeit des Auto in das Parkhaus zeitlich darstellt. Das Auto befindet sich nun auf dem Stellplatz. Die Transition, die auf die Stellplätze folgt feuert mit der Wahrscheinlichkeit  $\mu_{stand}$ , was eine Standzeit und die Ausfahrtzeit nachbildet. Es wird hierbei ein Token in die Stelle “Freie Parkplätze” zurückgeführt, da ja bei Aktivierung der Transition ein Stellplatz freigeworden ist.

Weitere Beispiele zu SPNs können in [MüC00] Kap. 2.3 und [BaK96], Kap. 7 eingesehen werden, zudem werden darin mathematische Analyse-Verfahren dargestellt. Hierzu siehe auch [Hav01], Kap. 14.

### Generalized Stochastic Petri Nets (GSPN)

Um den für die quantitative Analyse relevanten Zustandsraum einzuschränken, ohne die funktionale Beschreibung des Netzes zu beeinflussen, wurden SPNs Mitte der 80er Jahre verallgemeinert zu den Generalized Stochastic Petri Nets (GSPN).

Hier eine kurze Einführung und Definition:

Bei den GSPNs wird jetzt nicht mehr allen Transitionen eine exponentiell-verteilte Schaltdauer zugeordnet, sondern ein Teil der Transitionen kann als “zeitlos” (engl. immediate) definiert werden.

- Zeitlose Transitionen schalten (falls sie schalten!) unmittelbar nach Aktivierung, d.h. ohne Zeitverbrauch.
- Die anderen Transitionen werden weiterhin mit einer exponentiell-verteilten Schaltdauer assoziiert (timed transitions). Graphisch werden sie (meist) wie folgt unterschieden, wie in Abbildung 6.16 dargestellt.



Abbildung 6.16: Grafische Darstellung der GSPN-Transitionen (entnommen [MüC00] Kap. 2.4)

Zeitlose Transitionen  $t_j$  können mit einem Gewicht  $w_j$  (=Schalthäufigkeit) versehen werden, das im Falle von Schaltkonflikten Verwendung findet. Dieses Gewicht kann markierungsabhängig sein, d.h.  $w_i(m_k)$  beschreibt die Schalthäufigkeit der Transition  $t_i$  in Markierung  $m_k$ .

### Konflikte in GSPNs

In GSPNs kommen unterschiedliche Konfliktsituationen vor. Der einfachste Fall betrifft den Konflikt zwischen zeitbehafteten und zeitlosen Transitionen. Den Konflikt zwischen zeitbehafteten Transitionen kennen wir bereits von SPN. Die Konfliktauflösung unter den zeitlosen Transitionen kann durch die zusätzlich angegebenen Gewichte  $w_i$  differenzierter als bei Stellen-Transitions-Netzen betrachtet werden.

- Zeitlose Transitionen haben Priorität vor zeitbehafteten Transitionen. Zeitlose Zustände werden sofort, d.h. mit Aufenthaltsdauer 0 verlassen.
- Falls zeitbehaftete (exponentielle) Transition miteinander in Konflikt stehen, so wird wie bei den SPNs verfahren.
- Falls zeitlose Transitionen miteinander in Konflikt stehen, so werden mit Hilfe der Schalthäufigkeiten  $w_j(m_k)$  der aktivierten Transitionen Schaltwahrscheinlichkeiten berechnet.

GSPN lassen sich mathematisch durch Rückführung auf Markov-Ketten analysieren. Zum Prinzip dieser mathematischen Analyse von GSPN-Modellen sei an dieser Stelle auf [MüC00], Kap. 2.4.4 und [BaK96], Kap. 8 verwiesen.

### 6.6.2 Deterministische und stochastische Petri-Netze (DSPN)

Deterministische und Stochastische Petri-Netze wurden in den 80-er Jahren als Erweiterung der GSPN eingeführt. DSPN erlauben eine Verbindung zeitbehafteter Transitionen entweder mit deterministischer oder exponential-verteilter Feuer-Verzögerung. Deshalb sind DSPN gut dazu geeignet, Systembesonderheiten wie “time-outs”, Übertragungsverzögerungen oder die Zeit, die zum Neu-Booten eines Prozessors benötigt wird, die mit konstanten Verzögerungen verbunden sind. Erste Anwendungen fanden DSPN bei der Leistungsmodellierung eines Ethernet-Bus-Lokal-Netzwerk [ACF87].

DSPN enthalten somit folgende Komponenten.

- als Grundmodell die Stellen-Transitions-Netze (mit einfarbigen Marken), wobei die Menge der Transitionen  $T$  gegeben ist durch  $T = T^Z \cup T^E \cup T^D$ , wobei
- $T^Z$  zeitlose Transitionen
- $T^E$  zeitbehaftete Transitionen (mit exponentiell verteilter Schaltdauer)
- $T^D$  deterministische Transitionen sind
- für alle  $t \in T^Z$  können Prioritäten angegeben werden
- für alle  $t \in T^Z$  können markierungsabhängige Gewichte  $w_i(m_k)$  angegeben werden
- es gibt Verbotskanten (inhibitor arcs)

Abbildung 6.17 zeigt die grafische Repräsentation der Transitionen.

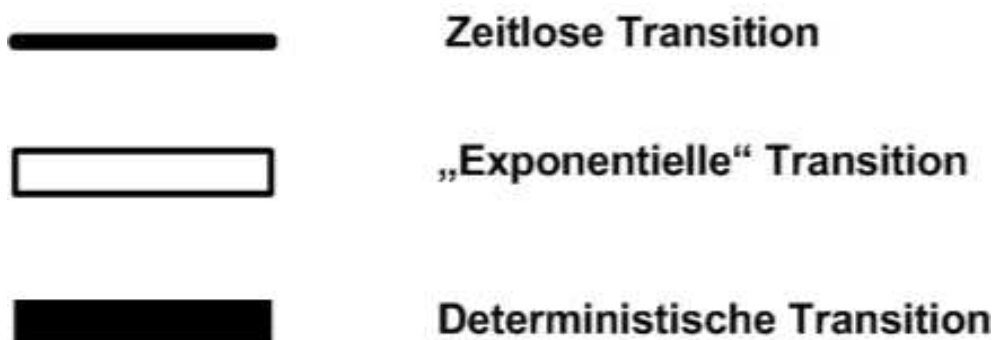


Abbildung 6.17: Grafische Darstellung der DSPN-Transitionen (entnommen [MüC00] Kap. 2.5)

Bei den Schaltregeln gelten folgende Vereinbarungen:

- zeitlose Transitionen haben Priorität vor zeitbehafteten Prioritäten
- im Falle von mehreren aktivierten zeitlosen Transitionen wird die Schaltreihenfolge durch die Transitionsprioritäten festgelegt
- bei gleichpriorigen zeitlosen Transitionen werden unter Verwendung der Gewichte wie Schaltwahrscheinlichkeiten berechnet.

Die Gewichtung von Transitionen wird wie bei den GSPN gehandhabt. Durch die Prioritätenangaben an den Transitionen können Schaltkonflikte beseitigt werden.

Zur numerischen Analyse von DSPN können wiederum Markov-Ketten herangezogen werden. Bei [Lin98] sind diese Methoden zur Analyse in Kapitel 3 beschrieben und zwar für DSPN mit und ohne nebenläufigen deterministischen Transitionen. Dort kann man weitere tiefgehendere Anwendungen und Beispiele zu DSPN finden, womit hier darauf verwiesen sei.

## **6.7 Zusammenfassung und Einordnung**

In den vorgestellten Möglichkeiten der Leistungsmodellierung wurde herausgestellt, dass die Modellierung eines Internet-Routing-Systems mit den Mitteln der Warteschlangentheorie und denen der Petri-Netze möglich ist, und teilweise sogar versucht und durchgeführt wurde. Aus der vorliegenden Literatur geht hervor, dass Ansätze der Analyse von verschiedenen Kommunikationsnetzwerken durchgeführt wurden, aber nicht, wie komplex eine Analyse eines Internet-Routing-Systems sein kann, wenn man numerische Ansätze mit Petri-Netzen oder Warteschlangennetzen wählt.

Die Einordnung dieser Seminararbeit in den Kontext des Seminars wird aus dem Titel klar: Durch Leistungsmodellierung sollen Internet-Routing Strategien in Modellen nachgebildet und anschließend bewertet werden. In der Seminararbeit wurden in der Hauptsache zwei Modellierungsmittel vorgestellt, die häufig zur Analyse von Systemen herangezogen werden. Diese weisen die Eigenschaften auf, die zur Modellierung von Internet-Routing-Strategien benötigt werden. Vor allem sind die vorgestellten Systeme mit nur geringen Einschränkungen auf Markov-Modelle zurückführbar, was eine numerische Analyse ermöglicht.

Somit kann auch eine Analyse dieser Strategien durchgeführt werden, was aber in einem weiteren Projekt realisiert werden muß, vor allem in der Hinsicht, ob es mit den vorgestellten Mitteln sinnvoll ist und in welcher Größe die Analyse durchgeführt werden kann.

# Literaturverzeichnis

- [ACF87] M. AJMONE MARSEN, G. CHIOLA AND A. FUMAGALLI: “An Accurate Performance Model of CSMA/CD Bus LAN”, in: G. Rozenberg (Ed.) *Advances in Petri Nets 1986, Lecture Notes in Computer Science 266*, pp. 146-161, Springer 1987
- [Ave99] RUDOLF AVENHAUS: *Skript zur Vorlesung “Quantitative Modelle für Rechen- und Kommunikationssysteme”*, Universität der Bundeswehr München, 1999
- [BaK96] F. BAUSE, P. KRITZINGER: *Stochastic Petri Nets - An Introduction to the Theory*, Vieweg & Sohn, Braunschweig/Wiesbaden 1996
- [Bor00] UWE M. BORGHOFF: *Skript zur Vorlesung “Einführung in die Informatik II”*, UniBw München 2000
- [Bch02] SÖNKE BRECHT: *Lastcharakterisierung für Netzverkehr. Seminararbeit im Rahmen des Seminars “Bewertung von Internet-Routing Strategien”*, durchgeführt durch Institut 4 der Fakultät für Informatik der Universität der Bundeswehr München, Neubiberg 2002
- [Bre02] ROBERT BRENDEL: *Performability-Modellierung. Seminararbeit im Rahmen des Seminars “Bewertung von Internet-Routing Strategien”*, durchgeführt durch Institut 4 der Fakultät für Informatik der Universität der Bundeswehr München, Neubiberg 2002
- [Hav98] BOUDEWIJN R. HAVERKORT: *Performance of Computer Communication Systems*. John Wiley & Sons, Chichester 1998
- [Jai91] RAJ JAIN: *The art of computer systems performance analysis: techniques for experimental design, measurement, simulation and modelling*. John Wiley & Sons, Inc. New York 1991
- [Käm90] S. KÄMPER: *PERGOS: Ein Konzept zur Entwicklung eines graphischen, objektorientierten Modellbildungs- und Simulationswerkzeuges auf der Basis von Petri-Netzen*. Dissertation, Fachbereich Informatik, Universität Hamburg 1990
- [Lan92] HORST LANGENDÖRFER: *Leistungsanalyse von Rechensystemen. Messen, Modellieren, Simulation*. Carl Hanser Verlag München Wien 1992
- [Leh00] AXEL LEHMANN: *Vorlesungsskript zu “Methoden und Werkzeuge zur Leistungsanalyse”*, Universität der Bundeswehr München, HT 2000
- [Lin98] CHRISTOPH LINDEMANN: *Performance Modelling with Deterministic and Stochastic Petri Nets*. John Wiley & Sons, Chichester 1998



- [Mey93] MEYER LEXIKONREDAKTION: *Duden Informatik*. Dudenverlag, Mannheim 1993
- [MüC00] BRUNO MÜLLER-CLOSTERMANN: *Skript zur Vorlesung "Systemmodellierung"*, Sommersemester 2000, Universität Essen, 2000
- [Nie77] G. NIEMEYER: *Kybernetische System- und Modelltheorie, System Dynamics*. Franz Wahlen, München 1977
- [Pag91] BERND PAGE: *Diskrete Simulation. Eine Einführung mit Modula-2*. Springer-Verlag, Berlin Heidelberg 1991
- [Pet62] CARL ADAM PETRI: *Kommunikation mit Automaten*. Institut für Instrumentelle Mathematik, Schriften des IIM Nr. 2, Bonn 1962
- [Sch82] B. SCHMIDT: *Die Bestimmung von Konfidenzintervallen in der Simulation stochastischer zeitdiskreter Systeme*. *Elektronische Rechenanlagen*, 24. Jg. 3/1982, S.118-124
- [Sch02] TILO SCHRÖDER: *Zuverlässigkeitsmodellierung*. Seminararbeit im Rahmen des Seminars "Bewertung von Internet-Routing Strategien", durchgeführt durch Institut 4 der Fakultät für Informatik der Universität der Bundeswehr München, Neubiberg 2002



# 7 Zuverlässigkeitsmodellierung

*Tilo Schröder*

## Inhaltsverzeichnis

---

<b>7.1</b>	<b>Einleitung</b> . . . . .	<b>148</b>
<b>7.2</b>	<b>Begriffsdefinitionen</b> . . . . .	<b>149</b>
7.2.1	Zuverlässigkeit und Sicherheit . . . . .	149
7.2.2	Zuverlässigkeitskenngrößen . . . . .	149
7.2.3	Verteilungsfunktionen am Beispiel der Exponentialverteilung . . . . .	152
<b>7.3</b>	<b>Zuverlässigkeitsanalyse - Der Weg zum Modell</b> . . . . .	<b>154</b>
7.3.1	Klassifikation von Modellen . . . . .	155
7.3.2	Der Modellbildungsprozess . . . . .	156
7.3.3	Vorstellung der ersten wichtigen Modellierungstechnik . . . . .	158
<b>7.4</b>	<b>Modellierungstechniken im Vergleich</b> . . . . .	<b>160</b>
7.4.1	Boolesche Modelle . . . . .	160
7.4.2	Fehlerbäume . . . . .	163
7.4.3	Markovketten . . . . .	165
7.4.4	Petri-Netze . . . . .	168
<b>7.5</b>	<b>Abschließendes Beispiel</b> . . . . .	<b>171</b>
<b>7.6</b>	<b>Zusammenfassung und Ausblick</b> . . . . .	<b>173</b>
	<b>Literaturverzeichnis</b> . . . . .	<b>173</b>

---

## Abstract

In order to evaluate the availability of technical devices, it becomes more and more important to make extensive reliability investigations.

Consequently it became an essential method to transform planned or already existing systems in models, in order to consider as much as possible different factors.

The purpose is to produce reliable and fully developed components, what becomes, referring to routing-systems in more and more growing networks, more important than ever.

Therefore, this work has the aim to introduce the term "reliability" with its metrics and with the modelling process in background, to present the most important methods to create models.

## Kurzbeschreibung

Um die Einsatzfähigkeit technischer Geräte umfassend beurteilen zu können, ist es in heutiger Zeit immer wichtiger, Zuverlässigkeitsbetrachtungen umfassend durchzuführen.

Dazu ist es zu einem unumgänglichen Mittel geworden, geplante oder schon existente Systeme in Modelle zu fassen, um möglichst viele Faktoren zu berücksichtigen.

Der Zweck ist, verlässliche und technisch ausgereifte Komponenten herzustellen, was gerade in Bezug auf Routing-Systeme in immer mehr wachsenden Netzwerken immer wichtiger wird.

Ziel dieser Seminararbeit soll somit sein, den Begriff Zuverlässigkeit mit seinen Metriken einzuführen, sowie aufbauend auf den Modellbildungsprozeß die wichtigsten Modellierungsverfahren zu präsentieren.

## 7.1 Einleitung

Der Einzug der Technik in alle Bereiche unseres Lebens hat im Laufe der letzten 50 Jahre das Dasein des Menschen komplett neu definiert.

Sind es zum Beispiel die Autos, die uns täglich zur Arbeit bringen, Flugzeuge, mit denen wir verreisen, oder medizinische Anlagen, die uns ein längeres Leben ermöglichen sollen.

All diesen Errungenschaften liegt jedoch der Gedanke zugrunde, daß sie neben der ihnen abgeforderten Leistung eines sind: **zuverlässig**.

Niemand will wegen kleinerer Defekte auf sein Auto verzichten. Noch größere Auswirkungen haben Fehler aufgrund mangelnder Zuverlässigkeit in sensiblen Bereichen wie Flugzeugen, da hier viele Menschenleben gefährdet werden können.

Doch wie „zuverlässig“ muß ein Flugzeug sein? Genügen hier 99,9% ?

Mit diesen Gedanken haben sich viele Wissenschaftler in den letzten Jahrzehnten beschäftigt und Techniken entwickelt, die Komponente „Zuverlässigkeit“ zu modellieren und mathematisch auswertbar zu machen.

Im Rahmen des Seminars „**Bewertung von Internet-Routing-Strategien**“ sollen deshalb Möglichkeiten untersucht werden, die Zuverlässigkeit speziell von Netzwerksystemen zu betrachten.

Zuvor werden jedoch im *Kapitel 3* die Begriffe „**Zuverlässigkeit**“, „**Sicherheit**“ sowie die dazugehörigen **Kenngrößen** eingeführt.

Beispielhaft für die jeweiligen Verteilungsfunktionen wird die Exponentialverteilung vorgestellt.

Abschließend erfolgt die Definition der **Redundanz** und ihre Bewertung im Rahmen der Zuverlässigkeitserhöhung.

Das *Kapitel 4* stellt die **Klassifikation von Modellen** und den eigentlichen formalen **Modellbildungsprozess** dar.

Als erstes Beispiel für eine Modellierungsform werden Zuverlässigkeitsblockdiagramme hervorgehoben.

Die eigentlichen Verfahren der Modellierung kommen dann erstmalig im *Kapitel 5* zur Geltung.

Hier werden im speziellen **Boolesche Modelle**, **Fehlerbäume**, **Markovketten** und **Petrinetze** gegenüber gestellt.

Vordergründig ist dabei die Vorstellung der Formalismen sowie die Darstellung der Vorteile und Grenzen der jeweiligen Techniken.

Dies erfolgt immer mit Bezug auf das Hauptthema des Seminars.

Den Abschluß der Arbeit bildet dann ein einfaches Beispiel, in dem anhand einer Markovkette die **Zuverlässigkeit in einem Rechnernetz** betrachtet werden soll.

## 7.2 Begriffsdefinitionen

### 7.2.1 Zuverlässigkeit und Sicherheit

Im Umgangssprachlichen wird der Begriff **Zuverlässigkeit (dependability)** oft gleichgesetzt mit Pünktlichkeit, Verlässlichkeit und Sicherheit.

Bezogen auf den technischen Kontext wird sie nach DIN 40041 jedoch definiert als:

„Beschaffenheit einer Einheit bezüglich ihrer Eignung, während oder nach vorgegebenen Zeitspannen bei vorgegebenen Anfangsbedingungen die Zuverlässigkeitsforderung zu erfüllen“.

Sie gilt also als Maß für die Fähigkeit eines Systems, seine vorgegebene Funktion in einem festgelegten Umfeld voll zu erfüllen.

Die Zuverlässigkeit steht in enger Verbindung mit dem Begriff **Sicherheit**, der nach [Ech01] folgendermaßen spezifiziert wird:

„Sicherheit (safety) ist das Nichtvorhandensein einer Gefahr, also der Möglichkeit eines Schadens. Sie ist nicht zu verwechseln mit dem Begriff „security““.

Zuverlässigkeit als auch Sicherheit können durch **Fehler** beeinflusst werden, die durch die Herstellung eines Systems oder durch den eigentlichen Betrieb verursacht werden können. Betriebsfehler sind im speziellen: Verschleiß, physikalische Fehler oder Bedienungs- und Wartungsfehler.

### 7.2.2 Zuverlässigkeitskenngrößen

Wie im vorhergehenden Abschnitt dargestellt, wird Zuverlässigkeit als Teil der qualitativen Betrachtung eines Systems verstanden und kann somit durch **Zuverlässigkeitskenngrößen** quantifiziert werden.

Unter Zuverlässigkeitskenngrößen versteht man nach [Ech01]

„eine Funktion der Beobachtungswerte, die eine Eigenschaft der Häufigkeitsverteilung eines Zuverlässigkeitsmerkmals charakterisiert“.

Maßgebliche Kenngrößen werden in der folgenden Abbildung dargestellt:

#### Die Überlebenswahrscheinlichkeit

Die wohl prägnanteste Kenngröße der Zuverlässigkeit eines Systems ist die **Überlebenswahrscheinlichkeit**, welche die Wahrscheinlichkeit dafür angibt, daß in einer Zeitspanne T kein Ausfall auftreten wird.

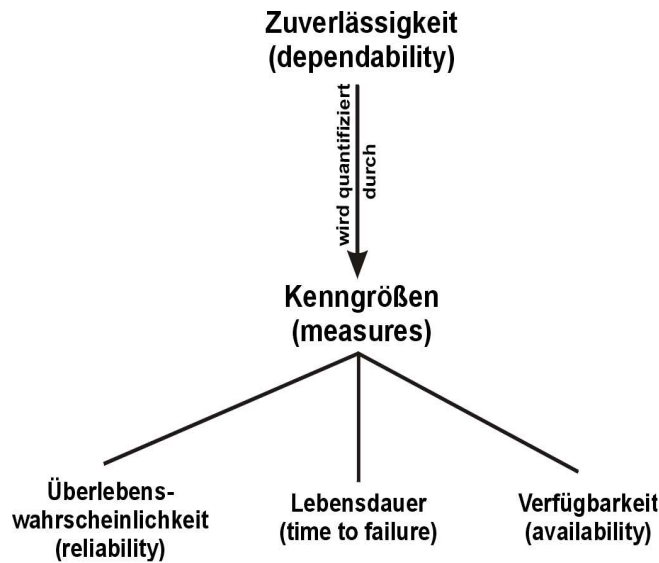


Abbildung 7.1: Zuverlässigkeitskenngrößen (in Anlehnung an [Ech90])

Sollte ein Gerät beziehungsweise eine Einheit seine geforderte Funktion nicht mehr erfüllen, so spricht man von einem **Ausfall**.

Zu Anwendungsbeginn geht die Überlebenswahrscheinlichkeit grundsätzlich von einem fehlerfreien System aus. Erst zu einem späteren Zeitpunkt wird dann die Wahrscheinlichkeit dafür angegeben, daß das System bis dahin ohne Unterbrechung fehlerfrei geblieben ist.

Gerade für **nicht reparierbare Systeme** wird die Überlebenswahrscheinlichkeit zur wichtigsten Zuverlässigkeitskenngröße.

Sie wird in den meisten Fällen durch eine **Zuverlässigkeitsfunktion**  $R(t)$  ausgedrückt, für die gilt:

$$R(t) = P(T > t) = 1 - F(t) \text{ mit } t \geq 0.$$

Die Größe  $F$  stellt dabei die Verteilungsfunktion der **Lebensdauer** dar, die nach [Ech90] die zweite wesentliche Zuverlässigkeitskenngröße symbolisiert.

### Die Lebensdauer

Die **Lebensdauer** definiert die Zeitspanne zwischen Betriebsbeginn und dem (ersten) Ausfall einer Einheit.

Sie wird auch als „Unzuverlässigkeit“ bezeichnet und ist daher die konträre Kenngröße zur Überlebenswahrscheinlichkeit.

Im allgemeinen wird die Lebensdauer als kontinuierliche Zufallsvariable mit einer exponentialverteilten Dichtefunktion angenommen.

Dabei bezieht sich die Lebensdauer immer auf den Anwendungsbeginn und nicht auf das Ende einer (möglichen) Reparatur.

Der Erwartungswert der Lebensdauer wird im allgemeinen als „**mittlere Lebensdauer**“ beziehungsweise als **MTTF** (mean time to failure) bezeichnet und durch den arithmetischen Mittelwert einer Stichprobe von Lebensdauern geschätzt.

In Bezug auf die Überlebenswahrscheinlichkeit errechnet sich die „mittlere Lebensdauer“ aus

der Funktion:

$$\text{MTTF} = \int_0^{\infty} R(t) dt$$

Im Gegensatz dazu rückt als weitere Teilgröße auch die **mittlere Betriebsdauer zwischen zwei Ausfällen** in den Vordergrund.

Sie gibt den Erwartungswert der Verteilung der Betriebsdauern zwischen zwei aufeinanderfolgenden Ausfällen wieder.

Diese Zeitspanne wird auch als **MTBF** (mean time between failure) bezeichnet.

Für **reparierbare Systeme** ist die mittlere Betriebsdauer zwischen zwei Ausfällen von großer Wichtigkeit.

### Die Verfügbarkeit

Während die Überlebenswahrscheinlichkeit die „Zuverlässigkeit“ in einem bestimmten Zeitraum darstellt, wird durch die Kenngröße **Verfügbarkeit A** (availability) die Wahrscheinlichkeit definiert, daß eine Komponente bzw. ein System zu einem festgelegten Zeitpunkt  $t$  unter Berücksichtigung der alternierenden Zeitintervalle von Betriebs- und Ausfallzeiten seinen Dienst erfüllt. Genauer betrachtet spricht man nach dieser Definition auch von der „**momentanen Verfügbarkeit**“.

In der Praxis benutzt man als Schätzwert für die Verfügbarkeit häufig die Gleichung:

$$A(t) = \frac{\text{Betriebsdauer}}{\text{Geforderte Anwendungsdauer}}$$

Im Gegensatz zur Überlebenswahrscheinlichkeit stellt die Verfügbarkeit die wichtigste Kenngröße für reparierbare Systeme dar.

Eine weitere Spezialisierung der Verfügbarkeit ist die „**stationäre Verfügbarkeit**“, die definiert ist als Anteil einer ununterbrochenen Zeitperiode, in der ein System oder eine Komponente zur operationellen Verfügung steht.

Die stationäre Verfügbarkeit  $A_0$  berechnet sich aus:

$$A_0 = \frac{\text{MTBF}}{\text{MTBF} + \text{Mean Down Time (MDT)}}$$

Sie sagt im allgemeinen jedoch wenig über die praktische Verfügbarkeit eines technischen Systems aus, da sie nur dann relevant ist, wenn sich das System rund um die Uhr im Einsatz befindet.

In allen anderen Fällen ist es notwendig, genaue Einsatzpläne sowie mögliche Ausfallauswirkungen nebst den genauen Instandsetzungsarbeitszeiten zu berücksichtigen, um aussagekräftige Prognosen bezüglich der Verfügbarkeit aufzustellen.

### Die Ausfallrate

Eng im Zusammenhang zur Zuverlässigkeit steht als weitere wichtige Verlässlichkeitskenngröße die **Ausfallrate**  $\lambda$  (hazard rate, failure rate). Sie stellt die Wahrscheinlichkeit dar, daß eine Komponente, die bis zum Zeitpunkt  $t$  überlebt hat, im Intervall  $[t, t + dt]$  ausfällt.

Die Ausfallrate wird in „Ausfällen pro Zeiteinheit“, also zum Beispiel durch  $3 \cdot 10^{-6}$  Ausfälle /

h, angegeben.

Im allgemeinen ist die Ausfallrate nicht konstant. Ein typisches Beispiel für die Ausfallrate ist die sogenannte „**Badewannenkurve**“, wie sie in Abbildung 3.2 dargestellt wird.

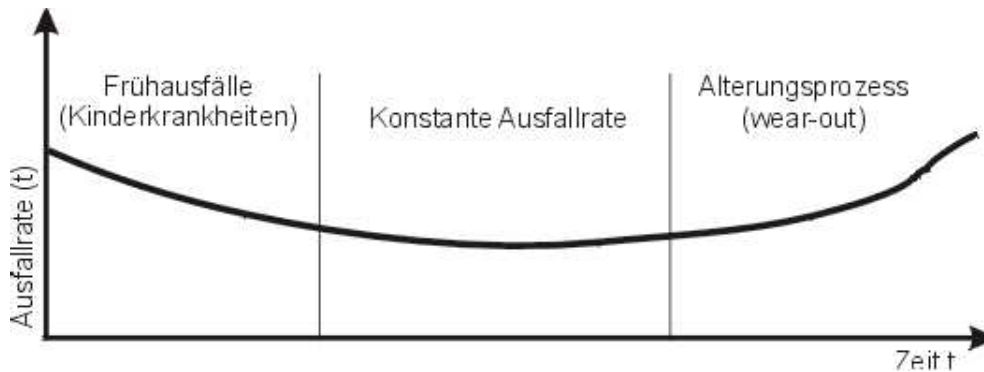


Abbildung 7.2: „Badewannenkurve“ der Ausfallkurve (in Anlehnung an [Ech01])

Für die Modellierung von Systemen wird jedoch häufig eine konstante Ausfallrate angenommen.

Sie errechnet sich aus dem Quotienten der beobachteten Betriebszeit und der Anzahl der Ausfälle in einem Beobachtungszeitraum.

### 7.2.3 Verteilungsfunktionen am Beispiel der Exponentialverteilung

Der **Exponentialverteilung** kommt in der Zuverlässigkeitstheorie eine besondere Bedeutung zu, da sie von einer konstanten Ausfallrate  $\lambda$  ausgeht und somit viele Überlegungen und Rechnungen vereinfacht, da Komponenten des zu betrachtenden Systems unter dieser Bedingung keiner Alterung unterliegen.

Für eine exponentialverteilte Zufallsvariable mit der **Lebensdauer L** und **konstanter Ausfallrate**  $\lambda$  mit  $\lambda > 0$  ergeben sich folgende Zuverlässigkeitskenngrößen:

$$\begin{aligned} \text{Ausfalldichte:} & \quad f(t) = \lambda \cdot e^{-\lambda \cdot t} \\ \text{Überlebenswahrscheinlichkeit:} & \quad R(t) = e^{-\lambda \cdot t} \\ \text{Mittlere Lebensdauer (MTTF):} & \quad E(L) = \frac{1}{\lambda} \end{aligned}$$

Grafisch dargestellt mit den Parametern  $z(t) = \lambda = 0.5$  Ausfälle/Zeiteinheit und der mittleren Lebensdauer von folglich 2 Zeiteinheiten ergibt sich folgendes Bild:

#### Redundanz als Mittel zur Erhöhung der Zuverlässigkeit

Wenn man sich die reinen Definitionen der Zuverlässigkeitskenngrößen betrachtet, stellt sich für jeden Systementwickler als auch Systemnutzer die Frage, welche Möglichkeiten es gibt, die Überlebenswahrscheinlichkeit eines Systems zu erhöhen.

Generell gilt, daß die Zuverlässigkeit eines (Serien-)Systems nur dadurch erhöht werden kann, wenn man die Überlebenswahrscheinlichkeit der einzelnen Komponenten erhöht. Dies kann



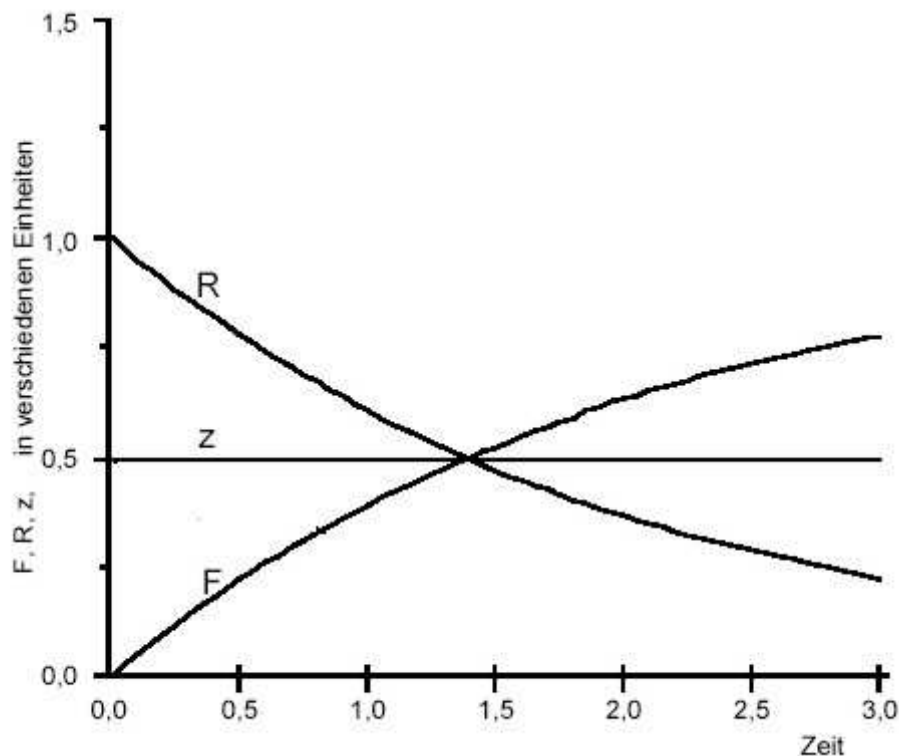


Abbildung 7.3: Funktionen der Zuverlässigkeitskenngrößen am Beispiel der Exponentialverteilung (in Anlehnung an [Ech01])

zum einen durch verbesserte Herstellungsverfahren oder auf der anderen Seite durch Steigerungen in der Materialqualität erfolgen.

Gerade in den schon angesprochenen kritischen Bereichen, wie der Luftfahrt, reicht diese Maßnahme jedoch nicht aus. Hier bedient man sich der Methode, daß Komponenten **mehrfach** als Reservemittel eingesetzt werden, die bei einem Ausfall die Funktionen der nicht mehr zur Verfügung stehenden Teile übernehmen. Solche Systeme werden als **redundant** bezeichnet.

Redundanz kann im Wesentlichen in drei Unterkategorien eingeteilt werden:

Man spricht von „**heißer**“ beziehungsweise **aktiver Redundanz**, wenn das zusätzliche technische Mittel ständig in Betrieb ist und der gleichen funktionsbedingten Beanspruchung unterliegt, wie die Primäreinheit. Als Beispiel dazu kann das Lüftersystem eines Servers genannt werden, welches zumeist aus mehreren Lüftern besteht und somit überdimensioniert ist, sodass bei einem Ausfall trotzdem die Systemtemperatur konstant gehalten werden kann.

Als nächste Stufe beschreibt man mit „**warmer Redundanz**“ bzw. leicht belasteter Redundanz Systeme, bei denen die zusätzliche Komponente bis zum Ausfall der Primäreinheit oder bis zum eigenen Ausfall einer geringen Belastung ausgesetzt ist.

Im Gegensatz dazu spricht man von „**kalter**“ oder **unbelasteter Redundanz**, wenn das zusätzliche technische Mittel bis zum Ausfall der Primärkomponente keiner Belastung ausgesetzt ist. Dies gilt zum Beispiel für unterbrechungsfreie Stromversorgungen in Rechenzentren, die erst aktiviert werden, wenn die Hauptstromversorgung ausfällt.

Charakteristisch ist jedoch, daß die Übergänge zwischen heißer und kalter Redundanz fließend sind, denn auch Komponenten, die im Rahmen der kalten Redundanz unbelastet sind, unterliegen trotzdem gewissen Beanspruchungen. Als Beispiel dazu kann die Alterung von Systemen angeführt werden, die sich negativ auf die Zuverlässigkeit auswirken kann.

Es gilt jedoch nicht immer, daß sich Redundanz positiv auf die Überlebenswahrscheinlichkeit auswirkt, wie der folgende Beweis zeigen soll:

Gegeben sei ein System mit 3 Komponenten gleicher Überlebenswahrscheinlichkeit, für das gilt, daß mindestens zwei Komponenten intakt sein müssen, damit das System funktionsfähig bleibt. Für ein Systembestandteil sei die Überlebenswahrscheinlichkeit  $R_1(t)$  definiert. Nach [Ech90] folgt dann:

$$R_3(t) = R_1(t)^3 + 3 \cdot R_1(t)^2 \cdot (1 - R_1(t)) \text{ und} \\ R_3(t) \geq R_1(t).$$

Daraus läßt sich beweisen:

$$\begin{aligned} R_3(t) &\geq R_1(t) \\ R_1(t)^3 + 3 \cdot R_1(t)^2 \cdot (1 - R_1(t)) &\geq R_1(t) \\ -2 \cdot R_1(t)^3 + 3 \cdot R_1(t)^2 - R_1(t) &\geq 0 \\ -2 \cdot R_1(t)^3 \cdot (R_1(t)^2 - 1,5 \cdot R_1(t) + 0,5) &\geq 0 \\ -2 \cdot R_1(t)^3 \cdot (R_1(t) - 1) \cdot (R_1(t) - 0,5) &\geq 0 \\ R_1(t) &\geq 0,5. \end{aligned}$$

Als Voraussetzung, damit sich Redundanz positiv auf die Gesamtzuverlässigkeit eines Systems auswirkt, muß also jede einzelne Komponente eine Überlebenswahrscheinlichkeit  $R(t)$  von mindestens 50% aufweisen.

### 7.3 Zuverlässigkeitsanalyse - Der Weg zum Modell

Bei der Entwicklung von neuen oder der Untersuchung von bestehenden Systemen in Bezug auf Zuverlässigkeitsaspekte ist es notwendig, genaue Erkenntnisse über das **Systemverhalten** zu erlangen, um eine Optimierung der Zuverlässigkeit zu bewirken.

Entscheidend bei vielen Entwicklungen ist jedoch, eine hohe Qualität (und somit auch Zuverlässigkeit) bei niedrigen Kosten zu erzielen.

Im Rahmen der Systemanalyse erscheint es aber oft als nicht sinnvoll, Untersuchungen am System selbst vorzunehmen, sondern ein **Modell** zu bilden, durch das die wesentlichen Eigenschaften des Ausgangsobjektes abgebildet werden.

Nach [Pag91] wird ein Modell somit bezeichnet als:

„ein materielles oder immaterielles System, welches andere Systeme so darstellt, das eine experimentelle Manipulation der abgebildeten Strukturen und Zustände möglich ist“.

Auch wenn Modelle vielfach den Nachteil aufweisen, daß sie durch vereinfachte Annahmen und Restriktionen die Orientierung am realen System verlieren, läßt es sich jedoch nicht

leugnen, daß die dadurch verminderte Komplexität eine Untersuchung am System erst möglich macht.

Diese formale Darstellung ist im speziellen Voraussetzung für den Einsatz **rechnergestützter Verfahren**, um die Systemeigenschaften in einer frühen Entwurfsphase zu kontrollieren. Sie bildet die Grundlage zum Einsatz von mathematischen Verfahren in Bezug auf den **Entwurf**, die **Dimensionierung** und der **Analyse** eines Systems.

Untersuchungen an realen Komponenten sind außerdem häufig äußerst teuer oder überhaupt nicht durchführbar, weil die internen Vorgänge zu schnell ablaufen, wie zum Beispiel bei Netzwerksystemen.

### 7.3.1 Klassifikation von Modellen

Modelle werden, wie in [Pag91] dargestellt, in Bezug auf die Zuverlässigkeitsbetrachtung nach zwei Gesichtspunkten charakterisiert.

Zum einen erfolgt die Einteilung nach der Art der **Untersuchungsmethode**, zum anderen anhand der **Zustandsübergänge**.

Wenn man Referenz auf die Untersuchungsmethode nimmt, gibt es **analytische Modelle** und **Simulationsmodelle**.

**Analytische Modelle** erlauben es, ein Gleichungssystem zur Widerspiegelung der vorhandenen Systembeziehungen aufzustellen, in welches angenommene Werte eingesetzt werden, um damit durch einen geschlossenen Lösungsdurchlauf den Zielsystemzustand zu ermitteln. Dies kann zum Beispiel die Überlebenswahrscheinlichkeit eines reparierbaren Systems nach einer bestimmten Zeitspanne sein.

Analytische Modelle haben den Vorteil, daß sie eine schnelle Berechnung zulassen und außerdem gute Möglichkeiten bieten, Verbesserungen am System zu analysieren.

Sie sind jedoch bezüglich ihrer Annahmen meist zu idealisiert.

Im Gegensatz dazu eignen sich **Simulationsmodelle** besonders zur Veranschaulichung von Systemverhalten, da hier der Modellzustand Schritt für Schritt fortgesetzt wird und somit als Zwischenzustand interpretiert werden kann.

Sie erscheinen zwar im Vergleich zu analytischen Modellen in den meisten Fällen realitätsnäher, jedoch ist ihre Verwirklichung deutlich aufwendiger.

Die Einteilung von Modellen anhand von Zustandsübergängen wird in der folgenden Grafik verdeutlicht:

Charakteristisch für **statische** Modelle ist, daß sie im Gegensatz zu **dynamischen** Modellen keine Zustandsänderungen aufweisen.

Daher lassen sich die Zustände bei **kontinuierlichen** Modellen durch stetige Funktionen beschreiben, während **diskrete** Darstellungen durch sprunghafte Wertzuweisungen der Zustandsvariablen aufgrund von auf der Zeitachse diskret verteilten Zeitpunkten charakterisiert werden können.

Kontinuierliche und diskrete Modelle können sowohl stochastisch als auch deterministisch definiert sein.

**Deterministisch** beschreibt, daß das Modell bezüglich der Reaktion auf eine bestimmte Eingabe von einem bestimmten Zustand ausgehend eindeutig festgelegt ist.

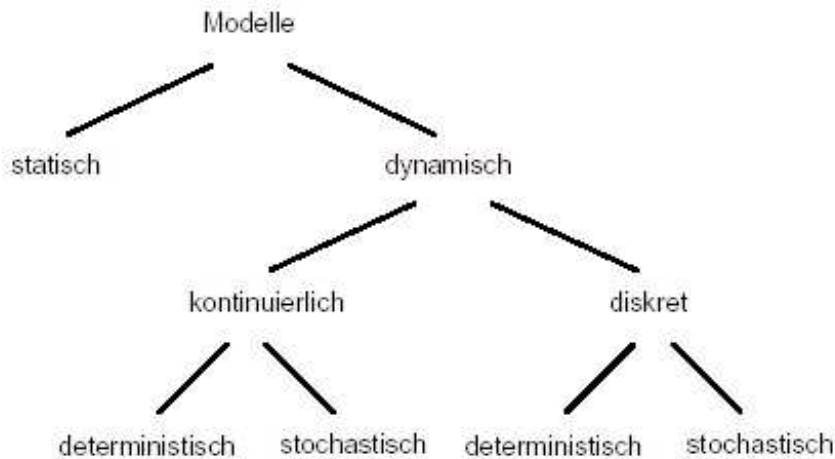


Abbildung 7.4: Modellklassifizierung nach Zustandsübergängen [Pag91].

Auf der anderen Seite sind Modelle, bei denen sich die Reaktionen durch Wahrscheinlichkeitsverteilungen beschreiben lassen, **stochastisch**.

### 7.3.2 Der Modellbildungsprozess

Während bislang nur Modelle im allgemeinen behandelt wurden, soll nun der eigentliche Prozeß der Modellbildung dargestellt werden.

Er besteht aus sechs einzelnen Zuständen und läuft zyklisch ab, wie die nachfolgende Abbildung zeigt:

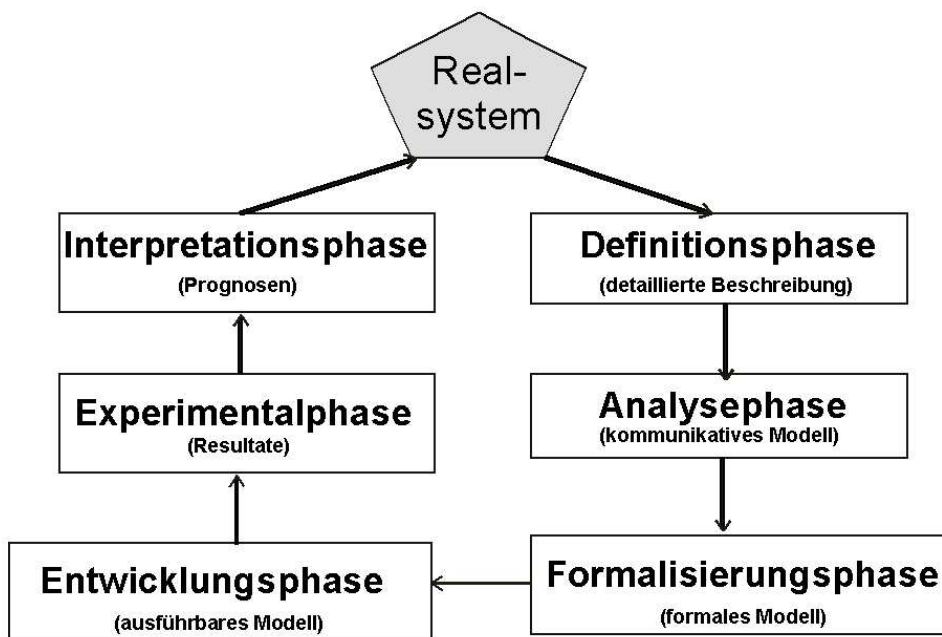


Abbildung 7.5: Der Modellbildungsprozess

### Die Definitionsphase

Im dargestellten Prozessdiagramm wird in der ersten Phase, der **Definitionsphase**, ausgehend von einer Idee beziehungsweise einem bereits existierenden realen System eine Problemdefinition durchgeführt.

In Bezug auf die Zuverlässigkeitsmodellierung werden anhand von Produktanforderungen, Analysen und Kostenerfordernissen bereits Grobziele für die Zuverlässigkeit des Systems definiert. Bereits in dieser Phase ist es notwendig, Vorentscheidungen bezüglich der Verfahren und Standards zur Modellierung der Zuverlässigkeit festzulegen.

Dadurch können frühzeitig Fehler vermieden werden.

### Die Analysephase

Die Kernaufgabe in der **Analysephase** besteht darin, ein **kommunikatives Modell** des Real-systems zu schaffen.

Dies bedeutet, daß das zu modellierende System in der Regel durch ein informales Modell dargestellt wird. Dieses Modell enthält bereits wesentliche Aussagen über die Komponenten des Systems, ihre Eigenschaften sowie deren Beziehungen zueinander.

Bezogen auf Internetroutingsysteme können dies Aussagen zum Ausfallverhalten einzelner Rechner oder zur Leitungsinfrastruktur sein.

Ein weiterer wichtiger Aspekt zu diesem Zeitpunkt besteht darin, eine Entscheidung über einen geeigneten Modelltyp zu treffen. Dabei fließen unter anderem Untersuchungen über diskretes oder stochastisches Verhalten an den einzelnen Komponenten mit ein.

Außerdem besteht die Möglichkeit, das komplette System in einzelne **Subsysteme** zu unterteilen, die dann isoliert untersucht werden können. Trotzdem müssen aber genau definierte Schnittstellen zum Gesamtsystem vorhanden sein, um am Ende eine Verknüpfung der Teilmodelle zum Gesamten gewährleisten zu können.

Zusätzlich erfolgt in dieser Phase bereits eine Entscheidung über die Relevanz bestimmter Systemaspekte, denn es wird abgegrenzt zwischen wichtigen und vernachlässigbaren Eigenschaften in der folgenden Modellierung.

Dadurch kann auch schon eine Aufwandsabschätzung in Bezug auf Arbeitszeiten, Ressourcen und Kosten abgegeben werden.

### Die Formalisierungsphase

Während in der vorhergehenden Phase die Systemanalyse in Form eines informalen (kommunikativen) Modells durchgeführt wurde, hat die **Formalisierungsphase** die Zielsetzung, die „formale Spezifikation des konzeptuellen Modells“ [Pag91] abzugeben. Das kommunikative Modell ist also in einen geeigneten **Formalismus** zu bringen.

Als logische Voraussetzung ist dazu jedoch anzumerken, daß die Systemgrößen, in diesem Fall also die Zuverlässigkeitskenngrößen der Teilsysteme, **quantifizierbar** sein müssen.

Im wesentlichen stützt sich der Entwickler jedoch auf empirische Daten, die auf das (Real-) System zurückgeführt werden können.

Anhand von Beobachtungsdaten können dann Gesetzmäßigkeiten zu den Schätzwerten (also z.B. zu den Beobachtungsparametern) definiert werden.

Im Rahmen der Abstraktion und Idealisierung des Prozesses werden jedoch wiederum vereinfachte Annahmen festgelegt, was zwar die Auswertung vereinfacht, jedoch nicht immer die Nähe zur Realität vereinbaren läßt.

## Zuverlässigkeitsmodellierung

Daher ist es die Aufgabe des Entwicklers, einen „gesunden“ Kompromiß zwischen Vereinfachung und Realitätsnähe zu finden.

Ziel dieser Phase ist letztendlich das formale Modell.

Geeignete Modellierungstechniken werden am Ende dieses Teils und im nächsten Kapitel über „Modellierungstechniken im Vergleich“ aufgegriffen und dargestellt.

### Die Entwicklungsphase

Das formale Modell aus der 3. Phase bildet die Grundlage der **Entwicklungsphase**. In ihr soll ein **ausführbares Modell** entwickelt werden, was in aller Regel in Form eines ablauffähigen Computerprogramms durchgeführt wird.

Ausgehend vom verwendeten Modelltyp ist dazu die geeignete Hard- und Software auszuwählen. Speziell die Wahl der Programmiersprache ist ein wesentlicher Entscheidungspunkt.

Das Ergebnis dieser Phase ist somit eine maschinell ausführbare Beschreibung des formalen Modells.

### Die Experimentalphase

Anhand des lauffähigen Computerprogrammes können nun Versuche durchgeführt werden. Diese **Modellexperimente** bestehen aus einem oder mehreren Durchläufen des Programmes. Bei jedem Durchlauf werden die Eingabewerte bzw. experimentellen Parameter variiert, was zu dynamischem Modellverhalten führt.

Hilfreich sind auch Experimente mit unterschiedlichen Modellvarianten, da dabei besonders Detailunterschiede zur Geltung kommen.

Wichtig ist jedoch nicht nur die Durchführung von verschiedenen Durchläufen, sondern auch die anschließende Aufbereitung der gewonnenen Daten.

Hilfreich sind dazu oft grafische Darstellungen, wie z.B. geeignete Diagramme.

### Die Interpretationsphase

Die **Interpretationsphase** ist die letzte Phase des Modellbildungsprozesse. Sie stellt wiederum die Verbindung zum (Real-)System her.

Da die Ergebnisse aus der vorhergehenden Phase bislang noch keine Antwort auf die zentrale Fragestellung der Modellstudie geben, ist nun eine umfassende Analyse der gewonnenen Daten notwendig, um eine Zielaussage zu gewinnen.

Es müssen somit die **Simulationsdaten** aus den verschiedenen Messungen verglichen werden, um letztendlich die Auswirkungen der Eingabewert-Variationen auf das Verhalten im Modell zu untersuchen.

Die Bewertung der Ergebnisse erfolgt im letzten Schritt immer unter Berücksichtigung der ursprünglichen Fragestellung der Modellstudie und verlangt auch, daß zwischenzeitlich vorgenommene Einschränkungen, wie vereinfachende Annahmen, ausreichend Berücksichtigung finden.

Nur mit dieser Grundlage ist eine **Zielaussage auf das Realsystem** möglich.

### 7.3.3 Vorstellung der ersten wichtigen Modellierungstechnik

Neben dem formalen Prozess der Modellbildung soll an dieser Stelle bereits exemplarisch eine einfache Modellierungstechnik vorgestellt werden, die in der Zuverlässigkeitstheorie große Be-

deutung hat.

Es handelt sich dabei um „**Zuverlässigkeitsblockdiagramme**“, die nach [Ave02] „von einem elektrischen Bild ausgehen“. Es wird angenommen, daß eine Komponente die Zustände „Strom durchlassen“ und „Strom nicht durchlassen“ annehmen kann. Diese Zustände werden mit „intakt“ und „defekt“ identifiziert. Jede Komponente wird durch ein Rechteck dargestellt, in das die Nummer des Elements geschrieben wird.

Formal ausgedrückt handelt es sich nach [Ech01] bei Zuverlässigkeitsblockdiagrammen um:

„Einen gerichteten Graph mit genau einem Eingangsknoten E und genau einem Ausgangsknoten A. Die übrigen Knoten sind entweder Hilfsknoten H (die als fehlerfrei angenommen werden) oder je einer Komponente  $k$  des Systems  $S = \{K_1, \dots, K_n\}$  zugeordnet. Die Systemfunktion wird wie folgt ausgedrückt: S ist genau dann fehlerfrei, wenn es einen Kantenzug von E nach A gibt, der nur über Knoten führt, die fehlerfreien Komponenten zugeordnet sind. Man beachte, daß einer Komponente mehrere Knoten des Zuverlässigkeitsblockdiagramms zugeordnet sein können“.

In Zuverlässigkeitsblockdiagrammen unterscheidet man aufgrund der vorgegeben Definition drei wesentliche Grundformen:

Zum einen gibt es **Seriensysteme**, welche aus  $n$  Komponenten bestehen und nur dann intakt sind, wenn alle  $n$  Bestandteile intakt sind.

Auf der anderen Seite redet man von **Parallelsystemen**, wenn die Funktionsfähigkeit schon dann gewährleistet werden kann, wenn mindestens eine Komponente intakt ist.

Das dritte System ist eine Mischform aus Parallel- und Seriensystemen.

Aufgestellt werden können Systeme auf Basis von Zuverlässigkeitsblockdiagrammen nach zwei Vorgehensweisen:

Zum einen gibt es das Verfahren „**Top Down**“ durch das die Zuverlässigkeitsvorgaben für das Gesamtsystem jeweils auf die Einheiten der darunterliegenden Hierarchiestufen aufgeteilt werden. Dabei besteht jedoch die Gefahr, daß zu hohe Anforderungen unter Umständen nicht erfüllt werden können.

Das zweite Verfahren „**Bottom-Up**“ sagt aus, daß die Zuverlässigkeit der jeweils übergeordneten Baugruppen ermittelt wird. Dies führt häufig jedoch dazu, daß die somit erhaltene Zuverlässigkeit nicht akzeptiert werden kann.

Die Berechnung der Zuverlässigkeit in den Blockdiagrammen erfolgt nach bestimmten Grundsätzen, die in der folgenden Tabelle dargestellt werden:

Zu beachten ist jedoch, daß die Strukturierung in den jeweiligen Diagrammen so gestaltet sein muss, daß jeder Block eine **funktionale Einheit** bildet.

Aufgrund der Serien- und Parallellform eignen sich Zuverlässigkeitsblockdiagramme vom Grundsatz her sehr gut zur Darstellung von Rechnernetzen. Auf der anderen Seite unterliegen sie der Einschränkung, daß z.B. keine Teilausfälle oder reparierbare System betrachtet werden können. Daher müssen weiterführende Modellierungstechniken angewandt werden.

Einige wichtige von ihnen werden im nächsten Abschnitt ausführlich vorgestellt.

Zuverlässigkeitsblockdiagramm	Zuverlässigkeitsfunktion R
	$R_{Sys} = R_1$
	$R_{Sys} = \prod_{i=1}^n R_i$
	$R_{Sys} = R_1 + R_2 - R_1 \cdot R_2$
	$R_{Sys} = \sum_{i=1}^n \binom{n}{i} \cdot R^i \cdot (1 - R)^{n-i}$
	$R_{Sys} = (R_1 R_2 + R_3 R_4 R_5 - R_1 R_2 R_3 R_4 R_5) \cdot R_6 R_7$
	$R_{Sys} = R_5 \cdot (R_1 + R_2 - R_1 R_2) \cdot (R_3 + R_4 - R_3 R_4) + (1 - R_5) \cdot (R_1 R_3 + R_2 R_4 - R_1 R_2 R_3 R_4)$

Abbildung 7.6: Berechnungsgrundlagen für Zuverlässigkeitsblockdiagramme [Www95]

## 7.4 Modellierungstechniken im Vergleich

Im vorhergehenden Kapitel sind Zuverlässigkeitsblockdiagramme als erstes Modellierungsverfahren bereits in Erscheinung getreten. Wie schon erwähnt, reichen die Erkenntnisse daraus jedoch oft nicht aus, da die Zuverlässigkeitsstruktur des Systems unter Umständen sehr komplex ist.

Daher resultieren aus unterschiedlichen Systemanforderungen unter Umständen auch unterschiedliche Zuverlässigkeitskenngrößen, deren Auswertung nach bestimmten Zuverlässigkeitsmodellen verlangt.

Nach [Hei97] unterscheidet man im wesentlichen nach drei Kategorien von Zuverlässigkeitsmodellen:

- Boolesche Modelle (inklusive Fehlerbäumen)
- Markov-Modelle
- Petrinetze

Da diese Modellierungstechniken zum Teil starke Unterschiede aufweisen, ist es Ziel dieses Kapitels, jede der genannten Modellkategorien kurz vorzustellen, wobei ein besonderes Augenmerk auf die Nachteile und Grenzen der jeweiligen Methode gelegt werden soll.

### 7.4.1 Boolesche Modelle

Die Boolesche Theorie geht auf den englischen Mathematiker George Boole (1815-1864) zurück und hat in der Zuverlässigkeitsanalyse von allen Theorien die größte Verbreitung gefunden.

Wie der Name schon vermuten läßt, wird hier zwischen zwei verschiedenen Zuständen einer



Systemkomponente unterschieden: entweder sie ist intakt oder ausgefallen.

Daher wird einem aus  $n$  verschiedenen Elementen bestehenden System eine **boolesche Variable**  $X_i$  zugeordnet:

$$X_i = \begin{cases} 1 & \text{falls das } i\text{-te Element} & \text{arbeitsfähig} \\ 0 & & \text{ausgefallen} \end{cases}$$

Die Arbeitsfähigkeit des Gesamtsystems wird durch die Arbeitsfähigkeit seiner Elemente zum gleichen Zeitpunkt bestimmt. Die **Boolesche Systemfunktion** wird aus diesem Grund folgendermaßen definiert:

$$S = S(X_1, \dots, X_n) = S(\vec{X}) = \begin{cases} 1 & \text{falls das } i\text{-te Element} & \text{arbeitsfähig} \\ 0 & & \text{ausgefallen} \end{cases}$$

Als Voraussetzung für weitere Betrachtungen wird angenommen, daß Boolesche Systeme monoton sind. Das heißt, daß ein System, welches beim Ausfall einer Menge  $M1$  von Komponenten noch arbeitsfähig ist, auch in Betrieb bleibt, wenn eine Teilmenge  $M2 \subset M1$  ausfällt.

Es gilt somit:

$$\begin{aligned} S(\vec{X}') &\geq S(\vec{X}'') && \text{falls } X' > X'' \text{ (Monotonie)} \\ S(\vec{1}) &= 1 && \text{Wenn alle Komponenten funktionsfähig sind,} \\ &&& \text{dann auch das Gesamtsystem} \\ S(\vec{0}) &= 0 && \text{Wenn alle Komponenten defekt sind,} \\ &&& \text{dann auch das Gesamtsystem} \end{aligned}$$

Dargestellt werden boolesche Systemfunktionen durch die Operatoren

- $\wedge$  für die **Konjunktion** (UND-Verknüpfung)
- $\vee$  für die **Disjunktion** (ODER-Verknüpfung)
- $\bar{N}$  für die **Negation** (NICHT)

Jede Boolesche Systemfunktion kann allein mit Hilfe dieser drei genannten Operatoren ausgedrückt werden.

Gerade bei weiterführenden Berechnungen mit den booleschen Funktionen sind die aufgeführten Operatoren etwas störend. Aus diesem Grund können sie durch die mathematischen Operationen Addition, Subtraktion und Multiplikation ersetzt werden.

Für die Konjunktion gilt somit:  $A \wedge B = A \cdot B$ .

Beim Ausschreiben wird dann üblicherweise jedoch weder das Konjunktionszeichen noch das Multiplikationssymbol benutzt.

Die Disjunktion  $A \vee B$  kann durch den Term  $A + B - AB$  ersetzt werden, wie die folgende Verifikation zeigt:

$$\begin{aligned} A \vee B &= \bar{A} \wedge \bar{B} && \text{(De Morgan)} \\ &= \overline{(1-A) \cdot (1-B)} && \text{(Konjunktionsregel)} \\ &= A + B - AB && \text{(arithmetische Umformungen)} \end{aligned}$$

Negationen  $\bar{A}$  können durch die Gleichung  $1 - A$  dargestellt werden.

Durch die zu den mathematischen Operationen äquivalenten booleschen Ausdrücken kann auch verifiziert werden, daß bei Klammerungen die Konjunktion immer Vorrang vor Disjunktionen hat („Punktrechnung vor Strichrechnung“).

In komplexen Systemen können boolesche Funktionen umfangreiche Formen annehmen. Hilfe bei der Umwandlung bietet der **Shannon'sche Entwicklungssatz**. Nach ihm gilt für jede boolesche Funktion:

$$S(X_1, \dots, X_i, \dots, X_n) = X_i \wedge S(X_1, \dots, 1, X_{i+1}, \dots, X_n) \vee \bar{X}_i \wedge S(X_1, \dots, 0, X_{i+1}, \dots, X_n)$$

Aus den Systemfunktionen  $S(1_i, \vec{X})$  und  $S(0_i, \vec{X})$  kann eine weitere Variable  $X_j$  herausgezogen werden. Nach  $n$ -facher Wiederholung dieser Prozedur hängt keine der Systemfunktionen auf der rechten Seite mehr von einer Variablen ab. Nach Fortlassung der Funktionen mit  $S(\vec{X}^k)$  erhält man ein Gleichungssystem in Form einer disjunktiven Verknüpfung von Konjunktionstermen.

Dieses System wird als „**disjunktive Normalform der Booleschen Systemfunktion**“ bezeichnet [ReU88].

Die „**konjunktive Normalform**“ ist demnach eine konjunktive Verknüpfung von vollständigen Disjunktionstermen.

Ausgerüstet mit diesen mathematischen Grundlagen der booleschen Modelle können weiterführende Aussagen in Bezug auf spezielle Zuverlässigkeitskenngrößen getroffen werden.

Ausgangspunkt dafür ist immer die Boolesche Systemfunktion.

In Anlehnung an [ReU88] soll nun im folgenden Beispiel die Überlebenswahrscheinlichkeit anhand der Systemfunktion ermittelt werden.

**Beispiel 7.4.1** Gegeben sei ein 2-aus-3-System. Es gilt somit (wie schon im Abschnitt 7.2.3 beschrieben), daß das System genau dann intakt ist, wenn mindestens zwei der drei Komponenten funktionsfähig sind.

Als Voraussetzung sei erwähnt, daß die Systemfunktion anhand der vorgestellten Möglichkeiten in einen Ausdruck von Produkten und Summen umzuwandeln ist, so daß kein Produkt dieselbe boolesche Variable in mehreren Faktoren enthält.

Die passende Boolesche Systemfunktion für das vorgegebene Beispiel lautet somit:

$$\begin{aligned} S(X_{2v3}) &= X_1 X_2 \vee X_1 X_3 \vee X_2 X_3 \\ &= (X_1 X_2 + X_1 X_3 - X_1^2 X_2 X_3) + X_2 X_3 - \\ &\quad (X_1 X_2 + X_1 X_3 - X_1^2 X_2 X_3) X_2 X_3 \quad (\text{Disjunktionsregel}) \\ &= X_1 X_2 + X_1 X_3 + X_2 X_3 - 2X_1 X_2 X_3 \quad (\text{Vereinfachung}) \end{aligned}$$

Nach 7.2.3 gilt für die Überlebenswahrscheinlichkeit im Falle von exponentialverteilter Zeit:

$$R_i(t) = e^{-\lambda \cdot t}$$

Durch einfaches Ersetzen erhält man dann den Wert für die Überlebenswahrscheinlichkeit:

$$S(X_{2v3}) = e^{-(\lambda_1 + \lambda_2)t} + e^{-(\lambda_1 + \lambda_3)t} + e^{-(\lambda_2 + \lambda_3)t} - 2e^{-(\lambda_1 + \lambda_2 + \lambda_3)t}$$

Für den Fall, daß die Ausfallrate für alle Elemente gleich ist, gilt:

$$S(X_{2v3}) = 3e^{-2\lambda t} - 2e^{-3\lambda t}$$

Anhand des dargestellten Beispielen werden schnell die Vorteile der Booleschen Zuverlässigkeitsermittlung klar: Die Modellierung ist sehr **übersichtlich** und **klar**, und es existiert eine hohe Übereinstimmung zwischen dem geschaffenen Modell und der Wirklichkeit.

Diese Einfachheit kann aber auch zum Nachteil werden, denn innerhalb der booleschen Theorie werden für jede Betrachtungseinheit nur die Zustände „**intakt**“ oder „**defekt**“ definiert. Dies hat zur Folge, daß unterschiedliche Konsequenzen von Ausfällen einer Komponente nicht berücksichtigt werden können.

Auch die zuvor definierte Grundvoraussetzung „Monotonie“ hat zur Folge, daß durch boolesche Modelle zum Beispiel keine „Selbsteilungseffekte“ von Komponenten dargestellt werden können.

Auch auf zeitliche Anforderungen nimmt die beschriebene Modellierungstechnik keine Rücksicht, denn die Theorie legt eindeutig die momentanen Zustände der Komponenten eines Systems zum gleichen Zeitpunkt fest.

Dadurch kann keine **zeitliche Reihenfolge** von Elementausfällen Berücksichtigung finden.

## 7.4.2 Fehlerbäume

Eine besondere Darstellungsform von booleschen Systemfunktionen und eine Erweiterung des zuvor vorgestellten Modells stellen die **Fehlerbäume** dar. Auch sie haben sich stark im Bereich der Zuverlässigkeitstechnik etabliert und sind aus diesem nicht mehr wegzudenken.

Doch was versteht man unter Fehlerbäumen beziehungsweise der **Fehlerbaummethode** eigentlich?

Nach [Red02] stellt die Fehlerbaummethode eine systematische Ermittlung der logischen Verknüpfungen von Komponenten- und Teilsystemausfällen dar, die zu unerwünschten Ereignissen führen können.

Im Klartext heißt dies, daß immer von einem negativen Zustand, einem sogenannten „**Top-Event**“ ausgegangen wird.

Diese Vorgehensweise wird als „**deduktiv**“ bezeichnet.

Die Darstellung der Ausfallkombinationen erfolgt durch einen endlichen Graphen mit endlich vielen Eingängen und einem Ausgang, der als Fehlerbaum bezeichnet wird.

Vor dem Aufstellen des Fehlerbaumes beginnt man mit der Festlegung des **unerwünschten Ereignisses**. Dies kann der Ausfall des Gesamtsystems oder einer bestimmten (Teil-)Funktion sein.

Im nächsten Schritt erfolgt die Definition der Ausfallarten und Reaktion des Systems auf Komponentenausfälle.

Dies bildet die Basis zum Aufstellen des grafischen Fehlerbaumes, der aus Gattern, passend zu den logischen Verknüpfungen, zusammengesetzt wird.

Die möglichen **Gatter** und **Zeichennormen** dafür veranschaulicht die folgende Tabelle:

Zu beachten ist jedoch, daß dieselbe Fehlerbaumfunktion  $S$  verschiedene Darstellungsformen beziehungsweise Arten haben kann.

Dies führt dazu, daß es verschiedene Fehlerbäume zu einer Funktion gibt, die jedoch im alle Ergebnis gleichwertig sind.

Als Beispiel zur grafischen Darstellung von Fehlerbäumen sei wieder ein 2-von-3-System angeführt, daß im jetzigen Fall jedoch defekt ist, wenn zwei der Einzelkomponenten außer Betrieb

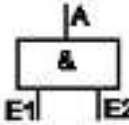
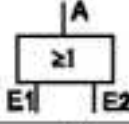
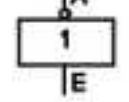
Verknüpfung	Bedeutung	Symbolik (DIN)
Konjunktion (UND)	$A = E1 \wedge E2$	
Disjunktion (ODER)	$A = E1 \vee E2$	
Negation (NICHT)	$A = \bar{E} = 1 - E$	

Abbildung 7.7: Schaltgatter in Fehlerbäumen (in Anlehnung an [Sch99])

sind.

Für die Realisierung gibt es nach [Sch99] 3 Möglichkeiten:

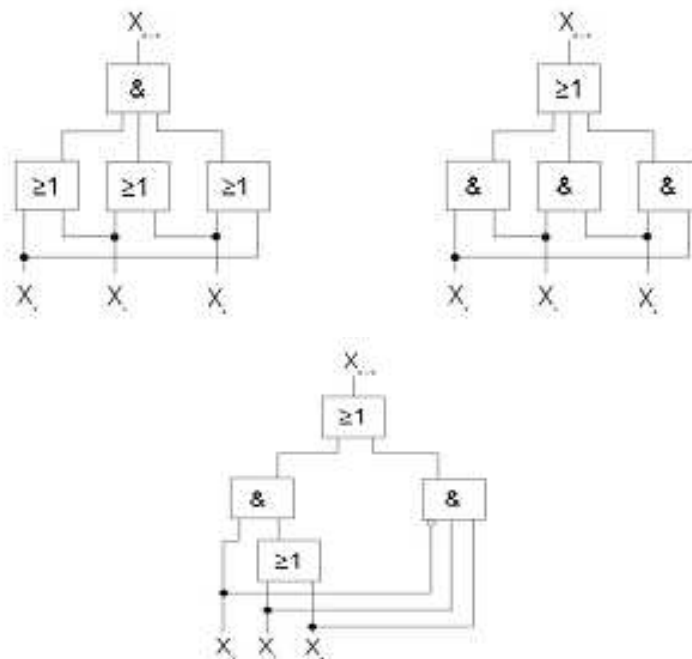


Abbildung 7.8: Drei mögliche Fehlerbäume des 2-von-3-Systems ([Sch99])

Anhand des aufgestellten Fehlerbaumes können nun **qualitative Aussagen** über die Bedeutung einzelner Ausfälle und Ausfallkombinationen gemacht werden.

Dies erfolgt durch das Aufstellen von sogenannten **Minimalschnitten**, was Ausfälle bzw. Ausfallkombinationen sind, die direkt zum „unerwünschten“ Ereignis führen. Sie enthalten keine anderen Ausfallkombinationen mehr, sind also deshalb „minimal“.

Somit kann der Fehlerbaum auf drei Ereignisebenen reduziert werden. In der obersten Ebene steht eine Oder-Verknüpfung und in der zweiten Ebene dann die Und-Verknüpfung der Primäreignisse der dritten Ebene.

Die Minimalschnitte können dann nach Anzahl der jeweiligen Ausfälle sortiert werden. Dadurch können im Speziellen Aussagen darüber getroffen werden, wo Redundanzen im System notwendig sind, da die Minimalschnitte mit den wenigsten Ausfällen im Vergleich eine höhere Wahrscheinlichkeit zur Auslösung des „Top-Ereignisses“, also des Ausfalls des Systems, aufweisen.

Anhand des Fehlerbaumes selbst besteht zudem die Möglichkeit, die bekannten Zuverlässigkeitskenngrößen zu errechnen. Bezüglich genauerer Erklärungen und Darstellungen der möglichen Berechnungsmethoden kann [Sch99] herangezogen werden.

Zu erwähnen ist jedoch noch, daß viele System sehr komplex sind, was wiederum zu äußerst umfangreichen Fehlerbäumen führen kann. Daher werden Auswertungen von Fehlerbäumen im wesentlichen rechnergestützt durchgeführt.

Wie auch schon im vorhergehenden Kapitel über die boolesche Modellierungstechnik erwähnt wurde, weist auch die Modellierung mittels Fehlerbäumen den entscheidenden Nachteil auf, daß sie nur bei stochastisch unabhängigen Komponenten angewandt werden kann.

Dazu kommt, daß in jedem Fall von **Sprungsausfällen** ausgegangen wird, denn zeitliche Abhängigkeiten zwischen verschiedenen Ausfällen finden keine Berücksichtigung. Es findet also generell eine sofortige Zustandsänderung statt.

Diese negativen Aspekte werden im nächsten Abschnitt über Markov-Ketten und im darauffolgenden Kapitel über die Modellierung mittels Petrinetzen noch einmal aufgegriffen.

### 7.4.3 Markovketten

Alle bisher beschriebenen Methoden haben bislang gute Möglichkeiten gegeben, den Ausfall von einfachen Systemen zu modellieren. Ein wichtiger Aspekt, der bislang jedoch noch unberücksichtigt geblieben ist, offenbart sich durch den Sachverhalt, daß ausgefallene Systeme oder Komponenten in gewisser Weise auch wieder **repariert** werden können und sollen. Es kommt also die Fragestellung auf, wie diese verschiedenen Zustände eines Systems in ein geeignetes Modell gebracht werden können.

Hier rücken die sogenannten „**Markovketten**“ als ein Formalismus in den Vordergrund, der die Möglichkeit bietet, Systemzustände, Zeiten und stochastische Abhängigkeiten auszudrücken und zu berechnen.

Markovketten basieren auf **Markovprozessen**, die wiederum stochastische Prozesse sind.

**Definition 7.4.1** *Nach [Mue02] sind stochastische Prozesse eine Familie von Zustandsvariablen  $\{X(t), t \in T\}$ .  $T$  stellt dabei den Parameterraum dar. Die Menge aller Werte, die  $X(t)$  annehmen kann, heißt Zustandsraum  $S$ .*

Stochastische Prozesse können bezüglich des **Zustandes S** (=State) und der **Zeit T** (=Time) klassifiziert werden.

Die Elemente  $t \in T$  werden meist als Zeitpunkte interpretiert.

In Bezug auf die Zustandsmenge unterscheidet man zwischen **zustandsdiskreten** und **zustandskontinuierlichen** Prozessen.

Im diskreten Fall ist die Zustandsmenge endlich oder zumindest abzählbar. Ein wichtiges Beispiel für diese Eigenschaft stellen die natürlichen Zahlen  $\mathbb{N}$  dar.

Ist die Zustandsmenge kontinuierlich, so ist sie überabzählbar unendlich.

Für den Fall der Zuverlässigkeitsmodellierung kommt hierbei jedoch nur die diskrete Betrachtung der Zustandsmenge in Frage, da die Anzahl von möglichen Zuständen für jedes System

endlich ist.

Mögliche Zustände können zum Beispiel sein:

- Z1: Das System ist intakt.
- Z2: Eine von zwei Komponenten ist außer Betrieb.
- Z3: Das System ist defekt.
- Z4: Das System ist in Reparatur.

Auch im Fall der Zeitbetrachtung unterscheidet man zwischen **zeitdiskreten** und **zeitkontinuierlichen** Prozessen.

Wird die Zeit durch  $T = \mathbb{N}$  definiert, dann ist sie diskret, während durch  $T = \mathbb{R}_+$  von einer kontinuierlichen Zeitachse ausgegangen wird.

Eine besondere Form von stochastischen Prozessen sind solche, bei denen die weitere Entwicklung des Prozesses, was sich im wesentlichen auf die Wahl des nächsten Zustandes bezieht, nur vom gegenwärtigen Zustand abhängt. Diese Art von „Gedächtnislosigkeit“ wird auch als Markov-Eigenschaft bezeichnet.

Daher spricht man auch von **Markovprozessen**, die nach [Mue02] folgendermaßen definiert werden:

**Definition 7.4.2** Ein stochastischer Prozess  $\{X(t), t \in T\}$  heißt Markov-Prozess, falls für  $n > 1$ , einer Folge von Zeitpunkten  $t_1 < t_2 < \dots < t_n < t_{n+1}$ ,  $t_i \in T$  und für eine beliebige Auswahl von Zuständen  $x_1, x_2, \dots, x_n$ ,  $x_i \in X$  gilt:

$$P[X(t_{n+1}) = x_{n+1} | X(t_1) = x_1, X(t_2) = x_2, \dots, X(t_n) = x_n] = P[X(t_{n+1}) = x_{n+1} | X(t_n) = x_n].$$

Da bei Markovprozessen in Verbindung zur Zuverlässigkeitsmodellierung nur von zustandsdiskreten Fällen auszugehen ist, spricht man in diesem speziellen Fall von „**Markovketten**“. Eine zeitkontinuierliche Markovkette kann sowohl durch ihren Zustandsübergangsgraphen, als auch durch eine **Ratenmatrix** dargestellt werden, die wie folgt definiert wird:

**Definition 7.4.3** Die Ratenmatrix einer zeitkontinuierlichen Markovkette ist definiert durch:

1.  $q_{ij}$  ist die Übergangsrate vom Zustand  $i$  zum Zustand  $j$ , wobei gilt  $i \neq j$ .
2.  $q_{ii} = - \sum_{j \neq i} q_{ij}$

Anhand des folgenden, an [Mue02] angelehnten, kleinen **Beispiels** soll der soeben beschriebene Sachverhalt einmal kurz verdeutlicht werden:

**Beispiel 7.4.2** Gegeben ist ein Rechnersystem aus 2 unabhängigen Computern, die ausfallen können. Daher können für dieses System 3 mögliche Zustände definiert werden:

- Z0: Beide Rechner sind defekt.
- Z1: Ein Rechner ist funktionsfähig, der andere wird repariert.

- Z2: Beide Rechner sind funktionsfähig.

Im fehlerfreien Zustand Z2 arbeiten beide Rechner gemeinsam. Beim Ausfall eines Rechners wird dieser repariert und steht eine gewisse Zeitspanne nicht zur Verfügung. Der verbleibende Rechner übernimmt nun die Aufgaben des defekten Computers. Dies erfolgt jedoch nur, wenn er zuvor für die Übernahme der Aufgaben konfiguriert werden konnte. Ist dies nicht der Fall, so gerät das System in den Zustand Z0 und keiner der Rechner ist einsatzfähig.

Nach der Reparatur eines Rechners springt das System in den Zustand Z1, und im Anschluß an die Reparatur des (möglichen) zweiten Rechners befindet es sich wieder im voll funktionsfähigen Status, dem Zustand Z2.

Da die Anzahl der Zustände (in diesem Fall sind es drei) endlich ist, und die Beobachtung über einen „kontinuierlichen“ Zeitraum stattfindet, kann hier eine Markovkette Anwendung finden. Die Darstellung erfolgt folgendermaßen:

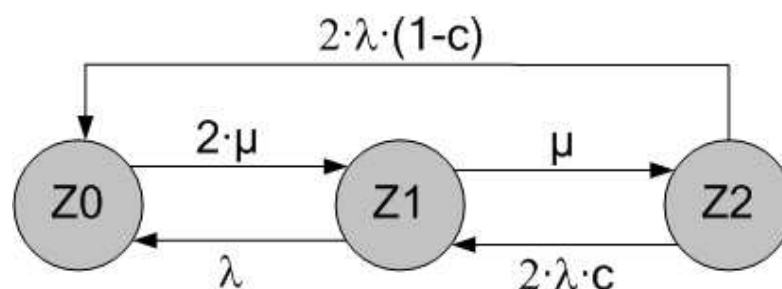


Abbildung 7.9: Markovkette für das gegebene Beispiel [Mue02]

In der grafischen Darstellung sind bereits Übergangsraten berücksichtigt worden. Grundsätzlich sollen Rechnerausfälle mit der Rate  $\lambda$  und Reparaturen mit Rate  $\mu$  durchgeführt werden. Da genügend Reparaturkapazitäten vorhanden sind, erfolgt der Übergang von Z0 nach Z1 mit der Rate  $2\mu$ .

Die Besonderheit in diesem System stellt der **Überdeckungsfaktor** (coverage factor)  $c$  dar, der die Wahrscheinlichkeit der erfolgreichen Neukonfiguration des Systems angibt.

Für die Beispielwerte  $\lambda = 10^{-5}/h$ ,  $\mu = 0.25/h$  und  $c=0.99$  ergibt sich die folgende Matrix:

$$P = \begin{pmatrix} -0.5 & 0.5 & 0.0 \\ 10^{-5} & -0.25001 & 0.25 \\ 2 \cdot 10^{-7} & 1.98 \cdot 10^{-5} & 2 \cdot 10^{-5} \end{pmatrix}$$

Die einzelnen stationären Zustandswahrscheinlichkeiten können nun mit Hilfe des Gaußschen Eliminationsverfahrens leicht errechnet werden.

Zusätzlich kann nun untersucht werden, inwieweit sich eine Änderung des Überdeckungsfaktors auf die einzelnen Werte auswirkt.

Neben der soeben vorgestellten Form von Markovketten findet in der Zuverlässigkeitsmodellierung eine weitere, spezifiziertere Form ihre Anwendung. Denn im Fall von konstanten Übergangswahrscheinlichkeiten zwischen den Zuständen spricht man von „**homogenen Markovketten**“, die folgendermaßen definiert werden:

**Definition 7.4.4** Eine Markovkette ist homogen, falls gilt:

## Zuverlässigkeitsmodellierung

$$P = P[X(t_{n+1} = x \mid X(t_n) = y)] = P[X(t_1 = x \mid X(t_0) = y)] := p_{xy}$$

für  $\forall n \in \mathbb{N}, \forall x, y \in S$

Da die Übergangsraten zwischen den einzelnen Zuständen im Fall von homogenen Markovketten konstant sind, erscheint es nicht sinnvoll, eine Ratenmatrix anzugeben.

Stattdessen bedient man sich in diesem Fall einer **Übergangsmatrix  $\mathbf{P}$** , welche die Übergangswahrscheinlichkeiten  $(p_{ij})_{i,j \in S}$  enthält.

Anhand der Wahrscheinlichkeiten in dieser Matrix, die in der Spaltensumme immer 1 ergeben müssen, können nun vielfältige Betrachtungen der Ausfallwahrscheinlichkeiten des Systems durchgeführt werden. Als Beispiel sei dazu auf die abschließende Aufgabe in dieser Seminararbeit hingewiesen.

Auch auf die weitere Auswertung von Markov-Ketten und den dazugehörigen Matrixdarstellungen soll an dieser Stelle nicht weiter eingegangen werden.

Weit in die Materie eingehende Lösungsmöglichkeiten mittels Differentialgleichungen werden in [Ave02], Kapitel 3 und [Koh87], Kapitel 13 und 14, vorgestellt.

Zusammenfassend kann man sagen, daß Markovmodelle eine gute Möglichkeit bieten, das reale Ausfallverhalten von vielen technischen Systemen darzustellen beziehungsweise eine Annäherung daran zu finden.

Nachteilig erweist sich jedoch, daß durch ein Zustandsmodell die gesamte Modellierung auf eine relativ niedrige Beschreibungsebene gebracht wird. Das heißt, man setzt sich in der Regel nicht mehr mit einzelnen Komponentenausfällen des Systems auseinander, sondern betrachtet das Gesamtobjekt (z.B. Z1: System ist ausgefallen, Z2: System ist nicht ausgefallen).

Würde man auf der anderen Seite in der Zustandsbeschreibung das System immer detaillierter betrachten oder sehr komplexe Systeme untersuchen, vervielfacht sich der damit verbundene Auswertungsaufwand. Auch die Auswertung von Systemen, in denen die Übergangswahrscheinlichkeiten von der Zeit abhängen, gestalten sich als sehr kompliziert und umfangreich.

Trotz dieser Nachteile haben Markovmodelle zurecht eine enorm große Anwendung in der Zuverlässigkeitsmodellierung gefunden.

### 7.4.4 Petri-Netze

Ein neuer Ansatz, um eine höhere Beschreibungsebene als bei Markovketten zu erzielen und um einige Nachteile des im vorhergehenden Kapitels beschriebenen Formalismus auszugleichen, wurde durch die sogenannten „Petri-Netze“ geschaffen.

Historisch betrachtet entstanden Petri-Netze bereits im Jahr 1962 im Institut für instrumentale Mathematik in Bonn. Dort dienten Sie im Rahmen der Dissertation von Carl-Adam Petri über die „Kommunikation mit Automaten“ zur Beschreibung von nebenläufigen, kommunizierenden Prozessen.

Definiert werden Petri-Netze nach [Syr02] folgendermaßen:

**Definition 7.4.5** *Ein Petri-Netz ist ein gerichteter, bipartiter Graph*

$$PN = (S, T, A, M')$$

mit

- einer endlichen Menge von Stellen (bzw. Plätzen)  $S = \{s_1, s_2, \dots, s_n\}$ ,
- einer endlichen Menge von Transitionen  $T = \{t_1, t_2, \dots, t_n\}$ ,



- einer endlichen Menge von gerichteten Kanten  $A = \{a_1, a_2, \dots, a_k\}$
- und einer Anfangsmarkierung  $M' = \{m'_1, m'_2, \dots, m'_n\}$ .

Ein Graph heißt bipartit (auch paar- oder zweigeteilt genannt), wenn:

1. die Knotenmenge  $S$  aus zwei disjunkten Teilmengen  $S'$  und  $S''$  besteht, so daß gilt:  $S = S' \cup S''$  und  $S' \cap S'' = \emptyset$ ,
2. es nur Kanten  $(s', s'') \in A$  beziehungsweise  $(s'', s') \in A$  gibt, mit  $s' \in S'$  und  $s'' \in S''$ , so daß gilt:  $a \subseteq S' \times S'' \cup S'' \times S'$ .

In Worten ausgedrückt bedeutet es, daß die Knotenmenge in zwei Klassen partitioniert wird und die Kanten des Graphen jeweils nur zwischen unterschiedlich klassifizierten Knoten liegen.

Grafisch werden Petrinetze folgendermaßen dargestellt:

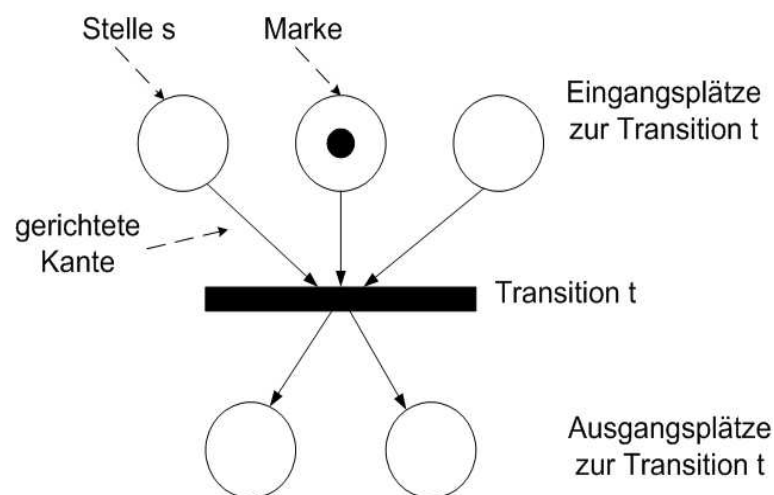


Abbildung 7.10: Grafiksymbole in Standard-Petrinetzen [Syr02]

Die Kreisknoten innerhalb des Petrinetzes werden als **Plätze** bezeichnet, die Rechteckknoten als **Transitionen**. Gerichtete Kanten verbinden Plätze mit Transitionen und umgekehrt.

Plätze können **Marken** beinhalten, die als schwarze Punkte gekennzeichnet sind.

Der Zustand eines Petrinetzes wird definiert durch die Anzahl der vorhandenen Marken an jedem Platz und drückt sich durch den Vektor  $M = \{m_1, m_2, \dots, m_n\}$  aus, dessen  $i$ -te Komponente die Anzahl der Marken in Platz  $s_i$  angibt.

Aus diesem Grund wird der Zustand des Petrinetzes auch als **Markierung** bezeichnet.

Der Anfangszustand des Petrinetzes wird durch die Anfangsmarkierung  $M'$  angegeben.

Die jeweiligen Markierungswechsel ergeben sich aus den **Schaltregeln**, welche die Schaltbereitschaft und das Schalten einer Transition definieren. Generell gilt eine Transition als schaltbereit bzw. aktiviert, wenn alle ihre Eingangsplätze markiert sind, also mindestens eine Marke enthalten.

Mit **Eingangsplätzen** bezeichnet man die Plätze, von denen aus ein gerichteter Pfeil zur Transition führt, während Ausgangsplätze die Stellen sind, zu denen ein Pfeil von der Transition ausgehend hinführt.

In Bezug auf die Zuverlässigkeitsmodellierung können nun Systeme dargestellt werden, in denen das Ausfallverhalten durch eine Kettenreaktion (**Kausalität**) charakterisiert werden kann, wie es in der Abbildung 4.4 gezeigt wird.

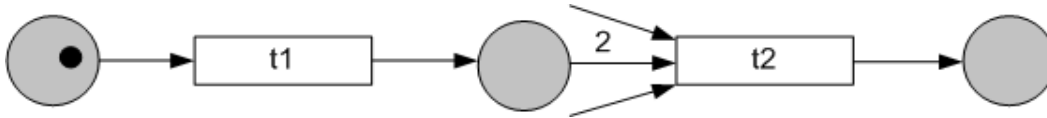


Abbildung 7.11: Kausalität in Petrinetzen [Uhr01]

Zum anderen lassen sich **Nebenläufigkeiten** modellieren. Das heißt, Transitionen können unabhängig voneinander, entweder in beliebiger Reihenfolge, oder zeitgleich schalten.

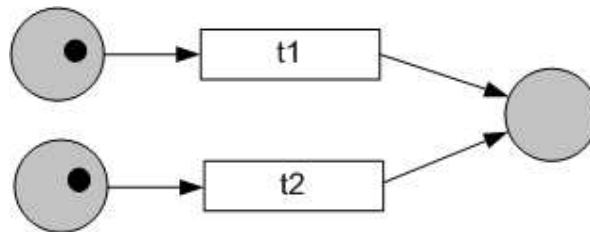


Abbildung 7.12: Nebenläufigkeit in Petrinetzen [Uhr01]

Dadurch können zum Beispiel, bezogen auf Netzwerke, Abhängigkeiten zwischen Komponentenausfällen, sowie voneinander unabhängige Fehler umfassend wiedergegeben werden.

Ein Problem, das bei Standard-Petrinetzen jedoch auftaucht, ist die fehlende Möglichkeit, zeitliches Verhalten quantitativ darzustellen. Bislang wurde das Systemausfallverhalten also ohne Berücksichtigung von Zeitfaktoren modelliert.

Daher ist es notwendig, das Konzept der Petrinetze sinnvoll zu erweitern. In der Regel geschieht dies durch eine Ergänzung der Schaltregel, so daß nun Verzögerungszeiten berücksichtigt werden können. Für die reale Modellierung von Systemen ist es an diesem Punkt sinnvoll, die jeweiligen Zeiten zufallsverteilt, also stochastisch anzugeben. Daher spricht man auch von „**Stochastischen Petrinetzen**“, die nach [Syr02] folgenden Definition unterliegen:

**Definition 7.4.6** Ein stochastisches Petrinetz ist ein gerichteter, bipartiter Graph  $SPN = (S, T, A, M', R)$  mit

- $S, T, A$  und  $M'$  wie bereits eingeführt und  $R = \{r_1, r_2, \dots, r_n\}$ ,

wobei  $R$  die Menge der Schaltraten mit  $r_i$  als Mittelwert der zur Transition  $t_i$  gehörenden, exponentiell verteilten Schaltraten darstellt.

Die Exponentialverteilung ist, wie schon im Kapitel 2.3 beschrieben wurde, die wichtigste und auch am leichtesten zu handhabende Verteilung. Sie besitzt als einzige kontinuierliche Verteilung die Markov-Eigenschaft (**Gedächtnislosigkeit**).

Oft ist es jedoch nicht notwendig, jeder Transition eine exponentialverteilte Schaltrate zuzuordnen. Daher ist es sinnvoll, sowohl zeitbehaftete, als auch zeitlose Transitionen zu integrieren. Petrinetze mit dieser Eigenschaft werden als „**Generalisierte Stochastische Petrinetze**“ (**GSPN**) bezeichnet.

Zusammengefaßt haben Petrinetze, neben den Möglichkeiten, Nebenläufigkeiten oder Kausalitäten darzustellen, den großen Vorteil, daß sie sowohl funktionale als auch grafische Lösungsmöglichkeiten bieten.

Zum einen lassen sie sich mit Rechnerunterstützung ausführen, zum anderen ist es jedoch immer möglich, daß der „Benutzer“ jeden Zustandswechsel (Schalten einer Transition) von Hand ausführen kann, was ein schrittweises Verfolgen des Netzablaufes ermöglicht.

(Stochastische) Petrinetze sind jedoch nur bei verhältnismäßig geringer Modellkomplexität praktisch auswertbar. Einen weiteren Nachteil offenbaren die exponential verteilten Schaltraten, da sie in Bezug auf das real vorliegende System nicht immer eine gute Approximation bieten.

In der Praxis erfolgt die Auswertung von Petrinetzen zumeist nicht direkt, sondern es findet eine Umwandlung in **Markovketten** statt, welche bereits Gegenstand des vorhergehenden Kapitels gewesen sind.

## 7.5 Abschließendes Beispiel

Zurückblickend auf die einzelnen vorgestellten Modellierungstechniken in der Zuverlässigkeitsbewertung erscheint nun natürlich die Fragestellung, welche Methode als am besten geeignet erscheint, das Ausfallverhalten von Routingsystemen zu modellieren und auswertbar zu machen.

Jedes Verfahren bietet in gewisser Weise Vorteile, jedoch auch deutliche Grenzen, die eine eindeutige Antwort unmöglich machen. An dieser Stelle soll mittels eines Beispiels noch einmal die Markovkette als eine Modellform aufgegriffen werden, die sehr häufig Anwendung in der Fragestellung der Zuverlässigkeitsbetrachtung von Netzwerksystemen findet.

Dazu sei folgende Problemstellung vorgegeben:

**Beispiel 7.5.1** Gegeben sei ein Rechnerring aus bestehend aus  $n$  Rechnern  $R_k$  ( $0 \leq k \leq n-1$ ,  $n \in \mathbb{N}$ ).

Zwischen zwei benachbarten Rechnern  $R_k$  und  $R_{k+1}$  existiert eine Datenverbindung, die nur die Übertragungsrichtung von  $R_k$  nach  $R_{k+1}$  zuläßt.  $R_{n-1}$  übertrage dabei nach  $R_0$ .

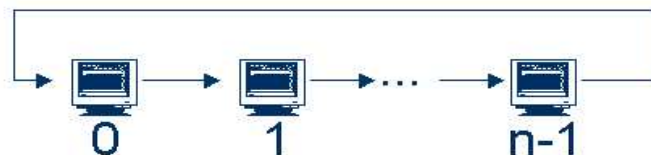


Abbildung 7.13: Rechnerring mit  $n$  Rechnern  $R_k$  ( $0 \leq k \leq n-1$ ,  $n \in \mathbb{N}$ )

Bei der Übertragung von Binärzeichen werden diese auf einer beliebigen Datenverbindungsstrecke mit einer konstanten Wahrscheinlichkeit  $q$  falsch übertragen.

## Zuverlässigkeitsmodellierung

Das Statusbit durchlaufe diesen Rechnerring kontinuierlich,  $x_0$  bezeichne das dabei zunächst gesendete Statusbit,  $x_i$  das am  $i$ -ten Rechner empfangene Statusbit.

Die Aufgabe soll sein, zu berechnen, wie groß die Wahrscheinlichkeit ist, daß das Statusbit an Rechner  $R_{k+3}$  richtig empfangen wird, wenn es am Rechner  $R_k$  richtig angekommen ist. Die Wahrscheinlichkeit  $q$  sei mit  $\frac{1}{3}$  gegeben.

Im ersten Schritt ist dazu das gegebene System zu betrachten. Es ist charakterisiert durch zwei Zustände:

- Zustand1: Das Statusbit wird richtig übertragen.
- Zustand2: Die Übertragung des Statusbits ist falsch.

Da die Übergangswahrscheinlichkeiten zwischen den einzelnen Zuständen konstant sind, ergibt sich für resultierende homogene Markovkette folgendes Bild:



Abbildung 7.14: Darstellung als homogene Markovkette

Anhand der Übergangswahrscheinlichkeiten ist es nun möglich, eine Übergangsmatrix  $M$  zu erstellen:

$$M = \begin{pmatrix} 1-q & q \\ q & 1-q \end{pmatrix}$$

Charakteristisch für  $M$  ist, das die Matrix ergodisch ist. Nach [Bro96] ist eine Markovkette ergodisch, wenn „es eine natürliche Zahl  $n \geq 1$  gibt, so daß alle Elemente der Matrix  $p^N$  positiv sind“.

Somit existiert auch eine **Grenzwertmatrix**  $\tilde{M}$ , die hier unabhängig von der Aufgabenstellung betrachtet werden soll.

Zur Erklärung: Die Grenzwertmatrix gibt die Grenzwerte der Übergangswahrscheinlichkeiten an, die für ein „eingefahrenes System“ zutreffen.

Es gilt also für die Matrix:

$$\tilde{M} = \begin{pmatrix} \pi_0 \\ \pi_1 \end{pmatrix} = M \cdot (\pi_0 \ \pi_1).$$

Daraus kann das Gleichungssystem:

$$\pi_0 = (1-q) \cdot \pi_0 + q \cdot \pi_1$$

$$\pi_1 = (1-q) \cdot \pi_1 + q \cdot \pi_0$$

gebildet werden.

Unter der notwendigen Voraussetzung  $\pi_0 + \pi_1 = 1$  erhält man als Lösung dann  $\pi_0 = \pi_1 = \frac{1}{2}$ .

Daraus ergibt sich die Grenzwertmatrix:

$$\tilde{M} = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix}$$

Bemerkenswert ist an dieser Stelle, daß die Grenzwahrscheinlichkeiten unabhängig von der gegebenen Übergangswahrscheinlichkeit  $q$  sind.

In der weiteren Betrachtung der ursprünglichen Aufgabenstellung errechnet sich die gesuchte Wahrscheinlichkeit aus der Potenzierung der Übergangsmatrix mit 3:

$$P_{00}^3 = M^3 = \begin{pmatrix} \frac{2}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{2}{3} \end{pmatrix} = \begin{pmatrix} \frac{14}{27} & \frac{13}{27} \\ \frac{13}{27} & \frac{14}{27} \end{pmatrix}$$

Die gesuchte Wahrscheinlichkeit kann dem Element der ersten Zeile und ersten Spalte entnommen werden, da nur der Übergang vom Zustand „richtige Übertragung“ zu sich selbst von Interesse ist.

Das Ergebnis beträgt also  $\frac{14}{27}$ , was ungefähr 51.9 Prozent entspricht.

Dazu ist zu erwähnen, daß an diesem Wert die Annäherung an die zuvor berechnete Grenzwahrscheinlichkeit von  $\frac{1}{2}$  sehr gut zu sehen ist.

## 7.6 Zusammenfassung und Ausblick

Die in dieser Arbeit dargestellten Möglichkeiten der Zuverlässigkeitsmodellierung haben gezeigt, daß es in vielen Fällen sehr schwer ist, für sein spezielles Problem das geeignete Modellierungsverfahren zu finden.

Gewisse Verfahren, wie die Auswertung mittels Markovketten, haben sich fest etabliert, aber dennoch stellen die hier vorgestellten Techniken nur einen Bruchteil der insgesamt existierenden Formen dar.

In regelmäßigen Abständen kommen neue Möglichkeiten hinzu oder existente Verfahren werden weiter verfeinert.

Auch die immer weiter wachsenden Rechnerkapazitäten tragen dazu bei, die zunehmenden Zuverlässigkeitbedürfnisse mittels komplexer Auswertungsverfahren zu befriedigen.

Dennoch sollte man generell, auch bei der Berechnung der Zuverlässigkeit von Internet-Routing-Systemen, berücksichtigen, daß ein Modell nie das real existierende System exakt wiedergeben kann, sondern immer Einschränkungen unterliegt.

# Literaturverzeichnis

- [Ave02] PROF. DR. RUDOLF AVENHAUS. *Vorlesungsskript „Zuverlässigkeitstheorie“*. WT 2002, Universität der Bundeswehr München.
- [Bro96] I.N. BRONSTEIN UND K.A. SEMENDJAJEW. *Teubner-Taschenbuch der Mathematik*. WT 2002, Teubner Verlagsgesellschaft, 1996.
- [Ech90] KLAUS ECHTLE. *Fehlertoleranzverfahren*. Springer-Verlag Berlin, 1990.
- [Ech01] KLAUS ECHTLE, IRENE EUSGELD. *Arbeitsblätter zur Vorlesung „Zuverlässigkeit von Hardware und Software“*. WS 2001/2002, Universität Essen.
- [Hei97] KLAUS HEIDTMANN. *Zuverlässigkeitsbewertung technischer Systeme*. Teubner Verlagsgesellschaft, 1997.
- [Koh87] JÜRIG KOHLAS. *Zuverlässigkeit und Verfügbarkeit*. Teubner Verlagsgesellschaft, 1987.
- [Leh02] AXEL LEHMANN. *Vorlesungsskript „Simulation“*. Universität der Bundeswehr München, 2002.
- [Mue02] BRUNO MÜLLER-CLOSTERMANN. *Vorlesungsskript „Stochastische Netze“*. Universität Essen, SS 2000.
- [Pag91] BERND PAGE. *Diskrete Simulation, eine Einführung mit Modula-2*. Springer Verlag Berlin Heidelberg, 1991.
- [ReU88] KURT REINSCHKE, IGOR ALEKSEEVIC USAKOV. *Zuverlässigkeitsstrukturen: Modellbildung, Modellauswertung*. R. Oldenbourg Verlag München Wien, 1988.
- [Red02] GEORG REDEKER. *Vorlesungsskript „Qualitätsmanagement“*. Universität Hannover, 2002.
- [Sch99] WINFRIED SCHNEEWEIS. *Die Fehlerbaummethode*. LiLoLe Verlag GmbH Hagen, 1999.
- [Syr02] MICHAEL SYRJAKOW. *Vorlesungsskript „Simulationstechnik“*. SS 2002, Berufsakademie Mannheim.
- [Tei01] JÜRIGEN TEICH. *Vorlesungsskript „Diskrete Ereignissysteme“*. WS 2001/2002, Universität Paderborn.
- [Uhr01] ADELINDE UHRMACHER. *Vorlesungsskript „Modellbildung und Simulation“*. WS 2001/2002, Universität Rostock.

[Www95] HUBERT BECKER. *Technische Zuverlässigkeit.* [http://home.t-online.de/home/becker2/log3\\_1.htm](http://home.t-online.de/home/becker2/log3_1.htm), 1995.





# 8 Performability Modellierung

*Robert Brendel*

## Inhaltsverzeichnis

---

<b>8.1 Einführung</b> . . . . .	<b>178</b>
8.1.1 Modelle . . . . .	178
8.1.2 Performability . . . . .	179
8.1.3 Mathematisches Modell versus Simulation . . . . .	179
8.1.4 Aufbau . . . . .	180
8.1.5 Danksagungen . . . . .	181
<b>8.2 Metriken</b> . . . . .	<b>181</b>
8.2.1 System-Antwortzeit . . . . .	181
8.2.2 Durchsatz . . . . .	183
8.2.3 Auslastung der Ressourcen . . . . .	183
8.2.4 Verlässlichkeit . . . . .	183
8.2.5 Verfügbarkeit . . . . .	183
8.2.6 Kosten . . . . .	183
<b>8.3 Grundlagen</b> . . . . .	<b>184</b>
8.3.1 Grundlegende Begriffe . . . . .	184
8.3.2 Markov-Ketten . . . . .	185
8.3.3 Übergangs- und Intensitätsmatrizen . . . . .	187
<b>8.4 Markov-Reward-Modelle</b> . . . . .	<b>187</b>
8.4.1 Das Markov-Reward-Modell . . . . .	188
8.4.2 Steady-State Analyse . . . . .	189
8.4.3 Die Transient-State Analyse . . . . .	189
8.4.4 Cumulative Rewards . . . . .	190
<b>8.5 Anwendungsbeispiele</b> . . . . .	<b>190</b>
8.5.1 Analysewerkzeuge . . . . .	191
8.5.2 Server mit Ausfällen . . . . .	193
8.5.3 Prozessoren und Speicher mit Ausfällen . . . . .	198

8.5.4	Prozessoren und Speicher mit Ausfällen und Reparatur . . . . .	200
<b>8.6</b>	<b>Zusammenfassung und Ausblick . . . . .</b>	<b>203</b>
8.6.1	Einordnung dieser Seminararbeit in den Kontext des Seminars . . . . .	204
<b>8.7</b>	<b>Bewertung und Ausblick . . . . .</b>	<b>204</b>
	<b>Literaturverzeichnis . . . . .</b>	<b>205</b>

---

## Abstract

This work is about performability-modelling using Markov-Reward-Models. It defines what performability is and gives an introduction into metrics used in such models. It is discussed why and when analytical methods are used and it explains the basic background of Markov-models. After this the Markov-Reward-Modell is defined and explained by three examples using the Sharp software package. A short overview of tools for mathematical analysis is also offered.

## Kurzbeschreibung

Diese Seminararbeit beschäftigt sich mit Performability-Modellierung am Beispiel des Markov-Reward-Modelles. Nach einer Definition von Performability wird diskutiert, wann es sinnvoll ist, eine mathematische Analyse eines Modells anzugehen und wann eine Simulation vorzuziehen ist. Im weiteren Verlauf wird eine kurze Einführung in die verschiedenen Metriken gegeben, die in der Performability-Analyse verwendet werden können. Nach einem Kapitel über die wichtigsten mathematischen Hintergründe wird das Markov-Reward-Modell motiviert und eingeführt. Anschließend wird es an drei Beispielen verdeutlicht. Zur Auswertung der Beispiele wird das Sharp Software Paket eingesetzt. Weiterhin wird ein kurzer Überblick über die wichtigsten Programme zur mathematischen Analyse gegeben.

## 8.1 Einführung

### 8.1.1 Modelle

Bei dem Versuch, Systeme zu untersuchen, wird man schnell feststellen, daß man nicht umhin kommt, Modelle zu erstellen. Die Gründe sind vielfältig. Untersucht man zum Beispiel die Auswirkungen von Giftstoffen auf unser Ökosystem, so wäre es ziemlich verwerflich, dies in der freien Natur zu tun. Man wird also auf ein Modell zurückgreifen müssen.

Will man neue Techniken bei Weltraumausflügen erproben, so wird man dies auf der Erde nicht bewerkstelligen können, da die Bedingungen, die man bräuchte, zum Beispiel die Schwerelosigkeit, nicht realisiert werden können. Was hierbei weiterhilft ist wieder ein Modell. So übt beispielsweise die NASA ihre Weltraumausflüge unter Wasser an Modellen im Maßstab eins zu eins, was den Bedingungen im Weltall schon sehr nahe kommt.

Soll ein neues Rechnersystem erprobt werden, das gerade erst in der Entwicklung ist, so hilft es auch hier, sich ein Modell, zum Beispiel in Form einer Emulation, zu schaffen, um nicht planlos viel Geld in eine Entwicklung zu investieren, die im Anschluß zu nichts zu gebrauchen ist.

Bei der Modellierung eines realen System muß man von vielen Details abstrahieren. Je weniger Details das Modell enthält, desto weniger komplex und damit einfacher wird es in der Regel, das System zu analysieren. Betrachtet man aber zu wenige Details, so stimmt das Modell oftmals

nicht mehr mit der Wirklichkeit überein. Die Kunst ist folglich, das reale System so einfach wie möglich nachzubilden, so daß die Realität dennoch bestmöglich wiedergegeben wird. Anschließend können die gewonnen Erkenntnisse auf das Originalsystem übertragen werden.

Man spricht von einem adäquaten Modell, wenn es die zuvor genannten Eigenschaften besitzt. Ob ein Modell adäquat ist, man folglich den gewonnen Erkenntnissen trauen kann, ist aber immer fraglich. Bei den Details zum Modellbildungsprozeß verweise ich an dieser Stelle auf die Seminarbeiträge über die Leistungsmodellierung [Sae02] und die Zuverlässigkeitsmodellierung [Soe02] von Andreas Schäfer und Tilo Schröder, die das Thema ausführlich dargestellt haben. Diese Seminararbeit wird sich mit dem Sinn und Zweck von Performability-Analyse am Beispiel des Markov-Reward-Modelles auseinandersetzen. Dabei wird auch ein Augenmerk auf Werkzeuge gelegt, die zu dieser Analyse unterstützend eingesetzt werden können. Insbesondere werden wir SHARPE 2000 für unsere Beispiele einsetzen.

### 8.1.2 Performability

Der Begriff “Performability” ist künstlich geschaffen und setzt sich aus den englischen Begriffen **Performance** und **Dependability** zusammen. Übersetzt bedeuten diese soviel wie Leistung und Zuverlässigkeit. Erklärtes Ziel der Performability - Analyse und Modellierung ist es, die beiden seit langem getrennt voneinander erforschten Interessengebiete, einander anzunähern und in einem einzigen, einheitlichen Modell zu vereinen, um den Abhängigkeiten zwischen Leistung und Zuverlässigkeit Rechenschaft zu tragen. Die Performability - Modellierung befaßt sich folglich sowohl mit den Merkmalen des Leistungsverhalten als auch mit denen der Zuverlässigkeit.

Die Gefahr, an der Wirklichkeit vorbei zu modellieren, wird auf diese Weise deutlich gemindert. Auch wird dazu beigetragen, die Modellbildung intuitiver zu gestalten. Dieser Prozeß hat sich gerade in letzter Zeit immer stärker entwickelt, nicht zuletzt aus dem Grund, daß heutzutage mathematisch gut handhabbare Verfahren zur Leistungs- und Zuverlässigkeitsanalyse, mitsamt Lösungsverfahren, bekannt und anwendbar sind. Zudem wird mit steigendem Technikfortschritt der Ruf nach solchen Methoden immer lauter. Ziel ist es, bekannte Verfahren noch effektiver zu gestalten, sowie neue (noch) bessere Verfahren zu entwickeln.

### 8.1.3 Mathematisches Modell versus Simulation

Die Gründe, die für eine Simulation beziehungsweise für die mathematische Analyse sprechen, sind am leichtesten in den Nachteilen des jeweiligen Verfahrens zu suchen.

Ist der Modellbildungsprozeß abgeschlossen, so kann, ausgehend von diesem Modell, relativ leicht ein ausführbares Modell für die Simulation erstellt werden. Warum sollte man also auf mathematische Analyseverfahren zurückgreifen, wo man doch so schön mittels Simulation ansetzen könnte?

Ganz allgemein gilt, daß Simulation meist rechenaufwendiger als die Analyse ist. Andererseits erlauben Simulationen eine wesentlich höhere Modellierungsfreiheit. Diese Problematik wird durch die rare-event Simulation zusätzlich noch verstärkt.

Der Grund für eine Entscheidung pro mathematischen Analyseverfahren liegt in der Häufigkeit der Ereignisse verborgen. Gerade bei der Performability-Analyse sind wir daran interessiert, Modelle zu betrachten, deren Komponenten mit Fehlern behaftet sein können. Diese Fehler treten aber im Vergleich zu den “normalen” Ereignissen äußerst selten auf. Ein bis zwei Fehler, zufällig verteilt auf die gesamte Laufzeit der Simulation, sind keine Seltenheit. So kann es also

bei unseren Simulationsläufen durchaus vorkommen, daß Fehler in den Komponenten gar nicht auftreten. Aber gerade am Einfluß dieser Fehler auf unser Modell sind wir doch interessiert. Ansonsten würde uns eine Leistungsanalyse vollständig genügen. Wollen wir möglichst aussagekräftige Simulationsergebnisse erhalten, so sind wir gezwungen die Simulationszeit möglichst lange (theoretisch sogar unendlich lang) zu wählen. Das widerspricht aber dem Wunsch die Ergebnisse möglichst schnell, beziehungsweise überhaupt in endlicher Zeit, zu erhalten. Simulationen werden eingesetzt, wenn die Parameterräume zu groß werden. Dies ist zum Beispiel bereits bei der Analyse mehrerer Warteschlangen mit begrenzter Kapazität der Fall. Die Simulation "spaziert", durch den Zufall geleitet, deterministisch durch den Zustandsraum. Durch die häufige Wiederholung mit immer neuen Zufallszahlen wird die Stochastische Aussagekraft sichergestellt.

Weitere Gründe, die gegen Simulation sprechen, sind die statistischen Unsicherheiten. Zufallszahlen können noch so gut sein, betrachtet werden immer einzelne Stichproben. Selbst unter der Berücksichtigung von Konfidenzintervallen verbleibt eine Restwahrscheinlichkeit mit der die Ergebnisse, von denen angenommen wird, sie wären korrekt, falsch sind.

Aus diesem Grund liegt es also nahe, eine mathematische Analyse und Lösung unserer Modelle anzustreben. Dabei kommen wir allerdings nicht umhin, Einschränkungen an die Modelle zu stellen. Nur so können wir deren mathematische Berechenbarkeit überhaupt sicherstellen. So können wir zum Beispiel nicht alle beliebigen Verteilungsfunktionen der Wahrscheinlichkeitstheorie zulassen oder müssen gar die Komplexität und Anzahl der Zustände beschränken, um die partiellen Differentialgleichungen, die oftmals bei der Beschreibung auftreten, überhaupt noch mit numerischen Mitteln lösen zu können.

Folglich muß eine Entscheidung für das eine oder das andere Mittel immer im Kontext des einzelnen Anwendungsfalles getroffen werden. Da im Kontext einer Performability-Analyse von Internet-Routing-Strategien immer von im Verhältnis äußerst selten auftretenden Fehlern auszugehen ist, werden wir im folgenden ausschließlich mathematisch-analytische Verfahren betrachten.

### **8.1.4 Aufbau**

Diese Seminararbeit besteht aus mehreren Teilen, die aufeinander aufbauen. Am Ende stehen Anwendungsbeispiele, die einen Eindruck geben, wie die in dieser Arbeit beschriebenen Mittel in einer Performability-Analyse eingesetzt werden können.

Zu Beginn gibt das Kapitel 2 einen Einblick in die verschiedenen Metriken, die in der Performability-Analyse Verwendung finden. Ziel ist es, klar zu machen, welche Faktoren und Kenngrößen auf das Modell wirken und so in der Modellbildung Berücksichtigung finden müssen.

Anschließend beschäftigen wir uns in Kapitel 3 mit den mathematisch, theoretischen Grundlagen, wie dem Begriff der Wahrscheinlichkeit, dem stochastischen Prozeß, Markov-Ketten und Übergangsmatrizen. Die hier geschilderten Themenbereiche sind speziell auf die Anforderungen der nachfolgenden Kapitel abgestimmt und geben daher nur die notwendigsten Informationen. Für einen tieferen Einblick in die Materie sei auf die einschlägige Literatur verwiesen.

In Kapitel 4 beschäftigt wir uns mit dem Markov-Reward-Modell. Zunächst definieren wir das eigentliche Modell, welches auf den Markov-Ketten aufbaut. Im Anschluß daran werden die verschiedenen Eigenschaften und Aussagen des Modells erläutert und diskutiert. Ziel ist es, die Faktoren und Metriken, welche auf das Modell wirken, herauszuarbeiten, denn diese müssen im Modellbildungsprozeß berücksichtigt werden.

Zuletzt betrachten wir in Kapitel 5 verschiedene Beispiele. Dabei sollen die vielfältige Einsatz-

möglichkeiten des Markov-Reward-Modelles aufgezeigt werden. Außerdem soll hier an Beispielen ein Gefühl für die Modellierung vermittelt werden.

### 8.1.5 Danksagungen

Zunächst danke ich Herrn Prof. Kishor Trivedi von der Duke University in Durham, North Carolina, für die kostenlose Bereitstellung, des von ihm und seinem Team entwickelten Modellierungswerkzeuges SHARPE 2000.

Des weiteren danke ich Herrn Prof. Haverkort von der TU Aachen, der mir viele Informationen zum Thema Markov-Reward-Modellen zur Verfügung gestellt hat.

Und nicht zuletzt danke ich meinem Betreuer Dr. Johannes Lüthi, sowie den vielen anderen Korrekturlesern, insbesondere Andreas Schäfer und Stefan Wagenbrenner, für die viele Arbeit und Zeit, die sie in meine Seminararbeit investiert haben.

## 8.2 Metriken

Wenn wir Leistung und Zuverlässigkeit messen wollen, müssen wir uns erst einmal geeignete Meßgrößen zurechtlegen. Nehmen wir zum Beispiel einen Gateway-Rechner (vergleiche Abbildung 8.1). Wenn wir diesem eine Anfrage (engl.: request) eines Dienstes schicken, so gibt es zunächst 2 verschiedene Ereignismöglichkeiten:

1. Die Anfrage wird erfolgreich ausgeführt, oder
2. sie schlägt fehl und kann nicht ausgeführt werden.

Ist die Anfrage erfolgreich ausgeführt worden, gilt es zwei weitere Fälle zu unterscheiden:

1. Die Anfrage ist korrekt ausgeführt worden, oder
2. bei der Bearbeitung der Anfrage ist ein Fehler aufgetreten und wir haben eine fehlerhafte Antwort erhalten.

Ist es einem Dienst gelungen, eine korrekte Antwort des Systems auf seine Anfrage zu erhalten, wird es interessant, nach den Leistungsmerkmalen des Systems zu fragen. In unserem Fall wären dies die Antwortzeit, der Durchsatz und die Auslastung der Ressourcen, also des Servers beziehungsweise des Netzwerkes, welches unseren Rechner, von dem wir die Anfrage zuvor geschickt haben, mit dem Server verbindet. Von Interesse ist also eine Analyse des Leistungsvermögens.

Ist jedoch zuvor ein Fehler aufgetreten, so ist es für uns von besonderem Interesse, zu betrachten, welcher Fehler aufgetreten ist, warum dieser aufgetreten ist, mit welcher Wahrscheinlichkeit bzw. Häufigkeit oder auch relativen Häufigkeit der Fehler auftritt und wieviel Zeit zwischen dem Auftreten zweier Fehler liegt. Unser Augenmerk liegt also auch auf der Zuverlässigkeit des Systems.

### 8.2.1 System-Antwortzeit

Die Antwortzeit in einem System (engl.: **Response Time**) ist definiert als die Zeit, die zwischen der Anfrage des Benutzers und der Antwort des Systems liegt.

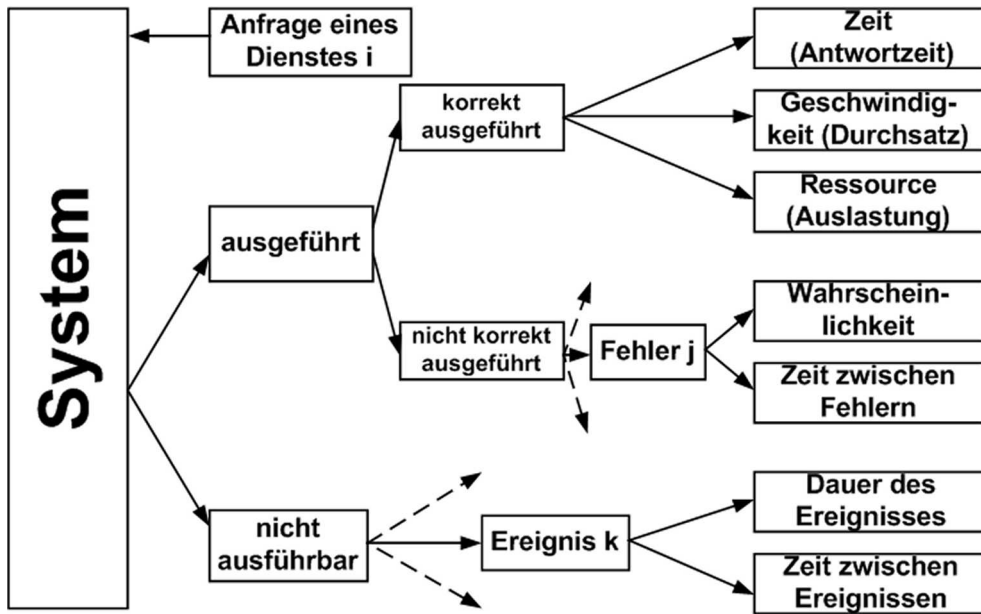


Abbildung 8.1: Metriken, die bei der Modellierung von Perfomability zu beachten sind (nach: [Jai91], Seite 37).

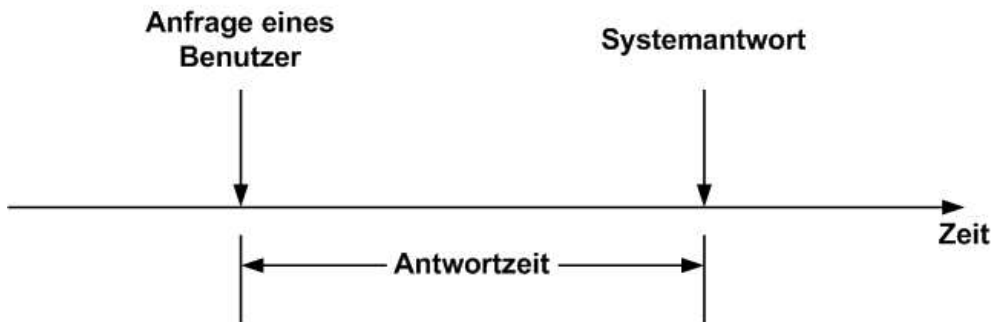


Abbildung 8.2: Ideale Anfrage und Systemantwort, also ohne jegliche Verzögerung (nach: [Jai91], Seite 37).

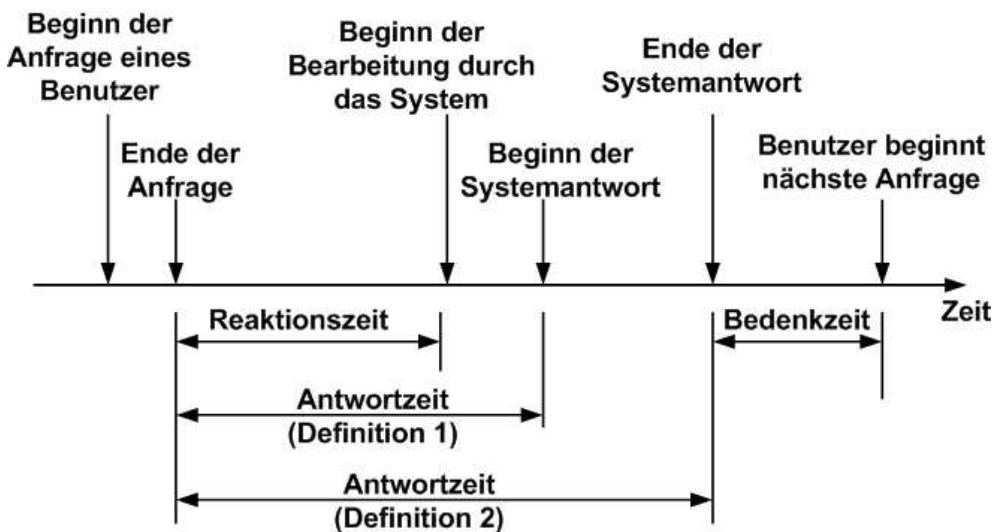


Abbildung 8.3: Realistische Anfrage und Systemantwort (nach: [Jai91], Seite 37).

Diese Definition ist solange eindeutig, wie keine Zeit für das Tippen der Anfrage durch den Benutzer oder die Ausgabe der Antwort durch das System benötigt wird, wie es in Abbildung 8.2 dargestellt ist. Betrachtet man jedoch ein reales System, so wie es in Abbildung 8.3 dargestellt ist, so können in diesem System sehr wohl Verzögerungen entstehen. In diesem Fall gibt es zwei mögliche, aber verschiedene Definitionen:

1. Die Antwortzeit kann als die Zeit zwischen Ende der Anfrage und dem Beginn der Antwort angesehen werden, oder
2. sie ist die Zeit zwischen Anfang der Anfrage und Ende der Systemantwort.

### 8.2.2 Durchsatz

Der Durchsatz (engl.: **Throughput**) beschreibt das Verhältnis von Anfragen pro Zeiteinheit, die ein System beantworten kann. Für stapelverarbeitende Systeme wird der Durchsatz in Jobs pro Sekunde gemessen. In interaktiven Systemen wird der Durchsatz in Anfragen pro Sekunde gemessen. Bei der CPU (Central Processing Unit, oder einfach Prozessor) spricht man von “Millions of Instructions per Second” (MIPS) oder “Million Floating-Point Operations Per Second” (MFLOPS) und der Durchfluß von Netzwerken wird typischerweise in “bits per second” (bps) gemessen.

### 8.2.3 Auslastung der Ressourcen

Die Auslastung charakterisiert den Anteil an der Zeit, in dem eine Ressource mit Anfragen beschäftigt ist. Definiert ist sie als Verhältnis von mit Arbeit beschäftigter Zeit zur gesamten verstrichenen Zeit im zu untersuchenden Zeitintervall. Die Zeit, in der eine Ressource nicht arbeitet, wird als “Idle-Zeit” (dt.: untätig) bezeichnet.

### 8.2.4 Verlässlichkeit

Die Verlässlichkeit (engl.: **Reliability**) beschreibt die Wahrscheinlichkeit, daß in einem betrachteten Zeitintervall  $[0; t]$  keine Fehler auftreten; das System also fehlerfrei arbeitet.

### 8.2.5 Verfügbarkeit

Die Wahrscheinlichkeit, daß ein System zu einem bestimmten Zeitpunkt  $t$  arbeitet, also funktionsfähig ist, wird Verfügbarkeit genannt (engl.: **Availability**). Unter der Voraussetzung, daß das System im Falle eines Fehlers nicht wieder repariert werden kann, fallen die Definitionen von Zuverlässigkeit und Verfügbarkeit zusammen.

### 8.2.6 Kosten

Kosten werden in Modellen häufig vernachlässigt, diese sind aber dennoch von hoher Bedeutung. Entscheiden sie doch meist über die Verwirklichung eines Projektes, das sich gerade in der Planung befindet. Zumindest stecken die Kosten aber den Rahmen ab, in dem sich das System bewegen wird. Dabei umfassen die Kosten die Preise für Hardware, Software, Installation und Wartung über Jahre hinweg. Angegeben werden sie zumeist bezogen auf ein Zeitintervall, zum Beispiel in tausend Euro pro Jahr.

Im Zusammenhang mit Zuverlässigkeit ist es aber auch interessant Ausfälle mit Kosten zu versehen. Diese Kosten können dann zum Beispiel Einbußen in Übertragungsgeschwindigkeiten sein oder auch das Aufwenden von Ressourcen für eine Reparatur.

## 8.3 Grundlagen

### 8.3.1 Grundlegende Begriffe

Zunächst beginnen wir mit ersten Definitionen des Wahrscheinlichkeitsbegriffs und vereinbaren eine einheitliche Syntax. Im weiteren legen wir grundlegende Begriffe, wie stochastischer Prozeß, Zustands- und Parameterraum fest. Diese Begriffe werden uns in den folgenden Definitionen und Sätzen immer wieder begegnen. Der Schwerpunkt dieses Kapitels liegt darin, die wichtigsten Eigenschaften von Markov-Ketten herauszuarbeiten.

Da hier nur die unbedingt notwendigen Begriffe erläutert werden können, sei an dieser Stelle auf das Nachschlagewerk von M. Fisz: "Wahrscheinlichkeitsrechnung und Mathematische Statistik" [Fis76] sowie das englischsprachige Buch von W. Feller: "An Introduction to Probability Theory and its Applications"[Fel67] verwiesen.

**Definition 8.3.1 (Zufallsexperiment)** *Ein Experiment, das unter Beibehaltung fest vorgegebener Bedingungen beliebig oft wiederholbar ist, ohne daß sich diese Wiederholungen gegenseitig beeinflussen, und dessen Ergebnis im Bereich gewisser Möglichkeiten ungewiß ist, heißt Zufallsexperiment. Die Menge  $\Omega$  aller möglichen Ergebnisse dieses Zufallsexperiments heißt Ergebnisraum des Zufallsexperiments.*

**Definition 8.3.2 (Ereignis)** *Jede Teilmenge des Ergebnisraumes  $\Omega$  eines Zufallsexperimentes heißt ein Ereignis des Zufallsexperimentes. Wenn das Ergebnis einer Durchführung des Zufallsexperimentes in der Teilmenge  $A$  der Ergebnismenge  $\Omega$  liegt, sagt man, das Ereignis  $A$  sei eingetreten.*

**Definition 8.3.3 ( $\sigma$ -Algebra)** *Eine Menge  $\mathcal{E} \subset \Omega$  von Ereignissen mit Ergebnisraum  $\Omega$  eines Zufallsexperimentes heißt  $\sigma$ -Algebra, genau dann, wenn gilt:*

1.  $\mathcal{E} \neq \emptyset$ ,
2.  $E_i \in \mathcal{E}, i \in \mathbb{N} \Rightarrow \bigcup_{i \in \mathbb{N}} E_i \in \mathcal{E}$ ,
3.  $E \in \mathcal{E} \Rightarrow \bar{E} \in \mathcal{E}$ .

**Definition 8.3.4 (Wahrscheinlichkeitsaxiome)** *Gegeben sei ein Zufallsexperiment mit der Ergebnismenge  $\Omega$  und  $\sigma$ -Algebra  $\mathcal{E}$ . Dann heißt  $p : \mathcal{E} \rightarrow \mathbb{R}$  Wahrscheinlichkeit(-smaß) über  $\Omega$ , genau dann, wenn gilt:*

1.  $0 \leq p(E) \leq 1$  für alle  $E \in \mathcal{E}$
2.  $p(\Omega) = 1$
3.  $p(A \cup B) = p(A) + p(B)$  für  $A \cap B = \emptyset$ , für  $A, B \subset \Omega$



**Definition 8.3.5 (stochastischer Prozeß)** Eine Familie  $\{X_t | t \in T\}$  bzw.  $\{X(t) | t \in T\}$  von Zufallsvariablen auf einem Wahrscheinlichkeitsraum  $(\Omega, \mathcal{E}, \omega)$  mit Parameterraum  $T$  heißt ein **stochastischer Prozeß**. Für jedes  $t \in T$  ist  $X_t$  eine Zufallsvariable. Unter dem **Zustandsraum**  $S$  des Prozesses verstehen wir die Menge aller möglichen Werte, die die Zufallsvariablen  $X_t$  annehmen können.

Um im folgenden doppelte Klammerungen der Form „({“ und „})“ zu vermeiden, werden die Klammern „{“ und „}“ häufig ausgelassen. Statt  $p(\{X = i\})$  wird also  $p(X = i)$  geschrieben. Des weiteren gilt der Ausdruck  $\{A|B, C\}$  immer als  $\{(A)|(B, C)\}$  geklammert. Auch diese Klammern lassen wir zumeist weg.

Stochastische Prozesse mit diskreten Zustands- und Parameterräumen heißen **stochastische Ketten**. Ereignisse der Form

$$\{X_t = i_t, X_{t-1} = i_{t-1}, \dots, X_0 = i_0\} \text{ für } t \in T, i \in S$$

werden **Basisereignisse** genannt, auf die sich die **Kettenformel**

$$\begin{aligned} & p(X_t = i_t, X_{t-1} = i_{t-1}, \dots, X_0 = i_0) \\ = & p(X_t = i_t | X_{t-1} = i_{t-1}, \dots, X_0 = i_0) \cdots p(X_1 = i_1 | X_0 = i_0) \cdot p(X_0 = i_0) \end{aligned}$$

anwenden läßt. Die Wahrscheinlichkeiten  $p(\{X_0 = i_0\})$  heißt **Anfangsverteilung** der stochastischen Kette, wobei  $\{A|B\}$  als  $A$  unter der Bedingung  $B$  zu lesen ist.

### 8.3.2 Markov-Ketten

Nachdem die wichtigsten theoretischen Grundlagen geschaffen worden sind, können wir dazu übergehen, Markov-Ketten zu definieren. Diese werden eine entscheidende Rolle in den später zu definierenden Markov-Modellen spielen.

**Definition 8.3.6 (Markov-Eigenschaft)** Eine Familie von reellwertigen Zufallsvariablen  $X_t$  über einem Zustandsraum  $S$  heißt **Markov-Kette** genau dann, wenn für alle  $i_0, i_1, i_2, \dots, i_{t-1}, i_t \in S$  und alle  $t_0, t_1, \dots, t_n \in T$ ,  $n \in \mathbb{N}$  gilt:

$$p(X_{t_n} = i_{t_n} | X_{t_{n-1}} = i_{t_{n-1}}, \dots, X_{t_0} = i_0) = p(X_{t_n} = i_{t_n} | X_{t_{n-1}} = i_{t_{n-1}}).$$

Die Menge  $T$  der Zeitpunkte  $t$  kann abzählbar oder überabzählbar sein. Im ersten Fall spricht man von einer **zeit-diskreten Markov Kette** (engl.: *discrete time Markov chain*), im zweiten von einer **zeit-kontinuierlichen Markov-Kette** (engl.: *continuous time Markov chain*).

Diese Ketteneigenschaft wird häufig auch als **Markov-Eigenschaft** bezeichnet.

**Satz 8.3.1 (Erweiterte Markov-Eigenschaft)** Sei  $X_t$  eine Markov-Kette. Für alle  $t \geq 2, n \geq 0$  und alle  $i_0, i_1, \dots, i_{t+n}$  gilt:

$$\begin{aligned} & p(X_{t+n} = i_{t+n}, \dots, X_t = i_t | X_{t-1} = i_{t-1}, \dots, X_0 = i_0) \\ = & p(X_{t+n} = i_{t+n}, \dots, X_t = i_t | X_{t-1} = i_{t-1}). \end{aligned}$$

## Performability Modellierung

*Beweis:* Beweis mittels vollständiger Induktion nach  $n$ .

$n = 0$ :

$$\begin{aligned} p(X_{t+0} = i_{t+0} | X_{t-1} = i_{t-1}, \dots, X_0 = i_0) &= p(X_t = i_t) \\ &= p(X_t = i_t | X_{t-1} = i_{t-1}) \\ &= p(X_{t+0} = i_{t+0} | X_{t-1} = i_{t-1}) \end{aligned}$$

$n \rightarrow n + 1$ :

$$\begin{aligned} \text{Setze: } A &:= \{X_{t+n+1} = i_{t+n+1}\} \\ B &:= \{X_{t+n} = i_{t+n}, \dots, X_t = i_t\} \\ C &:= \{X_{t-1} = i_{t-1}\} \\ D &:= \{X_{t-2} = i_{t-2}, \dots, X_0 = i_0\} \end{aligned}$$

$$\text{Somit ist zu beweisen: } p(B|CD) = p(B|C) \Rightarrow p(AB|CD) = p(AB|C)$$

$$\begin{aligned} p(AB|CD) &= p(A|BCD) \cdot p(B|CD) \\ \text{Markov-Eigenschaft } p(AB|CD) &= p(A|BC) \cdot p(B|C) = p(AB|C) \\ \text{Identität } \Leftrightarrow \frac{p(ABC)}{p(BC)} \frac{p(BC)}{p(C)} &= \frac{p(ABC)}{p(C)} \end{aligned}$$

□

Desweiteren gilt für Markov-Ketten die sogenannte erweiterte Markov-Eigenschaft für bedingte Ereignisse:

**Satz 8.3.2 (Erweiterte Markov-Eigenschaft)** Sei  $0 < t_1 < \dots < t_{k-1} < t_k < t$ ,  $k \in \mathbb{N}$  und  $X_t$  eine Markov-Kette, dann gilt für alle  $i, j_1, \dots, j_{k-1}, j_k \in S$ :

$$p(X_t = i | X_{t_k} = j_k, X_{t_{k-1}} = j_{k-1}, \dots, X_{t_1} = j_1) = p(X_t = i | X_{t_k} = j_k).$$

*Beweis:*

$$\begin{aligned} &p(X_t = i | X_{t_k} = j_k, X_{t_{k-1}} = j_{k-1}, \dots, X_{t_1} = j_1) \\ &= \sum_{i_{t-1}, j_1, \dots, i_{t_{k+1}}} p(X_t = i, X_{t-1}, \dots, X_{t_{k+1}} = i_{t_{k+1}} | X_{t_k} = j_k, \dots, X_{t_1} = j_1) \\ &= \sum_{i_{t-1}, j_1, \dots, i_{t_{k+1}}} p(X_t = i, X_{t-1}, \dots, X_{t_{k+1}} = i_{t_{k+1}} | X_{t_k} = j_k) \\ &= p(X_t = i | X_{t_k} = j_k) \end{aligned}$$

□

Die Wahrscheinlichkeit eines Basisereignisses einer Markov-Kette läßt sich also berechnen durch

$$\begin{aligned} &p(X_t = i_t, X_{t-1} = i_{t-1}, \dots, X_0 = i_0) \\ &= p(X_0 = i_0) \cdot p(X_1 = i_1 | X_0 = i_0) \cdot p(X_2 = i_2 | X_1 = i_1) \cdots p(X_t = i_t | X_{t-1} = i_{t-1}). \end{aligned}$$

Die Wahrscheinlichkeiten der Basisereignisse und damit alle Wahrscheinlichkeitsaussagen über Markov-Ketten lassen sich also genau dann bestimmen, wenn

1. die Anfangsverteilung der Zufallsvariablen  $X_0$ , **und**
2. alle bedingten Wahrscheinlichkeiten  $p(X_t = j | X_{t-1} = i)$

bekannt sind. Die Wahrscheinlichkeiten  $p(X_t = j | X_{t-1} = i)$  werden auch **Übergangswahrscheinlichkeiten** genannt.

### 8.3.3 Übergangs- und Intensitätsmatrizen

Übergangsmatrizen  $P$  werden zur Beschreibung von zeitdiskreten Markov-Ketten eingesetzt. Sie bestehen aus den einzelnen Übergangswahrscheinlichkeiten  $p_{ij}$  von dem Zustand  $i$  zum Zustand  $j$ . Die Parameter  $s$  und  $t$  sollen die Abhängigkeit des Systems von dem Zustand, in dem es sich zu diesem Zeitpunkt befindet, sowie von dem betrachteten Zeitpunkt  $t$  ausdrücken. In Matrixschreibweise zusammengefaßt sehen diese dementsprechen wie folgt aus:

$$P(s,t) = \langle p_{ij}(s,t) \rangle = \begin{pmatrix} p_{11}(s,t) & p_{12}(s,t) & p_{13}(s,t) & \dots \\ p_{21}(s,t) & p_{22}(s,t) & p_{23}(s,t) & \dots \\ p_{31}(s,t) & p_{32}(s,t) & p_{33}(s,t) & \dots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix}$$

Im Fall einer zeitkontinuierlichen Markov-Kette lassen sich analog zu den Übergangswahrscheinlichkeiten Übergangsraten definieren. Man erhält sie durch einen infinitesimalen Übergang von den zeitdiskreten zu zeitkontinuierlichen Markov-Ketten. Die Übergangsraten  $q$  läßt sich definieren durch:

$$q_{ij} = \lim_{\Delta t \rightarrow 0} \frac{p_{ij}(\Delta t)}{\Delta t}$$

$$q_{ii} = \lim_{\Delta t \rightarrow 0} \frac{p_{ii}(\Delta t) - 1}{\Delta t}$$

$q_{ij}$  beschreibt die Rate (Intensivität), für einen Zustandswechsel von  $i$  nach  $j$  und  $-q_{ii}$  die Rate (Intensivität) mit der der Zustand  $i$  "verlassen" wird. Mathematisch gesehen, stellen die  $q_{ij}$  Erwartungswerte exponentialverteilter Zufallsvariablen dar. Auch die Raten lassen sich zu einer Matrix  $Q$  zusammenfassen:

$$Q(s,t) = \langle q_{ij}(s,t) \rangle = \begin{pmatrix} q_{11}(s,t) & q_{12}(s,t) & q_{13}(s,t) & \dots \\ q_{21}(s,t) & q_{22}(s,t) & q_{23}(s,t) & \dots \\ q_{31}(s,t) & q_{32}(s,t) & q_{33}(s,t) & \dots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix}$$

## 8.4 Markov-Reward-Modelle

Im folgenden geht es um ein Modell, mit dem sowohl Aussagen zur Leistung als auch zur Zuverlässigkeit eines Systems getroffen werden können. Grundidee ist es ein Modell zu betrachten, das verschiedene Zustände einnehmen kann. Im zweiten Schritt wird versucht, dem

Modell mitzuteilen, welche Zustände als wünschenswert gelten sollen und welche Zustände zwar eingenommen werden können, aber negative Auswirkungen für das System haben.

### 8.4.1 Das Markov-Reward-Modell

Das Reward Modell besteht im wesentlichen aus einer Markov-Kette, es gibt also eine (endliche) Anzahl an Zuständen, die eingenommen werden können, sowie Übergänge von einem Zustand  $i$  in einen anderen Zustand  $j$ . Neu hinzu kommen sogenannte Rewards (engl.: Belohnung). Es werden zwei Arten von Rewards unterschieden:

1. **State Rewards** (dt.: Zustandsbelohnung)  $r_i, (i \in S)$ , die genau einem Zustand zugeordnet sind (für tiefer gehende Informationen siehe "Specification Techniques for Markov Modells"[Hav01]), und
2. **Impulse Rewards** (dt: Impulsbelohnung)  $v_{i,j}(i, j \in S)$ , die einem Zustandsübergang zugeordnet sind (Diese werden im weiteren nicht behandelt, tiefer gehende Informationen können "Dynamic Probabilistic Systems" [How71] entnommen werden).

Die State Rewards können als Kosten pro Zeiteinheit verstanden werden, die in einem Zustand fällig werden, oder aber als Gewinn, den das System in diesem Zustand pro Zeiteinheit erwirtschaftet. Mit Impulse Rewards können Gewichtungen während dem Übergang von einem Zustand in den nächsten vorgenommen werden. Negative Rewards sind durchaus erlaubt. Vollständig läßt sich das Modell im zeitdiskreten Fall mittels einer Übergangsmatrix  $P$  und dem Vektor  $\pi$  aller Anfangswahrscheinlichkeiten

$$\pi = p(0) = \begin{pmatrix} p_1(0) \\ p_2(0) \\ \vdots \\ p_n(0) \end{pmatrix}$$

beschreiben. Im zeitkontinuierlichen Fall wird statt der Übergangsmatrix  $P$  die Intensitätsmatrix  $Q$  genutzt.

Grundsätzlich gibt es zwei Untersuchungsansätze der Markov-Reward Modelle:

1. Die **Steady-State** (dt.: stationäre Zustands) Analyse, die im eingeschwungenen Zustand vorgenommen wird, und
2. die **Transient-State** (dt.: flüchtige Zustands) Analyse, die die gesamte Dynamik des Markov-Reward-Modelles mit in die Analyse einbezieht, deswegen aber auch erheblich komplexer ist.

Daher steigt der Aufwand zur Berechnung der transienten Analysen sehr schnell stark an. Die mathematisch-analytische Vorgehensweise wird daher sehr erschwert.

Stationäre Lösungen von Markov-Reward-Modellen lassen sich mittels numerischer Verfahren finden, aber auch eine diskrete Ereignissimulation ist als Lösungsverfahren durchaus denkbar. Dies bringt insbesondere den großen Vorteil, daß das zu untersuchende Markov-Reward-Modell zu Beginn der Simulation nicht explizit vorhanden sein muß.

Die Reward-Funktion wird häufig unter Zuhilfenahme verschiedener Leistungsanalyseverfahren bestimmt. Dazu wird zunächst angenommen, das System arbeite fehlerfrei. In diesem Fall entspricht die Konfiguration der Komponenten gerade dem Zustand  $i$  und  $r(i)$  beschreibt den Durchsatz des Systems im Zustand  $i$ . Auf diese Art und Weise läßt sich die gesamte Leistung des Systems im Zustand  $i$  bestimmen.

### 8.4.2 Steady-State Analyse

Steady-State-Analysen werden benutzt, um den Erwartungswert  $E\{r\}$  der Rewards  $r$  bei einem langen Versuch zu bestimmen. Betrachtet wird das Markov-Reward-Modell im eingeschwungenen Zustand. Errechnet wird das Verhalten ausgehend von einer stationären Markov-Kette. Voraussetzung ist die Unabhängigkeit des Erwartungswertes vom Anfangszustand.

$$E\{r\} = \sum_{i \in S} p_i \cdot r(i) \text{ im zeitdiskreten Fall} \quad (8.1)$$

beziehungsweise

$$E\{r\} = \sum_{i \in S} q_i \cdot r(i) \text{ im zeitkontinuierlichen Fall.} \quad (8.2)$$

Der Vektor  $p$  kann mittels des Ansatzes

$$p = p \cdot P \quad \text{mit} \quad \sum_{i \in S} p_i = 1$$

ermittelt werden. Der Vektor  $q$  läßt sich aus dem homogenen Gleichungssystem

$$p \cdot Q = 0 \quad \text{mit} \quad \sum_{i \in S} q_i = 1$$

bestimmen.

Die Komponenten  $p_i$  und  $q_i$  der Wahrscheinlichkeitsvektoren  $p$  und  $q$  können als diejenige Wahrscheinlichkeit interpretiert werden, mit der ein Zustand eingenommen wird.

Kleinere Gleichungssysteme, bis etwa 1000 Gleichungen, lassen sich mittels des Gaußschen Eliminierungsverfahren lösen. Bei größeren Gleichungssystemen muß auf andere Lösungsverfahren, wie die Gauß-Seidel-Iteration, zurückgegriffen werden. Für die Details der Lösungsverfahren sei auf [Kri80, Ste85] verwiesen.

### 8.4.3 Die Transient-State Analyse

Die Transient-State-Analyse untersucht den Erwartungswert  $E\{r(t)\}$  der Rewards zum Zeitpunkt  $t$ . Untersucht wird das dynamische Verhalten des Markov-Reward-Modelles. Analog zu den Gleichungen (8.1) und (8.2) lautet die Gleichung zur Bestimmung des Erwartungswertes im zeitdiskreten Fall

$$E\{r(t)\} = \sum_{i \in S} p_i(t) \cdot r(i) \text{ im zeitdiskreten Fall}$$

beziehungsweise

$$E\{r(t)\} = \sum_{i \in S} q_i(t) \cdot r(i) \text{ im zeitkontinuierlichen Fall.}$$

Der Ansatz zur Bestimmung von  $p(t)$  führt auf eine lineare Differentialgleichung 1. Grades:

$$\frac{dp(t)}{dt} = p(t) \cdot P \text{ im zeitdiskreten Fall} \quad (8.3)$$

beziehungsweise

$$\frac{dq(t)}{dt} = q(t) \cdot Q \text{ im zeitkontinuierlichen Fall.} \quad (8.4)$$

#### 8.4.4 Cumulative Rewards

Der **Cumulative Reward**  $CR$  (dt.: angehäuften Belohnung) drückt aus, wieviel Reward von einem System über einen definierten Zeitraum  $[0; t]$  hinweg "gesammelt" wird. Der Cumulative Reward wird berechnet, indem man über das Zeitintervall  $[0; t]$  integriert. So erhält man auch die nachfolgende Gleichung durch Integration von Gleichung (8.1) bzw. (8.2):

$$CR(t) = \int_0^t E\{r_s\} ds$$

Ähnlich läßt sich auch die Verteilungsfunktion  $l(t)$  der Cumulative Reward durch Integration der Differentialgleichung 8.3 bzw. 8.4 angeben. Auch Sie wird durch Integration auf dem Intervall  $[0; t]$  gewonnen. Sie lautet

$$\frac{dl(t)}{dt} = l(t)P + p(0) \quad \text{mit } l(0) = 0 \text{ im zeitdiskreten Fall}$$

beziehungsweise

$$\frac{dl(t)}{dt} = l(t)Q + q(0) \quad \text{mit } l(0) = 0 \text{ im zeitkontinuierlichen Fall.}$$

### 8.5 Anwendungsbeispiele

In diesem Kapitel beschäftigen wir uns mit einigen Beispielen. Diese sollen einen Eindruck vermitteln, wie man ein gegebenes System auf ein Modell abbilden kann, und welche Hilfsmittel man sich dabei zu nutzen machen kann. Für die Modellierung der Systeme werden wir Markov-Reward-Modelle einsetzen. Des weiteren sind die Systeme unter dem Gesichtspunkt einer Performability-Analyse ausgesucht worden. Wenn nicht gesondert erwähnt, sind die Beschreibungen der Programme den Konferenzbeiträgen aus "Computer Performance Evaluation" [Hav94] entnommen.

Das 1. Beispiel ist extrem einfach gewählt, mit nur 2 Zuständen, die beiden nachfolgenden Beispiele 2 und 3 sind deutlich komplexer, und geben einen schönen Eindruck in die Mächtigkeit der Modellierungsmöglichkeiten. Die Beispiele 2 und 3 sind stark an die Beispiele des Buches "Performance and Reliability Analysis for Computer Systems" [Sah98], Kapitel 12, angelehnt.

Da die Berechnung solcher Modelle relativ aufwendig ist und nur schwer von Hand zu berechnen sind und da bei einer transienten Betrachtung teilweise nur noch numerische Lösungsansätze möglich sind, widmen wir uns zunächst einigen Werkzeugen (Programmen) zur Modellierung und Berechnung. Erst anschließend werden wir uns mit den Beispielen auseinandersetzen.

### 8.5.1 Analysewerkzeuge

Auf dem Software-Markt existieren viele Software-Lösungen. Diese setzen unterschiedliche Schwerpunkte. Zur Zeit existieren wenige kommerzielle Lösungen. Die meisten Anwendungen werden vielmehr zur Forschungszwecken an Universitäten entwickelt. Es folgt eine kleine Übersicht, welche Programme existieren und für welchen Anwendungszweck sie geeignet sind:

#### **FiFiQueues**

Das Programm FiFiQueues ist zur Analyse von offenen Warteschlangen Netzwerken entwickelt worden. Es können verschiedene Warteschlangen mit begrenzter oder unbegrenzter Kapazität, sowie die Wege zwischen ihnen, nachgebildet werden. FiFiQueues wird an der Technischen Universität Aachen entwickelt. Die GUI ist in Java geschrieben. (Für weitere Informationen sei auf die Internetseite [Fif02] verwiesen.)

#### **Gallileo**

Gallileo ist ein Werkzeug zur Analyse von dynamischen Fehlerbäumen, die zur Zuverlässigkeitsanalyse eingesetzt werden. Es werden eine Reihe von speziellen Konstrukten für die sequenzielle Fehlerbetrachtungen in Kombination mit den Standard kombinatorischen Fehlerbäumen angeboten. Unabhängige Module werden automatisch erkannt, und die Restlichen durch Kombination bestimmt. Gallileo wird an der Universität Virginia in Zusammenarbeit mit der NASA entwickelt. Es ist vollständig in C++ geschrieben. Des weiteren sind in C++ programmierbare Schnittstellen implementiert. (Für weiter Informationen sei auf die Internetseite [Gal02] verwiesen.)

#### **Möbius**

Möbius ist ein leistungsstarkes Werkzeug für die Leistungs- und Zuverlässigkeitsanalyse. Es ist geeignet, große Zuverlässigkeitsmodelle zu validieren und unterstützt viele verschiedene Lösungsverfahren. Zudem werden viele verschiedene Spezifikationstechniken und Formalismen unterstützt. Das Entwicklerteam von Möbius sitzt an der Universität von Illinois. Das Projekt wurde in C++ verwirklicht.

#### **MRMSolve**

MRMSolve eignet sich zur Transient Analyse von besonders großen Markov-Reward-Modellen. Es können sowohl die über die Zeit angesammelten Rewards als auch das gesamte Zeitverhalten bestimmt und analysiert werden. Die Engine ist in C++ geschrieben, die GUI in Java. Die Entwickler sitzen an der Universität von Budapest. (Für weitere Informationen sei auf die Internetseite [Mrm02] verwiesen.)

## **SHARPE 2000**

Das SHARPE 2000 ist ein sehr weit entwickeltes und leistungsstarkes Werkzeug zur Zuverlässigkeits- und Leistungsanalyse. Die Abkürzung SHARPE steht für Symbolic Hierarchical Automated Reliability and Performance Evaluator (dt.: symbolischer, hierarchischer und automatisierter Zuverlässigkeits- und Leistungsüberprüfer) und existiert bereits seit 15 Jahren. Weiterentwickelt wird es an der Duke Universität von Durham, USA. Seit 2000 umfaßt das Softwarepaket auch eine in Java geschriebene GUI. Seine besondere Stärke steckt in der Unterstützung vieler Formalismen und Modelle, die auch miteinander kombiniert werden können. Ergebnisse können sowohl in numerischer Form, als auch in Form eines Graphen ausgegeben werden.

In den später folgenden Beispielen werden wir dieses Werkzeug für die Berechnung einsetzen. (Für weiter Informationen sei auf die Internetseite [Sha02] verwiesen.)

## **SPNP**

SPNP dient dem Lösen von Stochastischen Petri Netzen. Die Abkürzung SPNP steht für Stochastic Petri Net Package (dt.: stochastisches Petri-Netz Packet). Die Stochastischen Petri Netze werden in der Eingabesprache CSPL spezifiziert, welche C-basiert ist. Dementsprechend steht CSPL für C-based Stochastic Petri Net Language (dt.: auf C basierende Sprache zur Beschreibung stochastischer Petri-Netze). Eine graphische Benutzerschnittstelle, die diesen CSPL-Code erzeugt, ist ebenfalls erhältlich. Ebenfalls können auch die Parameter der Lösungsverfahren mit dieser GUI erzeugt werden. Neben analytisch numerischen Verfahren steht auch die Simulation als Lösungsverfahren zur Verfügung. Diese tritt automatisch immer dann in Einsatz, wenn die analytischen Verfahren fehlschlagen. SPNP wird an der Duke Universität von Durham, USA, entwickelt.

## **iSPN**

iSPN ist eine eine Weiterentwicklung der Eingabesprache CSPL. iSPN steht für integrated environment for modelling using Stochastic Petri Nets (dt.: eingebettete Umgebung zur Modellierung von stochastischen Petri-Netzen). Die Hauptkomponente des iSPN ist ein Editor für die graphische Modellierung und Eingabe von Petri Netzen. Des weiteren ist eine Sammlung von Routinen zur Visualisierung implementiert. iSPN ist unter anderem auch Bestandteil von SHARPE.

## **SREPT**

SREPT ist ein Software-Paket zur Bewertung und zur Vorhersage von Zuverlässigkeit. Die graphische Benutzerschnittstelle von SREPT ist in Java geschrieben. Die hinter der Benutzerschnittstelle steckende Engine ist zu Teilen in Java und zu Teilen in C geschrieben. SREPT steht für Software Reliability Estimation and Prediction Tool (dt.: Werkzeug für die Abschätzung und Vorhersage von SW-Zuverlässigkeit). Auch dieses wurde an der Duke Universität von Durham, USA, entwickelt. Die Möglichkeit zur ereignisorientierten Simulation steht ebenfalls zur Verfügung.



## STEADY

Der System Throughput Estimator for Advances Database sYstems (dt.: Durchsatzberechner für erweiterte Datenbanksysteme), STEADY, ist ein Werkzeug, das sowohl Entwickler als auch End-Benutzer unterstützen soll, die Leistung von parallelen, relationalen Datenbanken vorherzusagen. Die Vorhersage der Leistung, sowohl von Tabellen, als auch von Transaktionen auf spezifischen Maschinen, wird unterstützt. Es erlaubt dem Nutzer mit verschiedenen Einstellungen zu experimentieren. Eine graphische Benutzerschnittstelle ist leider nicht vorhanden. STEADY ist eine Entwicklung der Heriot-Watt Universität Riccarton in Edinburgh, Schottland.

## TANGRAM-II Environment

TANGRAM-II ist eine Modellierungs- und Experimentierungsumgebung für Computer- und Kommunikationssysteme. Sie wurde für Forschungs- und Lehrzwecke entwickelt. TANGRAM-I wurde in Prolog entwickelt, TANGRAM-II ist jedoch in C++ geschrieben und wesentlich leistungsfähiger. Es existiert eine graphische Oberfläche. TANGRAM-II ist eine Entwicklung der Universität von Rio de Janeiro, Brasilien.

## $\chi$ Prof-SDL

$\chi$  Prof-SDL ist ein Werkzeug, um Flaschenhalse im Leistungsverhalten von Software Spezifikationen zu erkennen. Das Programm gibt auch eine Anleitung zum Redesign. Die Designvorschläge müssen allerdings einer formalen Spezifikation genügen und in dem internationalen Telekommunikationsstandard SDL, Spezifikation and Description Language (dt.: Spezifikations- und Beschreibungssprache) niedergeschrieben werden. Eine, allerdings nicht sonderlich komfortable, Benutzeroberfläche ist vorhanden. Die Entwicklung des Programmes obliegt der Firma Telcordia Technologies, früher Bellcore, in Morristown NJ, USA.

### 8.5.2 Server mit Ausfällen

Für unser ersten Beispiel stellen wir uns ein kleines Firmennetz vor. Diese kleine Firmennetz besitzt eine Außenanbindung an das Internet über einen Router. Dieser Router hat die unangenehme Eigenschaft, daß er von Zeit zu Zeit wegen Fehlern ausfällt. Dann muss ein Techniker gerufen werden, um diesen Server zu reparieren.

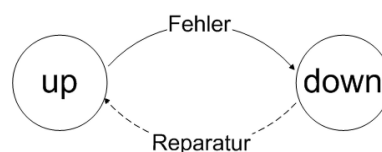


Abbildung 8.4: Router der ausfallen und wieder repariert werden kann

Mit einer Wahrscheinlichkeit von 80 Prozent arbeite der Router zu Beginn der Analyse korrekt. Im Zustand UP erhält die Firma von ihren Kunden 100 EUR pro Zeiteinheit. Fällt der Router allerdings aus entstehen der Firma Kosten in Höhe von 300 EUR pro Zeiteinheit.

Das System verbringe durchschnittlich 10 Zeiteinheiten im System up und  $3\frac{1}{3}$  Zeiteinheiten im Zustand down. Also beträgt die Rate für den Ausfall 0,1 und die Rate für eine Reparatur 0,3.

## Performability Modellierung

Unter Zuhilfenahme des SHARPE 2000 Software-Paketes werden wir im folgenden dieses Modell analysieren. Beispielhaft werden wir dazu die transiente Wahrscheinlichkeit des up und des down-Zustandes, den Erwartungswert des Rewards, die mittlere Zeit bis zum nächsten Fehler und die Verfügbarkeit im Zustand UP betrachten.

Die Abbildungen 8.5, 8.6, 8.7, 8.8 und 8.9 sind Screenshots aus der Modellierung des Beispiels mit SHARP 2000. Sie zeigen exemplarisch, wie die Eintragungen in SHARP 2000 gemacht werden. Das erste Bild zeigt das Programm, wie es sich gleich nach dem Starten präsentiert. Durch öffnen des Menüs File und auswählen des Menüpunktes New, gelangt man zu einem weiteren Fenster. In diesen läßt sich auswählen welches Modell zur Modellierung benutzt werden soll. Wir wählen hier den Menüpunkt Markov-Chain (vgl.: Abb 8.6).

Nun läßt sich das gewünschte Modell per Drag and Drop zusammenstellen. Wir modellieren hier für unser Beispiel zwei Zustände. Den einen benennen wir mit up und den anderen mit down. Im nächsten Schritt zeichnen wir die zwei Transitionen (Zustandsübergänge) ein und ordnen ihnen einen Impulse-Reward zu. Der Zustandsübergang von down nach up bekommt die Gewichtung 3 und der von up nach down eine einfache Gewichtung.



Abbildung 8.5: Das Hauptprogramm

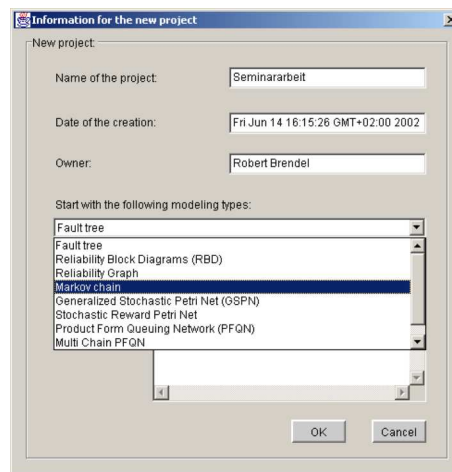


Abbildung 8.6: Neues Modell anlegen

Im nächsten Schritt wählen wir aus dem Menüpunkt Analysis Editor den Punkt Analyse aus. Es erscheint ein Fenster, in dem wir uns alle Punkte auswählen können, die wir gerne analysieren würden. Gemäß unsere Aufgabenstellung lassen wir die transiente Wahrscheinlichkeit zum Zeitpunkt  $t$ , Steady-State-Verfügbarkeit im UP-Zustand, die mittlere Zeit bis zu einem Fehler, die Zustandswahrscheinlichkeit im DOWN-Zustand und zuletzt noch den erwarteten Reward zum Zeitpunkt  $t$  berechnen. Natürlich interessiert uns nicht nur ein Wert zu einem Zeitpunkt  $t$  sondern möglichst viele, um das zeitliche Verhalten abschätzen zu können. Daher lassen wir uns

alle Werte zu den diskreten Zeitpunkten  $t \in \{1, 2, 3, \dots, 100\}$  ausgeben. Dazu müssen ein Startwert, ein Endwert und das Inkrement statt dem Zeitpunkt, durch Kommata getrennt, angegeben werden.

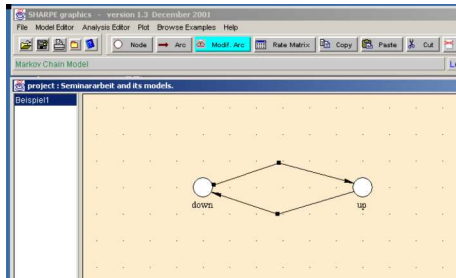


Abbildung 8.7: Das Modell wird per Drag & Drop zusammengestellt

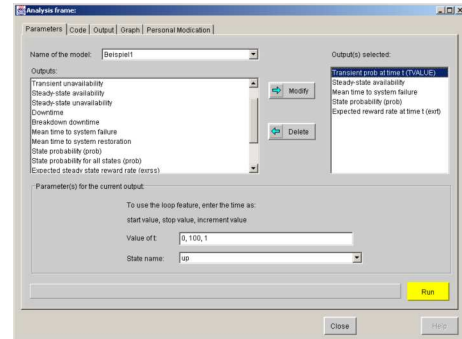


Abbildung 8.8: Die Analyse-Methoden-Auswahl

Mittels der graphischen Oberflächen läßt sich ein Code erzeugen, der von der SHARP Engine weiterverarbeitet werden kann. Dieser stammt noch aus den älteren textbasierten Versionen und kann auch direkt zur Eingabe verwendet werden.

In unserem Fall sieht der von SHARPE 2000 erzeugte Code wie folgt aus:

```
markov Beispiel1 readprobs down up 0.3 up down 0.1
* Reward configuration defined:
reward down rew_Beispiel1_down up rew_Beispiel1_up end
* Initial Probabilities defined:
down init_Beispiel1_down up init_Beispiel1_up end

* Reward configuration assigned:
bind
  rew_Beispiel1_down 0
  rew_Beispiel1_up 0
end

* Initial Probabilities assigned:
bind
  init_Beispiel1_down 0
  init_Beispiel1_up 0
end

echo
*****
echo ***** Outputs asked for the model: Beispiel1
*****
```

## *Performability Modellierung*

```
* Initial Probability: Anfangswahrscheinlichkeit
bind
  init_Beispiel1_up  0.8
  init_Beispiel1_down 0.2
end

func Transient_prob_at_Time_T(t) tvalue(t;Beispiel1, up) loop t,0,
100, 1 expr Transient_prob_at_Time_T(t) end

* UP configuration: up_states
bind
  rew_Beispiel1_up  1
  rew_Beispiel1_down 0
end

var SS_Avail exrss(Beispiel1) echo Steady_State Availability for
Beispiel1 expr SS_Avail

* UP configuration: up_states
bind
  rew_Beispiel1_up  1
  rew_Beispiel1_down 0
end

var MTTSF exrss(Beispiel1) / ((prob(Beispiel1, up) * 1)) echo Mean
time to system failure for Beispiel1 expr MTTSF

var State_Probability prob(Beispiel1, down) echo State probability
of the node: down expr State_Probability

* REWARD configuration: state_reward
bind
  rew_Beispiel1_up  1
  rew_Beispiel1_down -3
end

func Exp_Reward_Rate_T(t) exrt(t; Beispiel1) loop t,0, 100, 1 expr
Exp_Reward_Rate_T(t) end

end
```

Die von SHARPE berechneten Werte sind in den Tabellen 8.1, 8.2 und 8.3 zusammengefaßt.

$t$	$p(T_t)$
0.000000	8.0000e-001
1.000000	7.5092e-001
2.000000	7.5002e-001
3.000000	7.5000e-001
4.000000	7.5000e-001
5.000000	7.5000e-001
6.000000	7.5000e-001
7.000000	7.5000e-001
8.000000	7.5000e-001
9.000000	7.5000e-001
10.000000	7.5000e-001
⋮	⋮
50.000000	7.5000e-001
100.000000	7.5000e-001

Tabelle 8.1: *Transiente Wahrscheinlichkeit des UP-Zustandes*

$t$	$E\{T_t\}$
0.000000	2.0000e-001
1.000000	3.6631e-003
2.000000	6.7093e-005
3.000000	1.2288e-006
4.000000	2.2507e-008
5.000000	4.1223e-010
6.000000	7.5504e-012
7.000000	0.0000e+000
8.000000	0.0000e+000
9.000000	0.0000e+000
10.000000	0.0000e+000
⋮	⋮
50.000000	0.0000e+000
100.000000	0.0000e+000

Tabelle 8.2: *Erwartungswert der Reward-Rate zum Zeitpunkt  $t$*

Steady-State Verfügbarkeit im Zustand UP:	7.5000e-001
Mean Time to System Failure:	1.0000e+000
Wahrscheinlichkeit des DOWN-Zustandes:	2.5000e-001

Tabelle 8.3: *weitere Analyse-Ergebnisse*

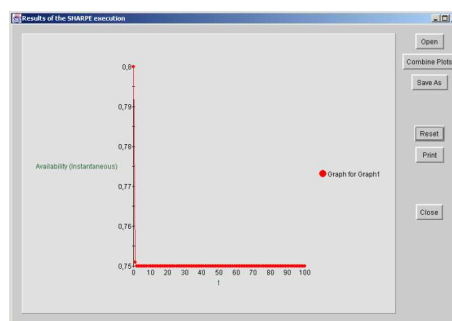


Abbildung 8.9: *augenblickliche Verfügbarkeit*

Als letztes lassen wir uns noch die augenblickliche Verfügbarkeit in Form eines Graphen von SHARPE ausgeben. Bild 8.9 zeigt das Ergebnis.

In Tabelle 8.1 läßt sich gut beobachten, wie das System einschwingt. Bereits nach der dritten Zeiteinheit ist dieser Zustand (Steady State) erreicht. In Tabelle 8.2 läßt sich ablesen, daß die Firma aus diesem Beispiel auf Dauer Insolvenz beantragen muß, da der erwartete Gewinn gegen Null geht.

### 8.5.3 Prozessoren und Speicher mit Ausfällen

Für unser nächstes Beispiel betrachten wir einen Rechner mit zwei Prozessoren. Dieser besitzt drei Speicher. Die Prozessoren können auf alle Speicher zugreifen. Das System ist so lange einsatzbereit, wie mindestens ein Speicher und ein Prozessor funktionstüchtig sind. Wenn ein Fehler im System auftritt, wird angenommen, daß der Speicher bzw. Prozessor abgeschaltet wird, so daß im System deswegen in keiner anderen Komponente ein Fehler auftreten kann. Speicher und Prozessor Fehlerraten werden als exponentialverteilt angenommen mit den Parametern  $\lambda_m$  und  $\lambda_p$ . Das System wird vollständig durch den Graphen in Abbildung 8.10 beschrieben. Dabei steht jeder Knoten  $ij$  für den Zustand des Systems, wenn  $i$  Speicher und  $j$  Prozessoren funktionieren. Die durchgezogenen Pfeile symbolisieren die Übergänge, wenn ein Fehler in einem Speicher oder einem Prozessor auftritt.

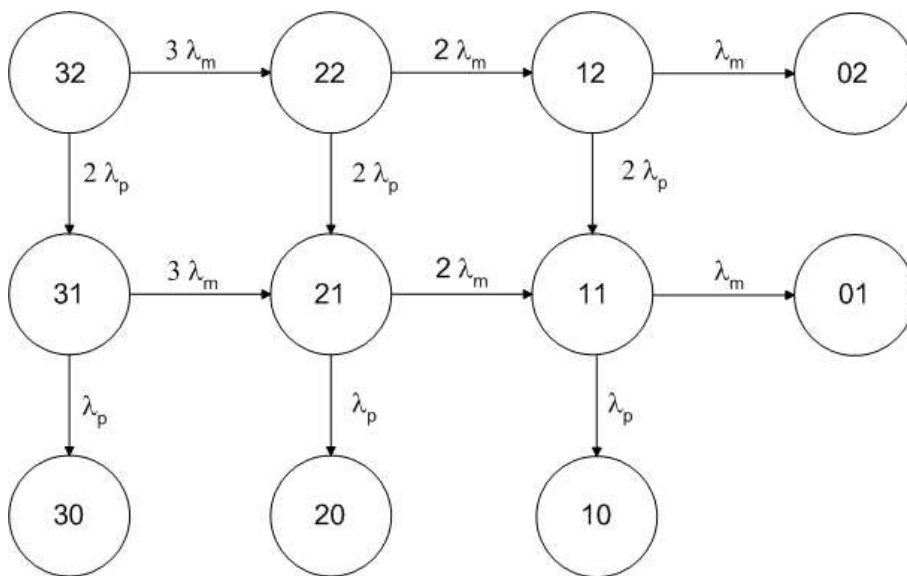


Abbildung 8.10: 2 Prozessoren und 3 Speicher; die Komponenten können ausfallen, eine Reparatur ist nicht möglich

Des weiteren besitzt das System eine Reward-Rate-Funktion, mit der das System bestimmen kann, welche Komponenten in welchem Zustand ausgefallen sind. Dafür setzen wir die folgende Funktion an:

$$r_{ij} = m \left( 1 - \left( 1 - \frac{1}{m} \right)^l \right) \quad \text{mit} \quad l = \min(i, j) \\ \text{und} \quad m = \max(i, j)$$

Der SHARPE-Code sieht dementsprechend so aus:

```

bind mlambda 1/(2*720)
bind plambda 1/720

32 22 2*mlambda
22 12 2*mlambda
12 02 mlambda

```

```

31 21 3*mlambda
21 11 2*mlambda
11 01 mlambda
32 31 2*plambda
31 30 plambda
22 21 2*plambda
21 20 plambda
12 11 2*plambda
11 10 plambda

```

```

reward
  32 r32
  22 r22
  12 r12
  31 r31
  21 r21
  11 r11
end 21 1.0

```

```

bind
  r32 15/9
  r22 3/2
  r12 1
  r31 1
  r21 1
  r11 1
end

```

```

reward (3mem-2proc) expr rvalue(4;3mem-2proc) expr exrt
(20,3mem-2proc) expr cexrt (20;3mem-2proc) eval (3mem-2proc ) 0 4
1 reward eval (3mem-2proc) 0 100 10 cexrt

```

```

expr sum (i,1,3,sum (j,1,2,(sreward(3mem-2proc,$(i)$ (j)) *
value(20;3mem-2proc,$(i)$ (j))) ) )

```

```

bind
  r21 1
  r22 1
  r12 1
  r31 1
  r21 1
  r11 1
end

```

```

cdf (3mem-2proc) reward (3mem-2proc)

```

```

end

```

## Performability Modellierung

Im folgenden sind die Ergebnisse in Tabellen zusammengefaßt. In Tabelle 8.4 befindet sich die Berechnung der angesammelten Rewards in Abhängigkeit vom Zeitparameter  $t$ . Tabelle 8.5 zeigt die Verteilungsfunktion der Rewards in Abhängigkeit des Reward  $r$ .

$t$	$E\{r(t)\}$
0.0000	0.0000 e+00
1.0000	1.6557 e+01
2.0000	3.2897 e+01
3.0000	4.9022 e+01
4.0000	6.4934 e+01
5.0000	8.0637 e+01
6.0000	9.6132 e+01
7.0000	1.1142 e+02
8.0000	1.2651 e+02
9.0000	1.4140 e+02
10.0000	1.5609 e+02

Tabelle 8.4: Erwartungswert des *cumulativen Rewards*

$r$	$l_r$
0.0000	0.0000 e+00
1.0000	1.1564 e-06
2.0000	4.6217 e-06
3.0000	1.0390 e-05
4.0000	1.8455 e-05

Tabelle 8.5: Verteilungsfunktion *der Rewards*

Zum Schluß wird auch noch die Time-to-Failure Verteilung berechnet und ausgegeben:

$$\begin{aligned}
 & 1.6667 \cdot 10^{-02} \cdot r(1) \cdot \exp(-2.7778 \cdot 10^{-03} r) \\
 + & 1.0000 \cdot 10^{+00} \cdot r(0) \cdot \exp(0.0000 \cdot 10^{+00} r) \\
 - & 1.8000 \cdot 10^{+01} \cdot r(0) \cdot \exp(-2.0833 \cdot 10^{-03} r) \\
 - & 9.0000 \cdot 10^{-00} \cdot r(0) \cdot \exp(-2.7778 \cdot 10^{-03} r) \\
 + & 2.0000 \cdot 10^{+01} \cdot r(0) \cdot \exp(-2.9167 \cdot 10^{-03} r) \\
 + & 6.0000 \cdot 10^{-00} \cdot r(0) \cdot \exp(-3.4722 \cdot 10^{-03} r)
 \end{aligned}$$

Erwartungswert:  $1.1349 \cdot 10^{03}$

Varianz:  $5.3153 \cdot 10^{05}$

Also fällt etwa alle 1100 Zeiteinheiten ein Speicher oder ein Prozessor aus. Aus Tabelle 8.4 zeigt ein immer langsames Zunehmen des Erwartungswertes mit der steigender Zeit  $t$ . Die Begründung hierfür liegt in der Tatsache begründet, daß ein Ausfall einer Systemkomponente mit steigender Zeitspanne  $[0; t]$  immer wahrscheinlicher wird.

### 8.5.4 Prozessoren und Speicher mit Ausfällen und Reparatur

Werfen wir noch einen weiteren Blick auf das Beispiel aus Kapitel 8.5.3. Diesmal bauen wir die Möglichkeit ein, daß Komponenten, die ausgefallen sind, wieder repariert werden können. Für



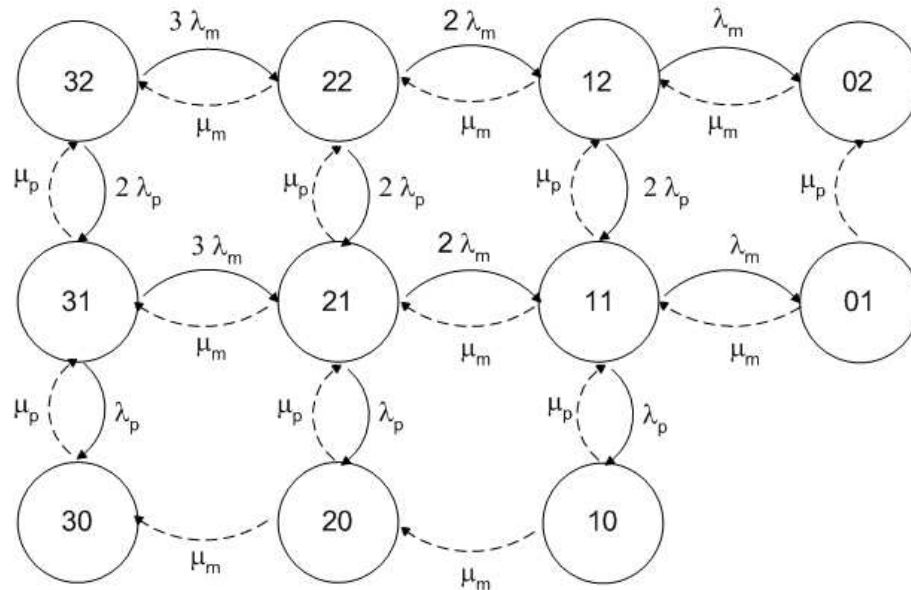


Abbildung 8.11: 2 Prozessoren und 3 Speicher; die Komponenten können ausfallen, eine Reparatur ist möglich

die Reparatur eines Processors und eines Speichers gibt es je eine Ressource. Die Reperaturzeiten für Prozessor und Speicher sind exponentialverteilt mit Parameter  $\mu_p$  und  $\mu_m$ . Das System ist aufgebaut wie in der Abbildung 8.11. Die durchgezogenen Pfeile symbolisieren die Übergänge beim Auftreten eines Fehlers in einem Prozessor oder einem Speicher. Die gestrichelten Pfeile repräsentieren die Übergänge bei denen ein Prozessor oder ein Speicher repariert wird.

Das Modell in Code gegossen sieht dann so aus:

```
markov 3mem-2proc readprobs
```

```
* mem failure
32 22 3*mlambda
22 12 2*mlambda
12 02 mlambda
31 21 3*mlambda
21 11 2*mlambda
11 01 mlambda
```

```
* processor failure
32 31 2*plambda
31 30 plambda
22 21 2*mlambda
12 11 2*plambda
11 10 plambda
```

```
* processor repair
```

## *Performability Modellierung*

```
30 31 pmu
31 32 pmu
20 21 pmu
21 22 pmu
10 11 pmu
11 12 pmu
01 02 pmu

* mem rapair
22 32 mmu
12 22 mmu
02 12 mmu
21 31 mmu
11 21 mmu
01 11 mmu

reward
    21 r32
    22 r22
    12 r12
    31 r31
    21 r21
    11 r11
end

32 1.0
end

bind
    mlambda 1/(2*720)
    plambda 1/720
    pmu 1/4
    mmu 1/2
    r32 15/19
    r22 3/2
    r12 1
    r31 1
    r21 1
    r11 1
end

expr exrss (3mem-2proc)
loop t,0,30,5
    expr exrt (t;3mem-2proc)
    expr cexrt (t;3mem-2proc)
    expr cexrt (t;3mem-2proc)/t
end
```

end

Tabelle 8.6 zeigt die Ausgabe des SHARPE-Programmes zusammengefaßt nach den Zeitpunkten  $t$ :

Zeit	Ergebnisse
$t = 0$	exrt (t;3mem-2proc): 1.6667e+00 cexrt (t;3mem-2proc): 0.0000e+00 cexrt (t;3mem-2proc)/t: 0.0000e+00
$t = 5$	exrt (t;3mem-2proc): 1.6607e+00 cexrt (t;3mem-2proc): 8.3153e+00 cexrt (t;3mem-2proc)/t: 1.6631e+00
$t = 10$	exrt (t;3mem-2proc): 1.6592e+00 cexrt (t;3mem-2proc): 1.6614e+01 cexrt (t;3mem-2proc)/t: 1.6614e+01
$t = 15$	exrt (t;3mem-2proc): 1.6587e+00 cexrt (t;3mem-2proc): 2.4909e+01 cexrt (t;3mem-2proc)/t: 1.6606e+00
$t = 20$	exrt (t;3mem-2proc): 1.6586e+00 cexrt (t;3mem-2proc): 3.3202e+01 cexrt (t;3mem-2proc)/t: 1.6601e+00
$t = 25$	exrt (t;3mem-2proc): 1.6586e+00 cexrt (t;3mem-2proc): 4.1495e+01 cexrt (t;3mem-2proc)/t: 1.6601e+00
$t = 30$	exrt (t;3mem-2proc): 1.6587e+00 cexrt (t;3mem-2proc): 4.9788e+01 cexrt (t;3mem-2proc)/t: 1.6596e+00

Tabelle 8.6: *Ergebnisse*

Die Tabelle 8.6 gibt zu jedem Zeitpunkt  $t$  den Erwartungswert des Reward-Rate (exrt), den Cumulativen Reward (cexrt) und den über die Zeit gemittelten Cumulativen Reward (cexrt/ $t$ ) wieder. Die Ergebnisse zeigen einerseits, daß das System zwar im Mittel weniger leistet, als es maximal möglich wäre, es aber andererseits auf lange Zeit gesehen gleichmäßig weiter arbeitet, obwohl Fehler auftreten.

## 8.6 Zusammenfassung und Ausblick

Abschließend drängen sich noch einige Fragen auf: Zum einen ist noch zu klären, ob und wie die mathematisch geprägten Ansätze der Performability-Analyse bei einer Analyse der Leistung und Zuverlässigkeit von Internet-Routing-Strategien Einfluß finden werden. Dazu müssen wir uns erst einmal eine Strategie zurecht legen, wie die Analyse angegangen werden kann. Auch müssen andere, weitere Methoden abgewogen werden. Erst wenn all diese Fragen geklärt sind, kann in die eigentliche Phase der Analyse übergegangen werden.

### **8.6.1 Einordnung dieser Seminararbeit in den Kontext des Seminars**

Die Bedeutung der Performability Modellierung für das Seminaroberthema “Bewertung von Internet-Routing-Strategien” wird aus der Fragestellung erkenntlich, welche Metriken an die Routingstrategien angelegt werden sollen. Bewertet werden soll das Leistungsverhalten und die Zuverlässigkeit von dynamischen Routing-Strategien. Leistung und Zuverlässigkeit sind wiederum miteinander verkoppelt. Im Einzelfall kann das zum Beispiel bedeuten, daß ein Ausfall des zu untersuchenden Systems die gesamte Leistung auf Null ziehen kann. Die Performability schafft genau diese geforderte Verbindung zwischen Leistung und Zuverlässigkeit und liefert Methoden und Modelle zur gleichzeitigen Analyse der beiden Gebiete.

Systeme werden häufig in Warteschlangennetzwerken oder Petri-Netzen modelliert. Zur Analyse können diese auf Markov-Reward-Modelle abgebildet werden. Diese Art der Modellierung und Umsetzung macht die Modellbildungsprozeß leichter, indem höhere Modellierungsmethoden genutzt werden, zugleich wird die Möglichkeit geschaffen die Vorteile, die aus der gleichzeitigen Betrachtung von Leistung und Zuverlässigkeit erwachsen, auszunutzen. Diese Möglichkeit ist aber dennoch mit Vorsicht zu genießen, da die Menge der Zustände und damit der aufzuwendene Rechenzeit sehr schnell in die Höhe schnell. Man spricht von einer Zustandsexplosion.

## **8.7 Bewertung und Ausblick**

Versuche in China, wo eines der modernsten Telekommunikationsnetzwerke geschaffen worden ist, zeigen, daß durch Messung am System alleine keine Erkenntnisse über Leistungsvermögen und Zuverlässigkeit gewonnen werden können. Im speziellen tritt bei ihnen das Problem in Erscheinung, daß ihre Netze nicht genügend Last tragen, als daß die gewonnen Werte auch nur im geringsten aussagekräftig werden könnten.

Schafft man künstlich Systemlast, so steht man einerseits vor dem Problem, daß bereits ein Modell für die Lastcharakteristik geschaffen werden muß, um die Last bei spezifischen Benutzerprofilen nachbilden zu können, andererseits brechen aber die Meßmethoden zusammen, wenn das Netz an die Grenzen seiner Kapazität gebracht wird.

Auch stehen dynamische Routing-Strategien gerade erst am Anfang ihrer Entwicklung und sind vielmals noch gar nicht implementiert. Aus diesen Gründen werden wir in unseren Bestrebungen, eine Performability Analyse der Routing-Strategien zu betreiben, auf Modelle zurückgreifen müssen.

Bei einer Analyse wird man folglich zunächst einmal das Problem haben, sich darauf einigen zu müssen, welche Metriken an das System anlegt werden müssen. Der nächste Schritt wird folglich sein, Modelle zu entwickeln, die die Systeme anhand dieser Metriken abbilden.

Im darauffolgenden Schritt muß die Entscheidung getroffen werden, ob die Modelle anhand von Simulation oder einer mathematischen Analysetechnik besser untersucht werden können. Diese Entscheidung wird wesentlich davon abhängen, wie fehleranfällig das jeweilige Routing-Protokoll ist, und wird für jedes einzelne Protokoll neu getroffen werden müssen.

Gerade die Analyse von dynamischen Routing-Protokollen zieht komplexe und große Modelle mit sich. Aus diesem Grund muß ein geeignetes Werkzeug ausgewählt werden. Dies trifft sowohl im Fall der Simulation als auch im Fall der mathematischen Analyse zu.

“Quality of Service” ist ein in diesem Zusammenhang gerade in letzter Zeit heiß diskutiertes Thema. Es beschäftigt sich mit der Fragestellung, wie man jedem Nutzer eine gewisse Grundqualität der von ihm genutzten Dienste zusichern kann. Das Thema insbesondere auch auf diese Fragestellung auszudehnen, wird lohnend sein, da gerade die Router einen wesentlichen Teil zu dieser Qualitätssicherung beitragen können. Zudem hat zum Thema “Quality of Service” schon ein Normungsprozeß begonnen, dessen Ergebnisse in die von uns zu entwickelnden Modelle zumindest als Metriken Einfluß nehmen werden.

Da es an Erfahrungen mit den dynamischen Internet-Routing-Strategien noch fehlt, sehe ich einen großen Bedarf an Analysen, sowohl an bestehenden Systemen, als auch an noch nicht implementierten Systemen, mit dem Ziel, mögliche Schwachstellen und Stärken in den jeweiligen Konzepten zu finden und so bessere, leistungsstärkere und zuverlässigere Konzepte zu finden. Nur so kann dem “Boom” im Netz begegnet werden und der bevorstehende Zusammenbruch abgewendet werden, der uns erwartet, wenn die Nutzung des Internet in der Form anhält, wie es sich momentan abzeichnet.

# Literaturverzeichnis

- [Ave98] R. AVENHAUS: *Skript zur Vorlesung Statistik, UniBw München, 1998*
- [Ave99] R. AVENHAUS: *Skript zur Vorlesung Quantitative Modelle für Rechen- und Kommunikationssysteme, UniBw München, 1999*
- [How71] R.A. HOWARD: *Dynamic Probabilistic Systems Vol. II: Semi-Markov and Decision Processes, Wiley & Sons, New York, 1971*
- [Fel67] W. WELLER: *An Introduction to Probability Theorie and Mathematical Statistics. Wiley & Sons, New York, 1967*
- [Fis76] M. FISZ: *Wahrscheinlichkeitsrechnung und Mathematische Statistik. 8. Auflage, VEB Deutscher Verlag der Wissenschaften, Berlin, 1976*
- [Fif02] INTERNETSEITE DER FAKULTÄT FÜR INFORMATIK AN DER TU AACHEN: *Beschreibung von FifiQueues,*  
*<http://www-lvs.informatik.rwth-aachen.de/tools/#FiFiQueues>*
- [Gal02] INTERNETSEITE DER FAKULTÄT FÜR INFORMATIK AN DER UNIVERSITÄT VON VIRGINIA: *Beschreibung von Gallileo,*  
*<http://www.cs.virginia.edu/~ftree/>*
- [Hav94] B. R. HAVERKORT, H. C. BOHNENKAMP, C. U. SMITH: *Computer Performance Evaluation, Modelling Techniques and Tools, 11th International Conference, TOOLS 2000, Schaumburg, IL, USA, Machrch 2000 Proceedings, Springer, 1994*
- [Hav98] B.R. HAVERKORT, K.S. TRIVEDI: *Spezifikation und Generation of Markov Reward Modells, Discrete-Event Dynamic Systems; Theory and Applications 3, S. 219-195, John Wiley & Sons, Berlin, 1998*
- [Hav98a] B.R. HAVERKORT: *Performance of Computer Communication Systems: A Modell based Approach, Wiley & Sons, Berlin, 1998*
- [Hav01] B. R. HAVERKORT, R. MARIE, F. RUBINO UND KISHOR TRIVEDI: *Performability Modelling, Techniques and Tools, John Wiley & Sons, New York, 2001*
- [Hei97] A. HEIN: *Conjoint Simulation - A Modeling Framework for Combined Performance and Dependability Analysis of Computer Systems, Society for Computer Simulation Int., Erlangen, 1997*
- [Jai91] R. JAIN: *The Art of Computer Systems Performance Analysis, Techniques and tools for experimental design, measurement, simulation and modeling, John Wiley & Sons, New York, 1991*

- [Kri80] U. KRIEGER, B. MÜLLER-CLOSTERMANN, M. SCITTNICK: *Modelling and Analysis of Communication Systems Based on Computational Methods for Markov Chains*, *IEEE Journal on Selected Areas in Communications*, Wiley & Sons, New York, 1990
- [Mrm02] POSTSCRIPTDOKUMENT VON DER UNIVERSITÄT VON BUDAPEST: *Beschreibung des RMRSolve-Werkzeuges*,  
<http://webspn.hit.bme.hu/~telek/cikkek/racz00f.ps.gz>
- [Sah98] R. SAHNER, K. TRIVEDI UND A. PULIAFITO: *Performance and Reliability Analysis of Computer Systems, An Example-Based Approach using the SHARPE Software Package*, Kluwer Academic Publishers, 1998
- [Sae02] ANDREAS SCHÄFER: *Leistungsmodellierung, Seminararbeit im Rahmen des Studienprojekt Bewertung von Internet-Routing-Strategien, UniBw München, 2002*
- [Sha02] INTERNETSEITE DER FAKULTÄT FÜR INFORMATIK AN DER DUKE UNIVERSITY: *Beschreibung und Dokumentation des SHARPE 2000 Software Packetes*,  
[http://www.ee.duke.edu/~kst/software\\_packages.html](http://www.ee.duke.edu/~kst/software_packages.html)
- [Soe02] TILO SCHRÖDER: *Zuverlässigkeitsmodellierung, Seminararbeit im Rahmen des Studienprojekt Bewertung von Internet-Routing-Strategien, UniBw München, 2002*
- [Ste85] W. J. STEWART, A. GOYAL: *Matrix Methods on Larger Dependability Models*, *IBM Research Report RC 11485*, 1985
- [Tri92] K.S. TRIVEDI, J.K. MAPPALA, S.P. WODET, B.R. HAVERKORT: *Composite performance Dependability Analysis, Performance Evaluation*, Wiley & Sons, New York, 1992
- [Tai96] A. T. TAI, J. F. MEYER, A. AVIZIENIS: *SOFTWARE PERFORMABILITY: FROM CONCEPTS TO APPLICATIONS*, Kluwer Academic Publishers, Norwell, 1996