Ein Mehrprozessorsystem mit rekonfigurierbarer Verbindungsstruktur für die parallele und verteilte ereignisgesteuerte Simulation

Von der Fakultät Elektrotechnik und Informationstechnik der Universität Stuttgart
zur Erlangung der Würde eines Doktor-Ingenieurs (Dr.-Ing.) genehmigte Abhandlung

vorgelegt von

Wolfram Seibold

geb. in Remshalden-Grunbach

Hauptberichter:	Prof. Dr.-Ing. Dr. h. c. mult. P. J. Kühn
Mitberichter:	Prof. Dr.-Ing. U. G. Baitinger
Tag der mündlichen Prüfung:	14 Juni 2002

Institut für Nachrichtenvermittlung und Datenverarbeitung der Universität Stuttgart

2002

A multiprocessor system with reconfigurable interconnection structure for parallel and distributed event driven simulation

Summary

Simulation is a very important technique for validation and evaluation of technical systems because it can be used in a flexible manner for systems where building a prototype is too expensive, time consuming or risky. Furthermore, rare situations can be studied in an acceptable time.

Simulation is done by building a model from reality and executing this model on a computer. Depending on the modelling of time, various types of simulations can be distinguished. In the case of the so called event driven simulation, the simulation model is only examined at the times of event occuring. In the context of communication networks typical events are packet arrivals, timeouts and start of a packet transmission.

The amount of computing time required for the execution of a simulation depends significantly on the complexity of the simulation model, the abstraction layer and, for statistical simulations, on the confidence level. Due to increasing system complexities, layer spanning simulation models and high confidence level requirements, fast simulation becomes more and more of a challenge.

A general and well-known method to accelerate simulations is the parallel and distributed simulation, where subfunctions and submodels are distributed over a set of computers or a multiprocessor system. Messages are used for communication and synchronization. The synchronization algorithms can be divided into two classes: The conservative and the optimistic algorithms. While using a conservative algorithm, a computing node will wait with the processing of the next local event as long as new local events with a smaller timestamp might be generated by arriving messages from other computing nodes.

The speed up, which can be obtained by using conservative parallel and distributed simulation, depends significantly on short waiting times. It has been shown, that a significant performance gain can be achieved by parallel and distributed simulation for simulation models, where an event will not cause other events on other computing nodes until a limited time has passed. This guaranteed delay time is called “look-ahead“. Another important point for a fast conserv-

ative simulation is, that enough messages are exchanged between the computing nodes to propagate the simulation.

A special simulator architecture was developed in this thesis to speed-up the conservative parallel and distributed simulation. The synchronization of the computing nodes is done by special purpose hardware to reduce the waiting time. Furthermore, the architecture comprises special simulation optimized hardware, a suitable communication protocol, and software libraries to hide the complexity. Focusing on speed-up instead of efficiency is possible due to a computing node architecture which is based on cheap standard chips.

This thesis is divided into seven chapters. Chapter 1 is a short survey on the various methods for specifying and evaluating technical systems. Strengths and weaknesses are presented. The objective of this thesis to speed up simulations by a simulation machine based on parallel computing and special hardware is justified. Chapter 2 contains an overview over the various simulation techniques. The simulation methods are classified and special purpose simulation machines are presented. Furthermore, costs and benefits of parallel and distributed simulations are evaluated and the requirements for a fast simulation are derived. Last but not least, alternative methods to speed-up simulations without using a multiprocessor computer are presented.

Both, requirements and hard- and software structure of the new simulation architecture are presented in chapter 3. This includes the architecture of the computing nodes of the multiprocessor computer, the communication protocol and the algorithms used for fast computing node synchronization. Furthermore, the functions of the control computers are presented.

Chapter 4 contains a short description of a prototypical realization of the simulator architecture presented in chapter 3. This realization was built up to proove the efficiency of the new simulation architecture.

Cost and benefits of the new simulator architecture are evaluated in chapter 5. The performance related issues are based on the prototypical realization described in chapter 4. The simulation models are taken from the context of communication networks. Models of a MAC protocol for Gigabit LANs and various ATM scenarios are presented, including the simulation results. It is shown, that a fast parallel and distributed simulation is possible with the new simulator architecture. Compared to conservative parallel distributed simulations without hardware support a several orders of magnitudes smaller look-ahead is shown to be sufficient. The new simulator can simulate networks with an arbitrary number of active network elements and

with a look-ahead of some meters in less time than the sequential simulator requires for a network with three active elements.

Conclusions and further studies are outlined in chapter 6. Chapter 7 contains a comprehensive reading list. Important terms used in this thesis are explained and defined in appendix A, the simulation parameters are listed in appendix B.

Inhaltsverzeichnis

Abkürzungen

AAL	ATM Adaptation Layer
ABR	Available Bit Rate
ADU	Atomic Data Unit
ALU	Arithmetic Logic Unit
ASIC	Application Specific Integrated Circuit
ATM	Asynchronous Transfer Mode
B-ISDN	Broadband ISDN
CBR	Constant Bit Rate
CCL	Calculus of Communication Systems
CDP	Control Data Packet
CP	Communication Processor
DBR	Deterministic Bit Rate
DESC	Discrete Event Simulation Computer
E/A	Ein-/Ausgabe
ERICA	Explicit Rate Indication for Congestion Avoidance
FIFO	First In First Out
FPGA	Field Programmable Gate Array
FPU	Floating Point Unit
FSM	Finite State Machine
FTP	File Transfer Protocol
GFC	Generic Flow Control
GSM	Global System for Mobile Communications
GVT	Global Virtual Time
GVZZ	Gleichverteilte Zufallszahlen
HAE	Hardware-Abstraktionsebene
HEC	Header Error Control
HSLAN	High Speed Local Area Network
IP	Internet Protocol
ISDN	Integrated Services Digital Network
ITU	International Telecommunication Union
KB	Kommunikationsbibliothek
LAN	Local Area Network
LIFO	Last In First Out
LSZ	Lokale Simulationszeit

MB	Minimumbilder
MIMD	Multiple Instruction Stream, Multiple Data Stream
MKZ	Minimale Kanalzeit
MPEG	Motion Picture Expert Group
NEZ	Nächste Ereigniszeit
NMP	Null Message Packet
NMPG	Null Message Packet Generator
nrt-VBR	Non-Real-Time Variable Bit Rate
PC	Personal Computer
PCI	Peripheral Component Interconnect
PRN	Paralleles Reduktionsnetzwerk
PT	Payload Type
RBC	Roll Back Chip
REL	Räumliche Ereignislokalität
RESTART	Repetitive Simulation Trial After Reaching Thresholds
RM	Resource Management
rt-VBR	Real-Time Variable Bit Rate
SBR	Statistical Bit Rate
SDH	Synchronous Digital Hierarchy
SDL	Specification and Description Language
SDP	Simulation Data Packet
SIMD	Single Instruction Stream, Multiple Data Stream
SK	Simulationsknoten
SKB	Simulationskomponentenbibliothek
SNNS	Schnelle Nullnachrichtenschleife
SONET	Synchronous Optical Network
SP	Simulationsprozessor
SPA	Simulationsprozessorausgang
SPE	Simulationsprozessoreingang
SPW	Simulationsprozessorwarteregister
SSB	Simulationssteuerungsbibliothek
St	Steuerung
StS	Steuerrechnerschnittstelle
SZE	Speicherzugriffseinheit
TCP	Transmission Control Protocol
UBR	Unspecified Bit Rate
VBR	Variable Bit Rate

VCI	Virtual Channel Identifier
VHDL	VHSIC Hardware Description Language
VHSIC	Very High Speed Integrated Circuit
VLT	Verkehrslenkungstabelle
VPI	Virtual Path Identifier
WAN	Wide Area Network
ZA	Zufallszahlenanforderung
ZEL	Zeitliche Ereignislokalität
ZGL	Zugriffslogik
ZÜE	Zeitüberwachungseinheit
ZUSI	Zufallssimulator
ZV	Zufallsvariable
ZVW	Zeitverwaltung
ZZ-CPU	Zufallszahlenmikroprozessor

Formelzeichen

$a_{Ek}(t)$	Anzahl der bis zur Simulationszeit t in einer Modellkomponente erzeugten Ereignisse
$a_{El}(i,t)$	Anzahl der vom logischen Prozeß i bis zur Simulationszeit t durch Vorausschau bearbeiteten Ereignisse
$a_{Eo}(i,t)$	Anzahl der vom logischen Prozeß i bis zur Simulationszeit t verschickten Ereignisse
$a_{Es}(t)$	Anzahl der im gesamten Simualtionsmodell bis zur Simulationszeit t erzeugten Ereignisse
$a_{E}(i,t)$	Anzahl der im logischen Prozeß i bis zur Simulationszeit t erzeugten Ereignisse
E_i	i-tes Ereignis (in einem logischen Prozeß)
$ED(t)$	Ereignisdichte zur Simulationszeit t
$g(E_i)$	Gewichtungsfunktion für die Komplexität von Ereignissen
$gED(t)$	gewichtete Ereignisdichte zur Simulationszeit t
KZ_x	Kanalzeit von Kanal x
$s(n)$	Speedup bei n Prozessoren
$e(n)$	Effizienz bei n Prozessoren
LP_i	Logischer Prozeß i
τ_i	Verzögerungszeit für Kanal i
$t_{0,i}$	lokale Simulationszeit im logischen Prozeß i
$t_{E,i}$	Zeitstempel von Ereignis E im logischen Prozeß i
$t_{N,i}$	Zeitstempel der i-ten Nullnachricht
$REL1(i,t)$	räumliche Ereignislokalität 1
$REL2(i,t)$	räumliche Ereignislokalität 2
$ZEL(i,t)$	zeitliche Ereignislokalität

Kapitel 1

Einführung

Technische Systeme [ISO95] haben einen hohen Stellenwert in der Gesellschaft des 20. und 21. Jahrhunderts, so daß der Entwicklung solcher Systeme wachsende Bedeutung zukommt. Aufgrund der steigenden Komplexität müssen dabei neue Wege beschritten werden, was insbesondere auch für die Bewertung und Validierung komplexer technischer Systeme gilt.

Wegen der engen Verknüpfung von Spezifikation, Validierung/Verifikation und Bewertung technischer Systeme wird im folgenden zunächst ein Überblick über die dafür verwendeten Verfahren und Methoden gegeben (Kapitel 1.1). Nach der Darlegung der Motivation für die vorliegende Arbeit (Kapitel 1.2) folgt in Kapitel 1.3 ein kurzer Überblick.

1.1 Formen der Bewertung komplexer Systeme

Eine sehr präzise Möglichkeit der Spezifikation und Bewertung stellt der Aufbau eines Referenzsystems dar. Von dieser Möglichkeit kann jedoch oft aus Zeit-, Kosten- und Gefahrengründen kein Gebrauch gemacht werden. Daher werden andere Methoden für die Modellierung und Untersuchung der Systeme benötigt. Da es keine allumfassende Methode gibt, haben sich verschiedene Verfahren etabliert, mit deren Hilfe sich unterschiedliche Aspekte beschreiben und bewerten lassen. Die Modellierung setzt eine präzise funktionelle Beschreibung des Systems und seiner Implementierung voraus. Gemeinsam für den Entwurf wie auch für die Modellierung eines Systems ist deshalb dessen Spezifikation.

1.1.1 Spezifikation in der Umgangssprache

Am Anfang einer Systementwicklung steht meistens eine mehr oder weniger strukturierte umgangssprachliche Beschreibung. Aufgrund der in der Umgangssprache enthaltenen Redundanzen und Mehrdeutigkeiten sind umgangssprachliche Beschreibungen komplexer Systeme sehr umfangreich, so daß die Wahrung der Konsistenz ein erhebliches Problem darstellt.

Eine wichtige Anforderung an Spezifikationen ist ihre Eindeutigkeit, d.h. sie dürfen insbesondere keinen Spielraum für Interpretationen bieten. Da es gegenwärtig nur eingeschränkte Möglichkeiten gibt, umgangssprachliche Beschreibungen maschinell auf Eindeutigkeit zu prüfen, stellt diese Forderung einen hohen Anspruch an die Ersteller der Spezifikation. Die maschinelle Unzugänglichkeit der umgangssprachlichen Spezifikation hat auch zur Folge, daß eine automatische Übersetzung in eine Implementierung durch Hard- oder Software unmöglich ist. Erfolgt diese manuell, kann es zu teilweise nur schwer erkennbaren Abweichungen zwischen Spezifikation und Implementierung kommen.

Zur Illustration komplexer Sachverhalte werden häufig auch bildhafte Darstellungen verwendet. Wegen fehlender Formalismen und mangelnder Eigenständigkeit werden diese jedoch ebenfalls den nichtformalen Methoden zugerechnet.

1.1.2 Formale Methoden der Spezifikation

Für eine Systemspezifikation ist es vorteilhaft, wenn sie sowohl von menschlichen Nutzer als auch von Maschinen verstanden bzw. bearbeitet werden kann. Dazu bedarf es spezieller Notationen, wofür sich verschiedene formale Spezifikationssprachen etabliert haben. Bild 1-1 zeigt eine Übersicht über die gebräuchlichsten Sprachen nach [Küh95].

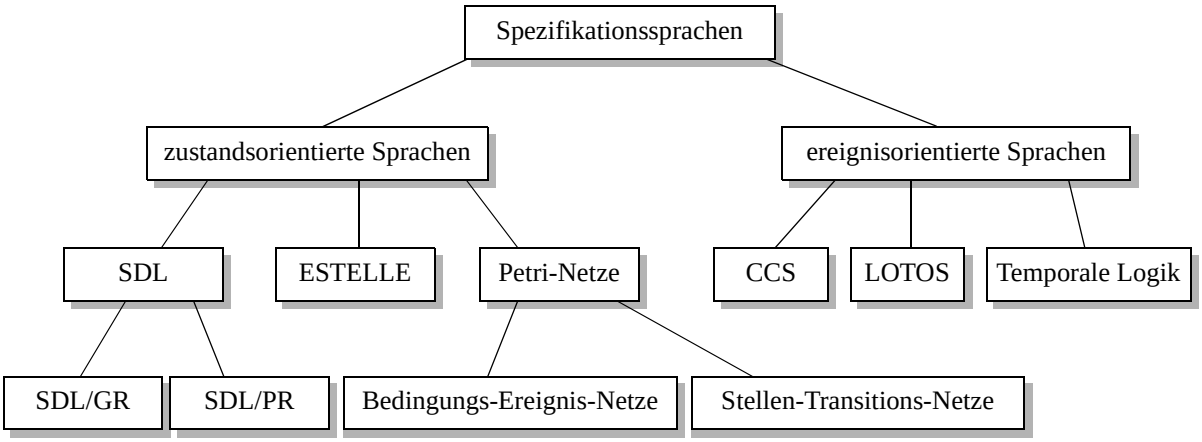


Bild 1-1: Klassifikation der Spezifikationssprachen

Auf die formale Spezifikation können Werkzeuge angewendet werden, welche die syntaktische und teilweise auch die semantische Korrektheit überprüfen. Mit Hilfe von Algorithmen der formalen Verifikation läßt sich eine Erreichbarkeitsanalyse durchführen und der Nachweis erbringen, daß Zusicherungen[1] eingehalten werden (Assertion Proofs). Ergänzend dazu kann eine Validierung der spezifizierten Vorgänge durch eine automatisch generierte Simulation

erfolgen, die aufgrund der automatischen Generierung, unter der Voraussetzung der Verwendung korrekter Werkzeuge, korrekt ist, d.h. mit der Spezifikation übereinstimmt.[2]

Ist eine formale Spezifikation vollständig, eindeutig und korrekt, so kann nach Hinzufügen von Realisierungshinweisen (z.B. Technologie, Programmiersprache, zeitliche Randbedingungen) daraus im Prinzip automatisch eine mögliche Realisierung gewonnen werden, die unter der Voraussetzung der Korrektheit der verwendeten Werkzeuge ebenfalls korrekt ist. Auf gleiche Weise kann ein Werkzeug für den Test auf Übereinstimmung von Spezifikation und Realisierung generiert werden. Eine Leistungsbewertung eines technischen Systems läßt sich nicht direkt aus der Spezifikation ableiten, da hierfür im allgemeinen wichtige Angaben fehlen. Sie läßt sich daher in der Regel nur über den problematischen Umweg der automatischen Generierung einer Realisierungsbeschreibung erstellen.

Trotz der zahlreichen Vorteile bei der Verwendung formaler Beschreibungsmethoden haben diese bisher nur relativ selten Eingang in den Verifikationsprozeß und die Leistungsbewertung gefunden [Dud93]. Dies liegt insbesondere daran, daß die meisten Probleme analytisch nicht zugänglich sind und daß Modelle realer Probleme im allgemeinen sehr viele Zustände und Zustandsübergänge aufweisen. Da für die formale Verifikation in der Regel der volle Zustandsraum aufgespannt werden muß, sind bereits Systeme mäßigen Umfangs mit heutigen Hard- und Software-Werkzeugen aufgrund von Speicherknappheit und mangelnder Rechenkapazität nicht mehr zu untersuchen. Für kleinere und ausgewählte Systeme stellen sie aber durchaus eine sinnvolle Möglichkeit dar, wie z.B. in [Rößß95] erfolgreich gezeigt wurde.

1.1.3 Analyse mit Hilfe von verkehrstheoretischen Methoden

Durch die Anwendung formaler Beschreibungssprachen und Methoden kann ein Korrektheitsnachweis für ein System erbracht werden, Aussagen über die Leistungsfähigkeit, die einen wesentlichen Beitrag zur Verwendbarkeit eines Systems liefert, lassen sich daraus aber normalerweise nicht ableiten.

Auf der Basis mathematischer Betrachtungen geben verkehrstheoretische Methoden Aufschluß über die Leistungsfähigkeit eines Systems. Sie stellen somit eine Ergänzung zu den formalen Beschreibungssprachen und -methoden dar. Für die verkehrstheoretische Untersuchung

1. Zusicherungen geben z.B. für Systemvariablen Wertebereiche und Konsistenzbedingungen an.
2. Correctness by construction.

eines Systems muß zunächst ein verkehrstheoretisches Modell erstellt werden, das die wesentlichen, d.h. die zu untersuchenden Abläufe nachbildet (Bild 1-2).

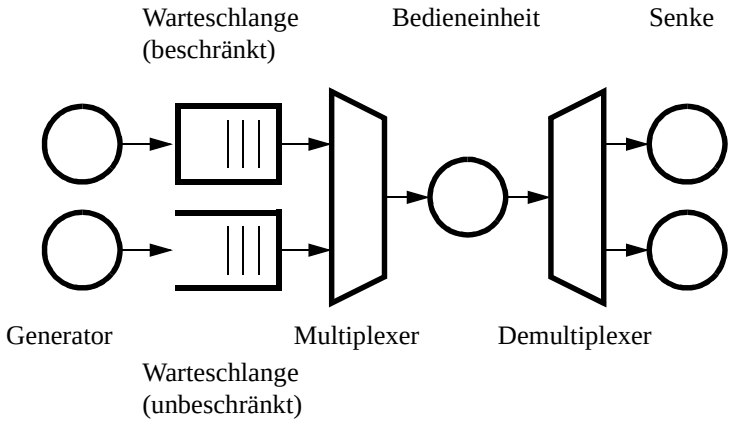


Bild 1-2: Beispiel für ein verkehrstheoretisches Modell

Der Nachrichtenfluß im Modell beginnt bei einem Generator, der zeitabhängig Nachrichten erzeugt. Dieser als „Verkehr“ bezeichnete Nachrichtenfluß kann beliebigen Verteilungsfunktionen genügen. Warteschlangen dienen der Zwischenspeicherung von Nachrichten. Gängige Strategien für die Nachrichtenspeicherung sind das Auslesen in Einschreibreihenfolge (FIFO), in umgekehrter Einschreibreihenfolge (LIFO) oder nach Prioritäten. Das Fassungsvermögen einer Warteschlange kann im Modell beschränkt oder unbeschränkt sein. Das Zusammenfassen bzw. Aufspalten von Verkehrsströmen wird mit Hilfe von Multiplexern bzw. Demultiplexern modelliert. Sie können beliebige Bedienstrategien aufweisen und damit das Simulationsmodell wesentlich charakterisieren. Zeitliche Abhängigkeiten werden durch Bedieneinheiten modelliert, die eine feste oder eine variable Bedienzeit haben. Eine variable Bedienzeit kann durch interne Zustände, durch Zufallsprozesse oder durch eine Kombination von internen Zuständen und Zufallsprozessen bestimmt werden.[3] Das Ende eines Verkehrsstroms wird stets durch eine Senke modelliert. Sie nimmt die Nachrichten auf und entfernt sie aus dem System.

Unter Verwendung der Grundelemente können Modelle für beliebig komplexe Systeme erstellt werden, doch der verkehrstheoretischen Analysierbarkeit sind, abhängig von den gesuchten Bewertungsgrößen, den Strategien für Bedieneinheiten, Multiplexer und Demultiplexer sowie den erzeugten Verkehrsströmen, relativ enge Grenzen gesetzt [AB82]. Dies gilt insbesondere dann, wenn als Bewertungsgrößen nicht nur Mittelwerte und Schranken sondern auch Vertei-

3. Ein Beispiel hierfür stellt ein Modell einer MPEG-II Quelle dar, bei welcher sich die Rahmenfolge aus dem Zustand der Quelle und ihrer Parameter ergibt, die Größe der einzelnen Rahmen aber zufallsabhängig ist [Ens98].

lungsfunktionen gefragt sind. Daher können für viele Systeme nur stark vereinfachte Modelle analysiert werden. Erschwerend kommt hinzu, daß in der Regel kein Nachweis für die Sinnhaftigkeit der Modellierung erbracht werden kann, so daß nicht sichergestellt werden kann, daß sich alle wesentlichen Aspekte des realen Systems im Modell widerspiegeln. Diese Probleme führen dazu, daß verkehrstheoretische Analysen sinnvollerweise ergänzend zu anderen Bewertungsmethoden eingesetzt werden. Es bietet sich an, Analysen durch Simulation zu validieren und umgekehrt.

Ausführliche Darstellungen der Verkehrstheorie, ihrer Möglichkeiten und Grenzen sowie Anwendungsbeispiele finden sich in [Kle75, GH81].

1.1.4 Analyse mit Hilfe der Simulation

Nach [Dud93] bezeichnet Simulation „die Nachbildung von Vorgängen auf einer Rechenanlage auf der Basis von Modellen. Sie wird meist zur Untersuchung von Abläufen eingesetzt, die man in der Wirklichkeit aus Zeit-, Kosten-, Gefahren- oder anderen Gründen nicht durchführen kann. (...) Jede Simulation beginnt mit der Entwicklung eines Simulationsmodells, das die wesentlichen Eigenschaften der zu simulierenden Vorgänge und ihre gegenseitige Beeinflussung widerspiegelt.“ Die Ergebnisse der Simulation gelten, wie bei der verkehrstheoretischen Analyse, zunächst nur für das Modell. Inwieweit sie auf die Wirklichkeit übertragen werden können, hängt entscheidend von der Qualität des Modells ab [LK91]. Ein Beispiel hierfür befindet sich in [Sei92].

Abhängig von der Zielsetzung werden zwei Arten der Simulation unterschieden. Bei der *funktionalen Simulation* geht es im wesentlichen darum, ein Modell und das zugrundeliegende technische System zu validieren oder zu veranschaulichen. Dies geschieht dadurch, daß die Reaktion des Modells auf unterschiedliche äußere Einflüsse demonstriert wird. Im Gegensatz dazu liegt bei der *statistischen Simulation* das Augenmerk auf der Leistungsbewertung, so daß sie sich für die Validierung von verkehrstheoretischen Analysen anbietet. Das Simulationsmodell wird hierfür meistens durch Pseudozufallszahlengeneratoren stimuliert. Während des Simulationslaufs werden dann die zu untersuchenden Größen protokolliert und am Ende dadurch statistisch ausgewertet, daß Mittelwerte, Streuungen, Korrelationen usw. gebildet werden. Damit präzisere Aussagen über die Zuverlässigkeit der gewonnenen Ergebnisse getroffen werden können, wird bei der statistischen Simulation ein Simulationslauf üblicherweise in

mehrere Teilläufe untergliedert, deren Ergebnisse wiederum statistisch ausgewertet werden können und die Angabe von Vertrauensintervallen ermöglichen [LK91].

Der Unterschied zwischen den beiden Simulationsarten läßt sich leicht am Beispiel eines Kommunikationsmoduls veranschaulichen. Bei der funktionalen Simulation wird geprüft, ob die Schnittstelle des Moduls korrekt arbeitet, ob das System die Nachrichten korrekt vermittelt und wie es auf Fehler, z.B. durch ungültige Nachrichten, reagiert. Mit Hilfe der statistischen Simulation wird untersucht, wie lange eine Nachricht im Mittel im System ist, ob Ober- und Untergrenzen dafür existieren, mit welcher Wahrscheinlichkeit bei gegebenem Verkehrsaufkommen Nachrichten verloren gehen usw.

Den Nachweis zu erbringen, daß ein Simulationsmodell die Eigenschaften der Wirklichkeit widerspiegelt, ist wie bei der verkehrstheoretischen Analyse im allgemeinen nicht möglich. Auch die automatische Erzeugung von Simulationsmodellen aus formalen Spezifikationen ist Gegenstand der Forschung. Hierbei stellt sich insbesondere das Problem, daß das Abstrahieren ein intelligenzerfordernder, kreativer Vorgang ist. Für spezielle Probleme, wie z.B. die automatische Simulationsprogrammgenerierung aus formalen Spezifikationen für MAC-Protokoll (Media Access Control-Protokoll), konnten jedoch schon Lösungen präsentiert werden [VT93].

1.1.5 Emulation

Unter einer Emulation versteht man die Nachbildung eines Systems unter Verwendung eines anderen Systems, wobei das Augenmerk normalerweise auf der Erbringung der Funktion bei maximaler Systemleistung liegt. Dies bedeutet, daß bei der Emulation das nachzubildende System nur bezüglich seiner Funktion nachgebildet wird. Das zeitliche Verhalten wird nur dann berücksichtigt, wenn es für die Erbringung der Funktionalität unerläßlich ist[4]. Die Emulation erbringt daher im Gegensatz zur Simulation die Funktion des nachzubildenden Systems und kann dieses vollständig ersetzen. Normalerweise erlaubt die Emulation aber keine Aussage über die Leistungsfähigkeit des nachgebildeten Systems.

Bei der funktionalen Simulation steht im Gegensatz dazu nicht die Erbringung der Funktionalität sondern das korrekte Verhalten des Systems im Vordergrund. Im Zusammenhang mit Kommunikationsnetzen bedeutet dies z.B., daß bei der funktionalen Simulation nicht unbe-

4. Z.B. Visualisierungen.

dingt eine Übertragung der Nutzdaten erfolgen muß. Dagegen ist bei der Emulation die Nutzdatenübertragung für die Erbringung der Funktionalität unabdingbar.

Emulation wird häufig bei Rechnersystemen angewendet. Entsprechende Emulatoren erlauben die Ausführung von Programmen, die für andere Rechnersysteme geschrieben wurden. Da bei den meisten Programmen die Ausführungsgeschwindigkeit von untergeordneter Bedeutung ist, kann sie bei der Emulation je nach Emulator und Rechnersystem höher oder niedriger sein.

1.2 Motivation und Zielsetzung der Arbeit

Das Ziel einer guten Qualitätssicherung ist es, bei der Entwicklung und Fertigung von Produkten die Entstehung von Fehlern zu vermeiden bzw. sie möglichst früh zu erkennen und zu beseitigen [Sch94]. Unter „Fehlern“ versteht man dabei ganz allgemein Abweichungen des Produkts von der Spezifikation. Um dies zu gewährleisten, werden die in Kapitel 1.1 beschriebenen Methoden der Spezifikation und Bewertung angewendet.

Fortschritte bei der Realisierung technischer Systeme erlauben immer komplexere und schnellere Systeme. Hinzu kommt, daß aufgrund der steigenden Zuverlässigkeit technische Systeme vermehrt Anwendung in sicherheitskritischen Bereichen finden, in denen Fehlfunktionen nicht oder nur sehr selten auftreten dürfen. Daher müssen auch die Methoden für die Systementwicklung weitergeführt und ergänzt werden, so daß eine Validierung und Leistungsbewertung trotz steigender Komplexität und Geschwindigkeit auch unter Berücksichtigung seltener Ereignisse hinreichend genau möglich ist.

Der Einsatz formaler Methoden und Analyseverfahren reicht aufgrund ihrer Einschränkungen (siehe Kapitel 1.1) nicht aus und bedarf der Ergänzung durch Simulation. Doch auch die herkömmliche Simulation von komplexen Systemen scheitert häufig an dem dafür notwendigen Rechenaufwand, an der verfügbaren Speicherkapazität bzw. an der geforderten statistischen Sicherheit. Um solche Systeme simulationstechnisch zu erschließen, müssen neue Wege beschritten werden. Neben einigen Sonderformen (vgl. Kapitel 2.5) liegt in der Parallelisierung und Verteilung des Simulationsmodells ein erhebliches Potential. Abhängig von den Eigenschaften der Simulationsmodelle sind für eine effiziente parallele und verteilte Simulation unterschiedliche Kommunikations- und Synchronisationsverfahren sinnvoll (siehe Kapitel 2.2 und 2.4).

Es sind verschiedene Ansätze bekannt, allgemeine Simulatoren zu entwickeln, die unterschiedliche Simulationsverfahren enthalten (z.B. [Ne98, Zbo95, tB95, MM93, DSYB90]). Da der Anwender sehr einfach zwischen den Verfahren wechseln kann, eignen sich diese Bibliotheken zur vergleichenden Simulation mit unterschiedlichen Simulationsverfahren und damit auch zur Untersuchung derselben. Allerdings hat sich herausgestellt, daß die Allgemeingültigkeit der Bibliotheken zu Lasten ihrer Leistungsfähigkeit geht. Daher wird nur sehr selten über ihren erfolgreichen Einsatz bei praxisrelevanten Problemen berichtet. Erfolge werden stattdessen dann erzielt, wenn das Simulationsverfahren an das jeweilige Simulationsmodell angepaßt wird.

Das Ziel dieser Arbeit ist die beschleunigte Simulation einer bestimmten Klasse von praxisrelevanten Simulationsmodellen aus dem Bereich der Kommunikationsnetze durch Verwendung eines parallel und verteilt arbeitenden Simulators. Es handelt sich dabei um Modelle, die bei der Simulation besonders hohe Anforderungen bezüglich der Anzahl der zu bearbeitenden Ereignisse stellen und bei deren Parallelisierung sich nur eingeschränkte Ereignislokalitäten erzielen lassen (vgl. Kapitel 2.4.4). Durch eine simulationsoptimierte Hardware wird es möglich, simulationstechnisch in neue Anwendungsgebiete und statistisch anders unzugängliche Bereiche vorzudringen. Für die Klassifikation sind zunächst die für die verteilte Simulation relevanten Eigenschaften von Simulationsmodellen aufzuzeigen.

Die Simulationsbeschleunigung soll im Rahmen dieser Arbeit dadurch erreicht werden, daß ein Multiprozessorrechner entwickelt wird, der die inhärente Parallelität des Simulationsmodells ausnutzt. Ein rekonfigurierbares Verbindungsnetz soll eine Abbildung der logischen Topologie des zu simulierenden Systems auf die des Multiprozessorrechners ermöglichen. Bild 1-3 zeigt hierfür ein Beispiel.[5]

Der Multiprozessorrechner soll ferner durch spezielle, simulationsspezifische Logik und ein angepaßtes Kommunikationsprotokoll, welches eine geeignete Bandbreite und kurze Latenzzeiten besitzt, optimiert werden. Ebenso ist es Ziel dieser Arbeit, die Handhabbarkeit eines solchen parallelen Simulationssystems zu untersuchen.

Da die Leistungsfähigkeit des zu entwickelnden Simulationssystems nicht aus der Spezifikation abgeleitet werden kann, gehört es zur Zielsetzung dieser Arbeit, diese durch prototypische

5. Die Shuffle-Netz-Topologie enthält eine ganze Anzahl weiterer Topologien als Untermenge. Für die Abbildung einer Ringstruktur mit n Knoten genügt auch ein Shuffle-Netz mit n Knoten [KSGL95].

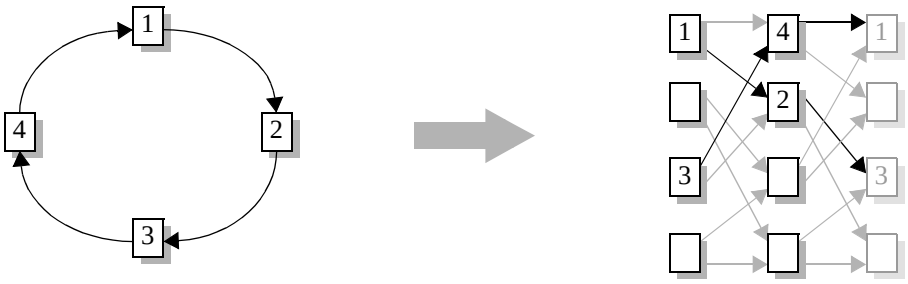


Bild 1-3: Beispiel für die Abbildung einer logischen Ringtopologie mit 4 Knoten auf eine physikalische Shuffle-Netz-Topologie mit 8 Knoten

Realisierung und Anwendung dieser Realisierung auf praxisrelevante Fragestellungen unter Beweis zu stellen.

1.3 Überblick über die Arbeit

Im Kapitel 2 erfolgt zunächst eine Einführung in die Simulationstechnik. Neben einer Klassifikation der Simulationsverfahren werden verschiedene Rechnerarchitekturen für die parallele und verteilte Simulation vorgestellt und ihre Chancen und Grenzen erläutert. Ferner werden alternative Verfahren und Methoden zur Simulationsbeschleunigung beschrieben. Im Kapitel 3 wird nach der Diskussion der Anforderungen an eine geeignete Simulatorarchitektur eine neue Architektur vorgestellt und begründet. Da eine Leistungsbewertung der Simulatorachitektur nur unter Berücksichtigung von Realisierungsaspekten möglich ist, erfolgt in Kapitel 4 die Beschreibung einer prototypischen Realisierung. Sie bildet die Grundlage für die Leistungsbewertung in Kapitel 5. Im Kapitel 6 folgt eine Zusammenfassung der Ergebnisse dieser Arbeit und ein Ausblick auf weitere Forschungsvorhaben. Eine Zusammenstellung und Erläuterung der wichtigsten in dieser Arbeit verwendeten Begriffe befindet sich im Anhang.

Kapitel 2

Simulation

Abhängig von den zu simulierenden Systemen und den zu untersuchenden Eigenschaften werden unterschiedliche Simulationsmodelle benötigt (siehe Kapitel 1.1.4). Aufgrund ihrer Verschiedenheit sind für die Bearbeitung der Modelle mit einem Rechnersystem jeweils geeignete Simulationsverfahren erforderlich. In diesem Kapitel wird durch Klassifikation ein Überblick über die gängigsten Verfahren gegeben.

2.1 Klassifikation nach der Simulationszeitsteuerung

In [Bai90] werden die Simulationsverfahren in die Klassen digitale und analoge Simulation eingeteilt (Bild 2-1). Bei der digitalen Simulation, die auch als zeitdiskrete Simulation bezeichnet wird, werden nur die Zustände betrachtet, die das zu simulierende System einnimmt. Im Gegensatz dazu ist bei der analogen (zeitkontinuierlichen) Simulation der kontinuierliche Verlauf von Bedeutung. Daraus folgt, daß bei der analogen Simulation das Fortschreiten der Simulationszeit über der realen Zeit (zumindest näherungsweise) eine stetige Funktion darstellt und somit das Simulationsmodell zeitkontinuierlich betrachtet werden muß. Demgegenüber wird das Simulationsmodell bei der digitalen Simulation nur zu diskreten Zeitpunkten betrachtet. Abhängig davon, wie diese Zeitpunkte ausgewählt werden, wird die Klasse der digitalen Simulationsverfahren unterteilt in zeit- und in ereignisgesteuerte Simulationsverfahren.

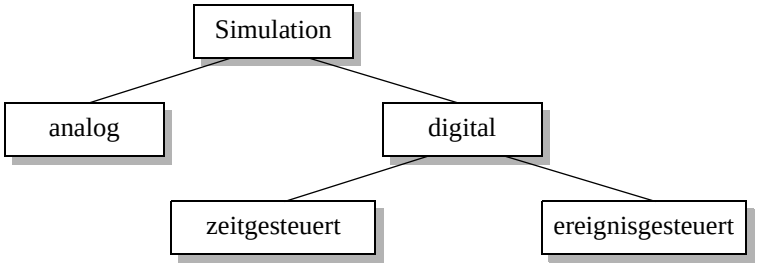


Bild 2-1: Einteilung der Simulationsverfahren nach der Simulationszeitsteuerung (Übersicht)

Die analoge Simulation eignet sich, wie der Name schon sagt, für analoge Vorgänge wie sie z.B. bei Wettervorhersagen, Strömungsuntersuchungen, mechanischen Systemen oder bei der analogen Signalverarbeitung betrachtet werden. Simulationsverfahren aus der Klasse der digitalen Simulation werden für Systeme verwendet, die sich durch endliche Automaten (Finite State Machine, FSM) beschreiben lassen.

2.1.1 Grundprinzip zeitgesteuerter Simulation

Bei zeitgesteuerten Simulationsverfahren wird das Simulationsmodell zu äquidistanten Zeitpunkten betrachtet. Zu jedem dieser Zeitpunkte wird geprüft, ob sich die äußeren Bedingungen (Eingangsgrößen) geändert haben und ob ein neuer Zustand eingenommen werden muß. Ereignisse, die im realen System zeitlich gesehen zwischen diesen Zeitpunkten auftreten, werden erst zum nächsten Betrachtungszeitpunkt bearbeitet (Bild 2-2).

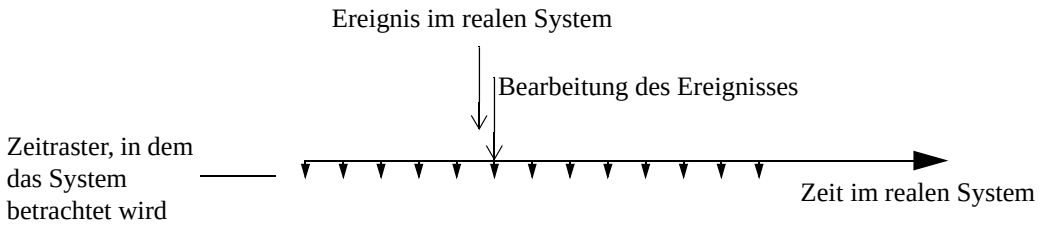


Bild 2-2: Zeitgesteuerte Simulation

Zeitgesteuerte Simulationsverfahren eignen sich besonders für die Simulation von getakteten Systemen, bei denen aufgrund des Taktes die Zustandsübergänge ohnehin nur in einem festen Zeitraster möglich sind. Sie sind besonders dann effizient, wenn zu den meisten Betrachtungszeitpunkten Zustandsänderungen auftreten. Ist dies nicht der Fall, wird das System unnötig oft betrachtet und das Simulationsverfahren damit ineffizient. Für solche Systeme ist ein ereignisgesteuertes Simulationsverfahren besser geeignet.

2.1.2 Grundprinzip ereignisgesteuerter Simulation

Die ereignisgesteuerten Simulationsverfahren sind dadurch gekennzeichnet, daß das zu simulierende Modell genau zu den Zeitpunkten betrachtet wird, an welchen Ereignisse auftreten. Wird zum Beispiel ein System simuliert, von welchem ein verkehrstheoretisches Modell entsprechend Kapitel 1.1.3 erstellt wurde, so sind typische Ereignisse die Erzeugung einer Nachricht durch einen Generator und das Bedienende in einer Bedieneinheit.

Damit die Ereignisse in chronologisch aufsteigender Reihenfolge bearbeitet werden können, wird ein Kalender benötigt, in den die Eintrittszeitpunkte (Zeitstempel) der Ereignisse eingetragen werden. Der Simulationsvorgang besteht dann darin, das Ereignis mit der kleinsten Zeitmarke aus dem Kalender zu entfernen und zu bearbeiten. Dadurch wird im allgemeinen der Systemzustand verändert und es werden neue Ereignisse generiert, die wiederum in den Kalender eingetragen werden. Dieser Vorgang wird so lange wiederholt, bis keine weiteren Ereignisse mehr vorliegen oder bis ein Abbruchkriterium erfüllt ist. Letzteres kann z.B. die Anzahl der bearbeiteten Ereignisse, das Überschreiten einer bestimmten Simulationszeit oder das Eintreten eines besonderen Ereignisses sein. Bild 2-3 zeigt das Grundprinzip ereignisgesteuerter Simulationsverfahren.

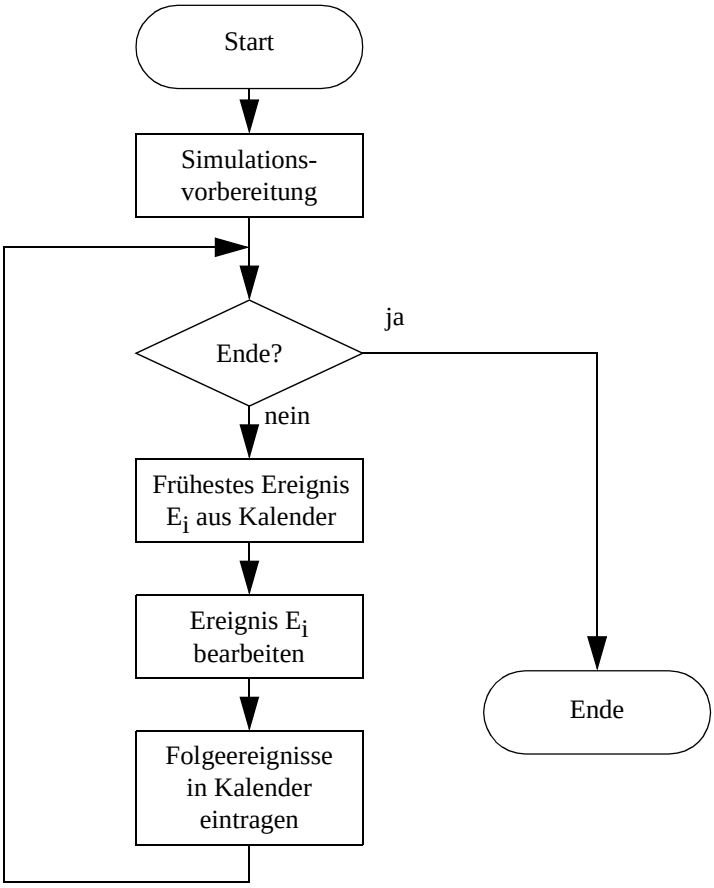


Bild 2-3: Grundprinzip ereignisgesteuerter Simulation

Da reale Systeme kausal sind, kann es in der Simulation nicht vorkommen, daß durch das Bearbeiten von Ereignissen andere Ereignisse mit kleinerer Zeitmarke erzeugt werden. Dadurch ist auch die Kausalität der ereignisgesteuerten Simulation garantiert.

Die ereignisgesteuerte Simulation besitzt gegenüber der zeitgesteuerten Simulation den Vorteil, daß die Simulationszeit immer direkt auf den Zeitpunkt des nächsten Ereignisses gesetzt werden kann. Diese Eigenschaft kommt insbesondere dann zum Tragen, wenn in dem zu untersuchenden System längere Phasen ohne zu bearbeitende Ereignisse auftreten. Während diese Phasen bei der zeitgesteuerten Simulation im Zeitraster durchlaufen werden, kann bei der ereignisgesteuerten Simulation die Simulationszeit direkt auf das Ende dieser Ruhephase gesetzt werden.

2.2 Klassifikation ereignisgesteuerter Simulationsverfahren nach der Ereignisbearbeitungsreihenfolge

Das Grundprinzip der ereignisgesteuerten Simulationsverfahren schreibt vor, daß immer das Ereignis mit der kleinsten Zeitmarke bearbeitet wird. Zur Optimierung einer Simulation bezüglich der Simulationsgeschwindigkeit kann von diesem Prinzip abgewichen werden. Im folgenden werden verschiedene Ereignisbearbeitungsreihenfolgen vorgestellt und ihre Vor- bzw. Nachteile kurz erörtert. Nicht alle der vorgestellten Bearbeitungsreihenfolgen schließen sich gegenseitig aus, sie können teilweise auch miteinander kombiniert werden.

2.2.1 Vorwärts versus rückwärts gerichtete Simulation

Bei Anwendung des Grundprinzips der ereignisgesteuerten Simulation wird die Simulationszeit zu Beginn auf Null gesetzt. Anschließend steigt sie monoton entsprechend den Zeitpunkten der bearbeiteten Ereignisse. Diese Vorgehensweise entspricht dem normalen Betrieb eines technischen Systems nach dem Einschalten und wird als vorwärtsgerichtete Simulation bezeichnet.

Ist man bei einer Simulation nur an den Zuständen bestimmter Modellkomponenten in einem bestimmten Zeitintervall interessiert (z.B. um auf kritische Zustände zu prüfen), dann kann dafür die rückwärts gerichtete (bedarfsgesteuerte) Simulation geeignet sein. Im Gegensatz zur vorwärts gerichteten Simulation werden bei diesem Verfahren eine Systemkomponente (bzw. deren Ausgangssignale) und ein Zeitintervall angegeben. Ausgehend von diesen Angaben wird geprüft, welche Größen die betrachtete Modellkomponente beeinflussen und welchen Wert diese im betrachteten Zeitintervall besitzen. Dazu werden die entsprechenden Werte von den erzeugenden Modellkomponenten angefordert, welche ggf. wiederum Anfragen an andere Modellkomponenten stellen. Dies wird so lange fortgesetzt, bis alle Größen eindeutig

bestimmt sind. Der Vorteil dieses Verfahrens ist, daß ausschließlich die Größen berechnet werden, die für das Simulationsergebnis benötigt werden.

Das Verfahren, das in [SMB87] vorgestellt wird, eignet sich nur für ganz bestimmte Anwendungen. Für die parallele Logiksimulation ist es in [Pfe92] ausgeführt, für ein Kommunikationssystem in [FBKA89].

2.2.2 Formen der Parallelisierung und Verteilung

Der in Kapitel 2.1.2 vorgestellte prinzipielle Ablauf einer ereignisgesteuerten Simulation stellt einen Algorithmus dar, der sequentiell von einer Recheneinheit zu bearbeiten ist. Bei der Simulation großer und komplexer technischer Systeme bringt die sequentielle Ausführung lange Simulationslaufzeiten mit sich. Eine vielversprechende Lösung dieses Problems stellt die Parallelverarbeitung dar. Sie kann auf unterschiedliche Weise eingesetzt werden.

2.2.2.1 Parallelisierung auf Parameterebene

Wenn bei einer statistischen Simulation mehrere Simulationsläufe mit unterschiedlichen Parametersätzen benötigt werden, erhält man die einfachste Form der Parallelisierung durch das Verteilen der Simulationsläufe auf die vorhandenen Recheneinheiten. In diesem Fall ist keinerlei Synchronisation und Kommunikation zwischen den Recheneinheiten notwendig. Ähnliches gilt, wenn die bei statistischer Simulation üblichen Teiltests auf mehrere Recheneinheiten verteilt werden.[1] Damit ergibt sich unter der Voraussetzung, daß alle Teilläufe gleiche Komplexität besitzen, eine relative Geschwindigkeitssteigerung (Simulationszeitgewinn, Speedup) *s(n)* entsprechend Gleichung 2-1 und eine Effizienz *e(n)* für die Nutzung der Recheneinheiten nach Gleichung 2-2.

$$s(n) = \frac{m}{\left\lceil \frac{m}{n} \right\rceil} \qquad \begin{array}{l} m\text{=Anzahl Läufe bzw. Teiltests} \\ n\text{=Anzahl verfügbarer Recheneinheiten} \end{array} \tag{2-1}$$

$$e(n) = \frac{s(n)}{n} = \frac{\frac{m}{n}}{\left\lceil \frac{m}{n} \right\rceil} \tag{2-2}$$

1. Für die Erstellung einer gemeinsamen Statistik ist eine Synchronisation am Simulationsende notwendig.

Diese Art der Parallelisierung ist sehr einfach und sehr effizient, da während der Simulation weder Prozeßsynchronisation noch Kommunikation notwendig sind. Sie wurde zum Beispiel in [BDDea96] erfolgreich angewendet. Den genannten Vorteilen stehen auch Nachteile gegenüber. Aus statistischen Gründen sind mehr als 20 Teiltests in der Regel nicht sinnvoll, weshalb auf diese Weise kein beliebig hoher Parallelisierungsgrad erreicht werden kann. Ferner sind die ersten Teilergebnisse nicht schneller verfügbar als bei sequentieller Simulation, da die Bearbeitungszeit für einen einzelnen Teillauf bei dieser Form der Parallelisierung nicht beeinflußt wird.

2.2.2.2 Funktionale Parallelisierung

Ein Simulationsprogramm, das nach dem Prinzip der ereignisgesteuerten Simulation arbeitet, enthält insbesondere bei statistischer Simulation neben dem Modul des eigentlichen Simulationsalgorithmus weitere Module mit Hilfsfunktionen. Solche Hilfsfunktionen können z.B. Pseudozufallszahlenerzeugung, Speicherverwaltung, Statistikerzeugung und bei verteilter Simulation auch Kommunikation und Synchronisation sein. Diese Hilfsfunktionen können teilweise weitgehend unabhängig von Ereignisbearbeitung und Kalender durchgeführt und somit auch leicht auf andere Recheneinheiten verlagert werden. Gibt es in einem Simulationsmodell nur wenige Module, die sich dafür eignen, kann für diese Modelle durch funktionale Parallelisierung nur eine stark begrenzte Geschwindigkeitssteigerung erzielt werden.

Wie eine funktionale Parallelisierung aussehen kann, soll am Beispiel eines Pseudozufallszahlenmoduls veranschaulicht werden. Bei der statistischen Simulation von verkehrstheoretischen Modellen werden für die Generatoren und Bedieneinheiten Zufallszahlen benötigt, welche z.B. Ankunftsabstand, Paketlänge und Bediendauer festlegen. Da die Berechnung guter Pseudozufallszahlen sehr aufwendig ist [L'E88, Wol99], kann es sinnvoll sein, diese getrennt nebenläufig zu berechnen und in einem FIFO-Speicher abzulegen (vgl. Kapitel 3.3.2). Das Simulationsmodul hat damit schnellen Zugriff auf Pseudozufallszahlen.

In [Leh79a], [SCM85] und [Lem89] wird jeweils eine Simulationsumgebung beschrieben, die von der funktionalen Parallelisierung Gebrauch macht. Dort gibt es z.B. unabhängige Module für die Erzeugung von Pseudozufallszahlen und für statistische Auswertungen. Auch die in dieser Arbeit entwickelte Simulatorarchitektur (Kapitel 3) verwendet u.a. diese Form der Parallelisierung.

2.2.2.3 Parallelisierung durch Modellaufteilung

Ein sehr hoher Parallelisierungsgrad kann erreicht werden, wenn das Simulationsmodell selbst aufgeteilt wird. Die einzelnen Teilmodelle werden auf die vorhandenen Recheneinheiten verteilt und können so gleichzeitig, aber nicht notwendigerweise unabhängig voneinander, bearbeitet werden. Dadurch kann die systemimmanente Parallelität genutzt werden. Hierin liegt insbesondere bei der Simulation umfangreicher und komplexer Systeme, die sich auf viele Bearbeitungseinheiten verteilen lassen, ein großes Potential.

Wird für die verteilten Teilmodelle ein gemeinsamer zentraler Kalender verwendet, so wird durch ihn auf einfache Weise die Ausführungsreihenfolge der Ereignisse eindeutig festgelegt [GPM81]. Andererseits stellt der zentrale Kalender, wenn viele Bearbeitungseinheiten zur Verfügung stehen, einen Engpaß dar. Simulationsverfahren mit zentralem Kalender arbeiten daher in der Regel transaktionsbasiert, d.h. eine zentrale Einheit identifiziert voneinander unabhängige Ereignisse bzw. Ereignissequenzen und gruppiert diese zu Transaktionen. Eine Transaktion wird dann auf einer beliebigen Recheneinheit zur Ausführung gebracht, wenn alle vorher auszuführenden Transaktionen abgeschlossen sind. Der transaktionsorientierte Ansatz eignet sich nur für Probleme, bei welchen sich die Transaktionen leicht finden lassen und der Kommunikationsaufwand im Verhältnis zum Aufwand für die Ausführung der Transaktion gering ist. In [CEGL82] wird ein Simulationssystem vorgestellt, das transaktionsorientiert arbeitet.

Die meisten Verfahren für parallele Simulation basieren auf verteilten Kalendern, d.h. für jedes unabhängige Teilmodell wird ein eigener Kalender geführt. Ein solches Teilmodell mit eigener Simulationszeitsteuerung wird als logischer Prozeß (LP) bezeichnet. Für die Synchronisation der Kalender wurden spezielle Verfahren entwickelt, die im Kapitel 2.2.3 vorgestellt werden.

Die Aufteilung des Gesamtmodells in logische Prozesse stellt bei Simulationsverfahren mit verteiltem Kalender eine feste Dekomposition dar. Dies bedeutet, daß der optimale Parallelisierungsgrad entscheidend von einer geeigneten Dekomposition abhängt, deren Findung ein nichttriviales Problem darstellt. Hilfestellung dazu kann die Analyse des kritischen Pfades geben [BJ85].

Abhängig von der Architektur des Simulationsrechners werden zwei Formen der Parallelisierung unterschieden. Wird ein Rechner verwendet, in dem alle Verarbeitungseinheiten die gleichen Instruktionen zeitgleich auf verschiedene Daten anwenden[2], so wird dies im engeren

2. Nach [Fly66] eine SIMD-Architektur (Single Instruction Stream, Multiple Data Stream).

Sinne als *parallele Simulation* bezeichnet. Am Beispiel der Simulation von Verbindungsnetzen in Parallelrechnern mit einem zeitgesteuerten Simulationsverfahren ist diese Art der Simulation in [Jur96] dargestellt.

Unter einer *parallelen und verteilten Simulation* versteht man eine Simulation, bei der das Simulationsmodell auf voneinander unabhängigen Bearbeitungseinheiten ausgeführt wird und somit keine Kopplung der Recheneinheiten bezüglich der Instruktionen oder der bearbeiteten Daten existiert[3]. Die parallele und verteilte Simulation erlaubt daher auch die modulweise Aufteilung von Simulationsmodellen mit verschiedenartigen Modulen (vgl. Kapitel 2.3.1).

Die parallele und verteilte Simulation stellt eine Sonderform der parallelen Simulation dar, bei der die Recheneinheiten untereinander eine größere Unabhängigkeit besitzen. Daher wird im folgenden, sofern nicht anders angegeben, parallele Simulation als Überbegriff verwendet, der insbesondere die parallele und verteilte Simulation mit einschießt.

2.2.3 Konservative versus optimistische verteilte Simulation

Wird ein Simulationsmodell in logische Prozesse (Module mit eigener Simulationszeit) aufgeteilt, dann hat jeder dieser logischen Prozesse Kanäle, über die Nachrichten ausgetauscht werden können (Bild 2-4).

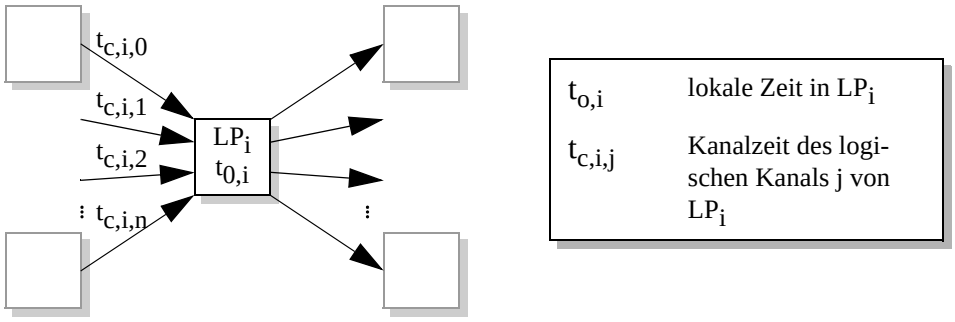


Bild 2-4: Beispiel für das Synchronisationsproblem bei paralleler Simulation

Wird von einem logischen Prozeß eine Nachricht empfangen, so kann dies zur Generierung von lokalen Ereignissen führen, deren Bearbeitungszeitpunkt vom sendenden und nicht vom empfangenden logischen Prozeß abhängt. Dadurch kann es vorkommen, daß nach dem Empfang einer Nachricht Ereignisse generiert werden, deren Bearbeitungszeitpunkt vor der aktuellen Simulationszeit des logischen Prozesses liegt. Anders ausgedrückt: Ist $t_{0,i}$ die aktuelle

3. Nach [Fly66] eine MIMD-Architektur (Multiple Instruction Stream, Multiple Data Stream).

Simulationszeit im logischen Prozeß i, so tritt dieses Problem genau dann auf, wenn (von außen initiiert) ein Ereignis im Prozeß i erzeugt wird, für dessen Zeitstempel $t_{E,i}$ gilt: $t_{E,i} < t_{0,i}$. Diese Situation kann im realen System nicht auftreten und stellt eine Kausalitätsverletzung dar.

Entsprechend dem Grundprinzip der vorwärtsgerichteten Simulation ist die Simulationszeit monoton steigend. Dadurch sind auch die Zeitstempel der Nachrichten, die über einen Eingangskanal bei einem logischen Prozeß ankommen, monoton steigend. Der Zeitstempel der zuletzt über Kanal j übertragenen Nachricht[4] wird als Kanalzeit $t_{c,i,j}$ bezeichnet. Solange die lokale Simulationszeit $t_{0,i}$ kleiner ist als die Kanalzeiten aller Eingangskanäle, kann es zu keinen Kausalitätsverletzungen kommen und das nächste lokal bekannte Ereignis kann bearbeitet werden, falls sein Zeitstempel kleiner ist als alle Kanalzeiten der Eingangskanäle.[5] Ein solches Ereignis wird als sicheres Ereignis bezeichnet.

Konservative und optimistische Simulationsverfahren unterscheiden sich bezüglich ihres Verhaltens, wenn das nächste lokal bekannte Ereignis nicht sicher ist, d.h. seine Zeitmarke größer ist als mindestens eine Kanalzeit eines Eingangskanals.

2.2.3.1 Konservative Simulation

Konservative Simulationsverfahren vermeiden die Entstehung von Kausalitätsverletzungen dadurch, daß unsichere Ereignisse nicht bearbeitet werden. Dies bedeutet, daß ein logischer Prozeß so lange fortschreiten kann, wie der kleinste Zeitstempel der lokal bekannten, noch nicht bearbeiteten Ereignisse kleiner ist als alle Kanalzeiten der Eingangskanäle. Andernfalls muß gewartet werden. Der in Bild 2-3 dargestellte prinzipielle Simulationsalgorithmus muß daher für die konservative parallele Simulation entsprechend Bild 2-5 erweitert werden.

Ein konservativer Algorithmus für die verteilte Simulation wurde erstmals 1979 in [CM79] vorgestellt. Bei der konservativen verteilten Simulation treten prinzipbedingte Schwierigkeiten auf, die einer besonderen Beachtung bedürfen und nachfolgend kurz beschrieben werden. Ausführlichere Darstellungen finden sich u.a. in [Fuj90a, FN92].

Verklemmungen bei konservativer verteilter Simulation

Verklemmungen können auftreten, wenn zwischen den logischen Prozessen zyklische Abhängigkeiten existieren. In diesem Fall ist es möglich, daß jeder logische Prozeß auf eine Nach-

4. Liegt Monotonie vor, so ist dies der größte übertragene Zeitstempel.
5. Die lokale Simulationszeit entspricht stets dem Zeitstempel des zuletzt bearbeiteten Ereignisses.

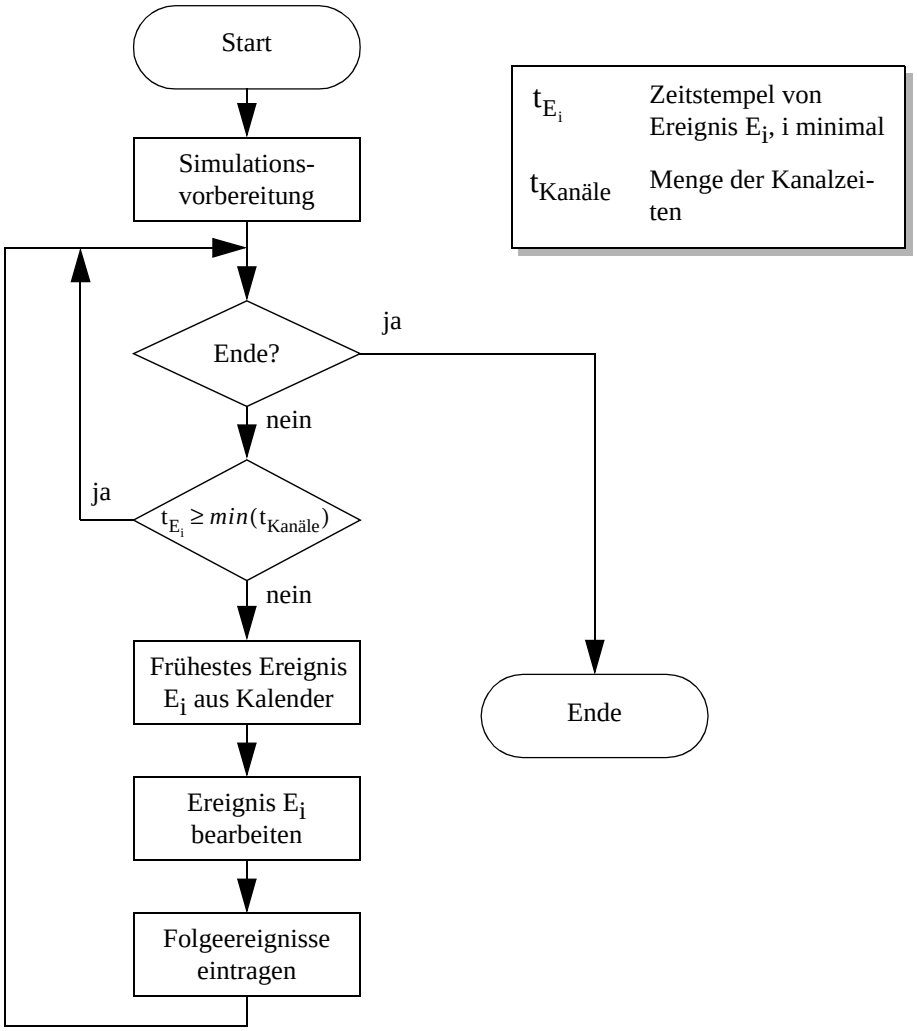


Bild 2-5: Grundprinzip ereignisgesteuerter konservativer paralleler Simulation

richt von seinem logischen Vorgänger wartet, so daß kein Fortschreiten der Simulationszeit mehr stattfinden kann. Bild 2-6 zeigt dies an einem Beispiel. Für sämtliche logische Prozesse gilt, daß ihre Kanalzeit kleiner ist als die Zeit des nächsten bekannten Ereignisses, wodurch ein Fortsetzen der Simulation nicht möglich ist.

In [Mis86] wird vorgeschlagen, das Verklemmungsproblem durch sogenannte „Nullnachrichten“ zu vermeiden. Nullnachrichten tragen keine simulationsrelevante Information und haben keinen Bezug zum realen System. Sie dienen ausschließlich zur Erhöhung der Kanalzeit. Die in Bild 2-6 dargestellte Blockierung würde sich dadurch lösen, daß LP_1 an LP_2 eine Nullnachricht mit dem Zeitstempel 17 schickt. Dieser könnte daraufhin seine lokale Simulationszeit auf 17 erhöhen und eine Nullnachricht mit eben diesem Zeitstempel an LP_3 schicken, worauf dieser sein nächstes lokal bekanntes Ereignis simuliert. Das Verfahren mit Nullnachrichten löst Verklemmungen in allen Zyklen, in denen die Zyklusumlaufzeit größer als Null ist, d.h. wenn

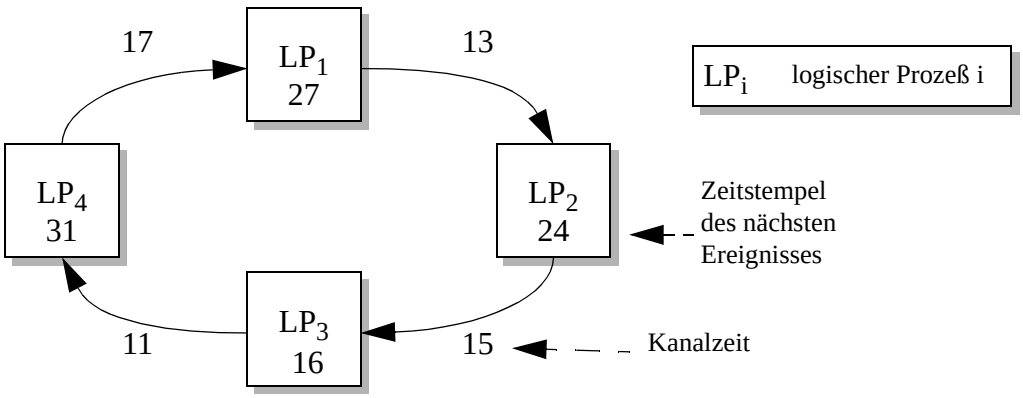


Bild 2-6: Beispiel für eine Blockierung aufgrund zyklischer Abhängigkeiten

wenigstens einer der beteiligten logischen Prozesse nach Erhalt einer Nullnachricht mit Zeitstempel $t_{N,i}$ vom Vorgänger im Zyklus eine Nullnachricht mit Zeitstempel $t_{N,j}$ weitergeben kann, so daß Ungleichung 2-3 gilt.

$$t_{N,j} > t_{N,i} + \varepsilon \qquad \text{mit } \varepsilon > 0,\ j > i \tag{2-3}$$

Unter ungünstigen Umständen, z.B. bei kleiner Zyklusumlaufzeit, können mehrere Umläufe von Nullnachrichten notwendig sein, um eine Blockierung aufzulösen. Daher ist es für eine schnelle Simulation wichtig, daß in einem Simulationsmodell keine Zyklen mit kleiner Umlaufzeit vorhanden sind. Die meisten Simulationsmodelle besitzen ausschließlich von Null verschiedene Zyklusumlaufzeiten, da in realen Systemen Bearbeitungszeiten in den Funktionseinheiten und Laufzeitverzögerungen bei der Nachrichtenweitergabe entstehen.

Die Verklemmungsvermeidung mit Hilfe von Nullnachrichten ist sehr einfach zu implementieren, erfordert bei stark verzweigten und zyklischen Abhängigkeiten der Prozesse während der Simulation jedoch einen wesentlichen Mehraufwand für die Nullnachrichtenerzugung und -auswertung. Dieser kann reduziert werden, indem Nullnachrichten nur auf Anforderung verschickt werden [Mis86]. Bei diesem Ansatz fordert ein logischer Prozeß immer dann Nullnachrichten an, wenn er blockiert ist. Dadurch wird verhindert, daß unnötige Nullnachrichten die Kommunikationskanäle belasten. Da andererseits aber weitere Nachrichten notwendig sind und gewisse Antwortzeiten anfallen, hängt die Eignung der Verfahren vom jeweiligen Simulationsmodell ab.

Alternativ zu den Verfahren mit Nullnachrichten können Verfahren zur Erkennung von Blokkierungen eingesetzt werden. Das in [Mis83] beschriebene Verfahren basiert auf der Verwen-

dung eines Markers. Dieser Marker zirkuliert durch alle Kanäle. Jeder logische Prozeß, der den Marker empfängt, muß ihn in endlicher Zeit weitergeben. Ein logischer Prozeß wird als „weiß“ bezeichnet, wenn er seit der letzten Markerankunft weder eine Nachricht gesendet, noch eine Nachricht empfangen hat. Andernfalls wird er als „schwarz“ bezeichnet. Unter der Voraussetzung, daß Nachrichten in den Kanälen nicht überholen können (FIFO-Kanäle), läßt sich zeigen, daß genau dann eine Verklemmung vorliegt, wenn die Anzahl der logischen Prozesse, die „weiß“ sind, größer oder gleich der Anzahl der im System vorhandenen Kanäle ist. Weitere Algorithmen für die Erkennung und Auflösung von Blockierungen finden sich z.B. in [CMH83, JaSi88, GT91].

Für verschiedene Beispiele [RMM88, Fuj89, WLB89] wurde gezeigt, daß die Verfahren mit Nullnachrichten auf Anforderung den beiden anderen unterlegen sind. Da die Beispiele jedoch nicht verallgemeinert werden können, muß für jeden Anwendungsfall abgewogen werden, welches Verfahren am besten geeignet ist.

Degenerierung zur sequentiellen Ausführung

Unter ungünstigen Umständen kann es trotz Verwendung eines konservativen verteilten Simulationsverfahrens zu einer sequentiellen Simulation kommen. Dies ist dann der Fall, wenn zu jedem Zeitpunkt genau ein Prozeß ein Ereignis bearbeiten kann und die anderen warten müssen. Dies tritt beispielsweise dann auf, wenn das zu simulierende System keine echte Parallelität besitzt, da ein zentrales Modul sequentiell andere Module nutzt. Konservative verteilte Simulationsverfahren sind für solche Systeme ungeeignet. Gleiches gilt für Systeme mit zyklischen Abhängigkeiten, bei denen sich Ereignisse nahezu zeitgleich in anderen logischen Prozessen auswirken können. Ist der mittlere zeitliche Ereignisabstand deutlich größer als der Vorausblick zwischen den logischen Prozessen, so ist das System im wesentlichen mit der Blockierungsauflösung bzw. -vermeidung beschäftigt, was letztlich wegen der engen Kopplung der logischen Prozesse ebenfalls zu einer sequentiellen Bearbeitung der Ereignisse führt.

Entkopplung der logischen Prozesse durch Vorausschauen

In einem großen technischen System ist es in der Regel so, daß sich Ereignisse nicht sofort auf das ganze System auswirken. Ihr direkter Wirkungsbereich beschränkt sich statt dessen auf das Modul ihrer Entstehung sowie auf wenige logisch „angrenzende“ Module, wobei jedoch eine gewisse Zeit vergeht, bevor sie sich dort auswirken. Der Grund dafür können z.B. Laufzeiten, Verzögerungszeiten und Bearbeitungszeiten im realen System sein. Bildet man im Modell die

Grenzen der logischen Prozesse entsprechend solcher Modulgrenzen, so wirkt sich dies bei der konservativen verteilten Simulation positiv aus, da dann die Simulationszeit des abhängigen logischen Prozesses gegenüber seinem Vorgänger nicht nur beliebig nachgehen, sondern um diesen Abstand auch vorauseilen kann (lookahead)[6]. Bild 2-7 zeigt dies am Beispiel eines zyklischen Systems. Für die Simulationszeit $t_{0,2}$ des logischen Prozesses LP_2 gilt in diesem System $t_{0,2} < t_{0,1}+\tau_1$. Aus lokaler Sicht kann $t_{0,2}$ gegenüber $t_{0,3}$ beliebig groß sein, aus dem Zyklus ergibt sich jedoch der maximale Abstand von $t_{0,2}$ und $t_{0,3}$ zu $\tau_3+\tau_4+\tau_1$. Daraus läßt sich ableiten, wie wichtig die Vorausschau für eine hohe Effizienz insbesondere in zyklischen Systemen mit kurzen Zyklen ist, wie sie z.B. bei direkter gegenseitiger Abhänigkeit zweier Module vorliegt.

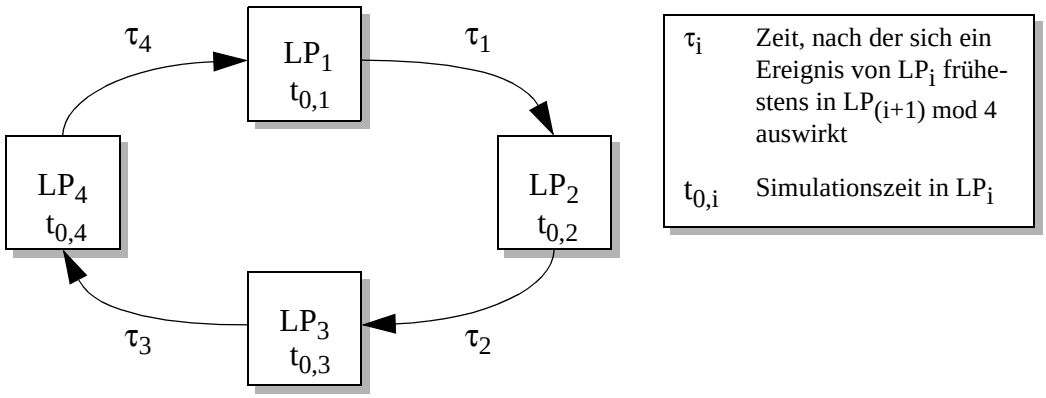


Bild 2-7: System mit Vorausschau (lookahead)

Die Bedeutung der Vorausschau wird in [Fuj88, Fuj89] simulativ anhand von Beispielen gezeigt und begründet. Eine erfolgreiche Anwendung eines konservativen Simulationsverfahrens mit Vorausschau auf ein reales System findet sich z.B. in [Lub89, KWWR93, RRRea94]. Eine Abgrenzung zu optimistischen Simulationsverfahren erfolgt im nachfolgenden Kapitel 2.2.3.2.

2.2.3.2 Optimistische Simulation

Ziel der optimistischen Simulationsverfahren ist es, die bei der konservativen Simulation aufgrund von Blockierungen auftretenden Leerlaufphasen zu nutzen. Jefferson und Sowizral haben dazu in mehreren Veröffentlichungen [JS82, JS83, Jef83, JS85] vorgeschlagen, die Simulation bei fehlenden Nachrichten bzw. Synchronisationsinformationen von anderen logi-

6. Der logische Prozeß kann „vorausschauen“, welche Ereignisse eintreffen werden.

schen Prozessen nicht anzuhalten, sondern sie mit (wahrscheinlichen) Annahmen fortzusetzen. Treten die Annahmen ein, so wird durch das Fortsetzen Zeit gewonnen, treten sie nicht ein, so muß die Simulation mindestens bis zur ersten inkorrekten Annahme zurückgesetzt werden (Rollback) und die falschen Aktionen sind rückgängig zu machen. Diese Vorgehensweise wird von Jefferson und Sowizral auch als „Virtual Time“ bzw. „Time Warp“ bezeichnet.

Damit ein logischer Prozeß im Bedarfsfall korrekt zurücksetzen kann, müssen alle Auswirkungen der falschen Annahmen rückgängig gemacht werden können. Dazu muß ein vor der Kausalitätsverletzung gültiger Prozeßzustand bekannt sein. Die Bereitstellung eines korrekten und aktuellen Systemzustands erfordert erheblichen Aufwand, da dazu der Zustand des Systems regelmäßig gesichert werden muß (Statesaving). Es sind verschiedene Verfahren zur Verbesserung des klassischen Vorgehens bekannt, das nach jeder Ereignisbearbeitung den Systemzustand sichert. [RA94] enthält eine Untersuchung des sogenannten „Sparse Checkpointing“, bei dem die Zustandssicherung erst nach mehreren bearbeiteten Ereignissen erfolgt. Die Anzahl der Ereignisse ist dabei fest oder hängt dynamisch von der „Komplexität“ der bearbeiteten Ereignisse ab. Ein anderen Weg für die Verringerung des zeitlichen Aufwands für die Zustandssicherung wird in [FTG88a] vorgeschlagen. Durch ein spezielles, von der Hardware unterstütztes Speichermanagement soll dabei die Zustandssicherung jeweils in eigene, getrennte Speicherbereiche erfolgen (vgl. Kapitel 2.3.2). Für logische Prozesse mit einem großen und selten geänderten Zustandsvektor oder mit geringer Rollback-Häufigkeit ist es sinnvoll, den Gesamtzustand des logischen Prozesses in größeren Abständen zu sichern. Zwischen diesen Sicherungen werden dann nur die Änderungen protokolliert (Incremental Statesaving) [BS93]. Dadurch wird sowohl der Speicherplatzbedarf als auch der Rechenzeitbedarf für die Zustandssicherung verringert, der Aufwand für die Wiederherstellung eines Zustands wird jedoch deutlich größer. Nachdem ein älterer Prozeßzustand wieder hergestellt wurde, können alle später gesicherten Zustände gelöscht werden.

Neben der Wiederherstellung des Prozeßzustands müssen alle Nachrichten an andere logische Prozesse, die nicht mehr zutreffen, storniert werden. Die Verfahren dafür lassen sich in zwei Klassen einteilen: Verfahren, die alle Nachrichten stornieren, die seit dem Wiederaufsetzpunkt gesendet wurden und Verfahren, die selektiv die falschen Nachrichten für ungültig erklären. Letztere vermeiden das erneute Senden von Nachrichten, die von der Kausalitätsverletzung nicht berührt werden, unter Inkaufnahme eines größeren Aufwands für die Stornierung.

Empfängt ein logischer Prozeß P_i eine verspätete Nachricht mit dem Zeitstempel $t_{E,i}$, so erfolgt das Rücksetzen der Simulation in zwei Phasen: Zuerst wird der letzte gültige gespeicherte Systemzustand wiederhergestellt. Für die zugehörige lokale Simulationszeit $t_{0,i}$ gilt damit Gleichung 2-4.

$$t_{0,\,i} \leq t_{E,\,i} \tag{2-4}$$

Für den Fall, daß nach der Wiederherstellung des Systemzustands die lokale Simulationszeit nicht gleich $t_{E,i}$ ist, schließt sich eine sogenannte „coast forward phase“ an, während der bis zum Erreichen von $t_{E,i}$ bereits korrekt simulierte Ereignisse erneut simuliert werden müssen[7]. Dadurch wird der Systemzustand zum Zeitpunkt $t_{E,i}$ rekonstruiert (Bild 2-8).

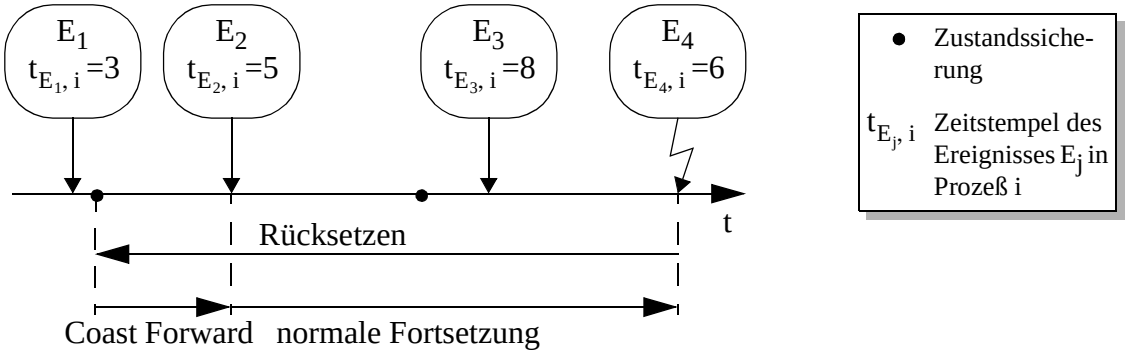


Bild 2-8: Rücksetzen nach Kausalitätsverletzungen bei optimistischer Simulation

Um den Speicherplatz für die Zustandssicherungen zu begrenzen und um das Simulationsende erkennen zu können, muß ermittelt werden, welche der gespeicherten Zustände auf keinen Fall mehr für das Zurücksetzen der Simulation benötigt werden. Dazu wird eine globale Sicht auf alle logischen Prozesse benötigt, da der Prozeß mit der kleinsten Simulationszeit bzw. die in Transit befindliche Nachricht mit dem kleinsten Zeitstempel direkt oder indirekt in allen anderen Prozessen Rücksetzvorgänge auslösen kann. Für die Bestimmung dieser Simulationszeituntergrenze (global virtual time, GVT) werden spezielle Algorithmen benötigt, wie sie z.B. in [Mat93] und [DFW94] beschrieben werden. Der dadurch bestimmte Wert kann auch zur Terminierung der Simulation herangezogen werden, da die bis zu diesem Zeitpunkt ermittelten Simulationsergebnisse gesichert sind.

7. Normalerweise werden in der coast forward phase keine Nachrichten verschickt, da nur Nachrichten mit Zeitstempel größer oder gleich $t_{E,i}$ storniert werden.

Für das beschriebene Basiskonzept der optimistischen Simulationsverfahren gibt es eine ganze Reihe von Optimierungsvorschlägen zur Reduktion der Nachrichtenanzahl, des Aufwandes für Zustandssicherung, GVT-Berechnung und Rücksetzen der Simulation sowie zur Begrenzung der falsch gesendeten Nachrichten. Letzteres wird in der Regel durch Fenstermechanismen erreicht, die dafür sorgen, daß die Simulationszeit eines logischen Prozesses nicht beliebig gegenüber den Simulationszeiten seiner logischen Vorgänger vorauseilen kann [Frö94]. Eine Übersicht über optimistische Simulationsverfahren findet sich beispielsweise in [Rös93].

Die Effizienz einer verteilten Simulation mit optimistischem Simulationsverfahren hängt wesentlich von der Qualität der Annahmen ab, unter denen die Simulation bei der Bearbeitung unsicherer Ereignisse fortgesetzt wird. Zusammen mit einer gleichmäßigen Lastverteilung tragen wahrscheinliche Annahmen zur Vermeidung von Rücksetzungen und damit zur Beschleunigung der Simulation bei.

2.3 Rechnerarchitekturen für die verteilte Simulation

Bei der parallelen Simulation stellt die Wahl einer geeigneten Rechnerarchitektur eine wesentliche Voraussetzung für eine schnelle Simulation dar. Welche Rechnerarchitektur geeignet ist, hängt von der Art der Parallelisierung, vom verwendeten Synchronisationsverfahren und von den Eigenschaften des Simulationsmodells ab.

2.3.1 Allgemeine Parallelrechner

Nach [Fly66] lassen sich Parallelrechner bezüglich ihrer Befehlssteuerung in zwei Klassen einteilen. Rechner der SIMD-Klasse (Single Instruction Stream, Multiple Data Stream) bieten sich besonders für die konservative zeitgesteuerte Simulation von Modellen an, die im wesentlichen durch Replikation eines Moduls entstanden sind. Da die mehrfach vorhandenen Module durch den gleichen Algorithmus beschrieben werden, ist eine einfache Abbildung auf die SIMD-Struktur möglich. Die Synchronisation der Module erfolgt bei einem SIMD-Rechner ohne Mehraufwand durch den gemeinsamen Instruktionsstrom. Rechner der Klasse MIMD (Multiple Instruction Stream, Multiple Data Stream) benötigen hierfür explizite Synchronisationsverfahren [Jur96]. Sie eignen sich jedoch aufgrund ihrer asynchronen und bezüglich dem Instruktionsstrom unabhängigen Recheneinheiten besser für die ereignisgesteuerte Simulation sowie für optimistische Simulationsverfahren. Rechner der Klasse SIMD sind nur bedingt für

die ereignisgesteuerte Simulation geeignet, da sie spezielle Algorithmen zur Unterstützung der asynchronen Ereignisbearbeitung benötigen [AB93].

Ein weiteres wesentliches Klassifizierungskriterium für Parallelrechner ist ihr Kommunikationssystem [SJ96]. In eng gekoppelten Parallelrechnern kommunizieren die Recheneinheiten über gemeinsame Speicherbereiche oder über dedizierte, schnelle Kommunikationskanäle mit sehr einfachen Protokollen. Bei lose gekoppelten Systemen finden für den Nachrichtenaustausch allgemeinere Kommunikationsnetze mit höheren Protokollschichten Anwendung.

Rechnersysteme mit einer losen Kopplung der Recheneinheiten, wie dies z.B. in lokalen Netzen der Fall ist, können erfolgreich bei einer Parallelisierung auf Parameter- oder Teiltestebene angewendet werden, da bei dieser Art der Parallelisierung die einzelnen Prozessoren weitgehend unabhängig voneinander arbeiten und keine zeitkritischen Abhängigkeiten bestehen.

Bei der Verteilung eines Simulationsmodells auf verschiedene Recheneinheiten ist ihre enge Kopplung wichtig für die schnelle Simulation. Dies gilt wegen den Prozeßblockierungen insbesondere für Modelle mit zyklischen Abhängigkeiten und bei der Verwendung eines konservativen Synchronisationsverfahrens (vgl. Kapitel 2.4.5). Aufgrund von physikalischen Beschränkungen ist eine vollvermaschte enge Kopplung von Recheneinheiten nur für eine sehr begrenzte Anzahl von Recheneinheiten möglich ist. Daher werden in eng gekoppelten Parallelrechnern häufig spezialisierte Kommunikationsnetze und -protokolle verwendet, die auf einer Teilvermaschung der Recheneinheiten beruhen. Um den Aufwand und die Wartezeiten, die bei der parallelen Simulation für die Kommunikation erforderlich sind, gering zu halten, müssen die logischen Prozesse so auf die Recheneinheiten verteilt werden, daß die Mehrheit der Nachrichten möglichst direkt vom Sender an den Empfänger weitergegeben werden kann.

2.3.2 Spezielle Rechnerarchitekturen für die Simulation

Um den unterschiedlichen Anforderungen, welche die Simulationsmodelle zur geschwindigkeitsoptimalen Simulation stellen, gerecht zu werden, wurden bereits in der Vergangenheit spezielle Simulatorarchitekturen entwickelt. Sie basieren auf dem Prinzip der funktionalen Parallelisierung oder unterstützen die konservative bzw. optimistische Simulation.

2.3.2.1 Funktionale Parallelisierung

Eine Simulationsbibliothek für die Realisierung der in Kapitel 2.2.2.2 beschriebenen funktionalen Parallelisierung mit Hilfe der Programmiersprache Ada wird in [SCM85] vorgestellt. Die Simulationsausführung wird bei dieser Bibliothek in vier Module aufgeteilt: Zufallszahlenerzeugung, Statistikgenerierung, Kalenderverwaltung und Simulationsmodellberechnung. Jedes Modul wird unter Verwendung eines speziellen Ada-Systems auf einer eigenständigen Recheneinheit eines universellen Multiprozessorrechners zur Ausführung gebracht.

An der Rheinisch-Westfälischen Technischen Hochschule Aachen wurden u.a. von Lehnert, Barel, und Kluth, jeweils unter Verwendung von Spezialrechnern, verschiedene Ansätze für die funktionale Parallelisierung untersucht. Für die statistische Simulation von Warteschlangennetzen wird von Lehnert in [Leh79a, Leh79b, Leh80] der Spezialrechner ZUSI1 beschrieben. Eigenständig arbeitende Zufallszahlenmodule erzeugen Pseudozufallszahlen verschiedener Verteilungen, wodurch die Simulationsausführungseinheit, die für die eigentliche Simulation zuständig ist, entlastet wird. Letztere ist optimiert auf die effiziente und schnelle Bearbeitung von Warteschlangensystemen. Sie besitzt dafür dedizierte Recheneinheiten und Listenverwaltungsmodule. Über den zentralen Datenbus des Systems werden während der Simulation die Werte der Zufallsvariablen der Simulationsausführungseinheit bekanntgegeben. Ein daran angeschlossenes Statistikmodul wertet die Zufallsvariablen statistisch aus und zeigt sie graphisch an.

Der Discret Event Simulation Computer (DESC bzw. DESC1) [Bar84, Bar83] besitzt, ähnlich wie ZUSI1, separate Module für Zufallszahlenerzeugung, Simulationsausführung, Listenverwaltung, Statistik und Ausgabe. Jedes dieser ggf. mehrfach vorhandenen logischen Module wird auf einer eigenständigen, dedizierten Recheneinheit zur Ausführung gebracht. Die Zufallszahlenmodule bestehen aus spezieller Hardware, die gleichverteilte Zufallszahlen erzeugt, und einem Mikroprozessor, der daraus beliebige Verteilungsfunktionen generieren kann. Über FIFO-Speicher werden die Zufallszahlen den anderen Modulen zur Verfügung gestellt. Die Steuerung des Ablaufs der Simulation und der notwendigen Berechnungen erfolgt durch die Simulationsausführungseinheit, die im Gegensatz zu den anderen Modulen nur einmal vorhanden sein kann. Sie besteht im wesentlichen aus einem universellen Mikroprozessor und Speicher. Für die in ereignisgesteuerten Simulationsprogrammen überwiegenden Listenoperationen wurde ein spezielles Modul entwickelt [Bar85], das über FIFO-Puffer Listenkommandos[8] und Daten entgegennimmt, so daß diese unabhängig von der

Simulationsausführungseinheit bearbeitet werden können. Die Generierung von Statistiken und deren Auswertung erfolgt parallel in den Statistikmodulen, die idealerweise mit sämtlichen Listen- und Zufallszahlenmodulen direkt verbunden sind.

In [Klu90] wird DESC2, eine Weiterentwicklung von DESC1, beschrieben. Der wesentliche architektonische Unterschied besteht in der Zusammenfassung von Listenmodul und Simulationsausführungseinheit. Beim praktischen Einsatz von DESC1 wurde festgestellt, daß durch die Nebenläufigkeit von Listenmodulen und Simulationsausführungseinheit keine signifikante Geschwindigkeitssteigerung möglich ist. Die Ursache dafür ist, daß die Simulationsausführungseinheit meistens sofort auf das Ergebnis der in Auftrag gegebenen Listenoperationen wartet. Durch die Integration der Listenfunktionen im Simulationsausführungsmodul wurde der Kommunikationsaufwand eliminiert und damit die Simulationsgeschwindigkeit erhöht. In [Klu90] werden zwei weitere Simulatorarchitekturen vorgestellt, die über keine simulationsspezifische Hardware verfügen. Sie basieren auf einem UNIX-Multiprozessorrechner bzw. auf einem Transputer-Netz. Die funktionale Aufteilung der Simulation deckt sich weitgehend mit der bei DESC2 vorgenommenen.

2.3.2.2 Spezielle Architekturen für die Bestimmung des globalen Zustands

In [Hos85] wird der Multiprozessorrechner PAX beschrieben, dessen Prozessorknoten zweidimensional in einem Netz vermascht sind. Dieses Netz hat einen hohen Gesamtdurchsatz und wird für die Nutzdatenübertragung verwendet. Zusätzlich besitzt der Rechner zwei Bussysteme, von denen eines der Verbreitung von Nachrichten dient, die an alle Knoten geschickt werden müssen (Broadcast), das andere wird ausschließlich für die Prozeßsynchronisation verwendet. Die Architektur von PAX erlaubt somit das schnelle Versenden von Nachrichten und Synchronisationsmeldungen. Für die Zeitberechnungen zur Synchronisation der logischen Prozesse gibt es zwei Alternativen: Entweder muß jeder Rechenknoten die Zeitinformationen von allen anderen Rechenknoten empfangen und selbst auswerten oder es muß eine Recheneinheit zentral und parallel zur Simulation die Berechnungen durchführen und über den Synchronisationsbus an die anderen Recheneinheiten schicken.

Eine Erweiterung der PAX-Architektur zur Unterstützung der funktionalen Parallelisierung durch dedizierte Hardware wird in [Rey91, RP92, RPS92, RPS93] beschrieben. Das im PAX-Rechner vorhandene Bussystem wird durch ein paralleles Reduktionsnetzwerk (PRN) ersetzt

8. Z.B. „Einfügen“, „Löschen“, „Suchen“, ...

(Bild 2-9). Mit Hilfe des PRNs wird parallel zur Simulation ein globaler Zustandsvektor ermittelt. Jeder Knoten legt dazu die einzelnen Werte seines Zustandsvektors[9] sequentiell an das PRN an. Jedem Element des Zustandsvektors wird eine logische oder arithmetische Operation zugeordnet, die in den entsprechenden Reduktionseinheiten (Arithmetic Logic Unit, ALU) des PRNs angewendet wird. Aufgrund der baumartigen Struktur des PRN liegt folglich bei n Knoten nach O(log(n)) der globale Zustandsvektor vor und wird allen Knoten zur Verfügung gestellt. Die Nutzdatenübertragung erfolgt völlig unabhängig davon über ein beliebiges zweites Netz, z.B. das Kommunikationsnetz eines allgemeinen Multiprozessorrechners.

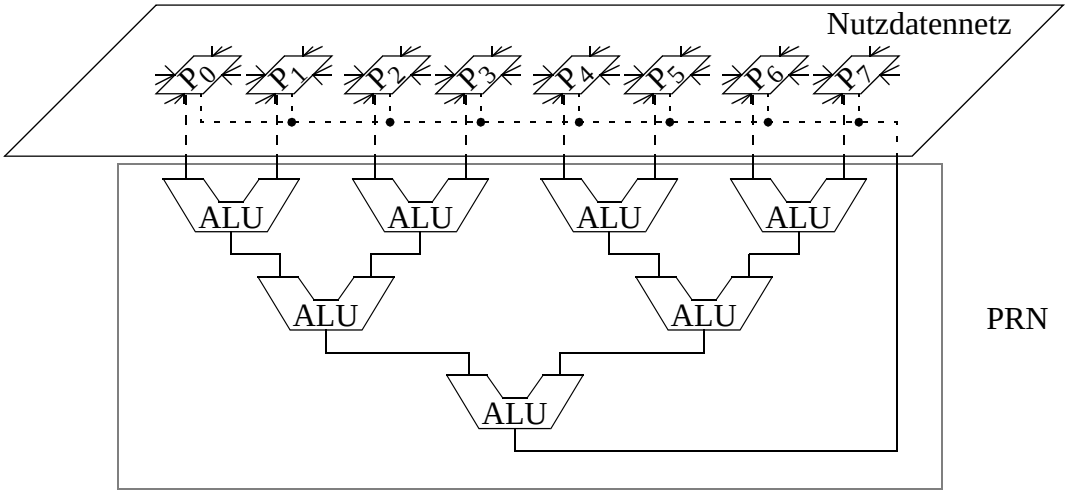


Bild 2-9: PRN zur Ermittlung eines globalen Zustandsvektors nach Reynolds

Eine weitere Verallgemeinerung des PRN wird im ArMen-Projekt der Universität in Brest (Frankreich) durch den Einsatz programmierbarer Logikbausteine verfolgt [BBCea94]. Anstelle einer fest vorgegebenen Anordnung arithmetisch-logischer Einheiten wird jedem Rechenknoten ein FPGA-Baustein (Field Programmable Gate Array) zugeordnet. Die FPGA-Bausteine werden modellspezifisch programmiert und sind untereinander ringförmig verbunden. Diese Anordnung wird von den Autoren als „verteilter Koprozessor“ bezeichnet und erlaubt neben einer schnellen Kommunikation beliebige simulationsunterstützende Berechnungen, wie z.B. Synchronisationsgrenzen bei konservativen oder optimierte GVT-Berechnung bei optimistischen Simulationsverfahren.

9. Der Zustandsvektor kann z.B. bei optimistischen Simulationsverfahren die Simulationszeituntergrenze (GVT) enthalten. Bei konservativen Simulationsfahren kann er zur Bestimmung von globalen Synchronisationspunkten verwendet werden.

Zusammenfassung und Bewertung

Konservative und optimistische Simulationsverfahren profitieren von einer schnellen Ermittlung und Verbreitung globaler Systemzustände. Bei konservativen Simulationsverfahren betrifft dies insbesondere das Erkennen und Vermeiden von Verklemmungen, da durch die globale Ermittlung des nächsten Ereignisses bei zyklischen Abhängigkeiten wesentlich weniger Nullnachrichten erzeugt werden müssen. Für die lokale Synchronisation ist die globale Zustandsbestimmung nicht hilfreich, d.h. solange keine Blockierungen auftreten wird die konservative Simulation durch die globalen Zustandsberechnungen nicht beschleunigt.

Bei den optimistischen Verfahren kann eine schnelle und häufige Ermittlung der Simulationszeituntergrenze (GVT) eine signifikante Verringerung des Speicherbedarfs bewirken. Dies führt zwar nicht direkt zu einer Steigerung der Simulationsgeschwindigkeit, ermöglicht durch den reduzierten Speicherbedarf aber unter Umständen den Einsatz weiterer Optimierungen (z.B. Roll Back Chip, Kapitel 2.3.2.4).

2.3.2.3 Spezielle Architekturen für konservative Simulatoren

Spezielle Rechnerarchitekturen für konservative Simulationsverfahren bieten in der Regel eine Unterstützung für die Prozeßsynchronisation zur Verkürzung von Leerlaufzeiten und ein optimiertes Kommunikationssystem.

In [THH80] wird ein Multiprozessorsystem für die zeitgesteuerte Simulation von (hierarchischen) Netzen beschrieben. Die einzelnen Prozessoren sind über ein aus zwei Hierarchieebenen bestehendes Bussystem gekoppelt, so daß lokale Kommunikationsvorgänge parallel ablaufen können. Die Synchronisation der Prozesse erfolgt davon unabhängig über ein spezielles Synchronisationsnetz. Jeder Prozessorknoten generiert ein Signal, das anzeigt, ob er gerade beschäftigt ist oder nicht. Diese Signale werden über das Bussystem logisch oder-verknüpft. Kann kein Prozessor mehr simulieren, wird dies von einer zentralen Station erkannt. Sie gibt daraufhin über dedizierte Leitungen ein Signal aus, das die Prozessoren auffordert, ihre lokale Simulationszeit um eine Zeiteinheit zu erhöhen und die zugehörigen Ereignisse zu bearbeiten.

Ebenfalls mit speziellen Leitungen und angepaßter Logik arbeitet das in [Con85, Con89] beschriebene hierarchische Simulationssystem, das ein beliebiges hierarchisches Bussystem und eingeschränkt die ereignisgesteuerte Simulation unterstützt.

Ein ähnlicher Ansatz, allerdings auf einem linearen Bussystem basierend, wird in [PGM83] beschrieben. Anstelle des speziellen Synchronisationsnetzes wird hier ein umfangreiches Unterbrechungssystem verwendet, mit dem die einzelnen Prozessoren einer zentralen Station, die für die Verteilung der Simulationsmodule zuständig ist, mitteilen, daß sie einen neuen Simulationsauftrag benötigen. Sofern sich für ein Simulationsmodell geeignete sichere Simulationszeitfenster definieren lassen, kann diese Architektur nicht nur für zeitgesteuerte, sondern auch für ereignisgesteuerte Simulationsverfahren genutzt werden.

Zur Minimierung des Kommunikationsaufwands wird in [Hab92] ein auf Transputern basierendes Multiprozessorsystem verwendet, bei dem sich der Graph der Kommunikationsbeziehungen im zu simulierenden Modell direkt in der Topologie des Multiprozessorrechners widerspiegelt. Dadurch wird der Aufwand für die Verkehrslenkung auf ein Minimum reduziert. Habermann hat diesen Simulator zur Untersuchung von ATM-Vermittlungseinheiten eingesetzt. Da die logischen Prozesse zyklenfrei und regelmäßig[10] über Nachrichten kommunizieren, benötigt der Simulator keine globale Synchronisation.

Zusammenfassung und Bewertung

Die Synchronisationsunterstützung der vorgestellten Architekturen dient zur schnellen Feststellung, wann die Simulation fortgesetzt werden kann. Da dies auf globaler Basis geschieht, stellen sie eine Spezialisierung der in Kapitel 2.3.2.2 vorgestellten Architekturen dar. Sie sind somit ebenfalls für die zeitgesteuerte Simulation geeignet, eine ereignisgesteuerte Simulation wird dagegen nicht bzw. nur eingeschränkt unterstützt. Läßt sich ein Simulationsmodell in mehr logische Prozesse aufteilen als Recheneinheiten vorhanden sind, ermöglicht die Architektur von Concepcion das dynamische Verteilen der Prozesse. Durch den zentralen Kalender ist dies aber nur bei wenigen Prozessoren sinnvoll. Ferner dürfen die logischen Prozesse nur einen kleinen Datenbereich aufweisen, da sonst bei der dynamischen Zuteilung zu viele Daten umkopiert werden müssen.

Der Simulator von Habermann ist im Gegensatz zu den anderen Architekturen für Systeme geeignet, bei denen die Simulationszeit nicht in allen Prozessen gleichzeitig fortschreiten muß, da keine globale Synchronisation erzwungen wird. Stattdessen erfolgt eine lokale Synchronisation durch die regelmäßig verschickten Nachrichten. In den von Habermann untersuchten

10. Mit jeder Zelle.

zyklenfreien Systemen kann es auch bei konservativer Simulation nicht zu Blockierungen kommen.

2.3.2.4 Spezielle Architekturen für optimistische Simulatoren

Neben den Optimierungsmöglichkeiten für die globale Zustandsbestimmung bieten insbesondere Architekturen zur beschleunigten Zustandssicherung und -wiederherstellung ein Potential zur Geschwindigkeitssteigerung optimistischer Simulationen [Fuj90a].

Ein spezieller Baustein zur Optimierung der Zustandsverwaltung bei optimistischen Simulationen ist in [FTG88a, FTG88b] beschrieben. Der als „Roll Back Chip“ (RBC) bezeichnete Baustein stellt im Prinzip eine spezialisierte Speicherverwaltungseinheit dar. Für die Speicherung der Zustände stehen Speicherbereiche fester Größe zur Verfügung, die nach dem Prinzip eines Stapels verwaltet werden. Schreibvorgänge auf Zustandsvariablen erfolgen stets in den Speicherbereich, der zuletzt auf den Stapel gelegt wurde. Eine Zustandssicherung erfolgt durch das Anlegen eines neuen, leeren Speicherbereichs auf dem Stapel. Damit dieser Vorgang schnell erfolgen kann, werden die Zustandsvariablen aus dem dabei verdeckten alten Zustand nicht kopiert. Daher muß beim Lesen von Zustandsvariablen stets der Stapel so lange von oben her durchsucht werden, bis ein gültiger Wert gefunden wird. Speicherbereiche von Zuständen, die für das Rücksetzen nicht mehr notwendig sind (siehe Kapitel 2.2.3.2), werden von der Unterseite des Stapels entfernt. Dabei muß darauf geachtet werden, daß Zustandsvariablen, die nur selten geschrieben werden, nicht verloren gehen, d.h. sie müssen ggf. vorher in den darüberliegenden Speicherbereich kopiert werden. Für das Erkennen solcher Situationen und zur Markierung gültiger Werte werden für jede Zustandsvariable Statusinformationen gespeichert.

Eine Untersuchung zur Leistungsfähigkeit des Roll Back Chips wurde in [BRF90] veröffentlicht. Die von den Autoren gemessene relative Geschwindigkeitssteigerung (Simulationszeitgewinn, Speedup) liegt abhängig vom Simulationsmodell zwischen 1,2 und 106. Es hat sich gezeigt, daß insbesondere bei häufigem Zustandssichern, großen Zustandsvektoren und kurzen Ereignisbearbeitungszeiten ein signifikanter Geschwindigkeitsvorteil gegenüber einer Software-Implementierung erzielt werden kann.

2.4 Aufwand und Nutzen ereignisgesteuerter verteilter Simulation

Für jedes der in Kapitel 2.2 vorgestellten Simulationsverfahren lassen sich Systeme und zugehörige Simulationsmodelle angeben, für die das jeweilige Verfahren bezüglich der Simulati-

onsdauer den anderen Simulationsverfahren überlegen ist. In der Regel stellt sich die Frage jedoch umgekehrt, d.h. es wird zu einem gegebenen Simulationsmodell das am besten geeignete Simulationsverfahren und eventuell ein passender Parallelrechner gesucht. Normalerweise ist hier die Simulationsdauer nicht allein ausschlaggebend, es muß auch der Aufwand für die Implementierung und für die Optimierung der Simulationsparameter berücksichtigt werden.

2.4.1 Einfluß auf die Implementierung

Werden parallele Simulationsverfahren miteinander verglichen, so steht meist die Simulationsdauer im Vordergrund. Sie hängt jedoch nicht ausschließlich vom verwendeten Simulationsverfahren ab. Wesentlichen Einfluß auf die Dauer hat z.B. auch die Optimierung der Implementierung. Bei einer sequentiellen Simulation kann die Weitergabe von Nachrichten zwischen den Komponenten des Simulationsmodells mit Hilfe von Zeigern wesentlich effizienter gestaltet werden als dies bei parallelen Verfahren möglich ist, bei denen Nachrichten beim Senden und beim Empfangen mindestens einmal kopiert werden müssen. Gleiches gilt für Berechnungen, die bei verteilter Simulation mehrfach identisch durchgeführt werden müssen sowie für den Prozeßwechsel zwischen Simulations- und Kommunikationsprozessen. In der Literatur finden sich Untersuchungen, bei denen die parallele Ausführung eines Simulationsprogramms auf n Prozessoren mit der Ausführung des gleichen oder eines nur unwesentlich veränderten Programms auf einem Prozessor verglichen wird. Dabei bleibt unter Umständen ein großes Potential von Optimierungsmöglichkeiten der sequentiellen Simulation unberücksichtigt, was zur Folge hat, daß sich ein verfälschtes Bild von der Leistung des parallelen Simulators ergibt.

Wird bei der Implementierung eines Simulationsprogramms eine geeignete Simulationsbibliothek oder eine speziell dafür entwickelte Simulationssprache verwendet, so kann bei konservativen Simulationsverfahren der Mehraufwand bei der Entwicklung der Modellkomponenten nahezu kompensiert werden [ULA85, Roz85, Ree85, MM93, MR93, Ne94, Ne98], da die Prozeßsynchronisation und der Nachrichtenaustausch durch Bibliothekskomponenten bzw. durch entsprechende Sprachkonstrukte erfolgt. Bei optimistischen Verfahren ist ein deutlicher Mehraufwand erforderlich, da sämtliche Komponenten über Mechanismen zur Zustandssicherung und -wiederherstellung verfügen müssen. Diese Funktionalität kann im allgemeinen aus Effizienzgründen nicht von einer Bibliothek oder einem Simulationslaufzeitsystem erbracht werden. Auch die Fehlersuche ist im Zusammenhang mit optimistischen Simulationsverfahren deutlich schwieriger, da die Wiederholbarkeit nicht gegeben ist und somit Fehler abhängig von Nach-

richtenlaufzeiten im System sporadisch auftreten können. Bei konservativen Simulationsverfahren erfolgt dagegen die Bearbeitung der Ereignisse innerhalb eines logischen Prozesses stets in der gleichen Reihenfolge.

2.4.2 Optimierung der Parameter des Simulators

Gegenüber den sequentiellen besitzen parallele Simulatoren zusätzliche Parameter zur Steuerung des Simulationsablaufs. Sie müssen für eine schnelle Simulation geeignet gewählt werden.

Während bei der konservativen parallelen Simulation nur wenige zusätzliche Parameter zur Steuerung der Nullnachrichtengenerierung existieren, gibt es bei der optimistischen Simulation wesentlich mehr Freiheitsgrade. Dazu gehören die Steuerung der Zustandssicherung[11], das Ungültigerklären von Nachrichten beim Rücksetzen[12] und die Mechanismen zur Begrenzung des Optimismus [Rös93]. Da sich die Parameter nicht direkt aus dem Simulationsmodell ableiten lassen, andererseits aber für eine schnelle Simulation von großer Bedeutung sind, müssen sie durch Testläufe geeignet bestimmt werden. Der dafür notwendige Aufwand ist signifikant und darf nicht unberücksichtigt bleiben.

2.4.3 Erzielbare Verkürzung der Simulationsdauer durch parallele Simulation

Durch Aufteilen eines Simulationsprogramms auf n Recheneinheiten kann die Simulationsdauer t_s bei konservativen Simulationsverfahren bestenfalls auf $t_{min}=t_s/n$ reduziert werden.[13] Diese untere Schranke gilt für alle Simulationsmodelle. Die in einem Simulationsmodell innewohnende Parallelität ist jedoch nicht immer so groß, daß dieser Wert erreicht werden kann. In [Liv85, BJ85] sind Untersuchungen zur Ermittlung der inhärenten Parallelität und damit zur Bestimmung einer genaueren unteren Schranke beschrieben. Die Verfahren beruhen auf der Suche nach der längsten sequentiell zu bearbeitenden Ereigniskette eines Simulationsablaufs.

11. Anzahl bzw. Komplexität der Ereignisse, nach der eine Zustandssicherung erfolgt sowie inkrementelle oder nicht inkrementelle Zustandssicherung.
12. Alle seit dem wiederhergestellten Zutand verschickten Nachrichten oder nur die falsch verschickten löschen.
13. Aufgrund von Nichtlinearitäten der Systemkomplexität bezüglich der Systemgröße kann es unter besonderen Umständen zu noch kürzeren Simulationszeiten kommen (super linear Speedup) [CPSV91]. Da solche Effekte nur selten dominieren, bleiben sie im folgenden unberücksichtigt.

Die Länge dieser Ereigniskette bestimmt die minimale Simulationsdauer, da alle weiteren Ereignisse parallel dazu bearbeitet werden können.

Viele Veröffentlichungen von Simulationen zeigen, daß die inhärente Parallelität in komplexen technischen Systemen groß genug ist, um bei geringer Anzahl an Prozessoren (< 10) relativ nahe an t_{min} heranzukommen. Bei Verwendung optimistischer Simulationsverfahren kann t_{min} erreicht werden, wenn die einzelnen logischen Prozesse ggf. unter Verwendung von Annahmen mit sehr hoher Eintreffwahrscheinlichkeit ihre Simulation fortsetzen können. Dies gilt allerdings nur für wenige der praxisrelevanten Simulationsmodelle.

In [LM90] wurden das konservative und das optimistische parallele Simulationsverfahren verglichen. Dort wird gezeigt, daß unter gewissen Vereinfachungen optimistische Simulationsverfahren bei n Recheneinheiten maximal um den Faktor n schneller als konservative Verfahren sein können. Andererseits können ihnen konservative Verfahren maximal um einen implementierungsabhängigen konstanten Faktor überlegen sein, der sich aus den Zeiten für einen Rücksetzvorgang und für die Bearbeitung eines Ereignisses ergibt.

Berücksichtigt man, daß es Mehrprozessorrechner mit mehreren hundert bis mehreren zehntausend Prozessoren gibt, so liegt es nahe, das Potential der Parallelverarbeitung für die Beschleunigung der Simulation nutzbar zu machen. Daß dies teilweise mit Erfolg möglich ist, wurde beispielsweise in [RRRea94, Fuj89, THH80, CPSV91, AILea95] nachgewiesen. Viele Untersuchungen zeigen jedoch, insbesondere bei Verwendung von optimistischen Simulationsverfahren, keine oder nur geringe Geschwindigkeitssteigerungen durch Parallelisierung [Zbo95, Fuj89, Bau94, GT91]. Eine ausführliche Diskussion über die erzielbaren Geschwindigkeitssteigerungen und die Grenzen der Synchronisationsverfahren für die parallele und verteilte Simulation sind in [Lin90] veröffentlicht, wobei für die Analyse einige gravierende Vereinfachungen vorgenommen wurden.

2.4.4 Ereignislokalität und Ereignisdichte in Simulationsmodellen

Eine wesentliche Voraussetzung für eine schnelle verteilte Simulation ist ein geeignetes Simulationsmodell sowie eine passende Aufteilung des Modells in miteinander kommunizierende logische Prozesse. Die Effizienz wird wesentlich durch die Nachrichten bestimmt, die zwischen den logischen Prozessen ausgetauscht werden [Fuj89], da durch sie bei konservativer Simulation Blockierungen aufgelöst werden und bei optimistischer Simulation das Risiko

begrenzt wird. Im folgenden werden daher verschiedene Maße zur Charakterisierung von Simulationsmodellen bezüglich ihrer Ereignisse definiert.

Definition der räumliche Ereignislokalität 1 eines logischen Prozesses

Die räumliche Ereignislokalität 1 gibt an, wie groß der Anteil der Ereignisse in einem logischen Prozeß ist, die keine (direkte) Auswirkung auf andere logische Prozesse haben, somit lokal bleiben und daher weder an andere logische Prozesse Nachrichten generieren noch zur Synchronisation der Prozesse dienen. Ist $a_{Eo}(i,t)$ die Anzahl der Ereignisse im logischen Prozeß *i* bis zur Simulationszeit t, die eine Nachricht an andere logische Prozesse generieren und $a_E(i,t)$ die Zahl der insgesamt im logischen Prozeß i erzeugten Ereignisse, so wird REL1(*i*,*t*) definiert zu

$$\mathrm{REL1}(i,t) \;=\; 1 - \frac{a_{Eo}(i,t)}{a_E(i,t)} \tag{2-5}$$

Definition der räumlichen Ereignislokalität 2 eines logischen Prozesses

Neben dem Anteil der Ereignisse, die durch Nachrichten in anderen logischen Prozessen Ereignisse generieren, sind die Kommunikationsbeziehungen zwischen den logischen Prozessen von Bedeutung. Daher wird die räumliche Ereignislokalität 2 entsprechend Gleichung 2-6 in Abhänigkeit von der Anzahl der vorhandenen logischen Prozesse p und der Anzahl der Prozesse $p_E(i)$ definiert, an die der logische Prozeß i Ereignisse schickt.

$$\mathrm{REL2}(i) \;=\; 1 - \frac{p_E(i)}{p} \tag{2-6}$$

REL2(i) gibt somit an, wie groß der Anteil der Prozesse ist, die in keiner (direkten) Kommunikationsbeziehung mit dem logischen Prozeß i stehen.

Definition der zeitlichen Ereignislokalität eines logischen Prozesses

Wie bereits in Kapitel 2.2.3.1 dargestellt, ist die Vorausschau bei konservativer Simulation sehr hilfreich. In der Literatur wird die Vorausschau meistens in Simulationszeiteinheiten angegeben (z.B. [Fuj90a, Fuj88]), da diese Größe direkt aus der Prozeßaufteilung des Simulationsmodells und aus den Randbedingungen des zu simulierenden Systems abgeleitet werden kann. Bezüglich der Effizienz einer verteilten Simulation ist jedoch nicht die absolute zeitliche Größe der Vorausschau wichtig, sondern die Anzahl der Ereignisse, die in dieser Zeit im Mittel

bearbeitet werden können. Ist $a_{El}(i,t)$ die Anzahl der Ereignisse, die aufgrund des Vorausblicks bearbeitet werden können, so sei die zeitliche Ereignislokalität ZEL

$$\mathrm{ZEL}(i,t) = \frac{a_{El}(i,t)}{a_E(i,t)}. \tag{2-7}$$

Die zeitliche Ereignislokalität stellt damit ein Maß für den Nutzen dar, den die verteilte Simulation aus der begrenzten „Ausbreitungsgeschwindigkeit“ der Ereignisse zieht.

Definition der Ereignisdichte in einem Simulationsmodell

Der Aufwand, der für die Simulation der Komponenten eines Simulationsmodells notwendig ist, hängt von der Anzahl und der Komplexität der in ihnen auftretenden Ereignisse ab. Um dies zu beschreiben, werden zwei Maße definiert: Die Ereignisdichte und die gewichtete Ereignisdichte. Die Ereignisdichte setzt die Anzahl der in der Komponente erzeugten Ereignisse in Relation zu den im gesamten Simulationsmodell erzeugten Ereignissen. Ist $a_E(k,t)$ die Anzahl der Ereignisse, die in der Komponente k bis zur Simulationszeit t auftreten, und $a_E(t)$ die Anzahl der Ereignisse, die im gleichen Zeitraum im gesamten Simulationsmodell erzeugt werden, so ist die Ereignisdichte

$$\mathrm{ED}(k,t) = \frac{a_E(k,t)}{a_E(t)}. \tag{2-8}$$

Die gewichtete Ereignisdichte berücksichtigt zusätzlich die Komplexität der Ereignisse und damit ihren Rechenzeitbedarf. Sie wird mit Hilfe einer Gewichtungsfunktion $g(E_i)$ für Ereignisse definiert:

$$\mathrm{gED}(k,t) = \frac{\sum_{i \in M_k(t)} g(E_i)}{\sum_{i=1}^{a_E(t)} g(E_i)} \quad \text{mit } M_k(t) = \{ i \mid E_i \text{ Ereignis in Komponente k} \} \tag{2-9}$$

Die Ereignisdichte in einem Simulationsmodell läßt sich auch auf logische Prozesse übertragen. Die Ereignisdichte für einen logischen Prozeß ergibt sich entsprechend aus dem Verhältnis der in ihm anfallenden Ereignisse und den im Simulationsmodell anfallenden Ereignissen.

2.4.5 Auswirkungen von Ereignisdichte und -lokalität auf die Effizienz der Simulation

Den Eigenschaften der in einem Simulationsmodell vorkommenden Ereignisse sowie der Modellaufteilung in logische Prozesse kommt bezüglich der Effizienz der verschiedenen Simulationsverfahren für die verteilte Simulation eine entscheidende Bedeutung zu. Dies liegt zum einen am Kommunikationsbedarf, zum anderen an der vom Kommunikationsverhalten abhängigen Wirksamkeit der verwendeten Synchronisationsverfahren, die sich bei konservativen Verfahren in der Blockierhäufigkeit und bei optimistischen Verfahren im Risikoanteil zeigt. Zur Formulierung einiger Zusammenhänge können die in Kapitel 2.4.4 definierten Maße herangezogen werden. Dabei ist zu beachten, daß für Aussagen bezüglich der Effizienz in aller Regel der jeweils ungünstigste Teil des Simulationsmodells betrachtet werden muß, da z.B. ein einzelner häufig blockierter Prozeß oder ein einzelner völlig überlasteter Prozessor zum geschwindigkeitsbestimmenden Engpaß der Simulation wird.

Ereignisdichte und logische Prozesse

Eine wesentliche Voraussetzung für eine effiziente verteilte Simulation ist die Aufteilung des Simulationsmodells in logische Prozesse sowie deren Verteilung auf die verfügbaren Prozessoren. Eine verteilte Simulation kann nur dann effizient sein, wenn sich die Summe der gewichteten Ereignisdichten der logischen Prozesse, die auf einem Prozessor bearbeitet werden, homogen über die verfügbaren Prozessoren erstreckt. Ändert sich bei einem Simulationsmodell die Ereignisdichte über der Zeit, so muß diesem Umstand durch dynamische Lastverteilung Rechnung getragen werden. Vorschläge dafür finden sich z.B. in [NR85, GT93, BD97].

Die Bildung der Prozeßgrenzen hat einen wesentlichen Einfluß auf den Umfang und die Eigenschaften der Ereignisse, die Nachrichten generieren. Abhängig davon entstehen für eine effiziente verteilte Simulation unterschiedliche Anforderungen bezüglich des verwendeten Kommunikationssystems. Umgekehrt gilt, daß bei gegebenem Parallelrechner und damit gegebenem Kommunikaitonssystem die Effizienz der Simulation von den räumlichen Ereignislokalitäten abhängt. Daher müssen Simulationsmodell, Rechnersystem und Simulationsverfahren aufeinander abgestimmt sein.

Räumlichen Ereignislokalität 1

Ist die räumliche Ereignislokalität 1 (REL1) sehr hoch, so werden keine besonderen Anforderungen an das Kommunikationssystem gestellt, da nur sehr wenige Ereignisse in Form von

Nachrichten an andere Prozesse weitergegeben werden müssen. Ist sie jedoch niedrig, so muß es unabhängig vom verwendeten Simulationsverfahren eine geringe Verzögerung und eine hohe Bandbreite sowie ein möglichst einfaches und damit schnelles Protokoll aufweisen. Die Bedeutung dieser Forderungen wurde für das in [Wit94] beschriebene Simlationsszenario gezeigt, das eine REL1 von weniger als 0,5 besitzt. In dieser Simulation wurde bei Verwendung von Ethernet mit einer Bandbreite von 10 Mbit/s und TCP/IP rund die Hälfte der Rechenzeit für die Kommunikation verwendet. Der Einfluß unterschiedlicher Kommunikationssubsysteme auf die Effizienz von verteilten Simulationen wird u.a. in [Gru94] diskutiert. [CFE94] enthält eine weitergehendere Untersuchung, bei welcher die Verzögerung, die bei der Nachrichtenübermittlung auftritt, gesondert berücksichtigt wird.

Räumlichen Ereignislokalität 2

Ereignisse sollen mit möglichst geringer Verzögerung zwischen den Prozessoren weitergegeben werden können. Dazu eignen sich am besten dedizierte Kanäle, so daß keine weitere Zwischenspeicherung und Verkehrslenkung erforderlich ist. Ist REL2 groß, so lassen sich dem Simulationsmodell entsprechende Kanäle auch bei einer großen Anzahl an Prozessoren realisieren bzw. auf die Knotentopologie eines vorhandenen Parallelrechners abbilden. Für kleine REL2 ist dies in der Regel nicht möglich, so daß hier mit einer verminderten Effizienz zu rechnen ist.

Zeitliche Ereignislokalität

Fujimoto hat in [Fuj88] den Einfluß des Vorausblicks auf die Effizienz verteilter konservativer Simulationsverfahren und in [Fuj90b] den Einfluß auf optimistische Simulationsverfahren anhand synthetischer Simulationsmodelle untersucht. Besteht zwischen den logischen Prozessen ein großer Vorausblick, so daß die logischen Prozesse eine große ZEL besitzen, kann mit konservativen Simulationsverfahren bei homogener Ereignisdichte effizient simuliert werden. Bei ausreichend großer Bandbreite und geringer Nachrichtenverzögerung des Kommunikationssystems gilt dies unabhängig von REL1. Beispiele für solche Simulationsmodelle finden sich u.a. in [RRRea94, CT97].

Zusammenfassung

Abhängig von der zeitlichen (ZEL) und räumlichen Lokalität 1 (REL1) sind für eine schnelle Simulation unterschiedliche Simulationsverfahren geeignet (Bild 2-10 zeigt den qualitativen Zusammenhang). Läßt sich ein Simulationsmodell nur so in logische Prozesse aufteilen, daß

deren ZEL und REL1 klein sind, so kann in der Regel durch Parallelisieren auf Modellebene keine oder nur eine unwesentliche Steigerung der Simulationsgeschwindigkeit erzielt werden, da bei konservativen Simulationsverfahren Blockierungen auftreten und optimistische Verfahren unnötig häufig Zustandssicherungen bzw. Rücksetzungen durchführen.

Der Einfluß, den die Ereignislokalitäten bei realistischen Simulationsmodellen auf die optimistischen Simulationsverfahren haben, zeigt sich u.a. an der unterschiedlichen Effizienz, die bei der Simulation verschiedener Modelle erzielt werden kann [Zbo95, tB95, Weg96, CJB97]. Aufgrund des hohen Aufwands für Zustandssicherung, GVT-Berechnung und Rücksetzen kann mit optimistischen Simulationsverfahren gegenüber den konservativen Simulationsverfahren nur dann eine Beschleunigung erzielt werden, wenn aus dem Fortsetzen der Simulation mit Annahmen ein signifikanter Zeitvorteil gegenüber den konservativen Verfahren erzielt werden kann. Folglich kann kein Gewinn bei Simulationsmodellen erreicht werden, die mit konservativen Simulationsverfahren effizient simuliert werden können. Dies wird insbesondere bei Modellen deutlich, die einen relativ großen Vorausblick (vgl. Kapitel 2.2.3.1) und damit eine große ZEL besitzen. Ist der Vorausblick groß genug, kann mit konservativen Verfahren ohne wesentliche Wartezeiten und ohne den Mehraufwand der optimistischen Verfahren effizient simuliert werden, da in diesem Fall die Simulationszeiten der logischen Prozesse ebenfalls weitgehend unabhängig voneinander sind. Dadurch besitzt die konservative Simulation mit großem Vorausblick die Vorteile der optimistischen Simulation, ohne deren Nachteile zu haben.

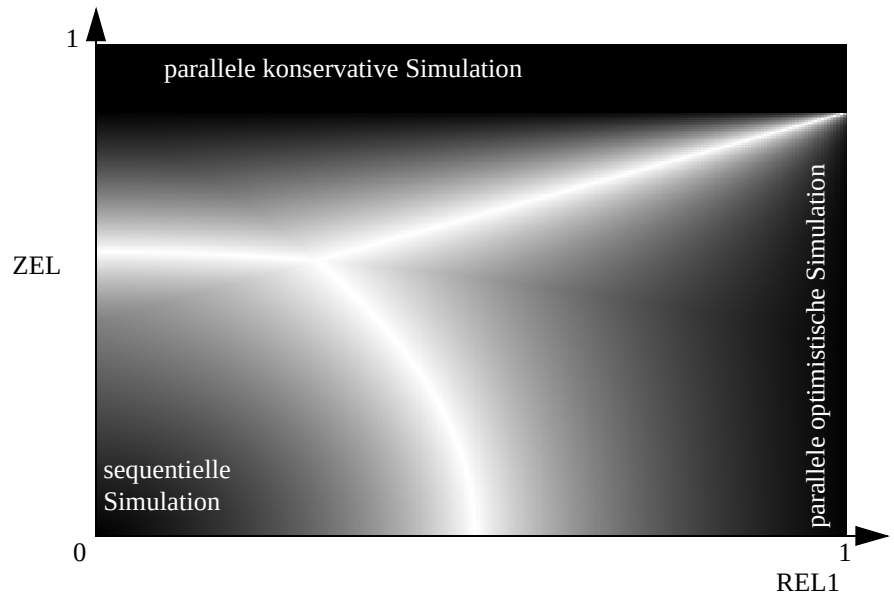


Bild 2-10: Eignung der Simulationsverfahren in Abhängigkeit von ZEL und REL1 (qualitativ)

2.4.6 Zusammenfassung

Die Abhängigkeit der Simulationslaufzeit von der Lokalität der Ereignisse führt dazu, daß es kein allgemeines, für alle Klassen von Simulationsmodelle effizientes und schnelles Simulationsverfahren geben kann, da stets eine Optimierung bezüglich der im Simulationsmodell auftretenden Ereignisse möglich und notwendig ist. Dies gilt in gleicher Weise auch für die bei der Simulation verwendeten Rechenanlagen. Wie eine Optimierung konkret aussehen kann, wurde im Kapitel 2.3.2 an einigen Beispielen gezeigt. Von den räumlichen und zeitlichen Ereignislokalitäten hängt ab, welche Simulationsbeschleunigung durch Parallelisierung erzielt werden kann. Sind sie ausreichend groß, wird eine signifikante Beschleunigung erzielt und die Effizienz liegt nahe bei Eins.

Prinzipiell ist die Integration verschiedener Verfahren und Optimierungen in eine Simulationsumgebung möglich, die aus einer Simulationsbibliothek und einer geeigneten Simulationsrechenanlage besteht [Ne98]. Eine solche Simulationsbibliothek erlaubt es, den durch die parallele Simulation notwendigen Mehraufwand bei der Implementierung eines Simulationsmodells auf ein Minimum zu reduzieren. Bei der konservativen parallelen Simulation sind keine Ergänzungen der Modellkomponenten erforderlich, bei der optimistischen Simulation sind die Mechanismen zur Zustandssicherung und -wiederherstellung zusätzlich zu implementieren. Ferner müssen bei der optimistischen Simulation zahlreiche simulationsverfahrenbezogene Parameter definiert und optimiert werden.

2.5 Simulationsbeschleunigung ohne Parallelverarbeitung

Neben der Parallelisierung gibt es noch weitere Verfahren zur beschleunigten Simulationsausführung, die mit parallelen Simulationsverfahren kombinierbar sind und teilweise neue Möglichkeiten für die Parallelisierung eröffnen. Eine umfangreiche Auflistung von Veröffentlichungen zu den nachfolgend beschriebenen Simulationsverfahren befindet sich in [Hee97a].

Eine Möglichkeit, Analyse und Simulation miteinander zu verbinden, bietet die Methode der Dekomposition und Aggregation [Küh77, Cou85]. Dabei wird das zu untersuchende Modell in kleinere und einfachere Teilmodelle zerlegt. Jedes dieser Teilmodelle wird getrennt untersucht und mit Hilfe der Interaktionen zwischen den Teilmodellen wird eine Näherung für das Verhalten des gesamten Modells bestimmt. Da die Teilmodelle unabhängig voneinander untersucht werden können, kann hierfür insbesondere die parallele und verteilte Simulation eingesetzt

werden. Problematisch ist bei dieser Methode die starke Modellabhängigkeit sowie die notwendige genaue Systemkenntnis, die oft erst aus der Untersuchung resultiert.

Zur Erhöhung der statistischen Sicherheit sind verschiedene Methoden bekannt, welche die Varianz der gesuchten Zielgrößen reduzieren. Sie basieren überwiegend auf Korrelationen zwischen Simulationsläufen, Korrelationen zwischen Zufallsvariablen[14] und funktionalen Abhängigkeiten[15]. Allen gemeinsam ist die starke Abhängigkeit vom Simulationsmodell sowie eine genaue Kenntnis der im Modell ablaufenden Vorgänge. Im allgemeinen läßt sich die erzielbare Varianzreduktion nicht vorhersagen. Ein Überblick über die Verfahren und ihre Vor- bzw. Nachteile befindet sich in [Gör98].

Bereits in [KM53] wird im Zusammenhang mit Monte-Carlo-Simulationen ein Verfahren diskutiert, bei dem das Simulationsmodell so verändert wird, daß der zu untersuchende Effekt häufiger auftritt und damit bei gleicher statistischer Sicherheit kürzere Simulationszeiten erzielt werden können. Die am modifizierten Modell aufgezeichnete Statistik muß dann für Aussagen über das ursprüngliche Modell transformiert werden [SS97]. Dieses als „Importance Sampling“ bezeichnete Verfahren, läßt sich im Prinzip auf die statistische Simulation übertragen und wurde dafür auch erfolgreich angewendet [OS94, DT93, FM93, LY88, Dav86, Man98]. Ein großes Problem bei Importance Sampling ist das Finden eines geeigneten, übertragbaren Simulationsmodells sowie der Umstand, daß dies meistens nur für einzelne Parameter des zu untersuchenden Systems gelingt. Sollen mehrere Systemgrößen gleichzeitig untersucht werden, stellt demzufolge Importance Sampling meist keine Alternative dar.

Eine weitere Methode zur Beschleunigung von Simulationen, bei denen es um das Auftreten seltener, einzelner Ereignisse bzw. Systemzustände geht, enthält [VAVA91]. Bei der dort vorgeschlagenen RESTART-Methode (Repetitive Simulation Trials After Reaching Thresholds) wird ausgenutzt, daß es Zustände gibt, aus denen die seltenen Ereignisse bzw. Zustände mit höherer Wahrscheinlichkeit folgen. Soll z.B. durch Simulation die Häufigkeit von Pufferüberläufen simuliert werden, so ist die Wahrscheinlichkeit dafür größer, wenn der Puffer weitgehend voll ist, d.h. sein Füllstand über einer bestimmten Schwelle liegt. Da dieser „kritische“ Zustand in der Simulation häufiger erreicht wird als der Pufferüberlauf, läßt sich die statistische Aussagesicherheit dadurch erhöhen, daß die Simulation mehrfach ab diesem Zustand mit

14. Z.B. durch Verwenden der gleichen Zufallszahlen für mehrere Systemkonfigurationen, so daß die Ergebnisänderungen nicht auf Änderungen der Simulationsstimuli basieren.
15. Indirekte Schätzung: Bestimmung eines Werts aufgrund funktionaler Zusammenhänge aus einem anderen Wert, der mit einer geringeren Varianz ermittelt werden kann.

anderen Zufallszahlen wiederholt und anschließend normal fortgesetzt wird. Mit Hilfe von bedingten Wahrscheinlichkeiten kann dann die Auftrittwahrscheinlichkeit des seltenen Zustands bzw. Ereignisses ermittelt werden. Bild 2-11 zeigt dies an einem Beispiel. Immer, wenn die Kenngröße den Schwellwert erreicht, wird die Hauptsimulation unterbrochen und es wird in die RESTART-Phase eingetreten. Im gezeigten Beispiel wird die Simulation ab diesem Punkt insgesamt dreimal mit anderen Zufallszahlen wiederholt und so lange fortgeführt, bis die Schwelle wieder unterschritten wird. Nach dem dritten Lauf wird die Hauptsimulation normal fortgeführt.

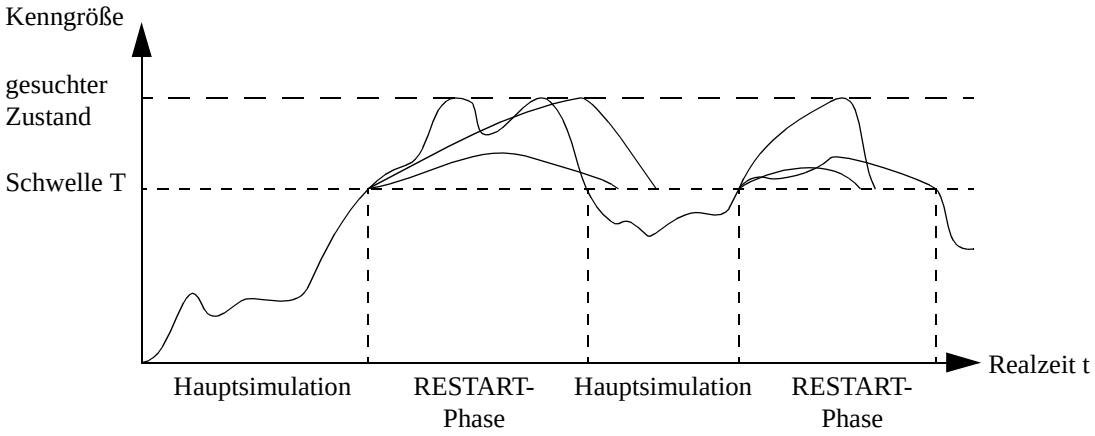


Bild 2-11: Beispiel für eine Simulation mit RESTART

Je Simulationslauf kann bei der RESTART-Methode immer nur ein seltener Zustand bzw. ein seltenes Ereignis untersucht werden. Ist dies der Fall, kann in Kombination mit anderen Verfahren ein erheblicher Simulationszeitgewinn erzielt werden [GF98]. Sind mehrere, voneinander unabhängige Größen gesucht, kann mit RESTART kein signifikanter Geschwindigkeitsvorteil erzielt werden [Köh94]. Eine weitere Schwierigkeit stellt die optimale Festlegung der Schwellwerte dar, deren Wahl Einfluß auf die Simulationsgeschwindigkeit und Genauigkeit hat [Hee97b, Hee98].

Da die einzelnen Wiederholungen der Simulation ab dem „kritischen“ Zustand voneinander unabhängig sind, bietet es sich an, diese zu einer weiteren Geschwindigkeitssteigerung auf mehrere Prozessoren zu verteilen. Dadurch ergibt sich die Möglichkeit, ähnlich wie bei der Parallelisierung auf Parameterebene (Kapitel 2.2.2.1), mit hoher Effizienz parallel zu simulieren.

Auf dem gleichen Prinzip wie das RESTART-Verfahren basiert der in [MBCea98] beschriebene Simulator, der aus der Kopplung von zwei Simulatoren mit unterschiedlichem Abstrakti-

onsniveau aufgebaut ist. Solange vorgegebenen Schwellwerte nicht erreicht werden, wird die Simulation auf hohem Abstraktionsniveau durchgeführt, so daß verhältnismäßig schnell größere Zeiträume durchschritten werden können. Sobald das System in „kritische“ oder „interessante“ Zustände kommt, was sich im Erreichen der Schwellwerte äußert, wird die Simulation bis zum Verlassen derselben mit dem detailierter arbeitenden Simulator fortgeführt. Der in [MBCea98] beschriebene Simulator wird für die Untersuchung von ATM-Netzen verwendet. Bei niedriger Netzlast erfolgt die Simulation auf Verbindungsebene, bei hoher Last erfolgt eine detailiertere Simulation auf der Zellebene.

Kapitel 3

Simulatorarchitektur

Gegenwärtig läßt sich das konservative parallele Simulationsverfahren nur auf zyklenfreie Simulationsmodelle und auf Modelle mit großer zeitlicher Ereignislokalität effizient anwenden (siehe Kapitel 2.4.5, Bild 2-10). Abhängig vom verwendeten Kommunikationssystem muß zusätzlich eine gewisse räumliche Ereignislokalität 2 (REL2) vorliegen. Die in diesem Kapitel vorgestellte Simulatorarchitektur erlaubt die Ausdehnung dieses Bereiches auf Modelle, die im Verhältnis zu den auf allgemeinen Parallelrechnern geeignet konservativ parallel simulierbaren Modellen eine deutlich geringere räumliche und zeitliche Ereignislokalität besitzen. Mit Hilfe dieser Architektur können Modelle, die bislang sequentiell oder optimistisch parallel simuliert werden mußten, mit einem parallelen und konservativen Simulationsverfahren sinnvoll und schnell bearbeitet werden (Bild 3-1).

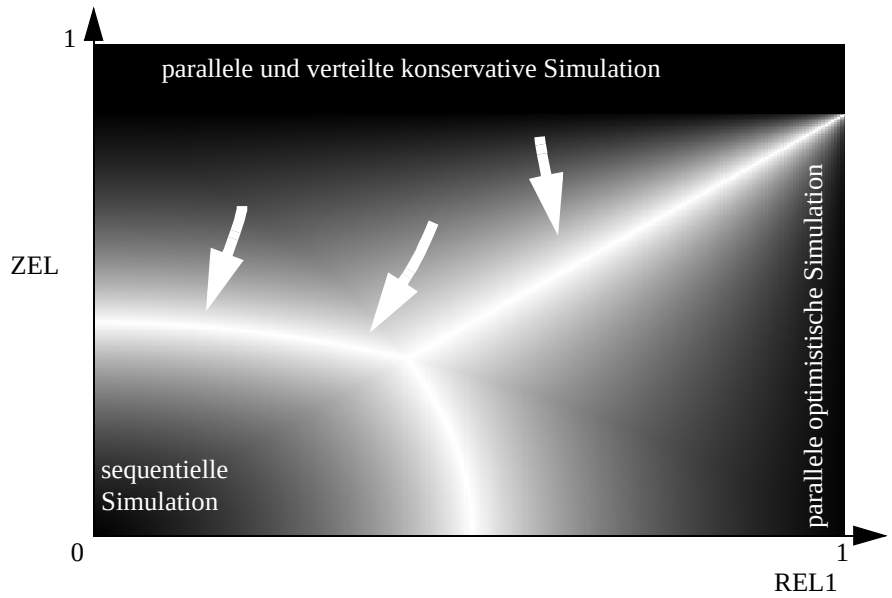


Bild 3-1: Ausdehnung des Bereichs der geeignet konservativ parallel simulierbaren Modelle entsprechend den eingezeichneten Pfeilen

3.1 Besondere Anforderungen an den parallelen Simulator

Die kürzeste Simulationsdauer für ein Modell kann mit einem speziellen, an das jeweilige Modell und seine Parameter angepaßten und entsprechend optimierten Simulator erzielt werden. Zur Erstellung eines solchen Simulators bedarf es einer genauen Kenntnis des zu simulierenden Modells. Da die in Kapitel 2.4.4 definierten Eigenschaften von Simulationsmodellen aber meistens nicht im voraus exakt angegeben werden können, sollte ein Simulator zumindest mit eingeschränkter Effizienz auch Modelle simulieren können, für die er nicht optimiert ist. Ferner ist die Entwicklung eines optimalen Simulators sehr zeitaufwendig. Daher gilt es allgemeinere Simulatoren zu entwickeln, die auf eine ganze Klasse von Problemen angewendet werden können. Außerdem ist es hilfreich, wenn der gleiche Simulator sowohl funktionale als auch statistische Simulation unterstützt, da dadurch beim Einsatz beider Simulationstechniken der Implementierungsaufwand reduziert wird.

Wissenschaftlich verwertbare Ergebnisse müssen reproduzierbar sein. Bei sequentiellen Simulatoren läßt sich diese Forderung auf die Zufallszahlenerzeugung zurückführen, da die deterministische Ereignisbearbeitungsreihenfolge immanent gegeben ist. Bei parallel arbeitenden Simulatoren muß zusätzlich durch geeignete Synchronisationsmaßnahmen dafür gesorgt werden, daß sowohl durch lokale Ereignisse als auch durch Nachrichten von anderen Prozessen erzeugte Ereignisse mit identischem Eintrittszeitpunkt stets in der gleichen Reihenfolge bearbeitet werden.

Das Ziel der nachfolgend vorgestellten speziellen parallelen Simulatorarchitektur ist die beschleunigte Simulationsausführung. Da die Kosten für eine Recheneinheit des Parallelrechners durch Serienfertigung, Verwendung von Standard-Bausteinen, programmierbarer Logik oder ASICs (Application Specific Integrated Circuit) relativ niedrig sind, ist die Anzahl der benötigten Recheneinheiten, sowie eine hohe Auslastung derselben, von untergeordneter Bedeutung.

Eine zukunftsorientierte Simulatorarchitektur darf nicht auf Eigenschaften aktueller Mikroprozessoren aufbauen. Stattdessen muß mit Hilfe einer Hardware-Abstraktionsschicht die Unabhängigkeit der Simulationsprogramme von der aktuell verwendeten Hardware erreicht werden. Nur so lassen sich in Verbindung mit einfachen Schnittstellen und klar definierten Subsystemen neue Prozessorgenerationen, Kommunikationsbausteine und andere Hardware-Elemente transparent einsetzen.

Bei der parallelen Simulation unterschiedlich großer und komplexer Systeme kommt einer guten Skalierbarkeit des Multiprozessorsystems große Bedeutung zu, da in diesen Fällen das System jeweils dem vorhandenen Parallelitätsgrad angepaßt werden kann. Dies gilt sowohl für die Programmierung (einfache und leicht änderbare Festlegung der Prozeßgrenzen) als auch für das Rechnersystem (variable und ausreichende Anzahl an Recheneinheiten), welches meist durch das Kommunikationssystem begrenzt ist. Für heutige Simulationen ist es ausreichend, wenn eine Skalierbarkeit bis zu einigen hundert Recheneinheiten gegeben ist.

Die Parallelisierung auf Parameter- und Teiltestebene wird nicht als gesonderte Anforderung an die Simulatorarchitektur definiert, da sie in trivialer Weise durch Replikation der verwendeten Rechenanlage erreicht werden kann. Beim Einsatz der nachfolgend vorgestellten Architektur ist diese Form der Parallelisierung ebenfalls möglich.

3.2 Systemarchitektur

Den Kern der Simulationsausführung bildet in jedem Knoten des Parallelrechners ein handelsüblicher Mikroprozessor, der im folgenden als „Simulationsprozessor“ (SP) bezeichnet wird. Dadurch wird ein hohes Maß an Flexibilität erreicht. Wie in [Bar83] und [Bar85] gezeigt, überwiegen in ereignisgesteuerten Simulationsprogrammen einfache Befehle zur Listen- und Datenmanipulation[1], wenn man von der Zufallszahlenerzeugung absieht. Daher eignet sich hierfür ein Prozessor, der in der Lage ist, einfache Datenmanipulationen schnell auszuführen. Die Fließkommaleistung des Prozessors ist von untergeordneter Bedeutung, wenn die Zufallszahlenerzeugung, wie nachfolgend vorgestellt, in einem gesonderten Modul erfolgt und alle Simulationszeiten in ganzen Zahlen als Vielfache von Elementarzeiteinheiten angegeben werden (vgl. Kapitel 3.3.1).

Jeder Rechenknoten des Multiprozessorrechners ist für die Bearbeitung genau eines logischen Prozesses der Simulation ausgelegt. Dies stellt keine Einschränkung dar, da die Prozeßgrenzen frei gewählt werden können. Die injektive Prozeßabbildung hat zusammen mit der Auslagerung der Kommunikationsfunktionen den Vorteil, daß nur ein einziger logischer Prozeß auf dem Simulationsprozessor ausgeführt wird. Prozeßwechselzeiten und Interprozeßkommunikation sowie der damit verbundene Mehraufwand entfallen somit. In [Sim95] wird gezeigt, daß dadurch ein signifikanter Geschwindigkeitsvorteil erzielt werden kann. Der Simulationsprozessor wird durch spezielle, simulationsoptimierte Hardware unterstützt, die im wesentlichen

1. Insbesondere für die Ereignisverwaltung und Statistik.

aus einem Zufallszahlen- und aus einem Kommunikationsprozessor (Kapitel 3.3) besteht. Letzterer ist auch für die Prozeßsynchronisation verantwortlich. Neben dem lokalen Speicher können optional weitere Schnittstellen zur lokalen Ein-/Ausgabe vorhanden sein. Den Gesamtaufbau eines Simulationsknotens zeigt Bild 3-2.

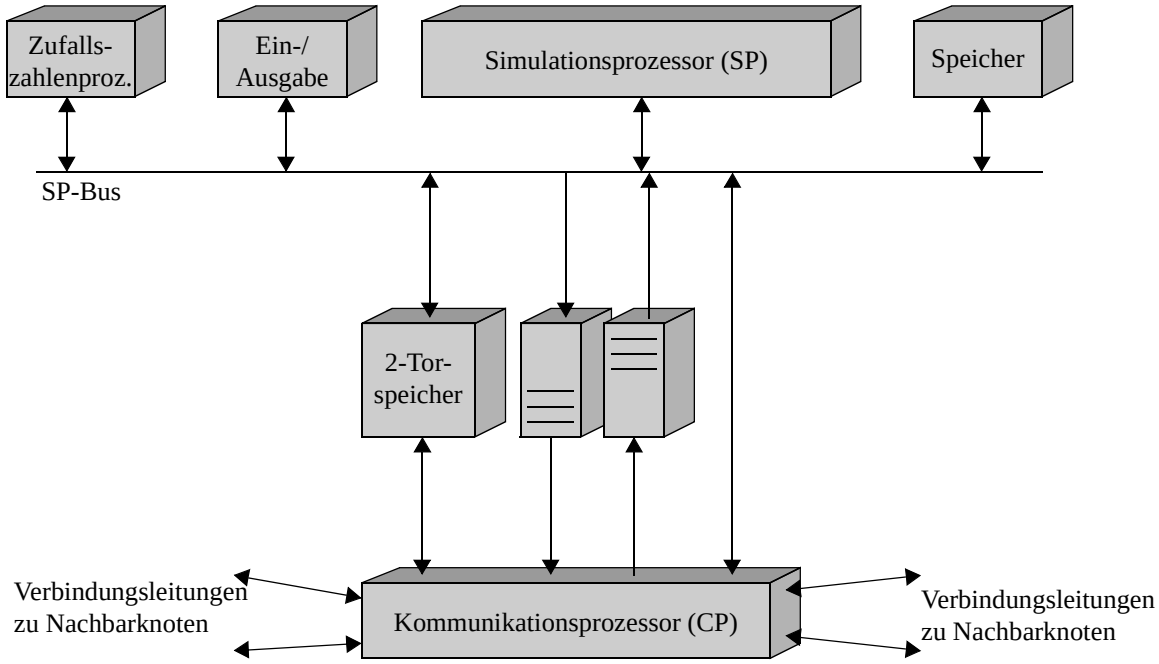


Bild 3-2: Architektur eines Simulationsknotens

Mit den insgesamt vier bidirektionalen physikalischen Kanälen zu Nachbarknoten lassen sich Bus-, Ring-, Binärbaum-, Shuffle- und Manhattan-Street-Topologien beliebiger Größe sowie Hypercubes bis 16 Knoten realisieren. Wie in [KSGL95] ausführlich gezeigt, stellen Bus- und Ring-Topologien Untermengen der Shuffle-Topologie dar. Gleiches gilt für den Binärbaum, wenn ein Knoten des Netzes so konfiguriert wird, daß alle eingehenden Meldungen direkt an einen Nachfolgerknoten weitergegeben werden (Transparentmodus, siehe Kapitel 3.3.1.3). Daher bietet es sich an, auf Grund der Skalierbarkeit und Universalität die Shuffle-Topologie als „Standard“-Topologie zu verwenden und andere Topologien ggf. durch manuelle Rekonfiguration bzw. mit Hilfe der Verkehrslenkungstabellen des Kommunikationsprozessors zu erzeugen. Bild 3-3 zeigt die Shuffle-Netzstruktur des Simulationssystems mit acht Simulationsknoten, wobei die erste Stufe zur Verdeutlichung der Struktur zweimal dargestellt ist.

Der Multiprozessorrechner ist ausschließlich für die Ausführung der Simulation zuständig. Die Initialisierung des Multiprozessorrechners, die Steuerung der Simulation sowie die Simulati-

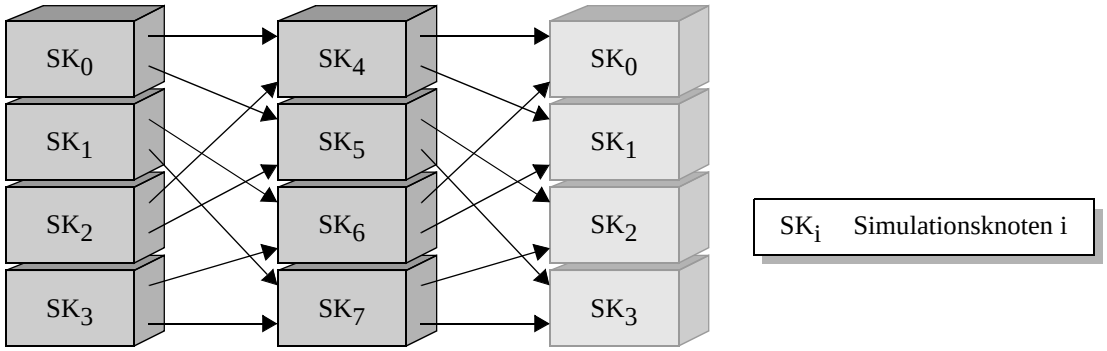


Bild 3-3: Simulationssystem mit acht Simulationsknoten und Shuffle-Netzstruktur

onsdatenauswertung erfolgt durch einen handelsüblichen Rechner (z.B. durch einen Personal Computer), da er keine besonderen Anforderungen erfüllen muß.

Der Anschluß des Steuerrechners an den Multiprozessorrechner kann prinzipiell auf verschiedene Weisen erfolgen. Der Anschluß eines standardisierten Schnittstellenbausteins an den Speicherbus des Simulationsprozessors erfordert den geringsten Entwicklungsaufwand. Dies hat aber den Nachteil, daß sämtliche Nachrichten anderer Simulationsknoten, die über diese Schnittstelle übertragen werden sollen, zunächst an den Knoten mit dem Schnittstellenbaustein adressiert werden müssen. Dort werden sie dann mit Hilfe des Simulationsprozessors an den Schnittstellenbaustein übergeben. Dies führt zu einer unnötigen Mehrbelastung des Simulationsprozessors. Deshalb wird die Schnittstelle für den Steuerrechner in einem physikalischen Kanal zwischen zwei Kommunikationsprozessoren zwischengeschaltet (Bild 3-4). Auf diese Weise können mehrere voneinander unabhängige Steuerrechner angeschlossen werden, was insbesondere dann sinnvoll ist, wenn bereits während der Simulation umfangreiche Datenauswertungen und -darstellungen erfolgen sollen.

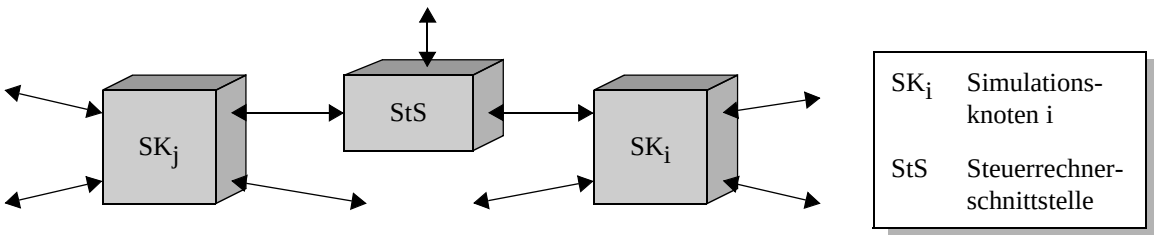


Bild 3-4: Anbindung des Steuerrechners

Über die Steuerrechnerschnittstelle wird der Multiprozessorrechner initialisiert. Zur Erzielung maximaler Flexibilität wird in der Initialisierungsphase durch den Steuerrechner die aktuelle Knotentopologie des Multiprozessorrechners ermittelt [Wag98]. Daher ist eine Topologieände-

rung durch einfaches Umstecken der physikalischen Verbindungskanäle möglich. Da die Ermittlung der Topologie vor dem Laden der Programme des Multiprozessorrechners erfolgen muß, ist es notwendig, daß die Steuerrechnerschnittstelle und die Kommunikationsprozessoren autonom, d.h. ohne Unterstützung durch die Mikroprozessoren des Multiprozessorrechners, betrieben werden können. Nach der Initialisierung werden durch den Steuerrechner das Simulationsmodell auf die vorhandene Multiprozessortopologie abgebildet und entsprechend dieser Abbildung die Simulationsprogramme geladen. Die nach dem Start der Simulation erzeugten Simulationsdaten werden mit Hilfe von Nachrichten zum Steuerrechner übertragen und dort ausgewertet bzw. angezeigt (siehe auch Kapitel 3.5). Für die Übertragung der Simulationsdaten und der Simulationsprogramme sowie für die Ermittlung der Topologie des Multiprozessorrechners wird ein paketorientiertes Kommunikationsprotokoll verwendet (Kapitel 3.3.1.1).

3.3 Spezielle Hardware-Komponenten für die Simulation

Um den Simulationsprozessor möglichst weitgehend zu entlasten, werden geeignete Teile der Simulation nach dem Prinzip der funktionalen Parallelisierung ausgelagert. Dafür bieten sich in erster Linie die Kommunikation (Kapitel 3.3.1.1) und die Prozeßsynchronisation (Kapitel 3.3.1.2) an, die durch die Parallelisierung der Simulationsausführung erforderlich werden. Die allgemeine Auslagerung von Listenoperationen, wie sie in [Leh79a] und [Bar83] vorgenommen wurde, hat sich nach [Klu90] nicht bewährt, weshalb im folgenden dafür keine eigene Hardware vorgesehen ist. Zur Untersuchung des Nutzens einer Hardware-unterstützten Zufallszahlenerzeugung ist ein entsprechendes Modul vorgesehen (Kapitel 3.3.2).

3.3.1 Kommunikationsprozessor

Bei der parallelen und verteilten Simulation besteht eine enge Kopplung zwischen Kommunikation und Synchronisation der logischen Prozesse. Eine Synchronisation muß genau dann erfolgen, wenn zwischen den logischen Prozessen eine Kommunikationsbeziehung besteht. Aufgrund der engen Kopplung wird in dem in dieser Arbeit vorgestellten Simulationsrechner ein gemeinsames, spezielles Modul für diese Aufgaben verwendet.

3.3.1.1 Kommunikationsunterstützung

Sind die räumlichen Ereignislokalitäten nicht groß, so bedeutet dies einen im Verhältnis zum Simulationsaufwand hohen Kommunikationsaufwand. Ohne spezielle Kommunikationsunterstützung wird dann ein signifikanter Anteil der Rechenleistung eines Prozessorknotens für die

Kommunikation verwendet. So konnte z. B. für die in [Wit94] beschriebene Simulation gezeigt werden, daß jeder der Prozessoren, die auch die Simulation ausführen, zur Hälfte mit den Basisdiensten der Interprozeßkommunikation beschäftigt ist.

Bei dem im Rahmen dieser Arbeit entwickelten Simulationsrechner werden daher die Simulationsprozessoren von den Kommunikationsaufgaben weitgehend entlastet, indem diese Funktionalität von speziellen Kommunikationsprozessoren (CP) erbracht wird. Diese kommunizieren über FIFO-Bausteine, einen Bus für Registerzugriffe, einen gemeinsamen Speicher mit Semaphorsteuerung und Unterbrechungsleitungen mit dem jeweils zugeordneten Simulationsprozessor (Bild 3-5).

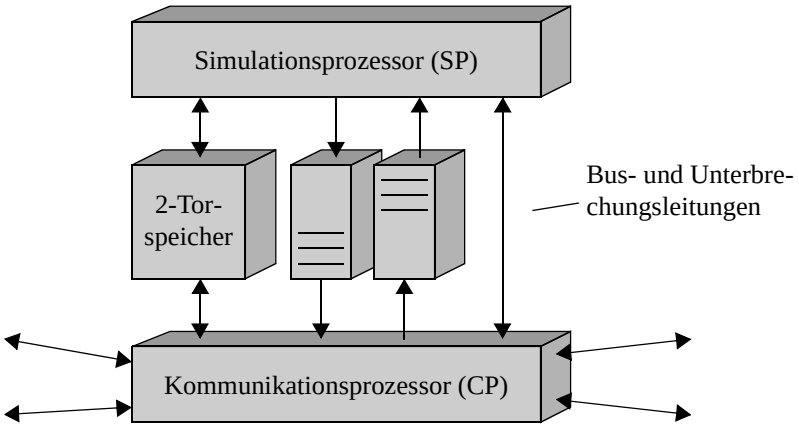


Bild 3-5: Kommunikation zwischen Simulations- und Kommunikationsprozessor

Ein gemeinsame Zweitorspeicher kann von beiden Einheiten beschrieben und gelesen werden. Der Zugriff auf den Speicher wird durch Semaphore gesteuert. Der Kommunikationsprozessor legt in diesen Speicher nach einem Neustart des Rechners den Urlader für den Simulationsprozessor ab und startet diesen anschließend. Da der Zugriff auf den gemeinsamen Speicher durch andere Simulationsknoten bzw. die Steuerrechner ohne Beteiligung des Simulationsprozessors erfolgt, ist der gemeinsame Speicher für die schnelle, simulationszeitunabhänige Kommunikation, für häufig vom Steuerrechner benötigte Daten und für die Fehlersuche geeignet.

Da der Simulationsrechner besonders für Simulationsmodelle mit niedriger räumlicher Ereignislokalität 2 (REL 2) optimiert ist, wird eine direkte Abbildung der Kommunikationsbeziehungen zwischen den logischen Prozessen des Simulationsmodells auf die Topologie des Kommunikationsnetzes des Multiprozessorrechners angestrebt. Dies ist bei einer Vollvermaschung zwischen den Simulationsknoten stets möglich, was jedoch der Anforderung nach Skalierbarkeit widerspricht. Darüber hinaus wird eine weitergehendere Unterstützung der

Kommunikation bei vielen physikalischen Kanälen äußerst komplex und ist nicht mehr sinnvoll realisierbar. Aus diesem Grund wird die Anzahl der Kanäle je Rechenknoten zunächst auf vier bidirektionale Kanäle begrenzt (vgl. Kapitel 3.2). Eine architekturbedingte Obergrenze für die Anzahl der Kanäle je Rechenknoten existiert aber nicht.

Die Paketübertragung zwischen Kommunikations- und Simulationsprozessor erfolgt über große FIFO-Speicher. Da der Kommunikationsprozessor für die Wegesuche, die Flußsteuerung und die Belegung der benötigten Kommunikationsressourcen verantwortlich ist, wird das Versenden von Paketen für den Simulationsprozessor im wesentlichen auf das Einschreiben der Pakete in den FIFO-Speicher reduziert. Ähnliches gilt für den Paketempfang, der ebenfalls über einen FIFO-Speicher abgewickelt wird. Gibt es nur einen logischen Vorgängerprozeß, so bilden die Zeitstempel der empfangenen Pakete eine monotone Folge, d.h. ein Sortieren der Pakete entfällt.

Protokoll für die Kommunikation im Simulationsrechner

Zur Beschleunigung der Kommunikation wird ein spezielles, auf die Belange der verteilten Simulation und den Multiprozessorrechner optimiertes paketorientiertes Kommunikationsprotokoll verwendet. Dadurch ist es möglich, eine schnelle, Hardware-unterstützte Paketübertragung bei gleichzeitiger Integration von Synchronisationsaufgaben zu erzielen.

Der Paketaufbau basiert auf einer Wortbreite von 32 Bit. Ein Paket besteht minimal aus einem und maximal aus achtzehn Datenworten. Alle Pakettypen besitzen einen einheitlichen, 32 Bit langen Paketkopf, der Ziel- und Absenderadresse sowie den Typ des Pakets enthält (Bild 3-6).

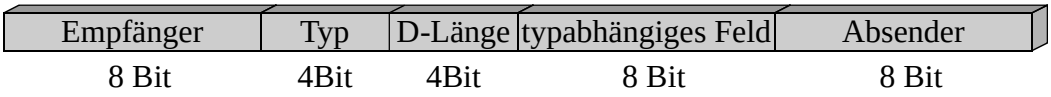


Bild 3-6: Aufbau des Paketkopfes

Das Protokoll ist für die Übertragung von zwei Hauptklassen von Informationen ausgelegt. Konfigurations- und Steuerdaten sind unabhängig von der Simulationszeit, so daß sie keine Zeitinformation benötigen. Im Gegensatz dazu haben zeitbehafteten Nachrichten einen Bezug zur Simulationszeit. Die Klasse der zeitbehaftete Nachrichten wird nochmals unterteilt in Nachrichten für den Informationsaustausch zwischen den Simulationsmodellkomponenten und in Nullnachrichten.

Für die Adressierung werden zunächst 8 Bit lange Adressen verwendet. Bei einem späteren Ausbau ist eine Erweiterung auf mehr Adressen möglich. Den Adressen 0_h und FF_h kommt jeweils eine besondere Bedeutung zu. Unter der Adresse FF_h werden sämtliche Stationen angesprochen (Broadcast). Mit der Adresse 0_h (implizite Adresse) wird stets die das Paket empfangende Station adressiert. Die implizite Adresse wird insbesondere für die Initialisierung des Kommunikationsnetzes benötigt, da sie auch dann verwendet werden kann, wenn einer Station noch gar keine eigene Adresse zugewiesen wurde.

Die Größe der während der Simulation durch Nachrichten übertragenen Datenstrukturen hängt vom Simulationsmodell und von den erzeugten Statistiken ab, weshalb sich eine paketorientierte Übertragung anbietet. Das Kommunikationsprotokoll des Simulationsrechners verwendet vier Bit des Paketkopfes für die Angabe der Datenlänge des Pakets. Da die Länge in Datenworten angegeben wird, können Pakete mit bis zu 15 Nutzdatenworten übertragen werden.

Im Paketkopf stehen vier Bit für die Codierung des Pakettyps zur Verfügung. Aktuell werden die drei Pakettypen CDP, SDP und NMP verwendet, so daß noch Raum für Erweiterungen besteht.

Pakettyp CDP

Konfigurations- und Steuerdaten werden mit CDPs (Control Data Packet) übertragen. Zur Initialisierung des Simulators sind CDPs für die Adreßzuweisung, für das Laden des Simulationsprogramms sowie für das Setzen und Abfragen von Verkehrslenkungstabelle und Zeitverwaltungsregistern vorgesehen. Ferner sind CDPs für die Simulationssteuerung, d.h. zum Starten, Anhalten und Fortsetzen der Simulation sowie für das Überspringen von großen, ereignislosen Zeiträumen (vgl. Kapitel 3.3.1.2) definiert. Das Setzen und die Abfrage von Daten und Ergebnissen der Simulation durch den Steuerrechner erfolgt ebenfalls unter Verwendung von CDPs. Mit Hilfe dieser Befehle können die Simulationsparameter übermittelt sowie akkumulierte und momentane Simulationswerte zum Steuerrechner gesendet und zur Anzeige gebracht werden.

CDPs haben keinen Bezug zur aktuellen Simulationszeit und besitzen daher keinen Zeitstempel. Im Paketkopf wird der jeweilige CDP-Typ codiert. Da die CDP-Bearbeitung typabhängig im Kommunikations- oder im Simulationsprozessor erfolgen muß, bietet es sich an, zur Unterscheidung eine geeignete Codierung zu verwenden, so daß an einem einzigen Bit erkannt wer-

den kann, ob der Kommunikations- oder der Simulationsprozessor für die Bearbeitung der Nachricht zuständig ist. Je nach CDP-Typ werden unterschiedliche Daten oder Parameter benötigt, die direkt nach dem Paketkopf übertragen werden. Eine vollständige Auflistung der CDP-Typen und der zugehörigen Paketstrukturen befindet sich in [KSGL95, Wag98].

Pakettyp SDP

Der Nachrichtenaustausch zwischen den logischen Prozessen der Simulation sowie die Übertragung zeitgebundener Größen der Simulation[2] zum Steuerrechner erfolget jeweils unter Verwendung von SDPs (Simulation Data Packet). Diese Pakete besitzen einen Zeitstempel, der aus einer 64 Bit breiten, nicht negativen Ganzzahl gebildet wird. Der Zeitstempel gibt die Simulationszeit in Vielfachen einer frei wählbaren, elementaren Zeiteinheit an. Der Paketanfang eines SDPs hat somit den in Bild 3-7 dargestellten Aufbau.

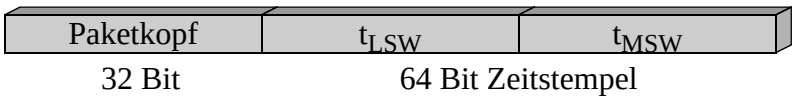


Bild 3-7: Paketkopf und Zeitstempel eines SDPs bzw. Aufbau eines NMPs

Die Struktur der zwischen den logischen Prozessen der Simulation verschickten Daten wird durch das Protokoll nicht eingeschränkt. Es können beliebige Daten in der jeweils geeigneten Form transparent übertragen werden.

Prinzipiell ist es möglich, daß auf einem Simulationsknoten mehrere Komponenten des Simulationsmodells simuliert werden. Bei der Simulation eines lokalen Netzes können beispielsweise zwei oder mehr Stationen auf einem Simulationsknoten simuliert werden. Dann ist es sinnvoll, daß gleiche Datenobjekte der Stationen auch die gleiche Kennung aufweisen. Für die Zuordnung der Pakete zu den einzelnen Komponenten wird im Paketkopf eine Subadresse übertragen. Diese hat auf die Verkehrslenkung im eigentlichen Kommunikationsnetz keinen Einfluß, sondern dient ausschließlich der lokalen Zuordnung innerhalb eines Knotens.

Pakettyp NMP

Als Simulationsverfahren wird ein konservatives paralleles Verfahren mit Nullnachrichten verwendet. Zur Übertragung dieser Nullnachrichten sind im Kommunikationsprotokoll NMPs (Null Message Packet) vorgesehen. Sie enthalten neben dem Zeitstempel keine weiteren Nutz-

2. z.B. Länge einer Warteschlange

daten und sind somit im Prinzip SDPs ohne Daten (siehe Bild 3-7). Da sie jedoch logisch von SDPs verschieden sind und anders bearbeitet werden müssen,[3] wird ihnen ein eigener Pakettyp zugeordnet.

3.3.1.2 Synchronisationsunterstützung

Die Synchronisation der Simulationsprozesse gliedert sich in eine lokale und eine globale Synchronisation. Die lokale Synchronisation entscheidet über das Bearbeiten von lokalen Ereignissen, d.h. ob das nächste lokal bekannte Ereignis nach den Regeln der konservativen parallelen Simulation bearbeitet werden darf oder nicht. Die globale Synchronisation sichert das Fortschreiten der Simulation und verhindert Verklemmungen. Bei der konservativen parallelen Simulation wird dies durch die geeignete Erzeugung und Bearbeitung von Nullnachrichten erzielt.

Globale Synchronisation

Wie in Kapitel 2.2.3.1 beschrieben, stellt das Versenden von Nullnachrichten eine Möglichkeit dar, Verklemmungen zu vermeiden bzw. aufzulösen. Wird die lokale Simulationszeit nach jeder Änderung mit Hilfe von Nullnachrichten an alle logischen Nachfolger geschickt, die noch keine Nachricht mit diesem Zeitstempel erhalten haben, so kann sich dies signifikant auf den Kommunikationsaufwand auswirken. Insbesondere ist es möglich, daß viele dieser Nullnachrichten unter Umständen überhaupt nicht notwendig gewesen wären und nur unnötigen Bearbeitungsaufwand erfordern. Andererseits wird aber die Simulationsgeschwindigkeit nicht beeinträchtigt, wenn momentan unbenutzte Kommunikationsressourcen Nullnachrichten bearbeiten. Werden in einem wenig ausgelasteten Kanal zusätzliche Nullnachrichten übertragen, so beeinträchtigt dies die anderen Nachrichten nur wenig. Durch das häufige Erhöhen der Kanalzeit kann aber die Blockierungsdauer des nachfolgenden Prozesses unter Umständen deutlich reduziert werden.

Die globale Synchronisation wird in der vorgestellten Simulatorarchitektur dadurch unterstützt, daß der Kommunikationsprozessor unabhängig vom Simulationsprozessor Nullnachrichten erzeugt, überträgt und auswertet. Durch spezielle Hardware wird parallel zur Simulation für jeden physikalischen Ausgangskanal unabhängig geprüft, ob durch das Versenden einer Nullnachricht die Kanalzeit erhöht werden kann oder ob bereits eine Nachricht mit

3. NMPs können vom Kommunikationsprozessor autonom bearbeitet werden, SDPs werden durch den Kommunikations- und den Simulationsprozessor bearbeitet.

dem aktuellen Zeitstempel verschickt wurde.[4] Ist eine Erhöhung der Kanalzeit möglich und stehen für diesen physikalischen Kanal keine weiteren Übertragungsanforderungen an, wird eine Nullnachricht erzeugt und verschickt. Beim Empfang einer Nullnachricht durch den Kommunikationsprozessor wird der Zeitstempel gespeichert und für die lokale Synchronisation zur Verfügung gestellt, die ebenfalls durch den Kommunikationsprozessor vorgenommen wird. Somit erfolgt die globale Synchronisation vollständig parallel zur Simulationsausführung ohne Interaktion mit dem Simulationsprozessor.

Lokale Synchronisation

Vor jeder Ereignisbearbeitung durch den Simulationsprozessor muß geprüft werden, ob das nächste zu bearbeitende Ereignis sicher ist, d.h. ob sein Eintrittszeitpunkt kleiner ist als alle Kanalzeiten der Kanäle zu den logischen Vorgängern. Der Aufwand, der für die lokale Synchronisation erbracht werden muß, wächst mit der Anzahl der Eingangskanäle. Für einen logischen Prozeß mit vier Eingangskanälen sind beispielsweise insgesamt vier Zeitstempelvergleiche notwendig, die sich wieder aus mehreren einzelnen Vergleichen zusammensetzt, wenn der Zeitstempel breiter ist als der verwendete Vergleicher.[5] Bei Simulationsmodellen, die einen Blick in die Zukunft (Lookahead) aufweisen, muß für die lokale Synchronisation die Verzögerungszeit berücksichtigt werden. Dies geschieht dadurch, daß das durch den Empfang einer Nachricht erzeugte Ereignis als Eintrittszeitpunkt die Summe aus Verzögerungszeit und Nachrichtenzeitstempel erhält.

Da der Kommunikationsprozessor sämtliche Kanalzeiten kennt, bietet es sich an, daß er auch die Addition der Verzögerungszeiten und den Zeitvergleich vornimmt. Als Resultat wird dem Simulationsprozessor mitgeteilt, ob das nächste lokale Ereignis bearbeitet werden kann oder nicht. Die Mitteilung kann mittels einer Unterbrechungsanforderung oder durch Setzen eines Statusregisters erfolgen. Da der Kommunikationsprozessor für die Synchronisation die Addition von Zeitstempel und Kanalverzögerungszeit ohnehin durchführen muß, werden die Pakete an den Simulationsprozessor zu dessen Entlastung mit dem bereits aktualisierten Zeitstempel weitergegeben.

---

4. Der Zeitvergleich zwischen der Kanalzeit und der Simulationszeit t_0 kann wahlweise zu Beginn oder nach der Bearbeitung sämtlicher Ereignisse mit dem Zeitstempel t_0 erfolgen. Findet er zu Beginn statt, wird bei freien Übertragungskapazitäten zu jedem Ereigniszeitpunkt eine Nullnachricht generiert, bei einem Vergleich am Ende dagegen nur, wenn zu dem betrachteten Zeitpunkt nicht ohnehin eine Nachricht verschickt wurde. Der optimale Vergleichszeitpunkt hängt vom jeweiligen Simulationsmodell ab. Sind die physikalischen Kommunikationskanäle nicht ausgelastet, so ist das frühzeitige Versenden der Nullnachrichten vorzuziehen.
5. Z.B. bei Verwendung eines 32 Bit Mikroprozessors und einem 64 Bit Zeitstempel.

Optimierungen

Bei der konservativen verteilten Simulation stellen zyklische Abhängigkeiten mit niedriger zeitlicher Ereignislokalität ein Problem dar, da in solchen Zyklen für den Fortgang der Simulation sehr viele Nullnachrichten verschickt werden müssen. Bild 3-8 veranschaulicht dies an einem Beispiel mit zwei logischen Prozessen. Um die Kanalzeit (und damit auch die lokalen Simulationszeiten) um 100 Zeiteinheiten zu erhöhen, müssen in diesem Beispiel 200 Nullnachrichten verschickt werden. Treten in dem Simulationsmodell häufig solche Verhältnisse auf, ist keine effiziente Simulation möglich. Um dem entgegenzuwirken, werden in dem speziellen Simulationsrechner zwei weitere Synchronisationsmechanismen verwendet: Die schnelle Nullnachrichtenschleife (SNNS) und der Zeitsprung.

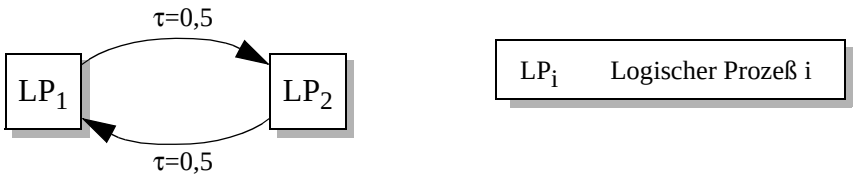


Bild 3-8: Beispiel für eine zyklische Abhängigkeit von logischen Prozessen

Mit Hilfe des Kommunikationsprozessors werden automatisch Nullnachrichten erzeugt, verschickt und beim Empfang bearbeitet, doch die Erhöhung der lokalen Simulationszeit kann ohne die schnelle Nullnachrichtenschleife nur durch den Simulationsprozessor erfolgen. Dies bedeutet, daß in Wartesitutationen, in denen der Simulationsprozessor nicht simulieren kann, der Simulationsprozessor mit jeder empfangenen Nullnachricht vom Kommunikationsprozessor aufgefordert wird, die lokale Uhr zu aktualisieren. Erst danach werden automatisch neue Nullnachrichten generiert. Dies läßt sich entscheidend beschleunigen, wenn der Kommunikationsprozessor direkt die Simulationszeit erhöht. Um Zugriffskonflikte zu vermeiden und die Konsistenz zu wahren, muß der Simulationsprozessor dem Kommunikationsprozessor mitteilen, daß er aktuell nicht simuliert. Dies geschieht über ein Register des Kommunikationsprozessors. Somit kann der Kommunikationsprozessor direkt nach dem Empfang einer Nachricht mit Zeitstempel eine aktualisierte Nullnachricht verschicken. Zur Vermeidung von Kausalitätsverletzungen muß dieser Modus beim Empfang eines SDPs wieder verlassen werden. Da während der schnellen Nullnachrichtenschleife die gesamte Zeitbearbeitung durch dedizierte Hardware des Kommunikationsprozessor erfolgt, können Blockierungen, die mit wenigen Nullnachrichten auflösbar sind, in sehr kurzer Zeit aufgelöst werden.

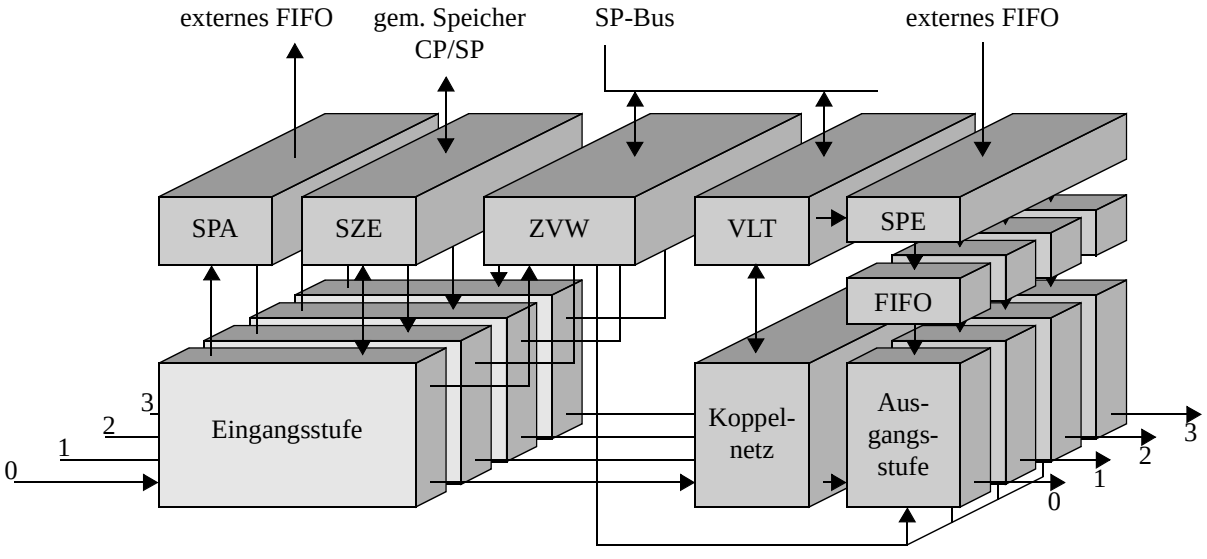
Die schnelle Nullnachrichtenschleife reduziert nicht die Anzahl der erzeugten und verschickten Nullnachrichten, sondern sie beschleunigt nur ihre Bearbeitung. Sie eignet sich daher nicht für das Überbrücken von sehr großen Zeiträumen. Dafür ist der sogenannte Zeitsprung vorgesehen. Tritt bei einem logischen Prozeß eine sehr große Differenz zwischen der minimalen Kanalzeit und der Zeit des nächsten lokal bekannten Ereignisses $t_{E,i}$ auf, so sendet er eine Nachricht entlang eines Hamilton-Pfads an alle anderen logischen Prozesse (Rundsendung), die seine nächste Ereigniszeit enthält. Wird die Nachricht von einem anderen logischen Prozeß empfangen, der simulieren kann und somit nicht im Wartezustand ist, wird die Nachricht von ihm verworfen. Andernfalls prüft er zunächst, ob die Nachricht noch aktuell ist, d.h. ob seine lokale Uhr nicht bereits den angeforderten Zeitpunkt überschritten hat.[6] Ist die angeforderte Zeit noch aktuell, schickt der logische Prozeß abhängig davon, ob sein nächstes, lokal bekanntes Ereignis einen größeren oder einen kleineren Zeitstempel aufweist, entweder die Nachricht unverändert weiter oder er generiert eine eigene Zeitsprunganforderungsnachricht. Durch dieses Vorgehen ist sichergestellt, daß ein logischer Prozeß, der seine eigene Zeitsprunganforderungsnachricht empfängt, gefahrlos seine lokale Simulationszeit auf die Zeit des nächsten lokal bekannten Ereignisses erhöhen kann, da dieses Ereignis den kleinsten Zeitstempel der noch unbearbeiteten Ereignisse im System besitzt.

Da die Möglichkeit zum Zeitsprung in Simulationsmodellen, die sich für die konservative verteilte Simulation eignen, nur relativ selten gegeben ist, wird dafür keine Hardware-Unterstützung vorgesehen. Die Bearbeitung erfolgt durch den Simulationsprozessor, der in solchen Situationen ohnehin nicht mit der Simulation beschäftigt ist.

3.3.1.3 Architektur des Kommunikationsprozessors

Die in den Kapiteln 3.3.1.1 und 3.3.1.2 beschriebenen Unterstützungsfunktionen und Optimierungen der parallelen und verteilten Simulation werden im wesentlichen durch den Kommunikationsprozessor zur Verfügung gestellt. Seine Architektur ist in Bild 3-9 dargestellt. Sie basiert auf voneinander weitgehend unabhängig arbeitenden Modulen. Dadurch kann, solange kein Ressourcenkonflikt besteht, ein hoher Parallelitätsgrad bei der Kommunikation und Synchronisation erzielt werden. Nachfolgend werden die wichtigsten Module vorgestellt.

6. Existieren abseits des Hamilton-Pfads weitere Kommunikationskanäle, so müssen für die Bestimmung des nächsten lokal zu bearbeitenden Ereignisses die in Transit befindlichen Nachrichten berücksichtigt werden. Dazu werden Algorithmen benötigt, wie sie bei der optimistischen Simulation für die GVT-Bestimmung Verwendung finden (z.B. [Mat93, DFW94]).



ZVW:	Zeitverwaltung	VLT:	Verkehrslenkungstabelle
SPA:	SP-Ausgang	SPE:	SP-Eingang
SZE:	Speicherzugriffseinheit für gem. Speicher CP/SP	FIFO:	FIFO-Speicher

Bild 3-9: Vereinfachte Architektur des Kommunikationsprozessors

Eingangsstufen

Innerhalb des Kommunikationsprozessors werden die vier bidirektionalen physikalischen Kanäle jeweils in unidirektionale Ein- und Ausgängskanäle aufgespalten. Die ankommenden Pakete werden in der entsprechenden Eingangsstufe dekodiert. Ist das Paket nicht für die betrachtete Station bestimmt, so wird es unverändert gemäß der Verkehrslenkungstabelle über die entsprechende Ausgangsstufe ausgegeben[7]. Die Behandlung der an die Station adressierten Pakete hängt vom Pakettyp ab. Besitzt ein Paket einen Zeitstempel (NMP, SDP), so wird dieser nach Addition der Verzögerungszeit durch die Eingangsstufe an die Zeitverwaltung (ZVW) zur Kanalzeitbestimmung und Synchronisation weitergereicht. Da der einzige Zweck von Nullnachrichten (NMP) die Erhöhung der Kanalzeit ist, können diese anschließend verworfen werden. Simulationsdaten werden über den Ausgang zum Simulationsprozessor (SPA), der im wesentlichen eine Zugriffsregelung realisiert, mit aktualisiertem Zeitstempel in einen gemeinsamen FIFO-Speicher geschrieben, der vom Simulationsprozessor gelesen wird. Wird in einer Simulation mehr als ein Kanal für die Simulationsdatenübertragung genutzt, muß der FIFO-

7. Die Architektur läßt auch zu, daß ein Paket gleichzeitig über mehrere Ausgangsstufen ausgegeben wird. Dies wird aber in der Regel nicht benutzt.

Speicher vom Simulationsprozessor vor der Ereignisbearbeitung vollständig geleert werden, da die Pakete dann nicht mehr notwendigerweise in chronologischer Reihenfolge vorliegen.

Die Bearbeitung der Steuerpakete (CDP) hängt von ihrem jeweiligen Typ ab. Pakete, die nicht vom Kommunikationsprozessor bearbeitet werden können,[8] werden wie die Simulationsdaten über das Modul SPA an den Simulationsprozessor ausgegeben. Die Bearbeitung von Steuerpaketen, die den Simulationsprozessor nicht benötigen, erfolgt in der Eingangsstufe. Dazu werden die Daten des Pakets ggf. entsprechend ihrer Bestimmung in ein internes Register des Kommunikationsprozessors übernommen[9] oder über die Speicherzugriffseinheit (SZE) nach Anforderung des zugehörigen Semaphors in den gemeinsamen Speicher bzw. in die Verkehrslenkungstabelle geschrieben. Muß ein Antwortpaket generiert werden, so erfolgt dies ebenfalls in der Eingangsstufe. Das erzeugte Paket wird dann, wie andere Pakete auch, über das Koppelnetz an die Ausgangsstufe weitergegeben.

Die Eingangsstufe eines Simulationsknotens kann so konfiguriert werden, daß alle Pakete direkt an einen definierbaren Nachfolger weitergereicht werden (Transparentmodus). Aus der Sicht der Simulation ist dieser Knoten somit nicht mehr vorhanden. Der Transparentmodus wird für die Simulation von Modellen verwendet, die nur eine Untermenge der vorhandenen Simulationsknoten benötigen. Besitzt beispielsweise das Verbindungsnetz des Multiprozessorrechners nur Zyklen mit einer geraden Anzahl an Knoten, das Simulationsmodell erfordert aber einen Zyklus mit einer ungeraden Anzahl an Knoten, so kann dennoch eine Abbildung der zu simulierenden Topologie auf die Topologie des Multiprozessorrechners erfolgen, wenn zusätzliche Knoten im Transparentmodus hinzugefügt werden.

Eingangsstufe für den Simulationsprozessor

Pakete, die der Simulationsprozessor erzeugt, werden von diesem in einen externen, beliebig großen FIFO-Speicher geschrieben. Sie werden von dort durch das Modul SPE ausgelesen und entsprechend der Verkehrslenkungstabelle in einen der vier internen FIFO-Speicher geschrieben. Jeder dieser internen Speicher kann genau ein Paket aufnehmen. Dadurch wird sichergestellt, daß nur vollständige Pakete in das Kommunikationsnetz gelangen und Verklemmungen aufgrund unvollständiger Pakete vermieden werden.

8. Z.B. Zeitsprunganforderungen, Setzen- und Abfragen von Objekten der Simulation.
9. Z.B. Zuweisung einer Stationsadresse.

Ausgangsstufe

Für jeden physikalischen Ausgangskanal ist eine separate Ausgangsstufe vorgesehen. Die Aufgabe der Ausgangsstufe ist das Multiplexen der Pakete vom Koppelnetz und der Pakete des Simulationsprozessors sowie die globale Synchronisation durch automatisches Erzeugen von Nullnachrichten. Dazu besitzt die Ausgangsstufe eine Architektur gemäß Bild 3-10.

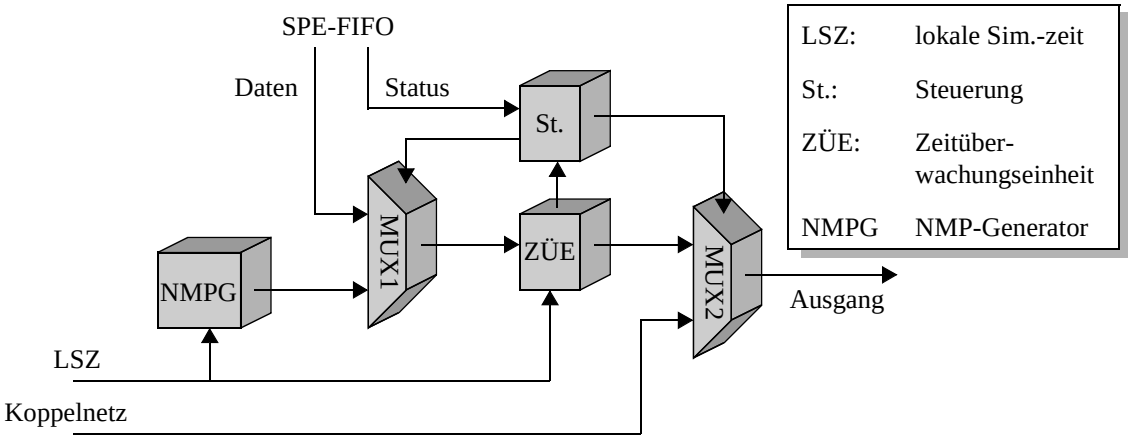


Bild 3-10: Architektur der Ausgangsstufe

Die Zeitüberwachungseinheit (ZÜE) ermittelt aus den lokal erzeugten und verschickten Paketen die Kanalzeit des Ausgangskanals.[10] Diese wird von der ZÜE mit lokalen Simulationszeit verglichen. Ist die lokale Simulationszeit größer, wird dies der Steuerung mitgeteilt, die daraufhin das Aussenden einer Nullnachricht veranlaßt, falls keine anderen Übertragungsanforderungen bestehen. Da bei Gleichheit von Simulations- und Kanalzeit keine Nullnachricht verschickt wird, ist sichergestellt, daß nicht wiederholt identische Nullnachrichten generiert werden.

Verkehrslenkungstabelle

Die Verkehrslenkungstabelle enthält die Zuordnung zwischen Adressen und Ausgangsleitungen. Auf sie kann sowohl der Simulations- als auch der Kommunikationsprozessor zugreifen. Daher wird ein Semaphor für die Zugriffsregelung benötigt. Damit dieser nicht bei jedem Paket neu angefordert werden muß, hält normalerweise der Kommunikationsprozessor den Semaphor. Der Semaphor wird vom Kommunikationsprozessor nur dann abgegeben, wenn er vom Simulationsprozessor angefordert wird. Zugriffe des Simulationsprozessors sind in der

10. Die Pakete vom Koppelnetz können dabei unberücksichtigt bleiben, da die Eingangsstufe lokal nur Steuerpakete erzeugt, die keinen Zeitstempel haben.

Regel weder bei der Initialisierung noch bei der Simulation notwendig, da die Verkehrslenkungstabelle vom Kommunikationsprozessor entsprechend empfangener Steuermeldungen gesetzt werden kann. Unter gewissen Umständen kann es jedoch für das Simulationsprogramm vorteilhaft sein, die Tabelle auszulesen und zu modifizieren.[11] Diese Operationen sind, da sie nur selten vorkommen, nicht zeitkritisch.

Zeitverwaltung

Das Modul Zeitverwaltung (ZVW) enthält die Logik für die lokale Synchronisation, d.h. im Modul Zeitverwaltung wird entschieden, ob das nächste, lokal bekannte Ereignis simuliert werden kann oder nicht. Dazu ist ein Vergleich mit sämtlichen Kanalzeiten notwendig. Die Architektur dieses Moduls ist vereinfacht in Bild 3-11 dargestellt.

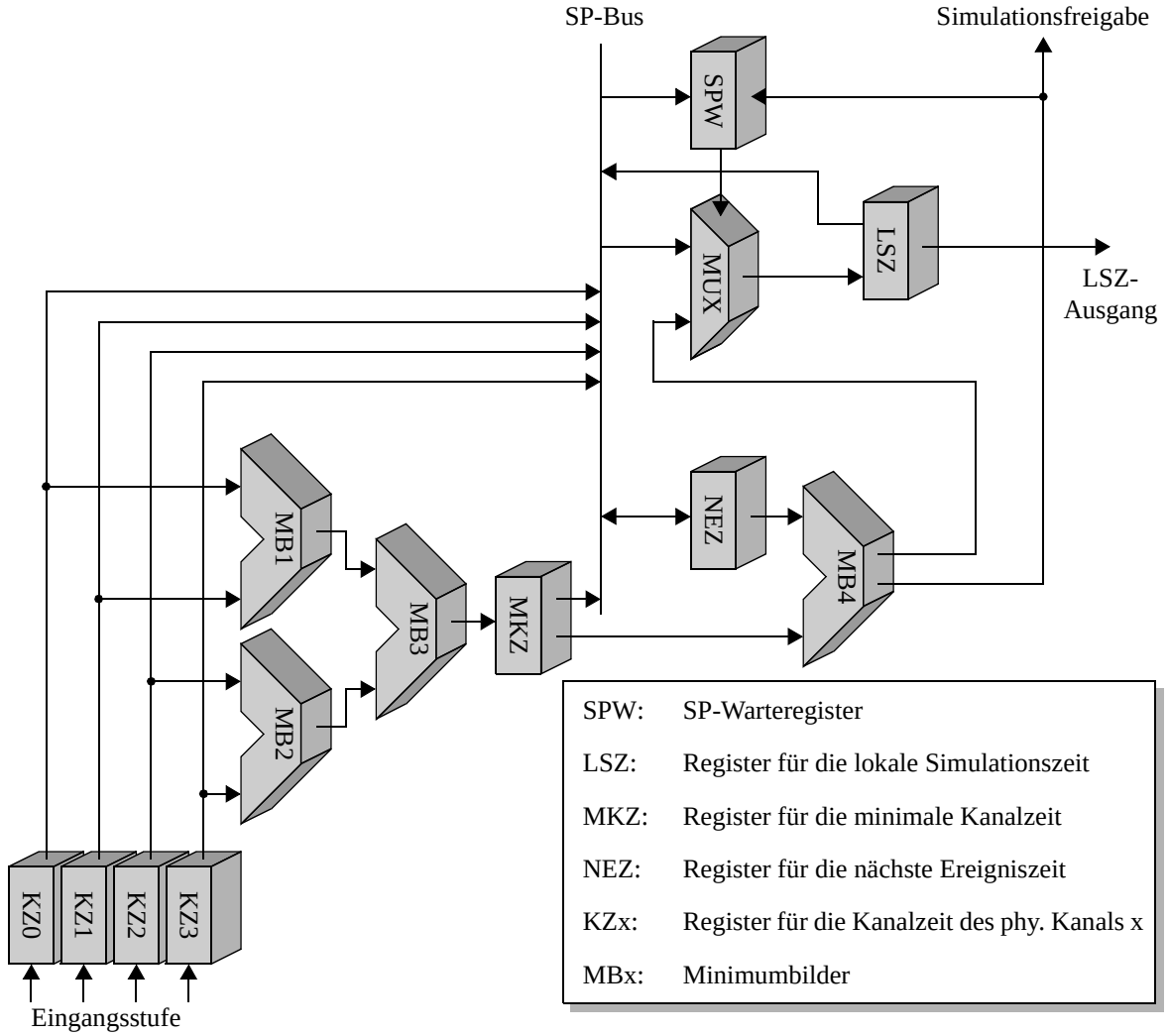


Bild 3-11: Architektur der Zeitverwaltungseinheit

11. z.B. bei der Simulation von rekonfigurierenden Netzen

Die von der Eingangsstufe gemeldeten Kanalzeiten werden in der Zeitverwaltungsstufe gespeichert und können durch den Simulationsprozessor abgefragt werden. Die Minimumbilder MB1 bis MB3 berechnen das Minimum der Kanalzeiten, das über das Register MKZ vom Simulationsprozessor gelesen werden kann, und zum anderen im Minimumbilder MB4 mit dem Zeitstempel des nächsten lokal bekannten Ereignisses verglichen wird. Ist die nächste Ereigniszeit (NEZ) kleiner als das Minimum der Kanalzeiten, so wird die Simulation freigegeben, andernfalls wird, sofern der Simulationsprozessor durch das Register Simulationsprozessorwarteregister (SPW) seinen Wartezustand angezeigt hat, die Kanalzeit als neue lokale Simulationszeit gesetzt.[12] Diese direkte Erhöhung kann so lange erfolgen, bis die Simulation wieder freigegeben wird. Daher muß mit diesem Signal das SPW-Register zurückgesetzt werden.

3.3.2 Zufallszahlenprozessor

Die statistische parallele und verteilte Simulation wird überwiegend für sehr komplexe und ereignisreiche Systeme verwendet, die eine Vielzahl von Zufallszahlen mit unterschiedlichen Verteilungen benötigen. Einer Ableitung dieser Zufallszahlen aus zufälligen Prozessen wie radioaktivem Zerfall oder elektrischem Rauschen steht die Forderung nach der Wiederholbarkeit von Simulationen entgegen. Daher wird die Zufallsfolge in der Regel pseudozufällig durch fest vorgegebene Algorithmen erzeugt [Säg85]. Dabei ist es von Bedeutung, daß die Periode der verwendeten Pseudozufallszahlengeneratoren sehr groß ist, so daß sich die Zufallszahlen innerhalb einer Simulation nicht wiederholen. Dies stellt insbesondere für parallele Simulatoren eine Herausforderung dar, da im Verhältnis zur sequentiellen Simulation längere Zeiträume simuliert werden können. Ferner dürfen die von den verschiedenen Prozessoren erzeugten Zufallszahlen nicht zu einer Korrelation von Ereignissen führen. Ein solcher Generator, der aus einer Kombination von zwei rekursiven Pseudozufallszahlengeneratoren besteht, wurde in [L'E96] vorgestellt. Er hat sich als geeignet für die parallele Simulation erwiesen und wird z.B. auch in [Ne98] verwendet.

Der im Rahmen dieser Arbeit entwickelte Simulationsrechner verwendet einen Zufallszahlenprozessor, dessen Aufbau in Bild 3-12 vereinfacht dargestellt ist. Den Kern bildet ein Modul, das auf dem in [L'E96] vorgeschlagenen Pseudozufallszahlenalgorithmus basiert (GVZZ). Damit wird es möglich, auf wiederholbare Weise mit hoher Geschwindigkeit gleichverteilte

12. Schnelle Nullnachrichtenschleife (SNNS)

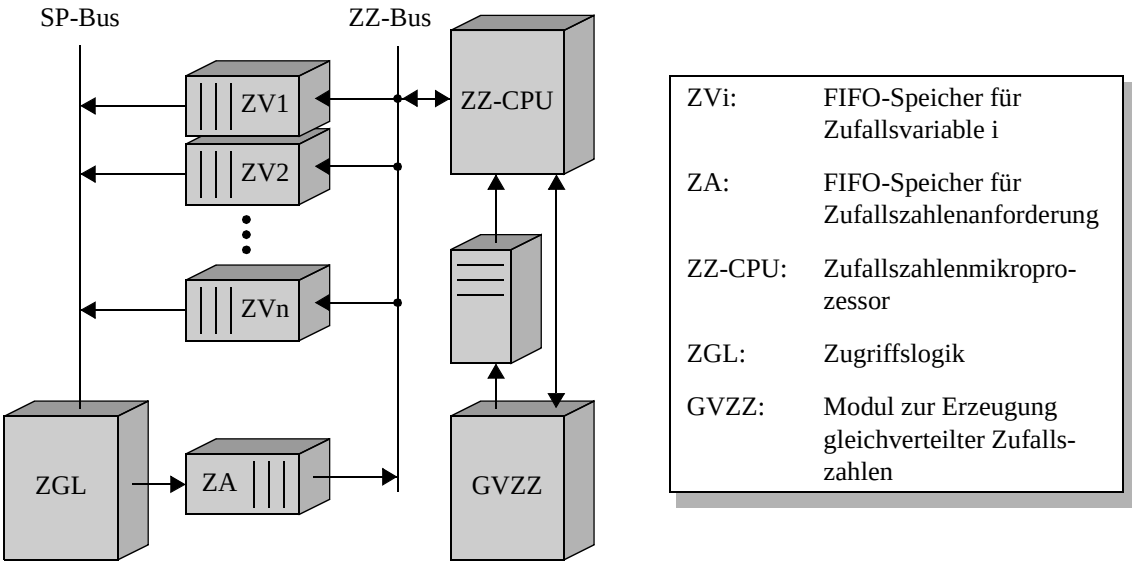


Bild 3-12: Architektur des Zufallszahlenprozessors

Pseudozufallszahlen zwischen 0 und 1 zu erzeugen. Die Startwerte für das Modul GVZZ werden von der ZZ-CPU generiert und können aus der Knotenadresse oder beliebigen Simulationsparametern abgeleitet werden.

Die von dem Modul GVZZ erzeugten Zufallszahlen werden in einem FIFO-Speicher abgelegt, mit Hilfe des Zufallszahlenmikroprozessors (ZZ-CPU) in die von der Simulation benötigten Verteilungen transformiert und getrennt nach Verteilungsfunktionen in FIFO-Speichern abgelegt. Diese FIFO-Speicher werden zu Beginn der Simulation vollständig gefüllt. Der Simulationsprozessor kann innerhalb eines Buszyklus durch einen Zugriff auf den entsprechenden FIFO-Speicher eine Zufallszahl auslesen. Er wird dadurch vollständig von der Zufallszahlenerzeugung entlastet. Damit die FIFO-Speicher in reproduzierbarer Weise aufgefüllt werden, generiert das Modul Zugriffslogik (ZGL) für jede gelesene Zufallszahl eine Zufallszahlenanforderung, die durch einen FIFO-Speicher gepuffert an den Zufallszahlenmikroprozessor weitergegeben wird. Dadurch ist sichergestellt, daß die Zuordnung der Zufallszahlen zu den verschiedenen Verteilungen sowie ihre Reihenfolge ausschließlich von der Lesereihenfolge des Simulationsprozessors abhängt.[13] Die Zugriffslogik wacht auch darüber, daß aufgrund leerer FIFO-Speicher keine ungültigen Zufallszahlen gelesen werden, indem sie ggf. in den Buszyklus des Simulationsprozessors Wartezyklen einfügt.

13. Durch diese Vorgehensweise ist die Wiederholbarkeit der Simulation gewährleistet.

3.4 Software-Architektur für den Multiprozessorrechner

Der Multiprozessorrechner ist ausschließlich für die Simulationsausführung zuständig. Die Darstellung der Simulationsergebnisse sowie die Benutzerinteraktionen erfolgen mit Hilfe des Steuerrechners und sind deshalb hier ohne Belang.

An die Software-Architektur des Multiprozessorrechners werden sämtliche Anforderungen gestellt, die auch für ein sequentielles Simulationssystem von Bedeutung sind [Ko94].[14] Hinzu kommt die Anforderung, daß durch die Parallelisierung kein signifikanter Mehraufwand bei der Simulationsprogrammerstellung entsteht. Für den Multiprozessorrechner wurde daher eine Software-Architektur entsprechend Bild 3-13 entwickelt, die auf drei sich gegenseitig ergänzenden Bibliotheken basiert.

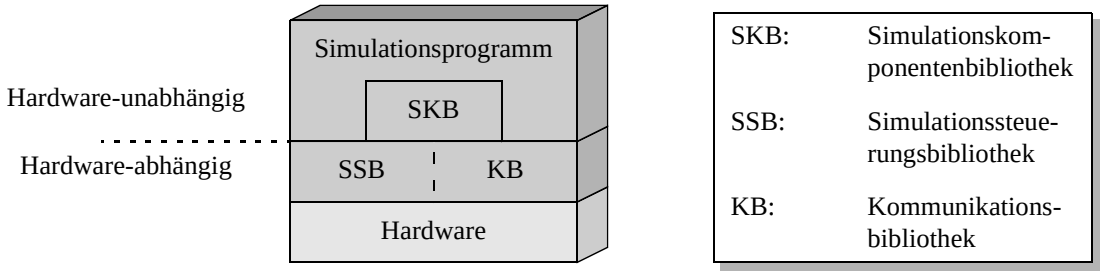


Bild 3-13: Software-Architektur für den Multiprozessorrechner

Der Algorithmus für die Ablaufsteuerung der verteilten Simulation ist problemunabhänig und läßt sich zusammen mit weiteren Simulationshilfsfunktionen in der Simulationssteuerungsbibliothek SSB zusammenfassen. Da Teile der Simulationssteuerung durch spezielle Hardware im Kommunikationsprozessor erfolgen, ist die Simulationssteuerungsbibliothek abhängig von der Hardware. Wird z.B. aus Aufwandsgründen ein vereinfachter Kommunikationsprozessor eingesetzt, so müssen die nicht realisierten Funktionalitäten durch die SSB und damit vom Simulationsprozessor erbracht werden. Die Schnittstelle der SSB ist so gestaltet, daß Hardware-Änderungen keinen Änderungsbedarf für das Simulationsprogramm nach sich ziehen. Gleiches gilt für die Kommunikationsbibliothek, die aufgrund der gegenseitigen Abhängigkeiten eng mit der Simulationssteuerungsbibliothek gekoppelt ist. Sie stellt dem Simulationsprogramm eine einfache, Hardware-unabhängige Kommunikationsschnittstelle zur Verfügung. Auch die Kommunikationsdienste werden gemeinsam von der Kommunikationsbibliothek und vom Kommunikationsprozessor erbracht. Die Simulationssteuerungbibliothek und die Kom-

14. Portabilität, Wiederverwendbarkeit von Simulationskomponenten, Eignung für komplexe Systeme,

munikationsbibliothek sind nahezu unabhängig vom verwendeten Simulationsprozessor. Sie können daher unabhängig vom Simulationsprozessor in einer beliebigen Hochsprache implementiert werden.

Die Hardware-unabhängige und damit portable Simulationskomponentenbibliothek (SKB) enthält häufig verwendete Modellkomponenten wie z.B. Warteschlangen und Verkehrsgeneratoren sowie Module für die Erfassung von Kenngrößen der Simulation für statistische Auswertungen. Mit Hilfe der Komponentenbibliothek lassen sich im allgemeinen große Teile von Simulationsmodellen einfach durch Zusammenfügen von geprüften und validierten Basiskomponenten erstellen. Sie enthält darüberhinaus Komponenten für die Interprozeßkommunikation, die aufbauend auf der Kommunikationsbibliothek die Verknüpfung von Modellkomponenten über die Grenzen logischer Prozesse hinweg erlauben. Bild 3-14 zeigt ein Beispiel. Ein Generator schreibt Nachrichten in einen begrenzten FIFO-Speicher. Dort werden sie von einer Bedieneinheit ausgelesen und mit Hilfe von Kommunikationskomponenten vom logischen Prozeß 1 an den logischen Prozeß 2 weitergegeben. Die Kommunikationskomponente im logischen Prozeß 1 wirkt dabei als Nachrichtensenke, während die Kommunikationskomponente im logischen Prozeß 2 als Nachrichtengenerator agiert, dessen Nachrichten wiederum in einen FIFO-Speicher geschrieben werden.

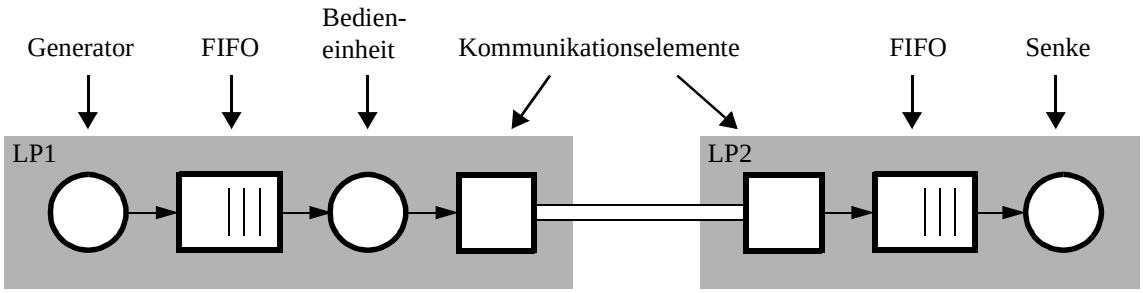


Bild 3-14: Beispiel für die Verwendung der Simulationskomponentenbibliothek

3.5 Software-Architektur für die Steuerrechner

Die Aufgaben des Steuerrechners umfassen im wesentlichen die Simulationssteuerung, die Interaktion mit dem Benutzer und die Anzeige der Simulationsergebnisse. Die dafür notwendige Software ist überwiegend unabhängig vom Einsatzgebiet des Simulationssystems, muß aber simulationsmodellabhängige Ergänzungen zulassen. Zur Trennung der unterschiedlichen Abstraktionsebenen wird eine Architektur mit drei Ebenen verwendet (Bild 3-15).

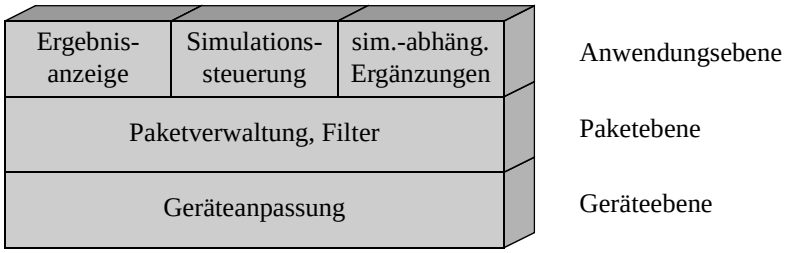


Bild 3-15: Software-Architektur für den Steuerrechner

Geräteebene

Wie in Kapitel 3.2 beschrieben, bestehen verschiedene Möglichkeiten für die Anbindung des Multiprozessorrechners an den Steuerrechner. Die Geräteebene dient der Abstraktion von dem verwendeten Ein-/Ausgabegerät, so daß die darüberliegenden Ebenen unabhängig davon sind. Die Geräteebene stellt der Paketebene eine paketorientierte Schnittstelle zur Verfügung, über die einzelne Pakete ausgetauscht werden können.

Paketebene

Die Paketebene stellt anwendungsunabhänige Basisdienste für die Kommunikation zur Verfügung. Dazu gehört u. a. die Pufferung von empfangenen bzw. zu sendenden Paketen sowie das Erzeugen und Vernichten von Paketen. In der Paketebene kann ein frei definierbarer Empfangsfilter eingerichtet werden. Empfangene Pakete werden entsprechend diesem Filter und ihrer Priorität sofort oder zu einem späteren Zeitpunkt an die Anwendungsprogramme übergeben. Dadurch muß nicht auf alle eingehenden Pakete sofort reagiert werden. Die graphische Anzeige von Simulationsdaten ist z. B. effizienter, wenn mehrere Datenpakete gemeinsam bearbeitet werden. Auf wichtige Pakete, wie z. B. Fehlermeldungen, muß dagegen sofort reagiert werden.

Anwendungsebene

Die Anwendungsebene gliedert sich in die drei Hauptmodule Simulationssteuerung, Ereignisanzeige und simulationsabhängige Ergänzungen. Die Simulationssteuerung ist für die Initialisierung des Multiprozessorrechners, für die Abbildung der logischen Prozesse auf die Simulationsknoten sowie für das Starten und Anhalten der Simulation zuständig.

Die Simulationssteuerung ist eng mit der Ergebnisanzeige verknüpft. Abhängig davon, ob eine funktionale, eine statistische stationäre oder eine statistische instationäre Simulation vorliegt, werden unterschiedliche Anforderungen an die Ergebnisanzeige gestellt. Sie muß sowohl

numerische als auch graphische Darstellungsformen unterstützen. Beides ist in Echtzeit (Momentanwerte) und nach Abschluß der Simulation (Statistiken) möglich.

Die Software-Architektur für den Steuerrechner bietet Raum für simulationsabhängige Programmerweiterungen. Dokumentierte Schnittstellen zwischen den Modulen der Anwendungsebene erlauben die einfache Integration weiterer Module, die z.B. eine dem zu simulierenden System angepaßte graphische Darstellung der Simulationsergebnisse zur Verfügung stellen oder logische Prozesse und zugehörige Parametersätze automatisch generieren.

Den grundsätzlichen Ablauf zur Ausführung einer Simulation zeigt Bild 3-16. Zunächst wird von der Simulationssteuerung die Topologie des Multiprozessorrechners ermittelt, wobei eine Initialisierung der Kommunikationsressourcen stattfindet. Anschließend werden die logischen Prozesse des Simulationsmodells und ihre Kommunikationsbeziehungen auf die Topologie des Multiprozessorrechners abgebildet. Entsprechend dieser Abbildung wird das Simulationsprogramm für jeden logischen Prozeß auf dem zugeordneten Prozessor zur Ausführung gebracht und die jeweils notwendigen Simulationsparameter übertragen. Die Simulation wird mit Hilfe einer Steuernachricht an alle logischen Prozesse gestartet. Während die Simulation ausgeführt wird, werden eingehende Simulationsergebnisse von der Simulationssteuerung zur logischen Zuordnung an die Prozeßverwaltung weitergereicht und ggf. unter Verwendung der Ergebnisanzeige dargestellt bzw. protokolliert. Das Simulation kann durch den Benutzer, die Simulationssteuerung oder durch einen logischen Prozeß terminiert werden.

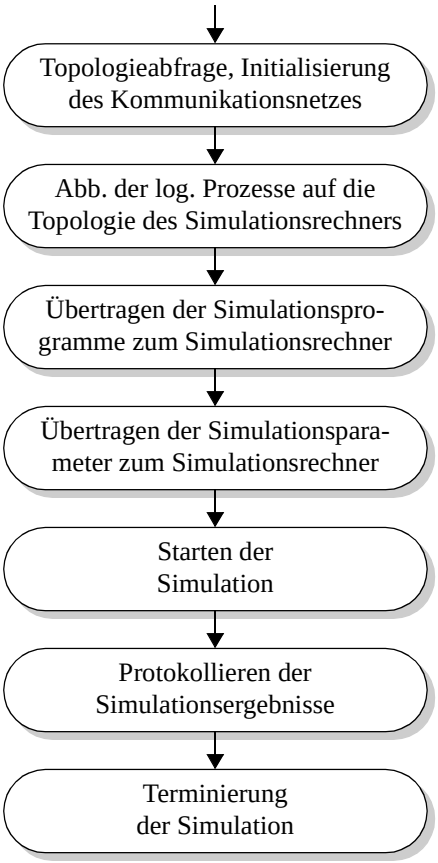


Bild 3-16: Ablauf der Simulationsausführung

Kapitel 4

Prototypische Realisierung der Simulatorarchitektur

Um die Realisierbarkeit und die Leistungsfähigkeit der im Kapitel 3 vorgestellten Simulatorarchitektur zu zeigen, wurde ein Prototyp des Simulationsrechners aufgebaut. Da es sich bei dem realisierten Simulationsrechner um einen Prototyp handelt, benötigt er nur die Komponenten, die für eine grundsätzliche Leistungsbewertung der Simulatorarchitektur notwendig sind (siehe Kapitel 5). Keinen Einfluß auf die Leistungsfähigkeit des Gesamtsystems haben Funktionalitäten, die dem Bedienungskomfort dienen, weshalb diese teilweise nicht in der prototypischen Realisierung enthalten sind.[1] Aufgrund der im Prototyp begrenzten Hardware-Ressourcen werden alle Funktion, die nicht zeitkritisch sind,[2] Software-gesteuert durch die SPU erbracht. Der Zweitorspeicher zwischen SPU und CP stellt eine Ergänzung der Simulatorarchitektur für wenige, spezielle Simulationsmodelle dar. Da für eine grundsätzliche Leistungsuntersuchung auf diesen Speicher verzichtet werden kann, ist er im Simuatorprototyp nicht enthalten.

Für den Nachweis der Skalierbarkeit des Simulationsrechners und der jeweils simulierbaren Modelle muß der Simulationsrechner mit unterschiedlich vielen Simulationsknoten betrieben werden. Der Simulatorprototyp enthält bislang drei Simulationsknoten, so daß durch den Vergleich von Simulationen mit zwei und mit drei Simulationsknoten Aussagen für größere Systeme abgeleitet werden können. Ferner können mit drei oder mehr Simulationsknoten Modelle untersucht werden, die keine direkte Rückkopplung zwischen den Simulationsknoten besitzen. Dies ist beispielsweise bei der in Kapitel 5.3.1 vorgestellten Simulation von CRMA-II der Fall, bei der die Simulationsknoten ringförmig miteinander verbunden sind und die Kommunikation nur unidirektional stattfindet.

Bei der Simulation von Kommunikationssystemen werden häufig bus- und ringförmige Topologien betrachtet. Bei der prototypischen Realisierung des Simulationsrechners reichen daher zwei physikalische Kanäle je Simulationsknoten für die meisten Simulationsmodelle aus. Bei drei Simulationsknoten entspricht die Ringtopologie der Vollvermaschung, so daß aus der

1. Z.B. Laden der Simulationsprogramme über den Zweitorspeicher.
2. Z.B. Initialisierung des Kommunikationsprozessors, Bearbeitung von Steuernachrichten.

Beschränkung auf zwei physikalische Kanäle keine Einschränkung für den Simulatorprototyp mit drei Simulationsknoten resultiert.

Bild 4-1 zeigt den Multiprozessorrechner mit drei Simulationsknoten ohne die Zufallszahlenprozessoren. Erweiterungen um weitere Simulationsknoten oder um zusätzliche simulationsspezifische Hardware sind bei der prototypischen Realisierung leicht möglich.

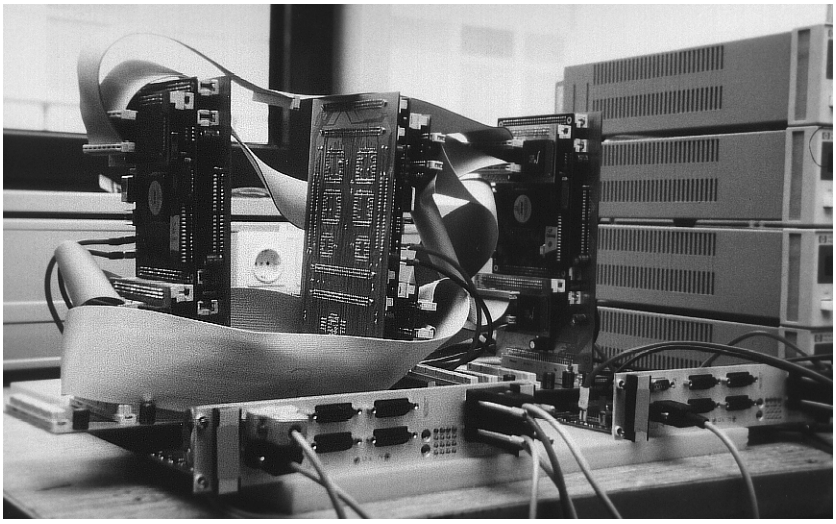


Bild 4-1: Prototypischer Multiprozessorrechner (Abbildung ohne Zufallszahlenprozessoren)

Die Hardware-Komponenten des Simulatorprototyps werden, soweit dies für die Leistungsbewertung von Belang ist, im Kapitel 4.1 erläutert. Die für den Betrieb des Simulationssystems erstellte Software wird in Kapitel 4.2 vorgestellt.

4.1 Hardware-Komponenten des Simulationssystems

Die Hardware des Simulationssystems besteht aus einem Steuerrechner und dem Multiprozessorrechner. Die beiden Rechner werden nachfolgend getrennt betrachtet, da sie, abgesehen von der zwischen ihnen stattfindenden Kommunikation, voneinander unabhängig sind.

4.1.1 Multiprozessorrechner

Der Multiprozessorrechner besteht aus den Simulationsknoten und aus physikalischen Kanälen, welche die Simulationsknoten untereinander verbinden. Der Aufbau der realisierten Simulationsknoten ist in Bild 4-2 dargestellt.

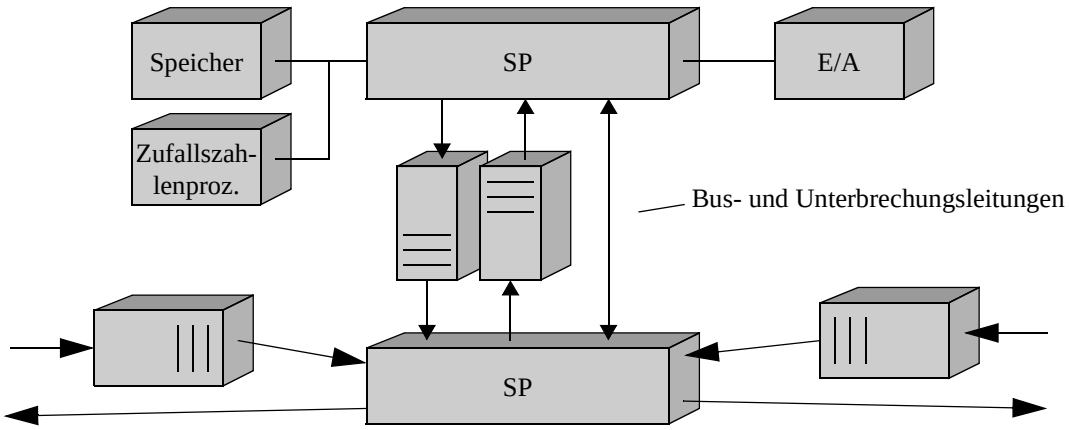


Bild 4-2: Struktur eines Simulationsknotens

4.1.1.1 Simulationsprozessor

Jeweils zwei Simulationsprozessoren befindet sich zusammen mit ihrer Peripherie auf einer eigenen Platine (in Bild 4-1 liegend dargestellt). Wie bereits in Kapitel 3.3 motiviert, eignet sich für die ereignisgesteuerte Simulation ein Prozessor, der einfache Listen- und Datenmanipulationen schnell durchführen kann. Gegenwärtig kommt als Simulationsprozessor ein TMS320C40 der Firma Texas Instruments zum Einsatz. Dieser Prozessor besitzt ein sehr einfaches und flexibles Busprotokoll und eine Fließkommaeinheit für einfache Fließkommaoperationen. Ferner hat er, ähnlich wie die bekannten Transputer der Firma Inmos, Kommunikationskanäle für den Aufbau von Multiprozessorsystemen ohne zusätzliche Hardware-Komponenten [TI91, INM88]. Der Prozessor eignet sich daher für die vergleichende Simulation mit und ohne simulationsspezifische Hardware. Während bei der Hardware-unterstützten Simulation die Kommunikation ausschließlich über den Kommunikationsprozessor erfolgt, werden für die parallele Simulation ohne Hardware-Unterstützung die Kommunikationskanäle der Prozessoren verwendet.

4.1.1.2 Kommunikationsprozessor

Die Umsetzung der in Kapitel 3.3.1 beschriebenen vollständigen Architektur des Kommunikationsprozessors in Hardware erfordert einen Schaltkreis mit ca. 270 Ein- bzw. Ausgängen, 40 kbit Schreib-/Lesespeicher und eine komplexe Struktur aus vielen interagierenden Automaten zur Steuerung der einzelnen Module und des Kommunikationsprotokolls. Aufbauend auf der Beschreibung und Synthese des vollständigen Kommunikationsprozessors unter Verwendung von VHDL wurde gezeigt, daß sich der Kommunikationsprozessor abhängig von der

konkreten Implementierung mit Hilfe von 60.000 bis 100.000 Gattern realisieren läßt. Somit ist die Logik zwar leicht in einer kundenspezifischen integrierten Schaltung (ASIC) unterbringen, doch eignen sich diese nicht für prototypische Realisierungen.

Die für den Simulatorprototyp ausreichende komplexitätsreduzierte Architektur läßt sich mit programmierbarer Logik (Field Programmable Gate Arrays, FPGAs) realisieren, so daß Modifikationen und Ergänzungen leicht möglich sind. Wesentlichen Einbußen bei der Simulationsgeschwindigkeit treten dadurch nicht auf. Die verwendeten anwenderprogrammierbaren Logikbausteine (Field Programmable Gate Arrays, FPGAs) besitzen integrierte Speicher, die für die Verkehrslenkungstabelle und die internen FIFO-Puffer verwendet werden. Dadurch wird eine Ein-Chip-Lösung des komplexitätsreduzierten Kommunikationsprozessors möglich, wenn man von der Adreßdekodierung und der Steuerung der externen FIFO-Puffer absieht, die einen schnelleren Baustein erfordern. Der Kommunikationsprozessor wird synchron zum Bustakt des Simulationsprozessors betrieben (20 MHz). Die maximale Anzahl an Paketen, die pro Sekunde durch den Kommunikationsprozessor bearbeitet werden können, hängt von der Paketlänge und von den benutzten Kommunikationskanälen ab. Im ungünstigsten Fall beträgt der Durchsatz 273.972 Pakete pro Sekunde[3], im günstigsten Fall ist eine maximale Paketrate von 20 Millionen Paketen pro Sekunde[4] möglich.

Die FIFO-Puffer zwischen den Kommunikationsprozessoren sind jeweils 2048 Worte groß und können somit abhängig von der Paketlänge 113 bis 2048 Pakete aufnehmen. Die beiden FIFO-Puffer zwischen dem Kommunikations- und dem Simulationsprozessor wurden doppelt so groß dimensioniert, da es dem Simulationsprozessor freisteht, längere Zeit kein Paket abzuholen bzw. büschelförmigen Verkehr zu erzeugen. Dies reicht für die meisten Simulationsmodelle aus, da aufgrund der maximal zulässigen Differenz zwischen den lokalen Simulationszeiten ohnehin die Anzahl der in Transit befindlichen Nachrichten begrenzt ist. Ist der FIFO-Puffer des Empfängers voll, so kommt es zum Rückstau in die FIFO-Puffer zwischen den Kommunikationsprozessoren bzw. in den Sendespeicher des entsprechenden Simulationsknotens. Damit auch im Extremfall keine Nachrichten verloren gehen, können sie unter Inkaufnahme einer verlangsamten Verarbeitungsgeschwindigkeit ggf. im Hauptspeicher des sendenden Simulationsknotens gepuffert werden.

---

3. Gilt bei maximaler Paketlänge (18 Worte zu je 32 Bit) und lokal versendeten Paketen, die ausschließlich über einen Ausgangskanal übertragen werden. Der minimale Durchsatz für Pakete, die nicht vom lokalen Simulationsprozessor erzeugt werden, beträgt 540.540 Pakete pro Sekunde.
4. Gilt bei minimaler Paketlänge (1 Wort mit 32 Bit) und drei sich gegenseitig nicht blockierenden Paketströmen.

Die schnelle Nullnachrichtenschleife ist so implementiert, daß sie programmgesteuert aktiviert und deaktiviert werden kann. Dadurch ist es bei der Leistungsbewertung möglich, den durch die schnelle Nullnachrichtenschleife erzielbaren Simulationszeitgewinn zu messen.

Für die Realisierung der Broadcast-Funktionalität gibt es verschiedene Möglichkeiten. Die schnellste Verteilung solcher Nachrichten erfordert, daß der sendende Simulationsknoten die Nachricht über sämtliche Ausgangskanäle ausgibt. Jeder empfangende Simulationsknoten kopiert die Nachricht und prüft absenderabhängig, über welche Ausgangskanäle die Nachricht zum Erreichen weiterer Simulationsknoten ausgegeben werden muß. Dazu ist eine eigene absenderabhängige Verkehrslenkungstabelle erforderlich. Da bei den in Kapitel 5 vorgestellten Simulationen nur wenige Broadcast-Nachrichten verwendet werden, enthält der prototypische Aufbau eine einfachere Realisierung der Broadcast-Funktionalität, bei der die Verkehrslenkung von Broadcast-Nachrichten, wie bei den übrigen Paketen, entsprechend der Zieladresse und den Eintragungen in den normalen Verkehrslenkungstabellen erfolgt. Die Einträge in den Tabellen werden so gesetzt, daß sich für die Adresse FF_h ein Hamilton-Pfad[5] ergibt. Durch Prüfen der Absenderadresse einer empfangenen Broadcast-Nachricht kann der Sender die Nachricht wieder entfernen.

4.1.1.3 Anbindung der Steuerrechner

Die Einheit für den Anschluß eines Steuerrechners an den Multiprozessorrechner wird in einem bidirektionalen Kanal des Multiprozessorrechners zwischengeschaltet. Die Anschlußeinheit filtert die Pakete für den Steuerrechner aus dem Kanal heraus und leitet sie an den Steuerrechner weiter. Umgekehrt werden Pakete des Steuerrechners entsprechend einer Verkehrslenkungstabelle in den Paketstrom des bidirektionalen Kommunikationskanals eingefügt. Da dabei weder Synchronisation, Zeitauswertung noch Nullnachrichtengenerierung notwendig sind, stellt die Anschlußeinheit eine stark vereinfachte Variante des Kommunikationsprozessors dar. Ihre Realisierung mit anwenderprogrammierbarer Logik (FPGA) basiert daher weitgehend auf Modulen des Kommunikationsprozessors [Nec98].

Die Paketübertragung zwischen Steuer- und Multiprozessorrechner erfolgt seriell mit 300 MBit/s [Cyp93]. Die hohe Datenrate und das auf die Anforderungen der Simulation opti-

5. Unter einem Hamilton-Pfad wird in diesem Zusammenhang ein geschlossener Pfad verstanden, der durch alle Rechenknoten des MIMD-Rechners genau einmal führt [Dud93]. Die Steuerrechner sind in diesem Pfad nicht enthalten.

mierte Übertragungsprotokoll machen für den Steuerrechner eine spezielle Schnittstellenkarte erforderlich. Diese ist rechnerunabhängig auf der Basis des PCI-Standards (Peripheral Component Interconnect) realisiert [Läu98].

4.1.1.4 Zufallszahlenprozessor

Der Zufallszahlenprozessor besteht aus zwei logisch getrennten Modulen, einem Modul zur Erzeugung von gleichverteilten Zufallszahlen (GVZZ) und einem nachgeschalteten Modul, das daraus beliebig verteilte Zufallszahlen generiert (ZZ-CPU) und sie für den Simulationsprozessor in FIFO-Puffer ablegt (Bild 3-12).

Erzeugung gleichverteilter Zufallszahlen durch spezielle Hardware

In [Wol99] wird gezeigt, daß sich ein sehr leistungsfähiges GVZZ-Modul unter Verwendung von FPGA-Bausteinen realisieren läßt. Die beiden im Algorithmus von L'Ecuyer [L'E96] verwendeten rekursiven Zufallszahlengeneratoren werden dazu ausschließlich aus kombinatorischer Logik aufgebaut. Die für die beiden Modulo-Operationen notwendigen Divisionen lassen sich, da ausschließlich durch Konstanten dividiert wird, so optimieren, daß für den Algorithmus insgesamt nur 24 Additionen durchgeführt werden müssen. Da diese teilweise parallelisierbar sind, kann bei der Realisierung mit anwenderprogrammierbarer Logik nach [Wol99] in jedem Systemtakt des GVZZ-Moduls (8 MHz) eine Zufallszahl erzeugt werden.

Erzeugung und Bereitstellung beliebig verteilter Zufallszahlen

Die Berechnung der Zufallszahlen entsprechend vorgegebener Verteilungsfunktionen erfolgt bei der prototypischen Realisierung der Simulatorarchitektur durch einen einfachen RISC-Prozessor (ZZ-CPU in Bild 3-12). Der Prozessor kann programmgesteuert Zufallszahlen nach acht verschiedenen Verteilungsfunktionen erzeugen und in getrennten FIFO-Puffern für den Abruf durch die SPU bereitstellen. Dabei werden jeweils maximal 4096 Zufallszahlen gespeichert. Ein gemeinsamer Speicherbereich der SPU und der ZZ-CPU wird für die Übergabe der Verteilungsfunktionen, der Parameter der Verteilungsfunktionen und der Aktivierung des Zufallszahlenmoduls verwendet.

Realisierung

Die in Kapitel 5 für die Bewertung herangezogenen Simulationsmodelle erfordern nicht die schnelle Zufallszahlenerzeugung, die mit einem aus spezieller Hardware aufgebauten GVZZ-Modul möglich ist. Daher werden die gleichverteilten Zufallszahlen im Rahmen der prototypi-

schen Realisierung nicht durch spezielle Hardware sondern durch die ZZ-CPU errechnet. Die der Realisierung entsprechende Architektur zeigt Bild 4-3.

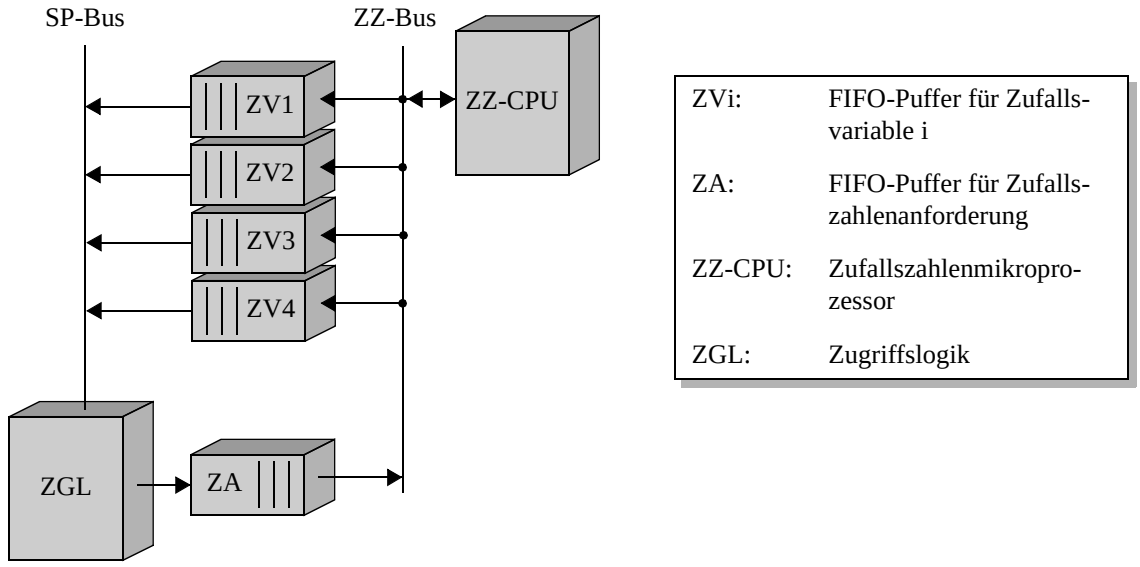


Bild 4-3: Architektur des realisierten Zufallszahlenprozessors

Aufgrund der großen FIFO-Puffer stehen auch bei kurzfristig hohem Bedarf an Zufallszahlen genügend Zufallswerte zur Verfügung. Wurden sämtliche Zufallszahlen einer Verteilung von der SPU ausgelesen, so daß der entsprechende FIFO-Speicher beim versuchten Einlesen einer Zufallszahl leer ist, so wird die SPU so lange angehalten, bis die ZZ-CPU wieder eine Zufallszahl dieser Verteilung erzeugt hat. Dadurch wird die Verwendung einer ungültigen Zufallszahl verhindert, was unter Umständen zu nicht reproduzierbaren Effekten führen würde.

4.1.2 Steuerrechner

Im Prinzip kann jeder beliebige Rechner als Steuerrechner verwendet werden, da keinerlei simulationsspezifische Ergänzungen notwendig sind. Für die prototypische Realisierung wird ein Personal Computer (PC) mit dem Betriebssystem Microsoft Windows NT verwendet. Der Steuerrechner verfügt sowohl über eine standardisierte Hardware mit PCI-Bus (Peripheral Component Interconnect), der für den Anschluß der seriellen Schnittstelle zum Multiprozessorrechner verwendet wird (siehe Kapitel 4.1.1.3), als auch über umfangreiche Möglichkeiten für graphische Ein-/Ausgaben. Ferner stehen geeignete Software-Entwicklungswerkzeuge zur Verfügung, die für den Entwurf der Steuerprogramme und der Programme für die Anzeige der Simulationsdaten notwendig sind.

4.2 Software-Komponenten

Die zum Betrieb des Simulationssystems notwendige Software läßt sich entsprechend Kapitel 3.4 in Software für den Parallelrechner und Software für den Steuerrechner aufteilen. Die Software für den Parallelrechner besteht aus den Bibliothekskomponenten und den simulationsmodellabhängigen Komponenten. Letztere werden zusammen mit den Leistungsuntersuchungen in Kapitel 5 erläutert.

4.2.1 Bibliotheken für den Parallelrechner

Die in Bild 3-13 dargestellte Software-Architektur wird bei der Realisierung um eine Schicht zur Abstraktion der eingesetzten Hardware erweitert (Bild 4-4). Diese Schicht ermöglicht es, den darüberliegenden Bibliotheken die konkrete Realisierung des Kommunikations- und Synchronisationsprozessors zu verbergen. Durch Anpassung der Hardware-Abstraktionsebene kann eine Simulation auf einem Multiprozessorrechner mit vollständiger Hardware-Unterstützung, mit einer komplexitätsreduzierten Architektur oder auf einem Multiprozessorrechner ohne simulationsspezifische Hardware erfolgen. Dadurch ist ein einfacher Vergleich der Leistungsfähigkeit der verschiedenen Architekturen möglich (siehe Kapitel 5).

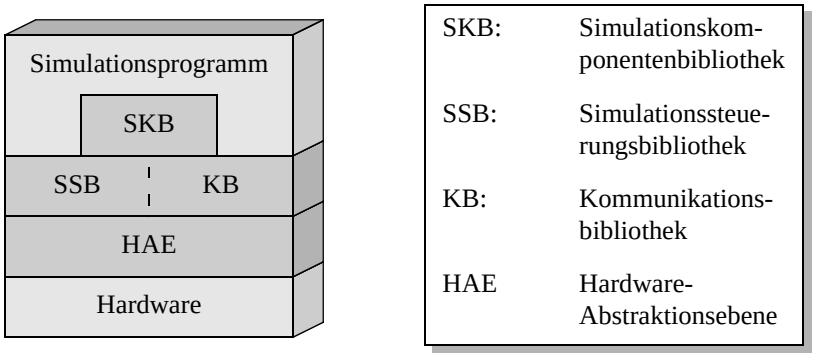


Bild 4-4: Bibliotheken für den Multiprozessorrechner

Die Implementierung der Bibliotheken erfolgte in der Sprache ANSI-C [Sch90], die für die meisten Mikroprozessoren verfügbar ist. Durch die Einführung der Hardware-Abstraktionsebene sind die darüberliegenden Bibliotheken portabel. Dadurch können auf einfache Weise andere Prozessoren eingesetzt werden.

4.2.2 Steuerprogramme

Aufbauend auf die in Kapitel 3.5 vorgestellte Geräte- und Paketebene existieren für die Steuerrechner im Rahmen der prototypischen Realisierung des Simulationssystems zwei voneinander unabhängige Anwenderprogramme [Wag98]. Beide basieren auf einer weitgehend plattformunabhängigen Oberflächenbibliothek [Zinc94a, Zinc94b] und einer objektorientierten Programmiersprache [ES90, Str91].

Das zu simulierende Modell wird vom Anwender in Komponenten aufgeteilt, die den logischen Prozessen entsprechen. Die Komponenten können als Knoten eines Graphen aufgefaßt werden, dessen Kanten die Kommunikationsbeziehungen zwischen den Komponenten darstellen. Daher wird ein allgemeiner, graphischer Netzeditor [Pro98] zur Eingabe solcher Graphen verwendet (Bild 4-5). Jedem Netzknoten und jeder Netzkante können beliebige Attribute zugewiesen werden, aus denen sich die Simulationsparameter ableiten.

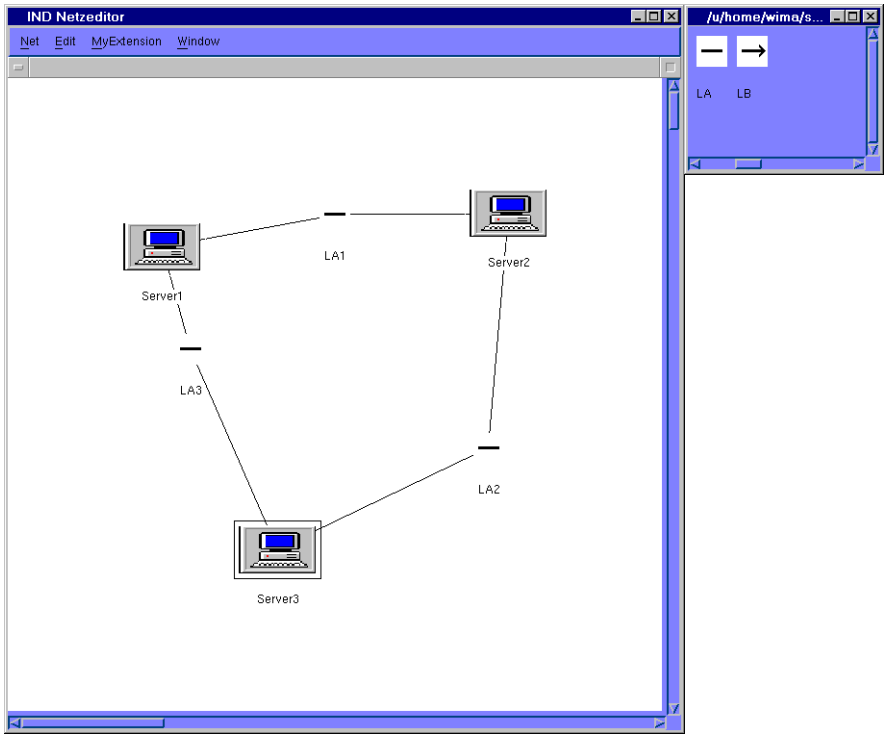


Bild 4-5: Netzeditor

Die Daten des zu simulierenden Netzes werden über eine Dateischnittstelle zum Simulationssteuerprogramm übertragen (Bild 4-6). Dieses steuert unter Verwendung der Paket- und der Geräteebene [Ern99] den Multiprozessorrechner und die Simulationsausführung. Die Anzeige von Simulationsdaten kann entweder auf der Basis der vom Multiprozessorrechner regelmäßig

erhaltenen Simulationsdatenpakete oder mit Hilfe von periodischen Abfragen durch das Steuerprogramm erfolgen. Für die graphische Darstellung der Simulationsdaten existieren spezielle Bibliotheken [Lei94, Por97], die sowohl während der Simulation den kontinuierlichen Verlauf von Simulationsgrößen anzeigen können, als auch in der Lage sind, im Anschluß an die Simulation statistische Auswertungen darzustellen.

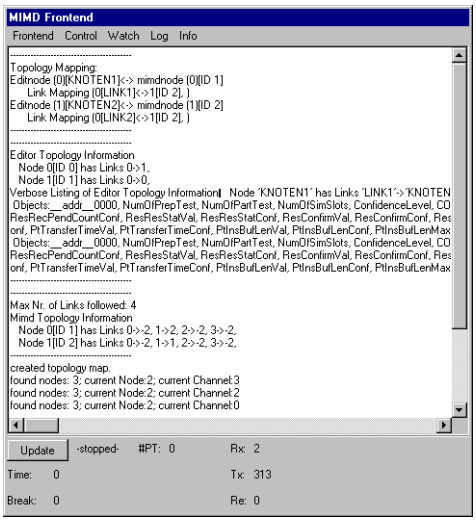


Bild 4-6: Simulationssteuerprogramm

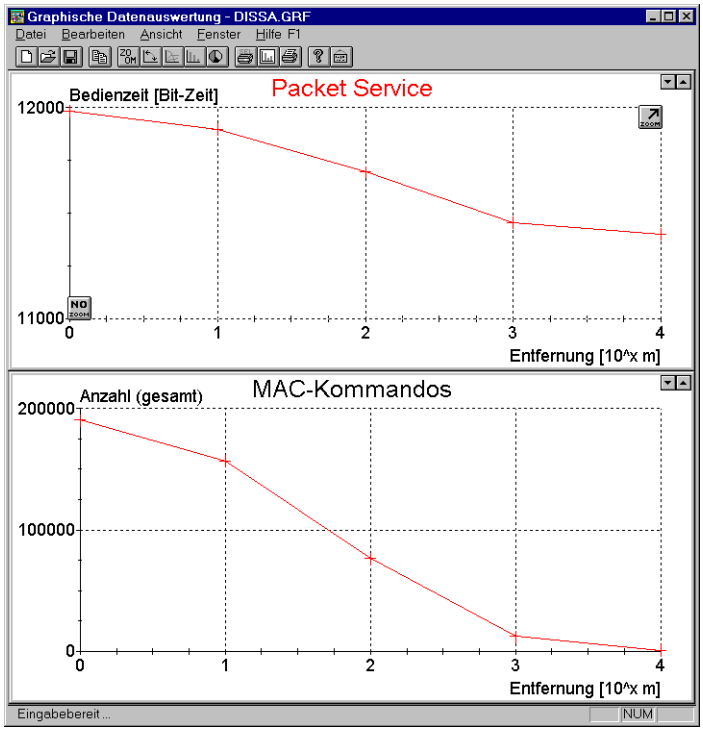


Bild 4-7: Graphische Darstellung von Simulationsergebnissen

Kapitel 5

Bewertung der Simulatorarchitektur und des Simulationsrechners

Die Bewertung einer Architektur gestaltet sich im allgemeinen schwierig, da es eine Vielzahl völlig unterschiedlicher Aspekte zu beurteilen und zu vergleichen gilt. Da eine Architektur in aller Regel keinen Selbstzweck, sondern die Grundlage einer Realisierung darstellt, interessieren bezüglich der Leistungsfähigkeit im allgemeinen die Auswirkungen der Architektur auf die Realisierung. Im konkreten Fall bedeutet dies für das Simulationssystem, daß für realisierungsabhängige Bewertungen die prototypische Realisierung aus Kapitel 4 herangezogen werden muß. Im folgenden wird daher zwischen den leistungsunabhängigen Maßstäben (Kapitel 5.1) und einer Leistungsuntersuchung (Kapitel 5.2) unterschieden. In Kapitel 5.3 wird die Anwendung auf praktische Simulationsprobleme aus der Kommunikationstechnik und die dabei erzielte Leistungsfähigkeit vorgestellt. In Kapitel 5.4 wird ein Ausblick auf weitere Simulationsprobleme gegeben.

5.1 Bewertung nach leistungsunabhängigen Maßstäben

Die wesentlichen leistungsunabhängigen Kriterien zur Bewertung eines Simulators sind seine Handhabbarkeit für den Anwender, die universelle Einsetzbarkeit sowie die Offenheit für Erweiterungen und neue Technologien. In diesem Kapitel erfolgt eine Bewertung der Simulatorarchitektur bezüglich dieser Maßstäbe. Dabei gilt als Referenz stets die sequentielle Simulation.

Handhabbarkeit für den Anwender

Die Handhabbarkeit eines parallelen Simulators hängt im Vergleich zum sequentiellen Simulator im wesentlichen von der Transparenz der parallelen Ausführung des Simulationsmodells ab. Während die Einteilung des Simulationsmodells in logische Prozesse noch als eine Form der Strukturierung und Modularisierung aufgefaßt werden kann, stellen die Parameter des parallelen Simulationsverfahrens und die Kommunikationsroutinen eine ungewollte Erhöhung der Komplexität für den Anwender dar.

Bei konservativen parallelen Simulatoren existieren üblicherweise Parameter für die Nullnachrichtengenerierung und das verwendete Kommunikationssystem.[1] Bei der vorgestellten Simulatorarchitektur wird das Kommunikationssystem unabhängig vom Anwender durch das Steuerprogramm und die Kommunikationsbibliothek konfiguriert. Die Nullnachrichtenerzeugung und -bearbeitung erfolgt vollständig transparent für den Anwender. Dadurch reduziert sich für ihn der Mehraufwand bezüglich der Kommunikation auf das Einfügen von Kommunikationsendpunkten aus der Simulationskomponentenbibliothek in das Simulationsmodell (siehe Kapitel 3.4). Somit verbleibt als einziger simulationsverfahrenspezifischer Parameter die Wahl des Zeitpunkts, zu welchem der Kommunikations- und Synchronisationsprozessor Nullnachrichten nach der Erhöhung der Simulationszeit generieren soll (siehe Kapitel 3.3.1.2).

Bei der parallelen Programmierung stellt normalerweise die Fehlersuche ein erhebliches Problem dar, da die fehlerhaften Konstellationen oft nicht zuverlässig reproduzierbar sind. Unter der Voraussetzung von korrekten Simulationsbibliotheken trifft dies für den Anwender dieses Simulators nicht zu, da die kritischen Programmteile ausschließlich Bestandteil der Simulationsbibliotheken sind. Die Module des Anwenders werden in eindeutiger und wiederholbarer Weise bearbeitet, da durch die Prozeßsynchronisation die Ereignissequenz in jedem logischen Prozeß im Wiederholungsfall identisch reproduziert wird.

Universalität

Aus der Sicht des Anwenders ist es wünschenswert, daß sämtliche Simulationsmodelle mit einem Simulator effizient und schnell bearbeitet werden können. Wie bereits in Kapitel 2.4 erläutert, ist dieses Ziel für die ereignisgesteuerte Simulation aufgrund der Verschiedenartigkeit der Simulationsmodelle nicht erreichbar. Daher muß die Forderung nach Universalität dahingehend eingeschränkt werden, daß die Simulationsmodelle einer Problemklasse unabhängig vom speziellen Problem geeignet simulierbar sind. Simulatoren für diese nicht notwendigerweise disjunkten Klassen haben unabhängig voneinander ihre Berechtigung.

Die vorgestellte Architektur wurde für die Klasse der Simulationsmodelle mit eingeschränkter Ereignislokalität optimiert. Alle Modelle, die diese Eigenschaften besitzen, sind schnell und effizient simulierbar, weitere Forderungen, wie z.B. zentraler Ereignistakt[2], werden nicht erhoben.

1. Bei optimistischen und hybriden Simulationsverfahren ist der Parameterraum ungleich größer, so daß diese Verfahren im allgemeinen eine weitaus höheren Anspruch an den Anwender stellen.
2. Simulationsereignisse treten synchron zu einem systemweit einheitlichen Takt auf.

Offenheit für neue Technologien und Erweiterungen

Die vorgestellte Architektur beschreibt im wesentlichen ein durch Protokolle und Anforderungen definiertes Simulationssystem. Da mit Ausnahme des zu Grunde liegenden Synchronisationsverfahrens keine Algorithmen und Implementierungen definiert werden, sind beliebige Realisierungen denkbar. Somit können leicht neue Programmiersprachen, Prozessoren und Kommunikationskanäle verwendet werden.

Erweiterungen, z. B. zur weiteren funktionalen Parallelisierung oder zur Berechnung spezieller Simulationsmodelle, können leicht in die Architektur eingefügt werden, da ihre Nutzung aufgrund der davon abstrahierenden Bibliotheken für das Anwendungsprogramm transparent ist.

5.2 Leistungsbewertung der Simulatorarchitektur

Für die Leistungsbewertung ist es notwendig, eine konkrete Realisierung der Architektur zu betrachten. Im folgenden wird daher die in Kapitel 4 vorgestellte prototypische Realisierung mit sequentiellen und parallelen Simulationen, jeweils mit und ohne spezielle Hardware verglichen. Daher wird im folgenden zunächst jedes Modul für sich betrachtet. Die Simulationsleistung des gesamten Simulators variiert mit den Eigenschaften des Simulationsmodells und kann daher nicht ohne weiteres angegeben werden. Somit müssen für eine Leistungsbewertung ohne das Vorhandensein eines Prototypen die jeweiligen Eigenschaften des Simulationsmodells analytisch oder durch Simulation ermittelt werden, die dann wiederum als Parameter einer weiteren Analyse bzw. Simulation bezüglich des Simulationssystems dienen. Da die Genauigkeit so gewonnener Ergebnisse nur schwer bestimmbar ist, wird an Stelle dieses Vorgehens in den Kapiteln 5.3.1 und 5.3.2 die prototypische Realisierung für die Simulation praxisrelevanter Modelle verwendet und mit anderen Möglichkeiten der Simulation verglichen.

In der Simulatorarchitektur sind die drei Module Synchronisation, Kommunikation und Zufallszahlenerzeugung definiert, die nach dem Prinzip der funktionalen Parallelisierung parallel zum Simulationsprozessor arbeiten. Jedes dieser Module erfüllt seine Aufgabe schneller und effizienter als dies der Simulationsprozessor mit Hilfe von Software könnte. Der dadurch erzielte Nutzen für die gesamte Simulation hängt von der Häufigkeit und der Verteilungsfunktion der Zugriffe auf die jeweiligen Module ab.

Synchronisation

Bezüglich der Synchronisation muß zwischen der lokalen und globalen Synchronisation entsprechend Kapitel 3.3.1.2 unterschieden werden. Der Aufwand für die lokale Synchronisation, mit der bestimmt wird, ob das nächste, lokal bekannte Ereignis sicher ist, hängt bei einer Software-Realisierung von der Anzahl der verwendeten Eingangskanäle ab. Für die Bestimmung des Minimums aus n Zahlen werden n-1 Vergleiche benötigt. Da der in der prototypischen Realisierung verwendete Prozessor TMS320C40 von Texas Instruments eine 32 Bit-Architektur besitzt, benötigt er für das Einlesen und Vergleichen von zwei Zeitstempeln im Mittel 14 Taktzyklen sowie für das abschließende Sichern des Minimums 2,5 Taktzyklen[3]. Allgemein ergibt sich somit die mittlere Anzahl der Taktzyklen a_t in Abhängigkeit von der Anzahl der Vergleiche a_v entsprechend Gleichung 5-1.

$$a_t = 14a_v + 2{,}5 \tag{5-1}$$

Da die Sicherheit von Ereignissen einzeln geprüft werden muß[4], stellt die Ausführung des Zeitvergleichs in Hardware eine signifikante Verkürzung dar, da sich der Aufwand für den Simulationsprozessor auf drei Lese- und zwei Schreibzugriffe auf den Synchronisations- und Kommunikationsprozessor bzw. den lokalen Speicher reduziert[5]. Dies bedeutet, daß sich bereits bei einem einzigen Eingangskanal der Zeitaufwand von durchschnittlich 16,5 Taktzyklen auf 5 Taktzyklen reduziert.

Ähnlich verhält es sich mit der Entscheidung für das Generieren von Nullnachrichten. Hierzu muß geprüft werden, ob die benötigten Kommunikationskanäle frei sind und ob bereits eine Nachricht mit dem entsprechenden Zeitstempel verschickt wurde. Da der Zustand der Kommunikationskanäle und sämtliche Daten für den Zeitvergleich im Synchronisations- und Kommunikationsprozessor verfügbar sind, wird der Simulationsprozessor von der Entscheidung zur Generierung einer Nullnachricht vollständig entlastet.

3. Nach der Bestimmung des Minimums wird in der Hälfte aller Fälle bedingt zu den Speicherbefehlen verzweigt.
4. Für spezielle Simulationsmodelle ist es möglich, die Anzahl der Vergleiche dadurch zu reduzieren, daß z.B. bei jedem Paketempfang die aktuelle Zeitgrenze ermittelt wird bzw. beim häufigen Auftreten von zeitgleichen Ereignissen diese gesondert behandelt werden.
5. Ein Lesezugriff wird zum Einlesen des Vergleichsergebnisses benötigt, die anderen Zugriffe für die Übertragung der 64 Bit breiten nächsten Ereigniszeit in den Kommunikations- und Synchronisationsprozessor (32 Bit breiter Datenbus).

Kommunikation

In einem parallelen Simulator ohne Hardware-unterstützte Kommunikation setzt sich der dafür notwendige Aufwand im wesentlichen aus Nachrichtenempfang, Nachrichtensenden, Flußsteuerung und Verkehrslenkung zusammen. Die Kommunikation findet in diesem Falle über gemeinsame Speicher oder spezielle Ein-/Ausgabe-Schnittstellen [INM88, TI91] statt.

Da auch bei Verwendung des Kommunikations- und Synchronisationsprozessors die Nachrichten in der Regel in den lokalen Speicher übernommen bzw. aus diesem heraus versendet werden, ist der Aufwand dafür im Regelfall nur unwesentlich geringer. Eine Entlastung des Simulationsprozessors kommt dadurch zustande, daß die Verkehrslenkung für Pakete, die für andere Simulationsknoten bzw. für einen der Steuerrechner bestimmt sind, nicht durch den Simulationsprozessor bearbeitet werden muß, wofür der Aufwand linear mit der Anzahl der Pakete wächst. Ferner werden Nullnachrichten weder an den Simulationsprozessor übertragen noch von ihm verschickt. Dies entspricht bezüglich der Kommunikation einer Entlastung von zwölf Taktzyklen je Nullnachricht. Hinzu kommt, daß die Auswertung bzw. die Erzeugung der Nullnachrichten entfallen. Positiv wirkt sich dies auch auf die entsprechenden FIFO-Puffer aus, so daß Verklemmungen seltener werden. Analog dazu ergeben sich Entlastungen für empfangene Pakete, die an andere Rechenknoten weitergereicht werden müssen.

Aufgrund der Synchronisations- und Kommunikationsunterstützung wird der Simulationsprozessor beim Senden und Empfangen von Nachrichten und den damit verbundenen Synchronisationsaufgaben nur geringfügig belastet. Dadurch hat die räumliche Ereignislokalität 1 (REL1) des Simulationsmodells einen wesentlich geringeren Einfluß auf die Simulationsdauer.

Zufallszahlen

Der Aufwand für die Erzeugung von Pseudozufallszahlen mit Hilfe von Software hängt stark von den benötigten Verteilungsfunktionen ab. Tabelle 5-1 zeigt die mittlere Anzahl an Taktzyklen, die der im Simulationsrechner eingesetzte Simulationsprozessor für die Erzeugung einiger Verteilungsfunktionen benötigt. Die Basis stellt dabei der in [L'E96] beschriebene Pseudozufallszahlenalgorithmus dar.

Der im Rahmen dieser Arbeit entworfene Zufallszahlenprozessor kann bei einer Realisierung durch spezielle Hardware vier Millionen gleichverteilte Pseudozufallszahlen je Sekunde erzeugen. Die Berechnung der Verteilungsfunktionen erfolgt durch einen eigenen Mikroprozessor parallel zum Simulationsprozessor. Dadurch kann letzterer durch einen einfachen Speicherzu-

Verteilungsfunktion	mittlere Anzahl Taktzyklen
gleichverteilt im Intervall [0,1)	81,2
negativ-exponentiell verteilt	144,5
verschoben geometrisch verteilt mit Mittelwert a und oberer Schranke 2a	308,2

Tabelle 5-1: Aufwandsübersicht für programmgesteuerte Pseudozufallszahlenerzeugung (Prozessor: TMS320C40)

griff eine Zufallszahl der gewünschten Verteilung einlesen. Tabelle 5-2 zeigt die mittlere Anzahl der Zufallszahlen, die durch den Zufallszahlenprozessor pro Sekunde erzeugt werden können.

Verteilungsfunktion	Zufallszahlen pro Sekunde
gleichverteilt im Intervall [0,1)	4.000.000
negativ-exponentiell verteilt	315.955
verschoben geometrisch verteilt mit Mittelwert a und oberer Schranke 2a	88.105

Tabelle 5-2: Pseudozufallszahlenleistung des speziellen Zufallszahlenprozessors

5.3 Anwendung der Simulatorarchitektur auf Kommunikationsnetze

Kommunikationsnetze bilden das Rückgrat der informationstechnischen Infrastruktur. Ihrer Entwicklung und Bewertung kommt daher eine hohe Bedeutung zu. Wie bei anderen technischen Systemen stellt die Simulation hierfür ein unerläßliches Hilfsmittel dar. Aufgrund der in heutigen Kommunikationsnetzen üblichen hohen Übertragungsraten bis hin zu mehreren Gbit/s und der begrenzten Ausbreitungsgeschwindigkeit der Signale treten z.B. sehr viele Ereignisse zwischen dem Absenden und der Ankunft von Paketen auf. Werden Systeme untersucht, bei denen Rückkopplungen von Bedeutung sind (z.B. ABR, Kapitel 5.3.2.3), so muß die Simulation über einen langen Zeitraum erfolgen, da unterschiedliche zeitliche Ebenen gleichzeitig betrachtet werden. Dies führt zu sehr langen Simulationslaufzeiten, so daß die Simulation von Kommunikationsnetzen eine Herausforderung und ein aktuelles Problem darstellt.

Kommunikationsnetze bestehen in der Regel aus aktiven Netzkomponenten, die über passive Netzelemente untereinander verbunden sind. In der Simulation stellen die aktiven Netzkomponenten im Verhältnis zu den passiven Elementen in der Regel Module mit einer sehr hohen

Anzahl an Ereignissen dar, von denen nur ein Teil Auswirkungen außerhalb des Moduls hat. Da die aktiven Netzelemente gegenüber den passiven Netzelementen eine hohe Ereignisdichte aufweisen, bietet es sich bei einer Parallelisierung der Simulation an, (zunächst) die aktiven Netzkomponenten zusammen mit dem dazugehörigen Teil des Übertragungsmediums in jeweils einem eigenen logischen Prozeß zu modellieren, der dadurch bedingt eine signifikante räumliche Ereignislokalität 1 besitzt. Auf Grund der im realen System vorhandenen Laufzeit der Nachrichten auf dem Übertragungsmedium haben die Logischen Prozesse darüberhinaus eine davon abhängige zeitliche Ereignislokalität (ZEL). Eine räumliche Ereignislokalität 2 liegt vor, wenn die modellierten aktiven Netzkomponenten nicht vollvermascht sind. Dies gilt insbesondere bei der Betrachtung von Punkt-zu-Punkt-Verbindungen.

Auf Grund dieser Eigenschaften fallen sehr viele Simulationsmodelle von Kommunikationsnetzen in die Klasse der Modelle mit eingeschränkter Ereignislokalität, für die das Simulationssystem optimiert wurde. Die Anwendung des Simulationsrechners auf die Simulation eines Medienzugriffsprotokolls folgt in Kapitel 5.3.1. Im Kapitel 5.3.2 werden Simulationen von Punkt-zu-Punkt-Verbindungen in ATM-Netzen vorgestellt, die mit und ohne die simulationsunterstützende Hardware durchgeführt wurden. Die nachfolgend zum Vergleich herangezogenen sequentiellen Simulationen werden auf einem einzelnen Knoten des Simulationsrechners ohne die Verwendung spezieller Hardware ausgeführt. Da für einen korrekten Vergleich von den erweiterten Optimierungsmöglichkeiten[6] bei sequentiellem Programmlauf Gebrauch gemacht werden muß, kommt für die sequentielle Simulation jeweils ein eigenständiges, optimiertes und unabhängiges Simulationsprogramm zum Einsatz.

5.3.1 Anwendung auf MAC-Protokolle für HSLANs

Die Entwicklung der paketorientierten lokalen Netze (LAN) verläuft von moderaten Übertragungsraten (≤ 100 Mbit/s) und relativ langen Rahmenstrukturen hin zu lokalen Hochgeschwindigkeitsnetzen (HSLAN) mit größerer geographischer Ausdehnung, Übertragungsraten von mehreren Gbit/s und kurzen Einheitszellen, die an die ATM-Technik angepaßt sind. Dadurch bedingt ergibt sich für die Leistungsbewertung solcher Netze und der zugehörigen Mediumzugriffsprotokolle (MAC Protokolle) mittels ereignisgesteuerter Simulation

6. Keine Überwachung von Ein-/Ausgabekanälen, Datenweitergabe durch Zeiger, direkter Zugriff auf die Datenstrukturen anderer Module.

zunehmend das Problem, daß die Anzahl der zu verarbeitenden Ereignisse zu groß für sequentielle Simulationsverfahren ist.

Die Simulation von lokalen Hochgeschwindigkeitsnetzen und den zugehörigen Mediumzugriffsprotokollen kann in geeigneter Weise durch das in dieser Arbeit vorgestellte Simulationssystem erfolgen. Dazu erfolgt sinnvollerweise eine injektive Abbildung der über das Hochgeschwindigkeitsnetz kommunizierenden Partner auf eine (Unter-)Menge der Simulationsknoten des Multiprozessorrechners.[7] Die Abbildung wird so gewählt, daß die Verbindungsstruktur des zu simulierenden Netzes direkt auf den Multiprozessorrechner abgebildet werden kann (siehe Kapitel 5.3.1.1), wodurch die parallele Funktionalität des realen Systems direkt in die parallele Simulation übernommen wird. Auf diese Weise wird eine relativ gleichmäßige Auslastung der Prozessoren und eine gute Skalierbarkeit erreicht, da typischerweise Netze mit bis zu hundert Kommunikationspartnern Gegenstand der Untersuchungen sind.

5.3.1.1 Einführung in lokale Hochgeschwindigkeitsnetze

Im Gegensatz zu den Weitverkehrsnetzen besitzen lokale Hochgeschwindigkeitsnetze in der Regel keine zentralen Vermittlungseinheiten und die an der Kommunikation beteiligten Stationen sind abgesehen von den Verwaltungsaufgaben gleichberechtigt. Wegen der Skalierbarkeit und zur Aufwandbegrenzung besitzen die Stationen untereinander keine Vollvermaschung sondern sie sind üblicherweise in einer Bus-, Ring-, Stern- oder Binärbaumtopologie angeordnet [KSGL95].[8]

Aufgrund der fehlenden Vollvermaschung der Stationen müssen Nachrichten gegebenenfalls über mehrere Stationen weitergereicht werden, so daß die Bandbreite der Kommunikationskanäle nicht exklusiv einer Station zur Verfügung stehen. Um Fairneß zu erzielen, muß der Zugriff auf die Übertragungsressourcen und das Übertragungsmedium durch ein Protokoll geregelt werden. Abhängig von den Übertragungsanforderungen, der Übertragungsrate und der geographischen Struktur eignen sich unterschiedliche Zugriffsprotokolle. Eine Übersicht über vorgeschlagene Protokolle und die ihnen zugrundeliegenden Mechanismen befindet sich in [vA94] und [LGS93].

7. Bei dieser Aufteilung kann auf allen Prozessoren des Multiprozessorrechners das gleiche Programm zur Ausführung gebracht werden.
8. Für den Medienzugriff und die Simulation ist die logische Topologie relevant, nicht die physikalische. Wird z.B. mit Hilfe eines Sternkopplers ein logischer Ring realisiert, so ist die physikalische Sternstruktur ohne Belang.

5.3.1.2 CRMA-II als Beispiel für ein HSLAN-Medienzugriffsverfahren

CRMA-II (Cyclic Reservation Multiple Access) wurde im IBM-Forschungslabor in Zürich für Hochgeschwindigkeitsnetze mit Übertragungsraten im Gbit/s-Bereich entwickelt [vALZZ91,vALZ92a]. CRMA-II ist ein zeitschlitzbasiertes Protokoll, welches zusammenhängende Zellen in einer Sequenz (Multi-Slots) übertragen kann. Es unterstützt die gängigen Bus- und Ringtopologien. Bei CRMA-II ist, abgesehen von Rundsendungen, stets der Empfänger einer Nachricht für deren Beseitigung vom Medium zuständig.[9] Der Zugriff auf das Medium erfolgt bei CRMA-II auf zwei verschiedene Weisen. Ohne Reservierung kann auf freie Zeitschlitze zugegriffen werden. Falls der Bedarf dadurch nicht gedeckt werden kann, müssen bei einer zentralen Station (Scheduler) Zeitschlitze reserviert werden. Diese zentrale Station, von welcher der Reservierungsmechanismus zyklisch angestoßen wird, berechnet auf der Basis der erhaltenen Reservierungen die optimale Anzahl der zu reservierenden Zeitschlitze [LvAS93] und markiert diese entsprechend. Stationen, die mehr Bandbreite als ihnen zusteht genutzt haben, dürfen auf die markierten Zeitschlitze nicht zugreifen und gegebenenfalls wird auch der Zugriff auf die freien Zeitschlitze eingeschränkt.

5.3.1.3 Simulation von CRMA-II

Für Rechnernetze mit dem Mediumzugriffsverfahren CRMA-II eignen sich aufgrund des höheren Durchsatzes und der größeren Fehlertoleranz besonders Doppelringtopologien, bei denen die Stationen jeweils an zwei gegenläufigen Ringen angeschlossen sind (Bild 5-1). Da die beiden Ringe unabhängig voneinander arbeiten, genügt es, für die simulative Untersuchung von CRMA-II nur einen einfachen Ring zu betrachten.

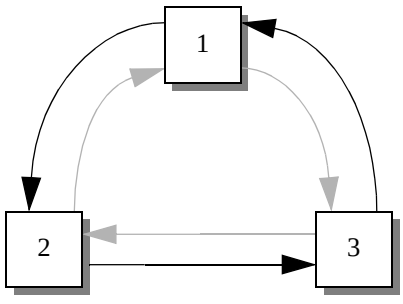


Bild 5-1: CRMA-II-Netz mit Doppelringtopologie und 3 Stationen

9. Destination Release. Bei Rundsendungen fällt das Entfernen der Nachricht in die Zuständigkeit der sendenden Station.

Bei den im folgenden betrachteten Simulationen werden CRMA-II-Stationen betrachtet, deren Verkehrsaufkommen jeweils 12,9% der Mediumbandbreite[10] entspricht und die an alle anderen Stationen mit gleicher Wahrscheinlichkeit senden. Eine detaillierte Auflistung der Simulationsparameter befindet sich im Anhang B. Weitere simulative Untersuchungen von CRMA-II befinden sich z.B. in [vALZ92b,BG93,MACea94].

Simulationszeitgewinn durch parallele Simulation mit Synchronisations- und Kommunikationsunterstützung

Bei der sequentiellen Simulation hängt die Simulationslaufzeit stark von der Anzahl der simulierten CRMA-II-Stationen ab. Bild 5-2 zeigt diesen Zusammenhang graphisch, wobei zwischen den einzelnen Datenpunkten zur Verdeutlichung des Verlaufs linear interpoliert wurde. Der durch die Verkehrsgeneratoren und das Medienzugriffsprotokoll verursachte Aufwand wächst linear mit der Anzahl der Stationen. Hinzu kommt ein nichtlinearer Anteil, der im wesentlichen durch die Kalenderverwaltung verursacht wird. Bei einer Heap-basierten Implementierung wächst der Aufwand mit O(n*log(n)), bei einem linearen Kalender quadratisch.[11]

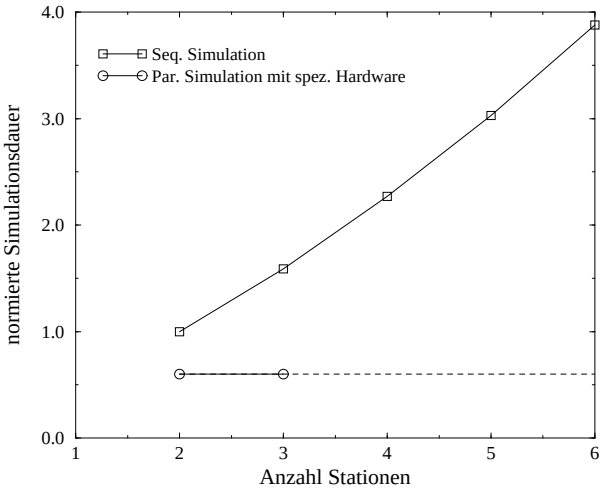


Bild 5-2: Simulationslaufzeit über der Anzahl der simulierten Stationen bei sequentieller Simulation (linearer Kalender)

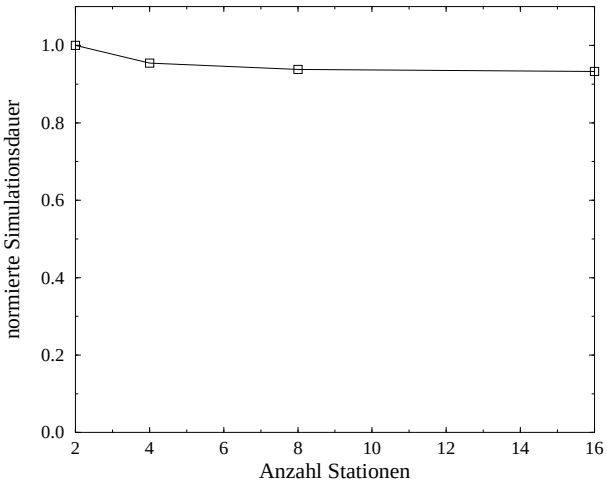


Bild 5-3: Simulationslaufzeit auf einem allgemeinen Parallelrechner mit bis zu 16 Stationen

Bei der Simulation unter Verwendung der prototypischen Realisierung stellt jede Station einen eigenen logischer Prozeß dar, der von einem exklusiv dafür verwendeten physikalischen Prozessor ausgeführt wird. Dadurch benötigt eine Simulation mit zwei Stationen etwa gleich viel

10. Die Bandbreite auf dem Medium beträgt 2,4 Gbit/s.
11. Für kleine n ist dennoch der lineare Kalender aufgrund des geringeren Aufwands bei wenigen Einträgen günstiger.

Zeit wie eine Simulation mit drei Stationen. Dies gilt auch bei der Hinzunahme weiterer CRMA-II-Stationen und ist unabhängig von der speziellen Hardware zur Unterstützung der Simulation. Da der Prototyp des Simulationssystems nur drei Simulationsknoten besitzt, wurde dies auf einem allgemeinen Parallelrechner[12] validiert (Bild 5-3[13]). Die Ausführungszeit für zwei identische parallele Simulationsläufe schwankt nur sehr geringfügig, so daß auf die Darstellung von Vertrauensintervallen in den Bildern 5-2 und 5-3 sowie in allen weiteren Bildern zur Darstellung von Simulationszeiten verzichtet wurde.

Zur Optimierung der parallelen Simulation werden zwischen den logischen Prozessen nur solche CRMA-II-Zeitschlitze verschickt, die relevante Informationen tragen. Die freien Zeitschlitze ohne Daten- und Steuerinformationen werden von dem empfangenden logischen Prozeß neu generiert. Dadurch können Phasen mit niedrigem Verkehrsaufkommen schneller simuliert werden, was sich insbesondere bei Verwendung der schnellen Nullnachrichtenschleife positiv auswirkt.

Die zeitliche Ereignislokalität wird bei der CRMA-II-Simulation durch den Abstand zwischen den einzelnen Stationen definiert, da durch die begrenzte Signalausbreitungsgeschwindigkeit eine Vorausschau möglich ist. Bild 5-4 zeigt die Simulationsdauer für die verschiedenen Simulationsverfahren in Abhängigkeit von der Stationsentfernung bei drei Stationen, Bild 5-5 den sich daraus ergebenden Simulationszeitgewinn (Speedup).

Die sequentielle Simulation zeigt bezüglich der Simulationsdauer nur eine geringfügige Abhängigkeit von der Stationsentfernung. Bei steigender Entfernung wird etwas weniger Zeit für die Simulation benötigt, da die Anzahl der CRMA-II-MAC-Kommandos von der Ringumlaufzeit und damit von der Ringgröße abhängt (Bild 5-6). Entsprechend werden bei kleinen Ringen viele Reservierungskommandos nicht benötigt (Bild 5-7), müssen aber dennoch in der Simulation bearbeitet werden. Die Wartezeit bis zum Beginn der Übertragung eines CRMA-II-Pakets ist dagegen nur wenig abhängig davon (Bild 5-8).

Bei der parallelen Simulation ohne spezielle Hardware zeigt sich eine starke Abhängigkeit von der Stationsentfernung. Bei drei Stationen wird erst bei einer Entfernung von ca. drei Kilometer ein Simulationszeitgewinn erzielt. Wird die spezielle Hardware-Unterstützung verwendet,

12. Intel Paragon.
13. Die längere Simulationsdauer bei wenigen Stationen ist u.a. darauf zurückzuführen, daß wenigen Stationen und konstantem Stationsabstand die Zyklusumlaufzeit kleiner ist und somit deutlich mehr CRMA-Steuerkommandos bearbeitet werden.

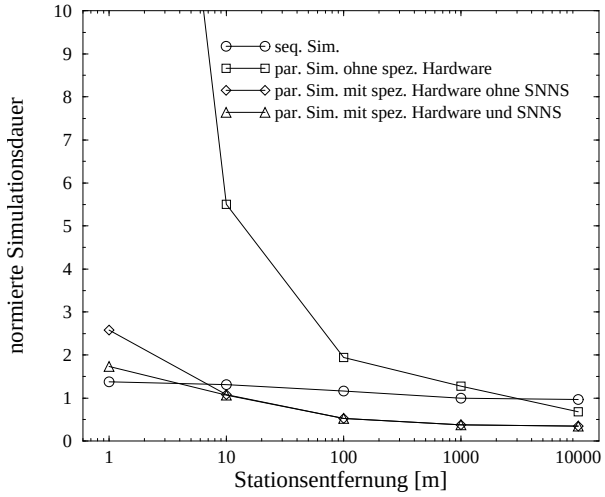


Bild 5-4: Normierte Simulationsdauer in Abhängigkeit von der Stationsentfernung

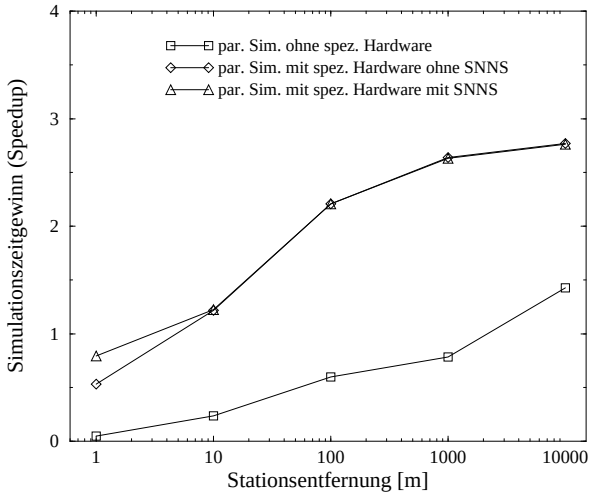


Bild 5-5: Simulationszeitgewinn in Abhängigkeit von der Stationsentfernung

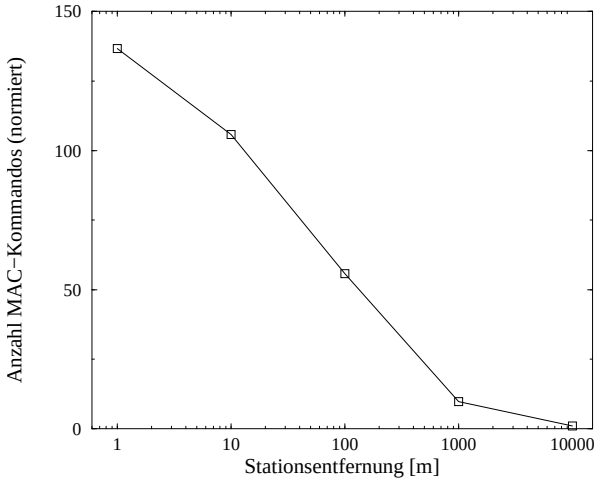


Bild 5-6: Anzahl der MAC-Kommandos in Abhängigkeit von der Stationsentfernung (Intervallänge des 95-Prozent Vertrauensintervalls nicht erkennbar klein)

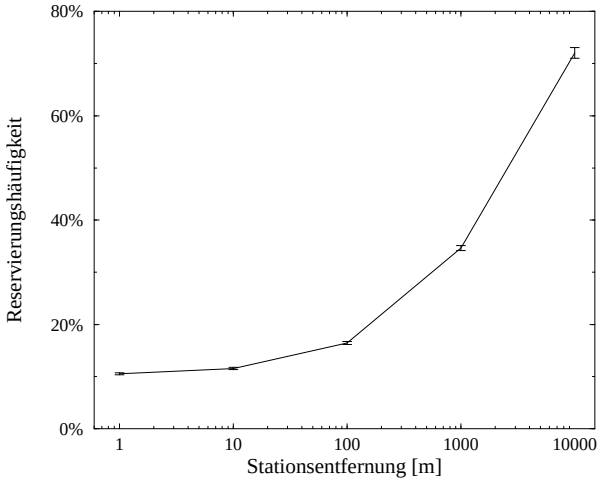


Bild 5-7: Reservierungshäufigkeit (Reservierungen je Zyklus) in Abhängigkeit von derStationsentfernung (mit 95-Prozent Vertrauensintervall)

so sinkt die Entfernung, ab der ein Simulationszeitgewinn erzielt wird, auf sieben bzw. vier Meter, je nach dem, ob die schnelle Nullnachrichtenschleife (SNNS) Anwendung findet oder nicht.

Wird keine spezielle Hardware zur Unterstützung der parallelen Simulation verwendet, so muß vom Simulationsprozessor immer dann eine Nullnachricht (NMP) generiert werden, wenn dadurch die Kanalzeit erhöht werden kann und wenn keine anderen Nachrichten zur Übertra-

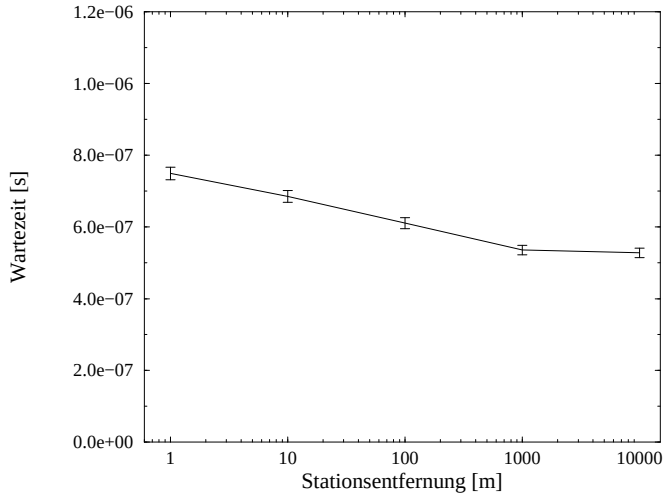


Bild 5-8: Wartezeit in Abhängigkeit von der Stationsentfernung (mit 95-Prozent Vertrauensintervall)

gung anstehen. Bei kurzer Entfernung zwischen den Stationen erhöht sich die Anzahl der erzeugten Nullnachrichten drastisch (Bild 5-9), da sich dann weniger Nachrichten zwischen den Stationen befinden können. Damit sind die Kommunikationskanäle häufiger frei und zum Fortsetzen der Simulation müssen Nullnachrichten geschickt werden. Dadurch werden die Simulationsknoten einerseits häufig blockiert, d.h. sie können keine Ereignisse bearbeiten, andererseits werden die Simulationsknoten durch die Nullnachrichten zusätzlich belastet.[14] Bild 5-10 zeigt den prozentualen Anteil der Rechenzeit eines Simulationsknotens (SPU) in Abhängigkeit von der Stationsentfernung, der für die Simulationsbearbeitung verwendet wird.

Simulationszeitgewinn durch Hardware-unterstützte Zufallszahlenerzeugung

Die Leistungsbewertung der Hardware-unterstützten Zufallszahlenerzeugung erfolgt anhand des Simulationsmodells, das auch für die Untersuchung der Kommunikations- und Synchronisationsunterstützung verwendet wurde, wobei das Modell wegen der Beschränkungen des prototypischen Simulationssystems um eine CRMA-II-Station reduziert wird. Der Abstand zwischen den Stationen beträgt 1 km. Der Simulationszeitgewinn, der mit der Hardware-unterstützten Zufallszahlenerzeugung möglich ist, hängt direkt von der Anzahl der benötigten Zufallszahlen und damit von den Verkehrsgeneratoren ab (Bild 5-11). In Bild 5-12 ist der Zusammenhang zwischen der Simulationsdauer und dem mittleren Zwischenankunftsabstand der zu übertragenden Pakete bei negativ-exponentiell verteiltem Verkehr dargestellt. Damit die

14. Der Nullnachrichten erzeugende Simulationsknoten wird relativ gesehen weniger belastet als der empfangende Simulationsknoten, da er nur dann Nullnachrichten erzeugt, wenn er keine Ereignisse bearbeiten kann und somit blockiert ist.

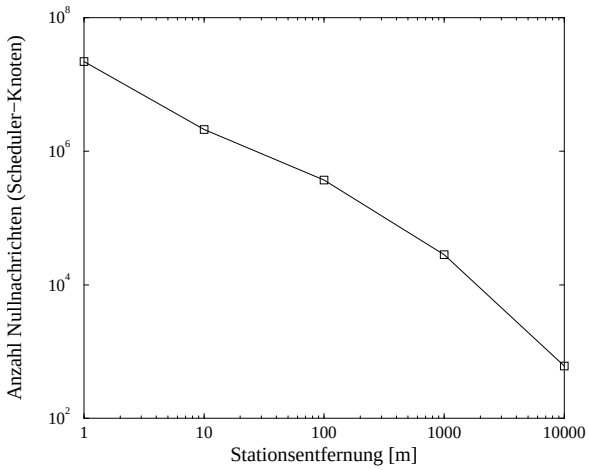


Bild 5-9: Anzahl der Nullnachrichten bei paralleler Simulation ohne spez. Hardware (Simulationsdauer: $1{,}1 \cdot 10^6$ CRMA-II-Zeitschlitze)

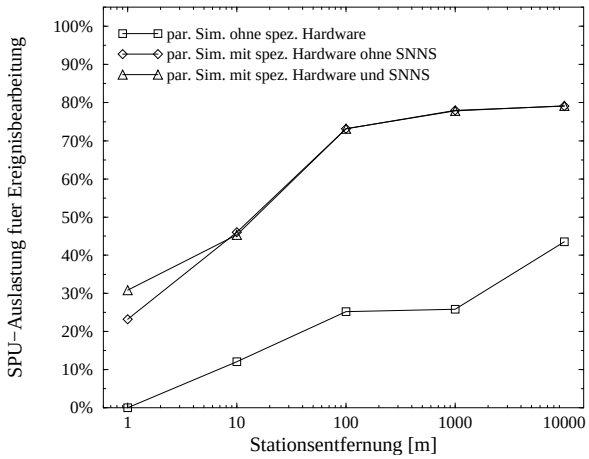


Bild 5-10: Auslastung der Simulationsknoten (Scheduler-Station, die anderen sind geringfügig weniger ausgelastet)

Zeitmessung nicht durch ein unterschiedlich stark ausgelastetes Übertragungsmedium und den korrelierenden Aufwand für das MAC-Protokoll beeinflußt wird, ist die mittlere Länge der Pakete umgekehrt proportional zum Zwischenankunftsabstand. Somit sendet jede Station unabhängig von der Paketrate mit 12,9% der Bandbreite des Übertragungsmediums.

Bei großen Paketen[15] werden so wenig Zufallszahlen benötigt, daß ihre Erzeugung keinen signifikanten Einfluß auf die Simulationsdauer hat[16]. Bei den kürzesten simulierten Paketen[17] ist jedoch ein erheblicher Aufwand für die Zufallszahlenerzeugung erforderlich, so daß die Simulation ohne Hardware-unterstützte Zufallszahlenerzeugung um nahezu 18% langsamer ist. Werden zwei CRMA-II-Stationen simuliert, ist bei der sequentiellen Simulation der absolute Simulationszeitgewinn im Verhältnis zur parallelen Simulation etwa doppelt so groß. Der Grund dafür ist, daß bei der sequentiellen Simulation der eine Prozessor doppelt so viele Zufallszahlen erzeugen muß als ein Prozessor bei der parallelen Simulation.

Zusammenfassung

Der Nutzen, der aus der Hardware-unterstützten Zufallszahlenerzeugung gezogen werden kann, hängt bei der CRMA-II-Simulation stark von den Verkehrsgeneratoren ab. Der erzielte Gewinn hängt bei paralleler und sequentieller Simulation linear von der Anzahl der erzeugten

15. Zwischenankunftsabstand 49,48 µs, Paketgröße 1920 Oktetts, entspricht 40 CRMA-II Zeitschlitzen.
16. Zufallszahlen werden für Paketgröße, Zieladresse und Zwischenankunftsabstand benötigt, d.h. für jedes Paket werden drei Zufallszahlen erzeugt.
17. Zwischenankunftsabstand 0.49 µs, Paketgröße 19,2 Oktetts, entspricht 0,4 CRMA-II Zeitschlitzen.

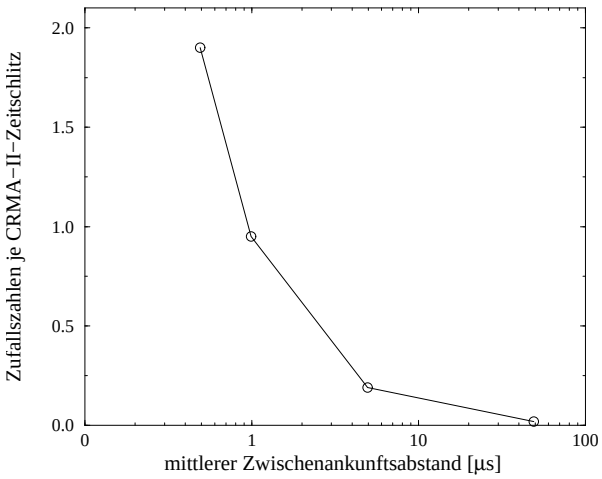


Bild 5-11: Umgekehrt proportionales Verhältnis von erzeugten Zufallszahlen und simulierten CRMA-II-Zeitschlitzen bei paralleler Simulation

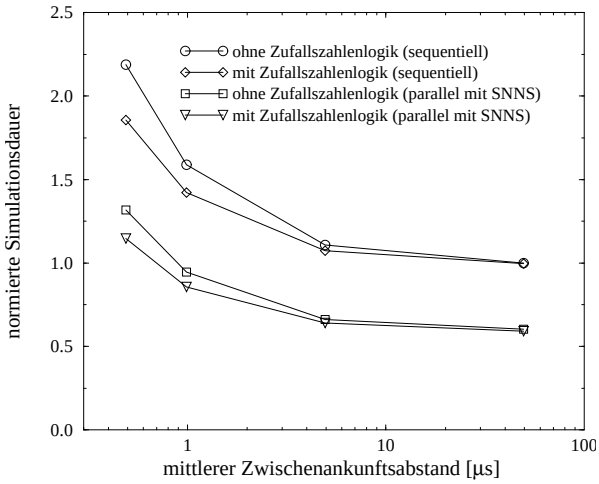


Bild 5-12: Simulationszeit in Abhängigkeit von der mittleren Zwischenankunftszeit der Verkehrsgeneratoren

Zufallszahlen ab. Wird für etwa jede zweite CRMA-II-Zelle, die verschickt wird, eine eigene Nachricht generiert, so wird ein Simulationszeitgewinn von ca. 15 Prozent erzielt.

Wird für jede CRMA-II-Station ein eigener Simulationsprozessor verwendet, so ist die Simulationsdauer unabhängig von der Anzahl der simulierten Stationen. Daher ist die parallele Simulation für sehr große Netze stets schneller als bei sequentieller Ausführung, bei der die Simulationszeit in Abhängigkeit von der Modellkomplexität mehr als linear wächst.

Die Hardware-Unterstützung bewirkt bei allen durchgeführten Simulationen einen deutlichen Geschwindigkeitsvorteil. Selbst bei Modellen, die aufgrund ihrer zeitlichen und räumlichen Ereignislokalität ohne Hardware-Unterstützung effizient simuliert werden können, wird ein Simulationszeitgewinn von rund 50 Prozent verzeichnet, was in etwa dem Kommunikationsaufwand bei der parallelen Simulation ohne spezielle Hardware entspricht. Dadurch ergibt sich für solche Simulationsmodelle eine Effizienz von nahezu Eins (vgl. Gleichung 2-2).

Bei Simulationsmodellen mit niedriger zeitlicher Ereignislokalität weist die parallele Simulation ohne Hardware-Unterstützung aufgrund des hohen Synchronisationsaufwands eine sehr hohe Simulationsdauer auf, so daß ihr Einsatz unter diesen Umständen nur bei sehr großen Modellen sinnvoll ist. Durch die Hardware-Unterstützung wird dagegen der Einsatz auch bei kleiner zeitlicher Ereignislokalität und kleinen Simulationsmodellen möglich und bereits bei mäßig großen Simulationsmodellen kann ein signifikanter Simulationszeitgewinn verzeichnet werden. Ohne spezielle Hardware benötigt beispielsweise die Simulation eines Netzes mit 50

CRMA-II-Stationen, die jeweils 1 m Abstand voneinander haben, etwas weniger Zeit als bei sequentieller Simulation. Unter Verwendung der speziellen Hardware läßt sich das gleiche Simulationsmodell dagegen 15 mal schneller simulieren.

5.3.2 Anwendung auf ATM-Netze

Der Asynchrone Transfermodus (ATM) wurde von der International Telecommunication Union (ITU) als Basis für das breitbandige, diensteintegrierende, digitale Netz (B-ISDN) ausgewählt und weitgehend standardisiert. Daneben haben sich interessierte Firmen im ATM-Forum zusammengeschlossen, von dem wichtige Impulse für die Weiterentwicklung, insbesondere für lokale und private ATM-Netze, ausgehen.

ATM-Netze werden sowohl als Weitverkehrsnetze (WAN) als auch als lokale Netze (LAN) eingesetzt. Sie besitzen in der Regel eine hohe Datenrate und teilweise komplexe Protokolle, was ihre Untersuchung mit Hilfe von sequentieller Simulation einschränkt. Da sie jedoch eine eingeschränkte Ereignislokalität aufweisen und die zeitliche Ereignislokalität durch die Distanz zwischen den aktiven Netzkomponenten bestimmt wird, lassen sich auf der Basis von ATM-Netzen leicht praxisrelevante Simulationsmodelle mit unterschiedlichen Ereigniseigenschaften erstellen und mit dem entworfenen Simulationssystem vergleichend simulieren.

Ein Überblick über die für die nachfolgenden Simulationen relevanten Eigenschaften von ATM ist in Kapitel 5.3.2.1 enthalten. Weiterführende Literatur stellen neben den ITU-Standards und den ATM-Forum-Empfehlungen ([ATMF99]) [Bla95] und [RW97] dar.

5.3.2.1 Simulationsrelevante Eigenschaften von ATM

In ATM-Netzen erfolgt die Nachrichtenübertragung in Paketen fester Größe und Struktur, die als Zellen bezeichnet werden. Die Zellen werden asynchron, d.h. ohne zeitlichen Bezug zueinander, übertragen (Bild 5-13). Dies ermöglicht das Multiplexen von Verkehrsströmen, die innerhalb der Bandbreite des zur Verfügung stehenden Kanals beliebige, nicht notwendigerweise konstante Bitraten besitzen können. Abhängig von der geforderten Verkehrsgüte kann unter Umständen durch das asynchrone Zeitmultiplexverfahren ein Multiplexgewinn erzielt werden.

ATM-Netze arbeiten verbindungsorientiert, d.h. zum Beginn einer Kommunikationsbeziehung erfolgt mit Hilfe spezieller Signalisiermeldungen ein Verbindungsaufbau [IT95b], bei wel-

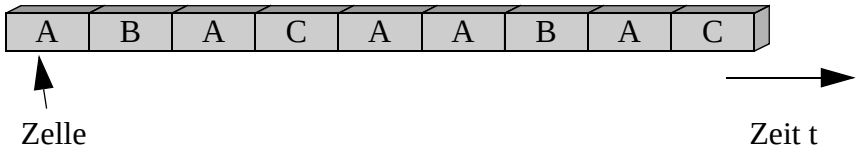


Bild 5-13: Beispiel für das asynchrones Zeitmultiplexverfahren

chem eine virtuelle Verbindung durch das Netz aufgebaut wird. Entlang des Pfades der virtuellen Verbindung werden die Zellen vom Sender zum Empfänger übertragen, wozu jeder Zelle entsprechende Informationen mitgegeben werden müssen.

In Anlehnung an [ISO94] wird in [IT91] ein Protokoll-Referenzmodell für ATM-Netze definiert (Bild 5-14). Die Basis bildet die Bit-Übertragungsschicht, für die in [IT93b] sowie in zahlreichen Veröffentlichungen des ATM-Forums eine große Vielfalt an Übertragungsmedien und -verfahren definiert wird, wobei insbesondere auch SONET/SDH-Systeme [IT98] zur Realisierung der Bit-Übertragungsschicht berücksichtigt werden.

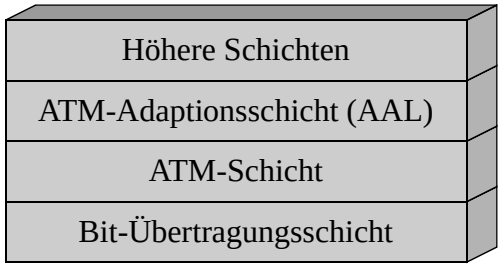


Bild 5-14: Vereinfachte Darstellung des Protokoll-Referenzmodells für ATM-Netze

In [IT95a] sind die grundlegenden Funktionen der ATM-Schicht, in der alle zellbezogenen Transportfunktionen wahrgenommen werden, sowie der Aufbau der ATM-Zellen definiert. Eine ATM-Zelle besteht aus einem 5 Oktett großen Zellkopf, der sämtliche Steuerinformationen enthält, und aus einem 48 Oktett großen Informationsfeld (Bild 5-15). Der Zellkopf ist in sechs Felder untergliedert. Das 4 Bit lange GFC-Feld kann für eine Flußsteuerung verwendet werden. Die Adreßinformation besteht aus einer Adresse für den virtuellen Pfad (VPI) und einer virtuellen Kanalnummer innerhalb des Pfades [IT93a]. Durch diese Zweiteilung vereinfachen sich zum einen die Managementaufgaben, da mit Hilfe der virtuellen Pfade mehrere virtuelle Verbindungen verwaltet werden können, zum anderen können sich einfachere Vermittlungssysteme auf das Bereitstellen von virtuellen Pfaden beschränken [Bla95]. Das PT-Feld ermöglicht die Kennzeichnung verschiedener Zelltypen. Neben den Zelltypen für die

Nutzdatenübertragung werden u.a. Management- und Signalisierzellen benötigt. Um im Überlastfall „wichtige“ Zellen von „weniger wichtigen“ Zellen unterscheiden zu können, gibt es im Zellkopf das CLP-Bit. Mit Hilfe dieses Bits können überlastete Netzkomponenten selektiv Zellen verwerfen. Der Zellkopf wird mit Hilfe einer Prüfsumme (HEC) gesichert, welche die Korrektur von 1-Bit-Fehlern erlaubt [IT93b].

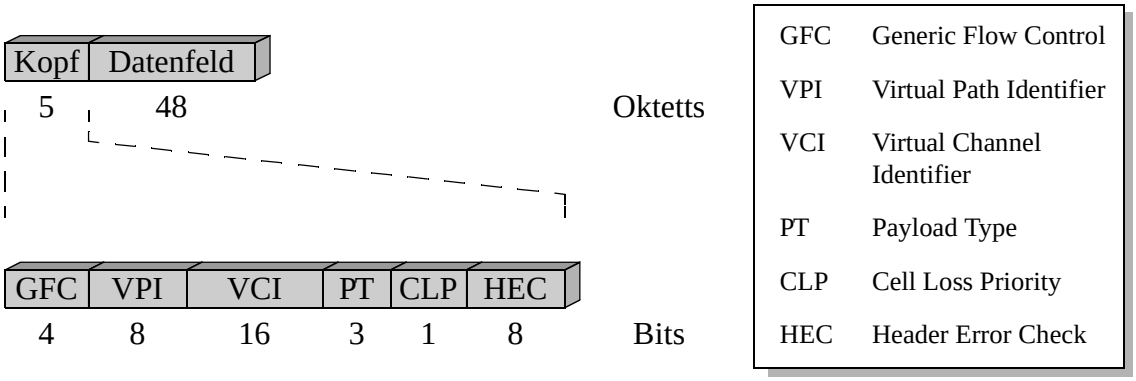


Bild 5-15: Aufbau einer ATM-Zelle

Die ATM-Adaptionsschicht (AAL) erweitert die Funktionalität der ATM-Schicht für die Nutzung durch die nächst höheren Protokollschichten. Um den unterschiedlichen Charakteristiken der Verkehrsströme gerecht zu werden, wurden insgesamt vier Adaptionstypen standardisiert [IT96a, IT96b, IT96c, IT97].

Neben den Adaptionsschichten wurden vom ATM-Forum Verbindungstypen standardisiert [ATMF99], die den unterschiedlichen Verkehrseigenschaften und Dienstgüteanforderungen Rechnung tragen. Der Verbindungstyp CBR[18] (Constant Bit Rate) garantiert für die gesamte Verbindungsdauer die festgelegte Übertragungsrate und ist daher besonders für Echtzeitanwendungen geeignet. Für Anwendungen, deren maximale Übertragungsrate deutlich über der mittleren Übertragungsrate liegt, ist der Verbindungstyp VBR (Variable Bit Rate) gedacht, da bei Verbindungen dieses Typs statistisches Multiplexen der Verkehrsströme sinnvoll ist. Dieser Verbindungstyp wird nochmals unterteilt in Verbindungen mit Echtzeitanforderungen (rt-VBR, Real-Time Variable Bit Rate) und in Verbindungen ohne Echtzeitanforderungen (nrt-VBR[19], Non-Real-Time Variable Bit Rate). Die Nutzung der Bandbreite, die aktuell nicht durch CBR- und VBR-Verbindungen belegt ist, erlaubt der Verbindungstyp ABR (Available Bit Rate). Die beim Aufbau von CBR-, VBR- und ABR-Verbindungen zwischen den Endgerä-

18. Entspricht weitgehend dem Verbindungstyp DBR (Deterministic Bit Rate) der ITU-T [IT96d].
19. Entspricht weitgehend dem Verbindungstyp SBR (Statistical Bit Rate) der ITU-T [IT96d].

ten und dem Netz jeweils ausgehandelte Dienstgüte wird dabei garantiert. Für Verbindungen, die keine Dienstgüteanforderungen besitzen, ist der Verbindungstyp UBR (Unspecified Bit Rate) vorgesehen. Der Verbindungstyp GFR[20] (Guaranteed Frame Rate) befindet sich gegenwärtig noch in der Standardisierung. Er ist für Applikationen gedacht, die eine Mindestbandbreite benötigen, höhere Bandbreiten aber gewinnbringend nutzen können. Im Gegensatz zu den anderen Verbindungstypen werden bei GFR von Netzkomponenten, die sich in Überlast befinden, ganze AAL Typ 5 Pakete verworfen und nicht nur einzelne Zellen.

In den nachfolgenden Kapiteln 5.3.2.2 und 5.3.2.4 werden die Mechanismen der für die Beispielsimulation relevanten Verkehrsklassen genauer erläutert.

5.3.2.2 Verkehrsklasse ABR

Die Verkehrsklasse ABR ist speziell auf Kommunikationsdienste abgestimmt, die sich schwankender Dienstgüte und verfügbarer Bandbreite anpassen können[21] (AAL Typ 5 [IT96c]). Durch die Bildung einer Regelschleife, über die fortlaufend die Senderate der Quelle entsprechend des beim Verbindungsaufbau ausgehandelten Verkehrsvertrags an die verfügbaren Netzkapazitäten angepaßt wird, erfolgt eine geeignete Aufteilung der Netzressourcen an die aktiven Verbindungen.

Die Regelschleife wird durch Ressource Management (RM) Zellen gebildet, die in regelmäßigen Abständen von der Quelle ausgesendet werden und den aktuellen Bandbreitenbedarf der Quelle enthalten. Sie werden entsprechend dem beim Verbindungsaufbau festgelegten Pfad von der Quelle zur Senke und von dort wieder zurück an die Quelle gesendet. Die an der Verbindung beteiligten Netzknoten und die Senke können durch zwei verschiedene Mechanismen auf die Senderate der Quelle Einfluß nehmen. Beim einfachen Binary Feedback Verfahren wird durch Setzen des Congestion Indication Bits in einer der Verbindung zugeordneten RM-Zelle eine Reduktion der Datenrate angeordnet. Ist das Bit bereits gesetzt, wird die Zelle unverändert weitergegeben, d.h. wenn mindestens ein Netzelement Überlastung signalisiert, wird dadurch die Quelle aufgefordert, die Senderate zu reduzieren.

Neben dieser binären Überlastungsanzeige haben die Netzelemente die Möglichkeit, explizit eine maximale Senderate vorzugeben und in der RM-Zelle einzutragen[22]. Jedes Netzelement

20. Wird auch als UBR+ bezeichnet.
21. Z.B. Videoströme mit bandbreiteangepaßter Auflösung, Dateiübertragung (FTP)
22. Explicit Rate Indication for Congestion Avoidance (ERICA) [ATMF99].

darf diese explizite Senderate im Rahmen des Verkehrsvertrags verringern, nicht aber erhöhen. Dadurch wird, ausgehend von der durch die Quelle vorgegebenen gewünschten Senderate, das globale Minimum entlang der Regelschleife gebildet, welches für die Quelle verbindlich ist.

5.3.2.3 Beispielsimulation: TCP über ATM-ABR

Die klassischen Internet-basierten Kommunikationsdienste[23] bauen auf dem Transmission Control Protocol (TCP) [Pos81,Bra89,Ste97,Sta98] und dem Internet Protocol (IP) [Bra89] auf. Da das Internet Protocol nur eine abschnittsweise, ungesicherte Datenübertragung und keine garantierte Dienstgüte anbietet, sind Ende-zu-Ende-Überwachung der Verbindung, Fehlererkennung, Fehlerbehebung, Flußsteuerung und Reihenfolgesicherung Aufgaben des darauf aufsetzenden Transmission Control Protocol. Durch die weite Verbreitung des Internets ist TCP zu einem De-Facto-Standard für die Rechnerkommunikation geworden. Daher ist die Integration von TCP und ATM für das Etablieren von ATM von großer Bedeutung. Sofern für die TCP-basierte Datenübertragung eine eingeschränkte Dienstgüte garantiert werden soll, bietet es sich an, den Verkehr über eine ATM-Adaptionsschicht vom Typ 5 und ABR abzuwikkeln. Dabei ist zu beachten, daß sowohl TCP als auch ABR eine komplexe Flußsteuerung beinhalten, deren Zusammenwirken die Leistungsfähigkeit des Gesamtsystems beeinflußt. Untersuchungen alternativer Übertragungsmöglichkeiten von TCP-Verkehr über ATM-Netze, z.B. unter Verwendung von UBR oder GFR, sowie der Einfluß einiger TCP-Parameter befinden sich in [Sta98,LSTea96,Bon97,Bon98], weitere Simulationen zu TCP über ATM-ABR in [Hau98,HOMM96].

Simulationsmodell

Für die Untersuchung von TCP über ATM-ABR wird ein Simulationsmodell entsprechend Bild 5-16 verwendet. Jeder der Generatoren G_i unterhält eine Kommunikationsbeziehung mit der ihm zugeordneten Senke S_i, wobei die Kommunikationskanäle zwischen den Vermittlungseinrichtungen zusätzlich durch Querverkehr belastet werden können. Den Aufbau der Generatoren zeigt Bild 5-17. Das TCP-Modul folgt neben den Anforderungen aus [Bra89] auch den aktuellen Empfehlungen aus [Ste97][24], das ABR-Modul unterstützt Binary Feedback und ERICA. Jeder Generator stellt gleichzeitig auch eine Senke und jede Senke gleichzeitig auch einen Generator dar, da die RM-Zellen von ABR und die TCP-Quittierungen in Gegen-

23. Dateiübertragung (FTP), Telnet, elektronische Post (email).
24. Slow Start, Congestion Avoidance, Fast Retransmit und Fast Recovery Algorithmen.

richtung zur Nutzdatenübertragung übermittelt werden müssen. Daher liegt keine unidirektionale Kopplung zwischen den Netzelementen vor, die eine parallele Simulation deutlich vereinfachen würde. Das Szenario ist auch nicht symmetrisch, da die Quittierungsgeneratoren in den Senken erheblich einfacher aufgebaut sind als die Verkehrsgeneratoren.

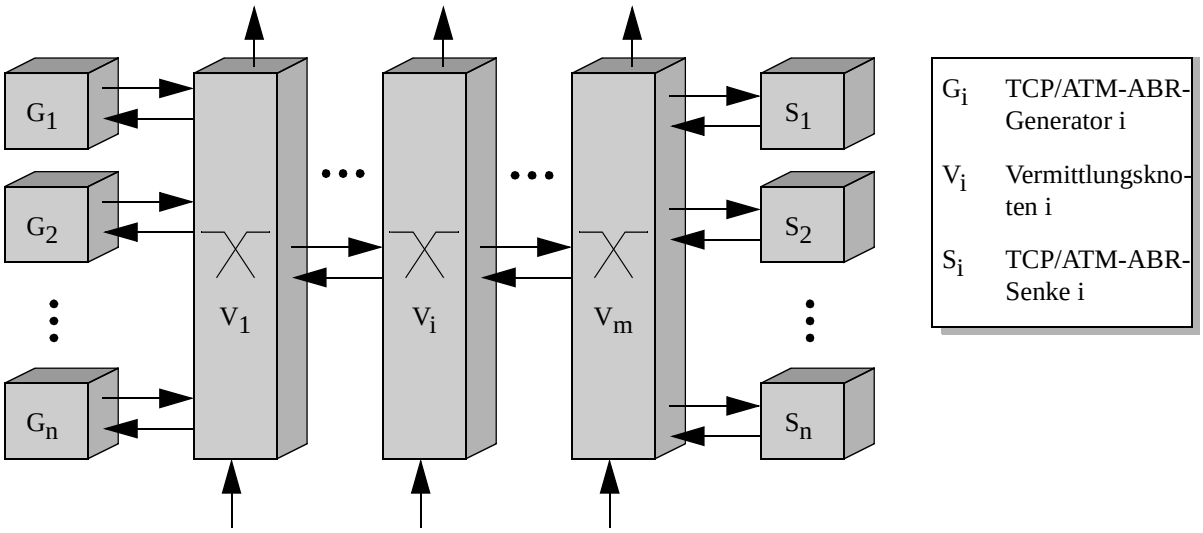


Bild 5-16: Simulationsmodell für die TCP über ATM-ABR-Simulation

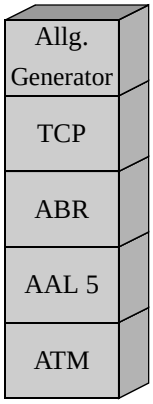


Bild 5-17: Schichtenmodell des TCP/ATM-ABR-Generators

Simulation

Zur Bewertung der in dieser Arbeit vorgestellten Simulatorarchitektur wird im folgenden das in Bild 5-16 dargestellte Modell mit vier Generatoren und vier Senken betrachtet. Für die Untersuchung der Flußsteuerungsalgorithmen werden zur Vermeidung eines stationären Zustands die Hälfte der Generatoren im 200 ms-Raster ein- bzw. ausgeschaltet. Sind die Generatoren ausgeschaltet, so können sich die übrigen Generatoren die gesamte Bandbreite der Kanäle teilen. Nach dem Wiedereinschalten liegt zunächst ein Überangebot vor, weshalb die

Ausgangswarteschlage im ersten Vermittlungsknoten anwächst. Durch die Regelmechanismen von TCP und ABR wird die verfügbare Bandbreite wieder fair unter den Verbindungen aufgeteilt. Da die Regelmechanismen davon abhängen, wie lange eine Nachricht vom Generator zur Senke und zurück benötigt, hängt auch die Größe der benötigte Warteschlange (Bild 5-18) und die Verteilung der Warteschlangenlänge davon ab (Bild 5-19). Die Verteilungsdichtefunktion weist größere Warteschlangenlängen nur sehr selten aus, woraus folgt, daß die Flußsteuerung im Verhältnis zum 200 ms Ein-/Ausschaltintervall sehr schnell reagiert.

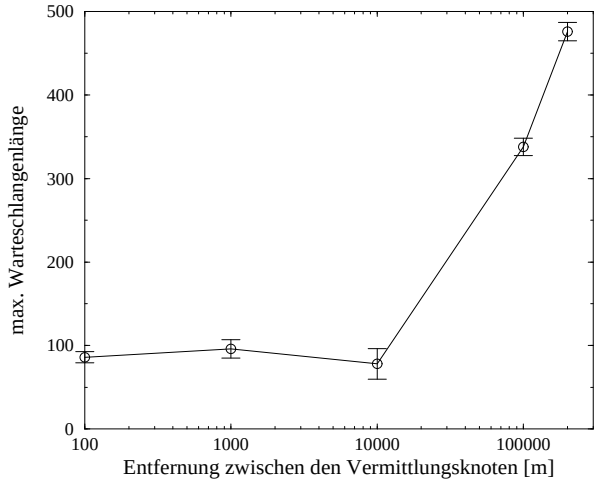


Bild 5-18: Maximale Warteschlangenlänge im ersten Vermittlungsknoten in Abhängigkeit von der Entfernung zwischen den Vermittlungsknoten

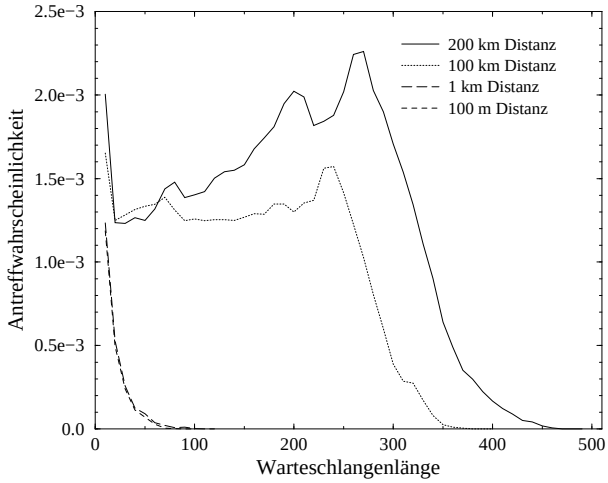


Bild 5-19: Antreffwahrscheinlichkeit für die Warteschlangenlänge bei der Ankunft einer Zelle im ersten Vermittlungsknoten für Warteschlangenlängen >0

Simulationszeitgewinn durch parallele Simulation mit Synchronisations- und Kommunikationsunterstützung

Bei Verwendung eines sequentiellen Simulationsverfahrens hängt die Simulationsdauer für das in Bild 5-16 dargestellte Simulationsmodell stark von der Anzahl der betrachteten Netzelemente ab (Bild 5-20). Bei der parallelen Simulation kann wie bei der Simulation von CRMA-II (vgl. Kapitel 5.3.1.3) die Simulation durch Nutzen entsprechend vieler Rechenknoten unabhängig von der Anzahl der Netzelemente in konstanter Zeit erfolgen, so daß der Simulationszeitgewinn (Speedup) mit der Anzahl der Stationen wächst (Bild 5-21). Bei der parallelen Simulation wurden die Generatoren und die Senken jeweils zusammen mit dem Vermittlungsknoten, an dem sie angeschlossen sind, simuliert. Bei vier Vermittlungsknoten wurden die beiden Vermittlungsknoten ohne Generatoren bzw. Senken zusammen auf einem Rechenknoten simuliert (Simulationsknoten 2), da die Auslastung dieser SPU dann in etwa der Auslastung

der SPU des Knotens mit den TCP-Generatoren entspricht (Simulationsknoten 1). Diese Konfiguration wurde auch für nachfolgende Laufzeitvergleiche herangezogen.

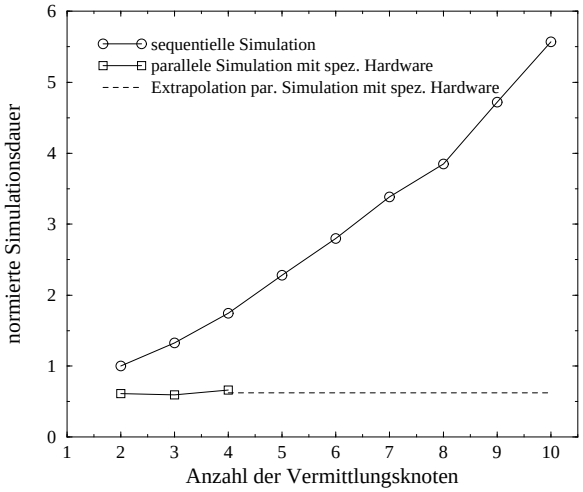


Bild 5-20: Simulationslaufzeit über der Anzahl der simulierten Vermittlungsknoten (Distanz der Vermittlungsknoten: 1 km)

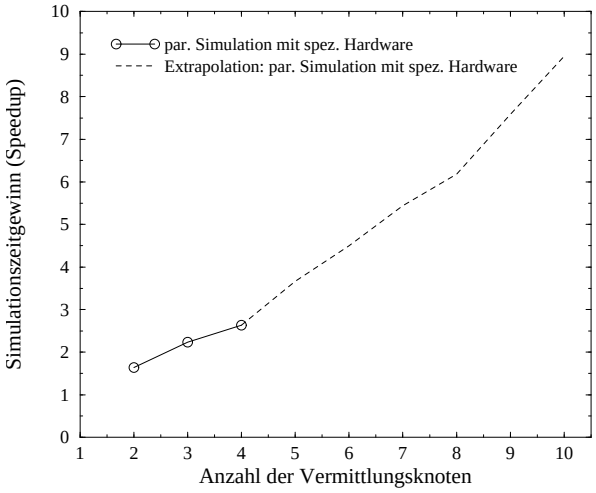


Bild 5-21: Simulationszeitgewinn über der Anzahl der simulierten Vermittlungsknoten (Distanz der Vermittlungsknoten: 1 km)

Die Bilder 5-22 und 5-23 zeigen den Einfluß des Abstands zwischen den Vermittlungsknoten auf die Simulationsdauer für unterschiedliche Simulationsverfahren. Ab einem Abstand von ca. 15 km zwischen den Vermittlungsknoten ist die parallele Simulation ohne Hardware-Unterstützung bei dem betrachteten Simulationsmodell mit vier Vermittlungsknoten schneller als die sequentielle Simulation. Sinkt die Entfernung zwischen den Vermittlungsknoten unter 1 km, dann steigt die Simulationsdauer bei paralleler Simulation ohne spezielle Hardware sehr stark an, da vermehrt Nullnachrichten für die Prozeßsynchronisation erzeugt werden müssen (Bild 5-24). Bei einer Distanz von 10 Metern werden vom mittleren Simulationsknoten, der für zwei Ausgangskanäle Nullnachrichten erzeugen muß, nahezu 60 mal so viele Nullnachrichten erzeugt als ATM-Zellen verschickt. Dadurch sinkt auch der Anteil der Prozessorleistung, der für die Ereignisbearbeitung verwendet wird (Bild 5-25).

Bei der Hardware-unterstützten parallelen Simulation kann dagegen der Simulationsprozessor selbst bei einer Entfernung von 10 Metern sinnvoll ausgelastet werden, insbesondere wenn die schnelle Nullnachrichtenschleife (SNNS) eingesetzt wird. Dadurch ist auch bei dieser kurzen Distanz und bei nur vier Vermittlungsknoten ein Simulationszeitgewinn um den Faktor 1,5 möglich (Bild 5-23). Entsprechend größer wird der Simulationszeitgewinn für umfangreichere Kommunikationsnetze. So läßt sich beispielsweise hochrechnen, daß die sequentielle Simula-

tion eines Modells nach Bild 5-16 mit 10 Vermittlungsknoten, die eine Distanz von 100 m aufweisen, ungefähr gleich lang dauert wie die parallele Simulation auf sechs Rechenknoten ohne Hardware-Unterstützung. Mit Hardware-Unterstützung erfolgt die Simulation dagegen in einem Sechstel der Zeit.

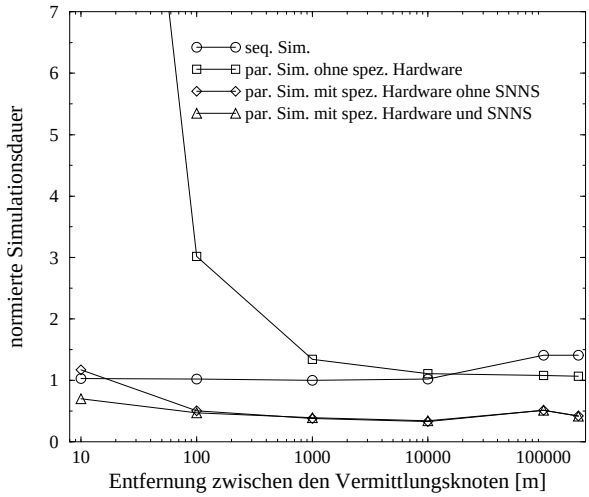


Bild 5-22: Normierte Simulationsdauer in Abhängigkeit von der Entfernung zwischen den Vermittlungsknoten

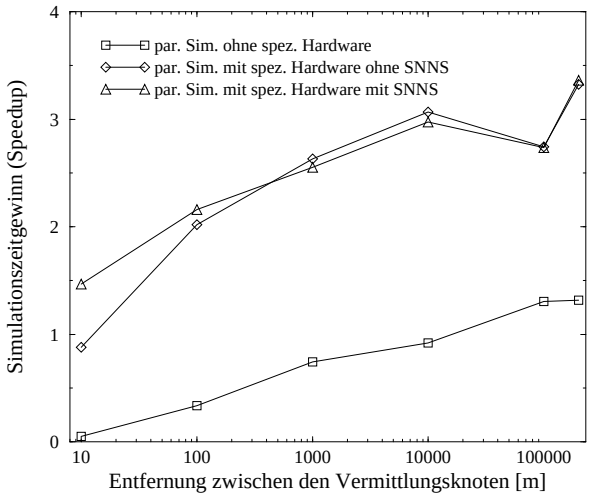


Bild 5-23: Simulationszeitgewinn in Abhängigkeit von der Entfernung zwischen den Vermittlungsknoten

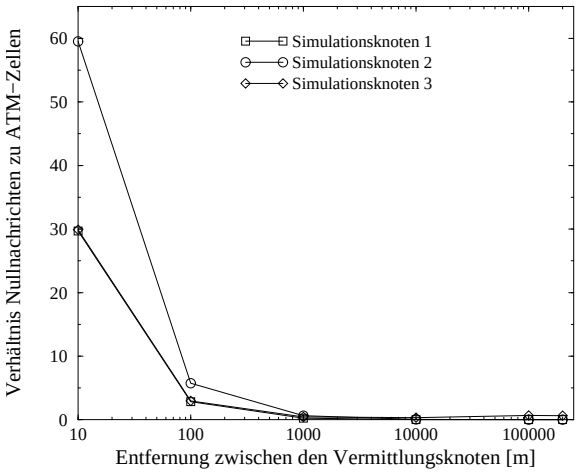


Bild 5-24: Verhältnis Nullnachrichten zu simulierten ATM-Zellen in Abhängigkeit von der Entfernung zwischen den Vermittlungsknoten (Simulation ohne spezielle Hardware)

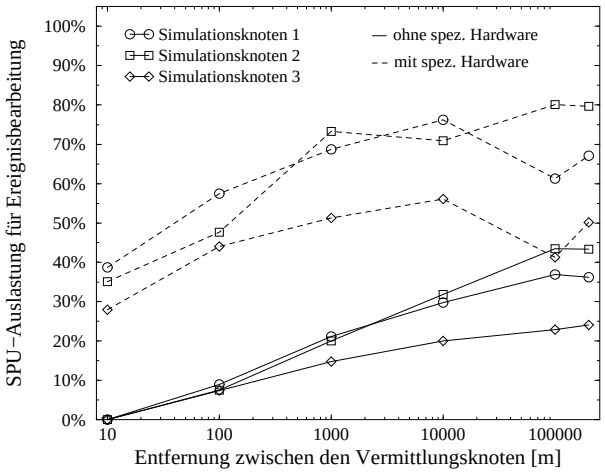


Bild 5-25: Auslastung des Simulationsprozessors in Abhängigkeit von der Entfernung zwischen den Vermittlungsknoten und dem verwendeten Simulationsverfahren

5.3.2.4 Integration von Verkehr mit niedriger Bandbreite

Der AAL Typ 2 ist für die Übertragung von kurzen Paketen mit variabler Länge bei verzögerungssensitiven Anwendungen gedacht [IT97].[25] Zur besseren Ausnutzung der ATM-Zellen wird das Multiplexen mehrerer Verbindungen in einer ATM-Zelle unterstützt. Da sich die AAL Typ 2-Pakete über die Grenzen der ATM-Zellen hinweg erstrecken können, wird in jeder ATM-Zelle als erstes Datum ein Startfeld übertragen, welches u.a. die Position des ersten Pakets in der Zelle angibt. Jedes AAL Typ 2-Paket wird mit einem 3 Oktett großen Kopf versehen, mit Hilfe dessen eine Zuordnung der Pakete zu virtuellen Kanälen möglich ist.

In einer ATM-Vermittlungseinrichtung, die keine Sonderbehandlung für AAL Typ 2-Verbindungen vorsieht, werden aufgrund der Zellvermittlung sämtliche AAL-2-Pakete einer ATM-Zelle auf dem gleichen virtuellen Pfad weitergegeben. Da dies einer effizienten Nutzung der ATM-Zellen entgegensteht, sollen zukünftige Vermittlungseinrichtungen die über AAL Typ 2-Verbindungen eingehenden Pakete gesondert vermitteln. Dazu werden die zu AAL Typ 2-Verbindungen gehörenden ATM-Zellen vor der ATM-Zellen-Vermittlung ausgekoppelt und vollständig unabhängig von den ATM-Zellen vermittelt (Bild 5-26). Die AAL Typ 2-Pakete eines jeden Ausgangskanals der Vermittlungseinrichtung werden durch Multiplexen zusammengefaßt und in ATM-Zellen übertragen.

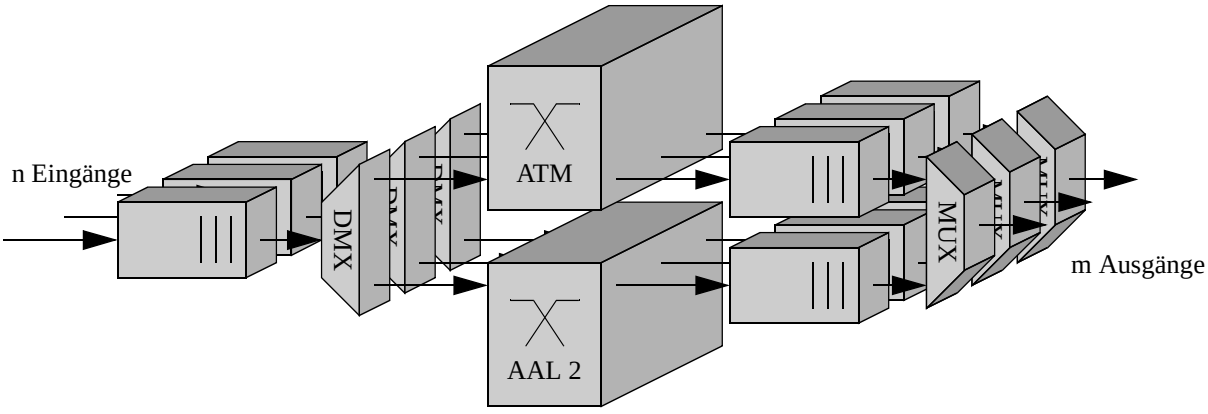


Bild 5-26: Vermittlung von AAL Typ 2-Paketen

25. Z.B. Sprache [NF98,GM98], komprimierte Bewegtbilder, Multimedia-Anwendungen [MB97].

5.3.2.5 Beispielsimulation: GSM-Sprachverkehr über AAL Typ 2

Der Einsatz der ATM-Anpassungsschicht Typ 2 (AAL 2) wird gegenwärtig vornehmlich im Zusammenhang mit Mobilfunkdiensten diskutiert, da die schmalbandige Luftschnittstelle hohe Datenraten verbietet. Aufgrund der wachsenden Bedeutung der Mobilfunknetze und ihres gesteigerten Verkehrsaufkommens wird vermehrt auf eine effiziente Übertragung der Mobilfunkdienste über das B-ISDN-Netz (ATM) geachtet. Daher werden für die nachfolgend dargestellten Simulationen Verkehrsgeneratoren und Sprechermodelle entsprechend dem GSM-Standard (Global System for Mobile Communications) verwendet.

Simulationsmodell

In [ETSI90] und [MP92] wird der sogenannte „full-rate“-Kodierer für diskontinuierliche Sprachübertragungen im GSM-Netz beschrieben. Die Sprache wird von dem Kodierer in 36 Oktett großen Paketen verschickt. Ist der Sprecher aktiv, so werden die Pakete im Abstand von 20 ms erzeugt, bei Inaktivität im Abstand von 480 ms[26]. Für die Simulation wurde der Kodierer mit Hilfe von drei Verkehrsgeneratoren modelliert (Bild 5-27). Der Generator „Sprecher“ hat zwei Zustände und schaltet die beiden anderen Generatoren wechselseitig ein bzw. aus. Die Verteilungsfunktionen können für die beiden Zustände unterschiedlich sein.

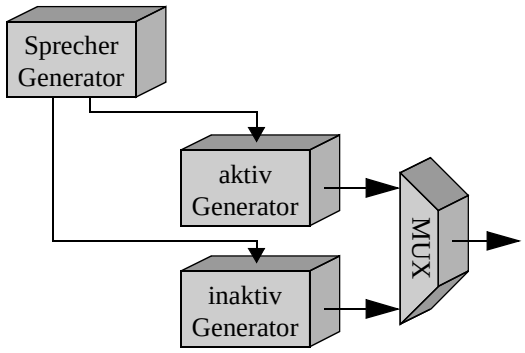


Bild 5-27: Simulationsmodell für einen GSM-Verkehrsgenerator

Bild 5-28 zeigt den Aufbau des gesamten Simulationsmodells einschließlich der Verkehrslenkung in Vorwärtsrichtung (Rückwärtsrichtung analog), wobei die Vermittlungsknoten entsprechend Bild 5-26 aufgebaut sind. Für die statistische Bestimmung der Dienstgüte wird ein Referenzpfad, der beim Generator $G_{1,1}$ beginnt und in der Senke am letzten Knoten endet,

26. Durch das Übertragen von Paketen bei Inaktivität wird erreicht, daß Hintergrundgeräusche übertragen werden können, die dem Gegenüber ein Weiterbestehen der Verbindung anzeigen.

betrachtet. Zur Erzeugung von Hintergrundverkehr werden weitere GSM-Generatoren verwendet. Beim ersten und beim letzten Vermittlungsknoten wird sämtlicher Verkehr von anderen Knoten zur Senke weitergereicht, bei den anderen Vermittlungsknoten wird die Hälfte des Verkehrs der nichtlokalen Generatoren zur Senke vermittelt, der Rest entsprechend der Verkehrslenkungstabelle an Nachbarknoten weitergegeben. Damit ergibt sich in jedem Vermittlungsknoten eine andere Abbildung der AAL Typ 2-Pakete auf die verfügbaren ATM-Zellen.

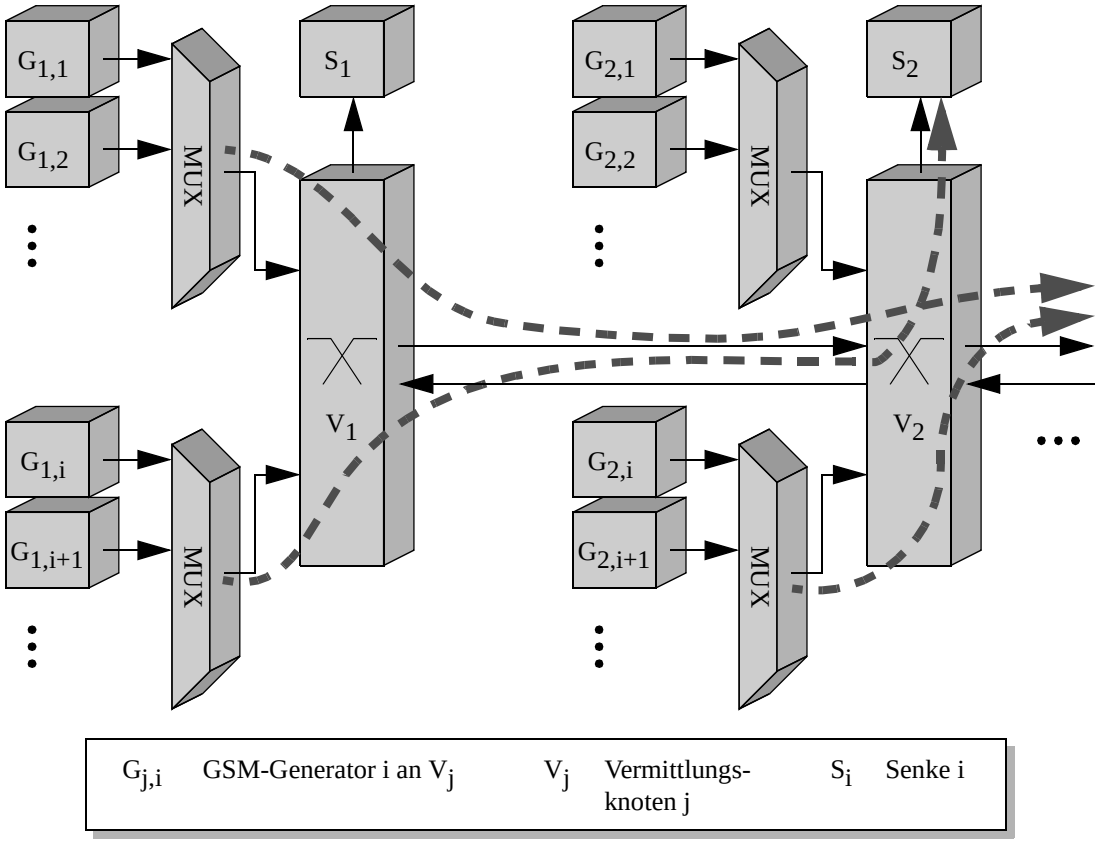


Bild 5-28: Simulationsmodell für die Simulation von GSM über AAL Typ 2 (mit Verkehrslenkung in Vorwärtsrichtung, Rückwärtsrichtung analog)

Im folgenden werden Simulationsmodelle betrachtet, die entsprechend Bild 5-28 aus drei Vermittlungsknoten und acht bis 64 GSM-Generatoren aufgebaut sind. Für die Pufferdimensionierung im Mobiltelefon und für die erzielbare Dienstgüte sind die Übertragungszeitschwankungen (Jitter) der einzelnen AAL Typ 2-Pakete von Bedeutung. Bild 5-29 zeigt die Verteilungs- und Bild 5-30 die Verteilungsdichtefunktion der Übertragungszeitschwankungen, während der GSM-Sprecher aktiv ist. Es zeigt sich eine deutliche Abhängigkeit von der Anzahl der Quellen, die auf einen ATM-Kanal gemultiplext werden. Bei wenigen Quellen

ergibt sich eine hohe Streuung, da die ATM-Zellen oft nicht voll werden und daher erst nach Auslösen der Zeitüberwachung verschickt werden. Sind mehr Quellen aktiv, tritt dieser Effekt weniger oft ein. Signifikante Verzögerungen durch kurzfristige Überlastsituationen treten bei der betrachteten Dimensionierung nicht auf, da die Verbindungen zwischen den Vermittlungsknoten zu maximal 55 Prozent ausgelastet werden[27].

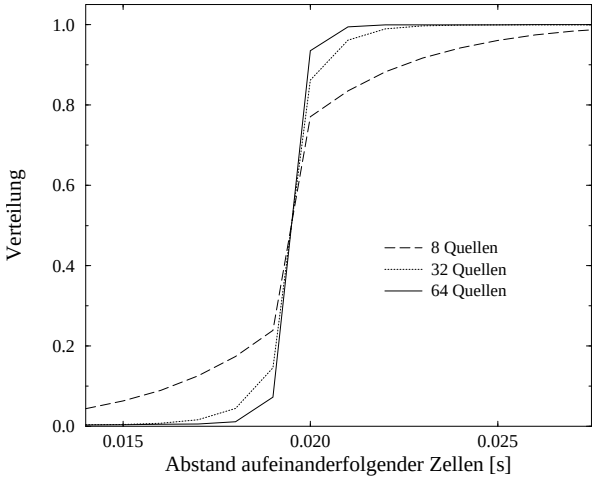


Bild 5-29: Verteilungsfunktion der Übertragungszeitschwankungen bei der Übertragung von GSM-Verkehr über AAL Typ 2

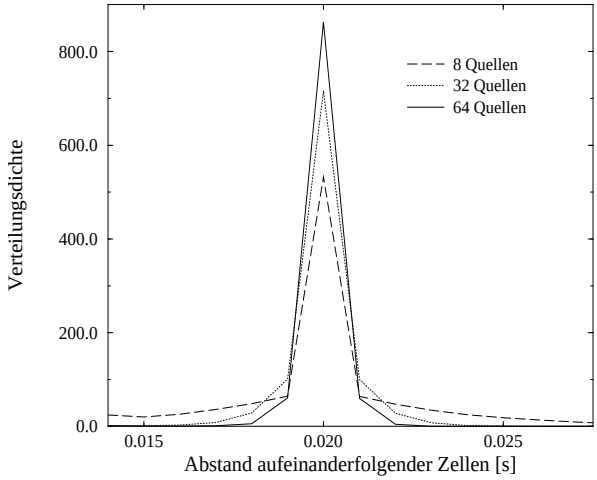


Bild 5-30: Verteilungsdichtefunktion der Übertragungszeitschwankungen bei der Übertragung von GSM-Verkehr über AAL Typ 2

Simulationszeitgewinn durch parallele Simulation mit Synchronisations- und Kommunikationsunterstützung

Die Anzahl der an einem Vermittlungsknoten angeschlossenen GSM-Verkehrsgeneratoren bestimmt bei der betrachteten Simulation wesentlich die Komplexität und damit den Rechenaufwand für das Simulationsmodell. Dies gilt sowohl für die sequentielle als auch für die parallele Simulation, bei der jeweils ein Vermittlungsknoten und die direkt an ihn angeschlossenen GSM-Generatoren von einem Prozessor simuliert werden (Bild 5-31). Aufgrund des nichtlinearen Zusammenhangs zwischen der Modellkomplexität und der Simulationsdauer ist der erzielbare Simulationszeitgewinn bei komplexen Modellen geringfügig höher (Bild 5-32).

Der Abstand zwischen den Vermittlungseinrichtungen hat gegenüber der Modellkomplexität bei Verwendung eines parallelen Simulationsverfahrens einen ungleich höheren Einfluß auf den erzielbaren Simulationszeitgewinn. Bild 5-33 zeigt die Simulationsdauer bei einem

27. 64 GSM-Verkehrsgeneratoren, die sich alle im Zustand „aktiv“ befinden.

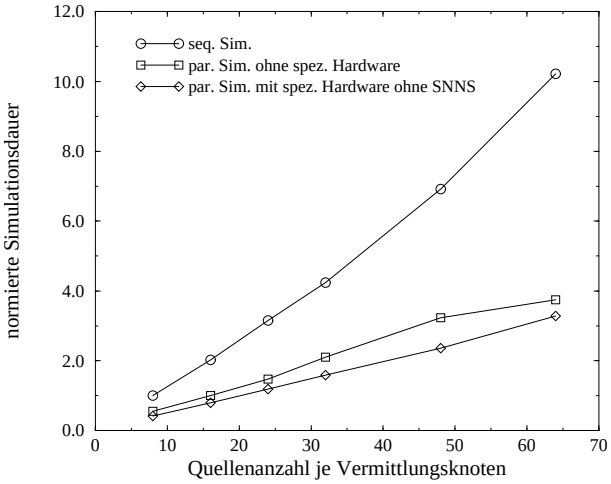


Bild 5-31: Normierte Simulationsdauer in Abhängigkeit von der Anzahl der GSM-Verkehrsgeneratoren (Vermittlungsknotendistanz: 200 km)

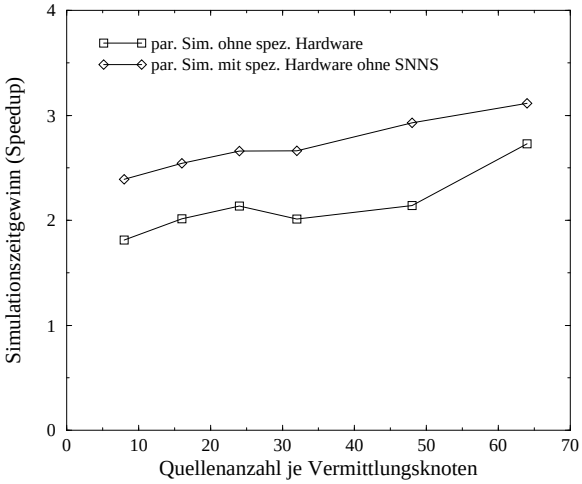


Bild 5-32: Simulationszeitgewinn in Abhängigkeit von der Anzahl der GSM-Verkehrsgeneratoren (Vermittlungsknotendistanz: 200 km)

Modell mit 32 GSM-Verkehrsgeneratoren und drei Vermittlungsknoten für unterschiedliche Simulationsverfahren. Bei den parallelen Simulationsverfahren wurden drei Rechenknoten verwendet, die jeweils einen Vermittlungsknoten und die direkt daran angeschlossenen GSM-Verkehrsgeneratoren simulieren. Aufgrund der im Simulationsmodell niedrigen Datenrate von 2 Mbit/s zwischen den Vermittlungsknoten und des hohen Aufwands für die Vermittlung der AAL Typ 2-Pakete sind die sequentielle und die parallele Simulation bereits bei einer Distanz von etwa 100 Metern zwischen den Vermittlungsknoten gleich schnell. Bei dem Simulationsmodell für die Simulation von TCP-Verkehr über ATM ist dies dagegen erst bei ca. 15 km der Fall (vgl. Kapitel 5.3.2.3, Bilder 5-22 und 5-23).

Wird bei der parallelen Simulation die spezielle Hardware-Unterstützung verwendet, so wird, wie in Bild 5-34 dargestellt, selbst bei einer Entfernung von 10 Metern zwischen den Vermittlungsknoten ein signifikanter Simulationszeitgewinn (Faktor 1,4) möglich. Ab einer Entfernung von ca. einem Kilometer wirkt sich eine weitere Distanzvergrößerung nicht mehr wesentlich auf die Simulationsdauer aus. Die Wartezeiten zur Prozeßsynchronisation und der Anteil der Rechenzeit, der ab dieser Entfernung für die Synchronisation der Prozesse aufgewendet werden muß (Bild 5-35), fallen nicht mehr ins Gewicht, und die Auslastung der Rechenknoten steigt entsprechend mit einer weiteren Vergrößerung der Entfernung zwischen den Vermittlungsknoten nicht mehr an (Bild 5-36). Ist die Entfernung jedoch so groß, daß nicht mehr alle gesendeten Nachrichten in den FIFO-Speichern zwischen den Rechenknoten gepuffert werden können, so müssen diese im Hauptspeicher des sendenden Prozessors zwi-

schengepuffert werden. Beim betrachteten Simulationsmodell ist dies bei einer Distanz von 200 km der Fall, was sich, wenn auch nur geringfügig, nachteilig auf die Leistungsfähigkeit des Simulationssystems auswirkt.

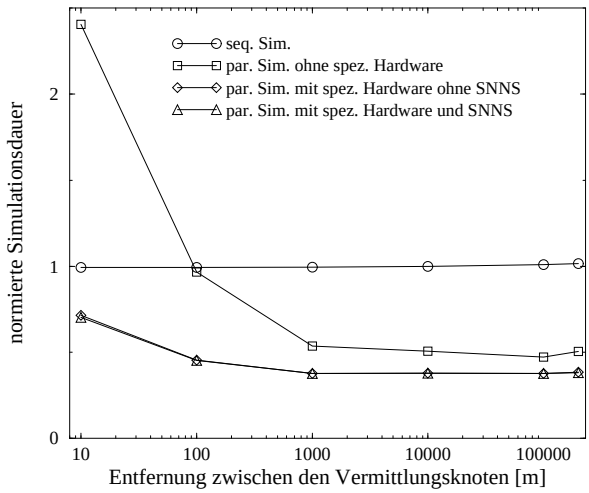


Bild 5-33: Normierte Simulationsdauer in Abhängigkeit von der Entfernung zwischen den Vermittlungsknoten

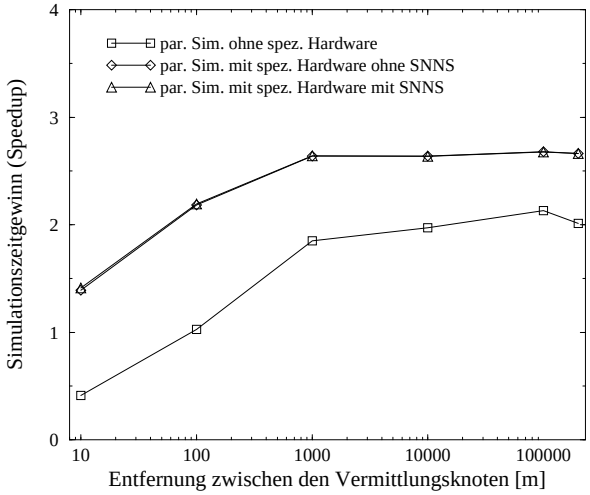


Bild 5-34: Simulationszeitgewinn in Abhängigkeit von der Entfernung zwischen den Vermittlungsknoten

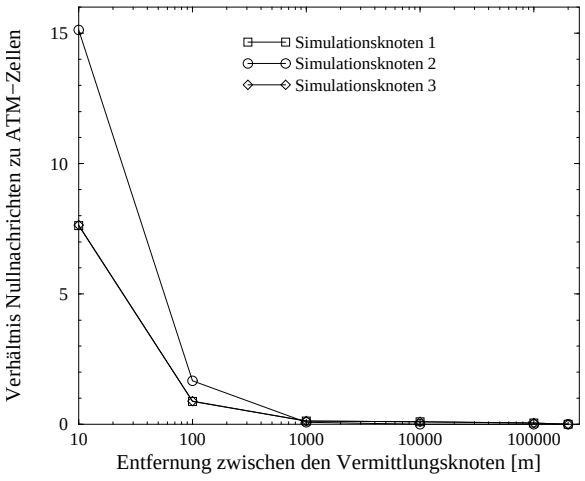


Bild 5-35: Verhältnis Nullnachrichten zu simulierten ATM-Zellen in Abhängigkeit von der Entfernung zwischen den Vermittlungsknoten (Simulation ohne spezielle Hardware)

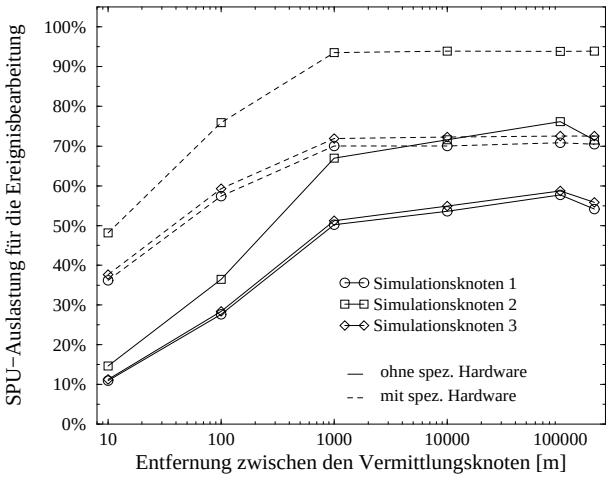


Bild 5-36: Auslastung des Simulationsprozessors in Abhängigkeit von der Entfernung zwischen den Vermittlungsknoten und dem verwendeten Simulationsverfahren

Bei großen Netzen ist der erzielbare Simulationszeitgewinn durch die parallele Simulation weitaus höher als in dem betrachteten Fall mit nur drei Vermittlungsknoten. Wie in Bild 5-37 ist die Simulationszeit bei paralleler Simulation weitgehend unabhängig von der Anzahl der

simulierten Vermittlungsknoten, sofern für jeden Vermittlungsknoten im Simulationsmodell ein Rechenknoten verfügbar ist. Bei einer Distanz von 10 km zwischen den Vermittlungsknoten ergibt sich aufgrund des überlinearen Anstiegs der Simulationsdauer bei sequentieller Simulation bei ca. acht Vermittlungsknoten ein Simulationszeitgewinn, der in etwa der Anzahl der verwendeten Recheneinheiten entspricht. Somit wird bei dieser Distanz ab acht Vermittlungsknoten eine Effizienz von mindestens Eins erreicht. Beträgt die Distanz nur 10 m, so hat dies auf die sequentielle Simulation keine wesentliche Auswirkung, die parallele Simulation mit Hardware-Unterstützung benötigt aber ungefähr die doppelte Zeit, wodurch sich die Effizienz halbiert.

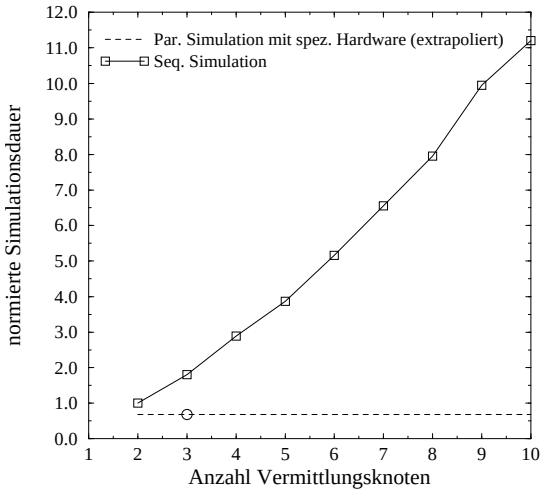


Bild 5-37: Simulationslaufzeit über der Anzahl der simulierten Vermittlungsknoten (Distanz der Vermittlungsknoten: 10 km)

Zusammenfassung

Die Simulation von GSM-Verkehr über ATM zeigt im Vergleich zur TCP über ATM-Simulation bereits bei kleineren Entfernungen deutliche Simulationszeitgewinne bei Anwendung paralleler Simulationsverfahren. Dies liegt insbesondere an der höheren räumlichen Ereignislokalität 1. Bei Entfernungen unter einem Kilometer zeigt sich ein klarer Nutzen durch die Hardware-Unterstützung bei paralleler Simulation. Bei großen Distanzen ist die parallele Simulation ohne spezielle Hardware ca. 25 Prozent langsamer als die Hardware-unterstützte Simulation. Diese Werte sind weitgehend unabhängig von der Anzahl der GSM-Verkehrsgeneratoren und damit der Komplexität des Simulationsmodells, da diese sich gleich stark auf die parallele und auf die sequentielle Simulation auswirkt.

5.4 Weitere Anwendungsbeispiele

In Kapitel 3 wurde gezeigt, welche Anforderungen ein Simulationsmodell erfüllen muß, damit es sinnvoll mit dem vorgestellten Simulationssystem bearbeitet werden kann. Neben den vorgestellten Simulationsmodellen aus dem Bereich der Kommunikationstechnik eignen sich für die Simulation auf dem Simulatorprototyp aufgrund der geforderten räumlichen Ereignislokalität 2 alle Modelle für Punkt-zu-Punkt-Betrachtungen in Kommunikationsnetzen. Bei einer Realisierung mit mehr Verbindungskanälen können auch beliebig vermaschte Netze simuliert werden, solange eine Abbildung ihrer Topologie auf das Simulationssystem möglich ist.

Bedingt durch die steigenden Taktfrequenzen und die wachsende Komplexität moderner Rechenanlagen stellt ihre Simulation vermehrt eine Herausforderung dar. Sie besitzen eine großen Anzahl an zu bearbeitenden Ereignissen, die sich auf verschiedene Verarbeitungseinheiten[28] verteilen. Da die zur Verbindung der Verarbeitungseinheiten eingesetzten System- und Peripheriebusse eine vergleichsweise lange Laufzeit haben, ergibt sich bei einer entsprechenden Aufteilung in logische Prozesse eine zeitliche Ereignislokalität, die für eine schnelle Simulation durch das vorgestellte Simulationssystem ausreicht. Dies gilt bei entsprechend höheren Taktfrequenzen in gleicher Weise für die Simulation von Logik, insbesondere wenn diese auf mehrere Bausteine verteilt ist. Dadurch ist ergänzend zur Simulation einzelner Bausteine die Simulation eines ganzen Rechnersystems möglich, wenn ein leistungsfähiges Simulationssystem mit der vorgestellten Architektur verwendet wird.

28. Prozessor, Peripherie, ...

Kapitel 6

Zusammenfassung und Ausblick

Zusammenfassung

In dieser Arbeit wurden einleitend verschiedene Methoden und Verfahren für den Entwurf und die Bewertung komplexer technischer Systeme vorgestellt sowie bezüglich ihrer Möglichkeiten und Grenzen miteinander verglichen. Dabei wurde auf die Bedeutung der Simulation für die Validierung und Leistungsbewertung hingewiesen.

Nach einer Gegenüberstellung verschiedener sequentieller und paralleler Simulationsverfahren wurden Eigenschaften von Simulationsmodellen aufgezeigt und zusammengestellt, die einen erheblichen Einfluß auf den Nutzen bei der Parallelisierung einer Simulation haben. Anhand dieser Eigenschaften läßt sich abschätzen, welche Simulationsverfahren sich jeweils eignen.

Auf der Basis der Erkenntnis, daß für eine beschleunigte parallele Simulation mit Hilfe eines konservativen Simulationsverfahrens und eines herkömmlichen Parallelrechners eine signifikante zeitliche Ereignislokalität gegeben sein muß, wurde im Kapitel 3 eine neue Simulatorarchitektur vorgestellt, die durch den Einsatz spezieller Hardware den Anwendungsbereich der konservativen parallelen Simulation erheblich erweitert.

Die in der Simulatorarchitektur enthaltene spezielle Hardware läßt sich drei Bereichen zuordnen: Der Zufallszahlenerzeugung, der Synchronisation und der Kommunikation. Die letzteren beiden sorgen dafür, daß der durch die Parallelisierung im Vergleich zur sequentiellen Simulation hinzugekommene Mehraufwand nicht durch den Simulationsprozessor erbracht werden muß, sondern daß diese Funktionen durch dedizierte, unabhängige und optimierte Einheiten zur Verfügung gestellt werden. Die Simulatorarchitektur enthält aufgrund der engen Verknüpfung von Kommunikation und Synchronisation einen speziellen Prozessor, der anhand der übertragenen Nachrichten selbständig die Prozeßsynchronisation durchführt. Diese besteht aus der lokalen Synchronisation, mit Hilfe derer die Sicherheit lokaler Ereignisse bestimmt wird, und aus der globalen Synchronisation, die durch Versenden und Auswerten von Nullnachrichten und anderen Nachrichten Verklemmungen auflöst bzw. vermeidet. Sämtliche Synchronisationsfunktionen werden für den Anwender des Simulators vollständig transparent durchgeführt, so daß hieraus kein erhöhter Aufwand bei der Simulationsprogrammerstellung

resultiert. Dies gilt in gleicher Weise auch für die Zufallszahlenerzeugung, die ebenfalls durch dedizierte Hardware und für den Anwender transparent erfolgt.

Die Umsetzbarkeit und Realisierbarkeit der vorgestellten Simulatorarchitektur wurde durch die prototypische Realisierung (Kapitel 4) unter Beweis gestellt. Dieses Simulationssystem bildet auch die Grundlage für die Leistungsuntersuchungen in Kapitel 5. Die dort dargestellten Simulationen zeigen anhand von praxisrelevanten Simulationsmodellen das Potential der in dieser Arbeit entwickelten Simulatorarchitektur. Die gezeigten Ergebnisse bezüglich der Synchronisations- und Kommunikationsunterstützung lassen sich entsprechend den Bildern 6-1 und 6-2 zusammenfassen.

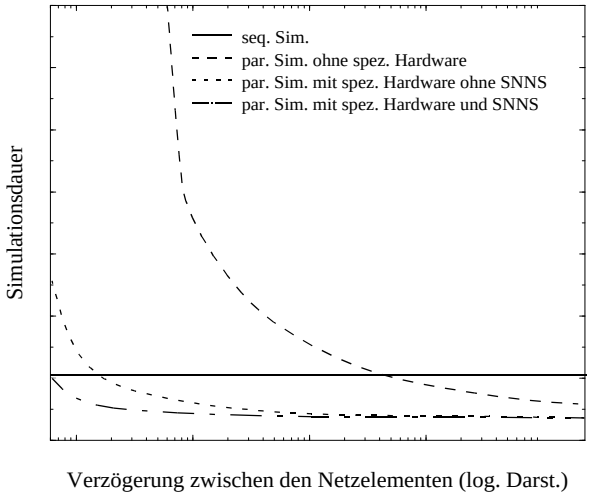


Bild 6-1: Qualitative Abhängigkeit der Simulationsdauer von der Verzögerung zwischen den Netzelementen für verschiedene Simulationsverfahren

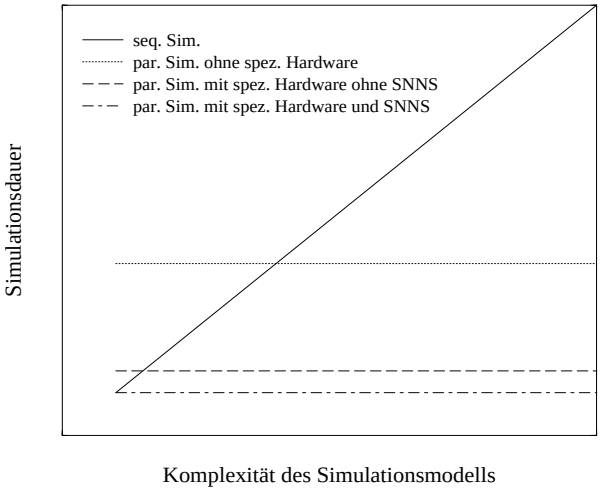


Bild 6-2: Qualitativer Zusammenhang zwischen der Modellkomplexität und erforderlichen Simulationsdauer für verschiedene Simulationsverfahren

Die Abhängigkeit der Simulationsdauer von der zeitlichen Ereignislokalität, die bei den betrachteten Modellen aus der Verzögerung zwischen den Netzelementen resultiert, wird durch die Hardware-Unterstützung signifikant verringert. Modelle mit um zwei bis drei Größenordnungen kürzeren Verzögerungszeiten lassen sich somit parallel konservativ simulieren. Die Simulationsdauer gegenüber der parallelen Simulation ohne Hardware-Unterstützung ist bei solchen Modellen um mehrere Größenordnungen kleiner. Bei der parallelen Simulation hängt die Simulationsdauer nicht von der Modellkomplexität ab, sofern das Modell sich geeignet auf ausreichend viele Simulationsknoten verteilen läßt. Im Kontext der Kommunikationsnetze bedeutet dies, daß die Netzgröße keinen Einfluß auf die Simulationsdauer hat. Die genaue Lage der in Bild 6-2 dargestellten Halbgeraden und damit der Punkt, ab dem sich die parallelen

Simulationsverfahren simulationsbeschleunigend auswirken, hängt von den Ereignislokalitäten des Simulationsmodells ab.

Der Aufwand des Simulationsprozessors für die Erzeugung einer Zufallszahl wird durch die Hardware-Unterstützung bei den betrachteten Verteilungsfunktionen um den Faktor 80 bis 300 reduziert. Der dadurch erzielbare Simulationszeitgewinn hängt unabhängig vom verwendeten Simulationsverfahren linear von der Anzahl der benötigten Zufallszahlen ab. Für die betrachteten Kommunikationsnetze und Verkehrsprofile wurde gezeigt, daß ohne die Hardware-unterstützte Zufallszahlenerzeugung die Simulation um bis zu 15 Prozent länger dauert.

Die Simulatorarchitektur besitzt keine Elemente, die von allen Simulationsknoten gemeinsam genutzt werden. Sie ist somit in weiten Bereichen skalierbar, da keine architekturbedingte Obergrenze existiert. Daß aber auch mit wenigen Knoten eine signifikante Steigerung der Simulationsgeschwindigkeit möglich ist, beweisen die in Kapitel 5 dargestellten Simulationen.

Die Eigenschaften der vorgestellten Simulatorarchitektur erlauben es somit, Simulationsmodelle, die bislang sequentiell simuliert werden mußten, effizient parallel zu simulieren. Ferner können Simulationsmodelle, die für eine beschleunigte parallele Simulation bislang ein optimistisches Simulationsverfahren benötigt haben, mit Hilfe der vorgestellten Simulatorarchitektur einfacher und schneller konservativ parallel simuliert werden.

Ausblick

Die im Rahmen dieser Arbeit prototypische Realisierung der Simulatorarchitektur erlaubt es, verschiedene Simulationsverfahren modellbezogen miteinander zu vergleichen. Um daraus den Einfluß der Modelleigenschaften und insbesondere der Ereignislokalitäten auf die Leistungsfähigkeit des Simulationssystems quantitativ bestimmen zu können, müssen die Eigenschaften des Simulationsmodells genauer untersucht werden. Hierzu sind entsprechende Statistiken notwendig, die sich jedoch bei einer Erzeugung durch den Simulationsprozessor aufgrund des dafür notwendigen Rechenaufwands auf die Ergebnisse der zu erzeugenden Statistik auswirken.[1] Daher müssen diese Statistiken durch entsprechende Hardware des Kommunikations- und Synchronsiationsprozessors erstellt werden, der Zugriff auf alle für die Bestimmung der Ereignislokalitäten notwendigen Größen hat.

1. Insbesondere läßt sich nicht ohne erheblichen Aufwand bestimmen, welche Auswirkung die Statistikerzeugung auf die synchronisationsbedingte Wartezeit der SPU hat.

Werden Modelle simuliert, bei denen ein Simulationsprozessor mehr als einen logischen Vorgänger und damit mehr als einen Eingangskanal hat, so müssen die ankommenden Pakete entsprechend ihrer Zeitstempel sortiert werden. Dies geschieht bei der vorgestellten Architektur durch den Simulationsprozessor. Verhalten sich die nachrichtenerzeugenden Modellkomponenten stark unterschiedlich, so können die Kanalzeiten der Eingangskanäle stark voneinander abweichen. In Verbindung mit einer stark unterschiedlichen Sendehäufigkeit der Modellkomponenten kann dadurch ein signifikanter Sortieraufwand für den Empfänger resultieren. Dies läßt sich durch eine Erweiterung des Kommunikationsprozessors in Verbindung mit einer Integration der Kalenderverwaltung verhindern. Dazu wird für jeden logischen Vorgängerprozeß ein eigener FIFO-Speicher vorgesehen. Beim Empfang eines Paketes über einen beliebigen Pfad wird dieses abhängig von der im Paketkopf gespeicherten Absenderadresse in den entsprechenden FIFO-Speicher eingetragen. Eine Zeitvergleichsstufe schreibt die Pakete aus den FIFO-Speichern, die den logischen Vorgängerprozessen zugeordnet sind, sortiert in einen FIFO-Empfangsspeicher des Simulationsprozessors. Gleichzeitig wird das jeweils älteste Paket in den Kalender eingetragen. Mit dieser Erweiterung des Kommunikationsprozessors wird auch die Forderung der räumlichen Ereignislokalität 2 sowie der Abbildbarkeit der zu simulierenden Topologie auf die des Simulationssystems aufgehoben, da dadurch beliebig viele logische Vorgängerprozesse unterstützt werden können.

Eine weitere Optimierung der Architektur ist durch Integration von schneller Nullnachrichtenschleife und Zeitsprung (Kapitel 3.3.1.2) möglich. Die schnelle Nullnachrichtenschleife vermindert nicht die Anzahl der Nullnachrichten, die für das Auflösen einer Verklemmung benötigt werden, sie erzeugt diese nur sehr schnell. Dagegen ist der programmgesteuerte Zeitsprung erheblich langsamer und eignet sich daher nur für sehr große Zeitsprünge. Durch die Definition eines weiteren Nullnachrichtenformats[2], das neben der aktuellen lokalen Simulationszeit auch die Eintrittszeit des nächsten lokal bekannten Ereignisses enthält, läßt sich der Zeitsprung mit begrenztem Aufwand im Kommunikationsprozessor durch dedizierte Hardware integrieren. Damit ist eine erheblich schnellere und häufigere Ausführung eines Zeitsprungs möglich. Simulationsmodellen, die keine große zeitliche Ereignislokalität besitzten, können auf diese Weise in noch kürzerer Zeit simuliert werden.

Unabhängig von den Erweiterungen der Simulatorarchitektur ist durch Simulationen, die nicht dem Umfeld der Kommunikationsnetze entstammen, der Nutzen der Architektur für weitere

2. Im hier vorgestellten Kommunikationsprotokoll gibt es dafür einen reservierten Pakettyp.

Aufgabenfelder zu zeigen. Hierzu zählt insbesondere die Simulation komplexer Rechnersysteme und ihrer Komponenten.

Um auf der Basis der vorgestellten Simulatorarchitektur ein kommerzielles Produkt zu entwikkeln, sollte der Kommunikations- und Synchronisationsprozessor höher integriert werden. Ferner ist eine Realisierung sinnvoll, bei welcher jeder Simulationsknoten aus einem standardisierten Mikrorechner, z.B. PC, und einer Einsteckkarte, welche die simulationsspezifische Hardware enthält, besteht. Ein solches System eignet sich zwar nicht so gut für die Bewertung der Simulationsverfahren wie die in Kapitel 4 beschriebene prototypische Realisierung, da durch das Betriebssystem und andere Prozesse weitere unbekannte Einflüsse auf die Simulationsdauer hinzu kommen, doch lassen sich so leichter Systeme schaffen, die aktuellen Prozessorentwicklungen folgen können und Anwendern eine gewohnte Bedienoberfäche bieten.

Kapitel 7

Literaturverzeichnis

[AB82] Akyildiz, I. F.; Bolch, G.: *Möglichkeiten und Grenzen der analytischen Modellbildung von Warteschlangensystemen*, Informatik Fachberichte Band 56, Springer Verlag Hamburg, D, 1982, S. 66–78.

[AB93] Ayani, R.; Berkman, B.: *Parallel Discrete Event Simulation on SIMD Computers*, Journal of Parallel and Distributed Computing, Vol. 18, USA, 1993, pp. 501–508.

[AILea95] Aynai, R.; Ismailov, Y.; Liljenstam, M.; Popescu, A.; Rajaei, H.; Rönngren, R.: *Modeling and Simulation of a High Speed LAN*, Simulation, Vol. 64, No. 1, USA, January 1995, pp. 7–14.

[ATMF99] ATM-Forum: *Traffic Management Specification Draft Version 4.1*, ATM-Forum, February 1999.

[Bai90] Baitinger, U. G.: *Skript zur Vorlesung „Rechnergestützter Schaltungsentwurf“*, Universität Stuttgart, D, 1990.

[Bar83] Barel, M.: *Ein Multiprozessorsystem für die stochastische ereignisorientierte Simulation*, Dissertationsschrift, RWTH Aachen, D, 1983.

[Bar84] Barel, M.: *The discrete event simulation computer - DESC*, ACM Simuletter, Vol. 15, No. 2, USA, April 1984, pp. 9–15.

[Bar85] Barel, M.: *A high speed list processor for discrete event multiprocessor-simulators*, ACM Simuletter, Vol. 16, No. 4, USA, 1985, pp. 20–27.

[Bau94] Bauer, H.: *Verteilte diskrete Simulation komplexer Systeme*, Berichte aus der Informatik, Verlag Shaker, Aachen, D, 1994.

[BBCea94] Beaumont, C.; Boronat, P.; Champeau, J.; Filloque, J. M.; Pottier, B.: Reconfigurable Technology: *An innovative Solution for Parallel and Discrete Event Simulation Support*, Proceedings of the 8th Workshop on Parallel and Distributed Simulation (PADS ’94), Edinburgh, Scotland, GB, July, 6-8, 1994, pp. 160–163.

[BD97] Boukerche, A.; Das, S. K.: *Dynamic Load Balancing Strategies for Conservative Parallel Simulations*, Proceedings of the 11th Workshop on Parallel and Distributed Simulation (PADS ’97), Lockenhaus, A, June 10-13, 1997, pp. 20–28.

[BDDea96] Bartoli, A.; D'Andrea, N. A.; Dini, G.; Lottici, V.; Luise, M.; Prete, C. A.: *Parallelization of the Simulation Tool Wing-Space for the Analysis and Design of Transmission Systems*, International Journal of Communication Systems, Vol. 9, No. 4, John Wiley & Sons, GB, July/August 1996, pp. 213-222.

[BG93] Bach, C.; Grebe, A.: *Comparison and performance evaluation of CRMA-II and ATMR*, Proceedings of the EFOC&N '93, The Hague, NL, June 30-July 2, 1993.

[BJ85] Berry, O.; Jefferson, D.: *Critical path analysis of distributed simulation*, Distributed Simulation, Vol. 15, No. 2, San Diego, CA, USA, January 1985, pp. 57–60.

[Bla95] Black, U.: *ATM: Foundation for Broadband Networks*, Prentice Hall Series in Advanced Communications Technologies, Prentice Hall, Englewood Cliffs, USA, 1995.

[Bon97] Bonaventure, O.: *A simulation study of TCP with the proposed GFR service category*, Proceedings of the Dagstuhl Seminar 9725: High-Performance Networks for Multimedia Applications, Dagstuhl, June, 15-20, 1997, D, Kluwer Academic Publishers, June 1997.

[Bon98] Bonaventure, O.: *A flexible buffer acceptance algorithm to support the GFR service category in ATM switches*, Sixth IFIP workshop on Performance Modelling and evaluation of ATM networks, July, 20-22, 1998, Part 2 - Research Papers, Ilkley, GB, pp. 71/1–71/10.

[Bra89] Braden, R.: *Requirements for Internet Hosts – Communication Layers RFC 1122*, Internet Engineering Task Force, October 1989.

[BRF90] Buzzell, C. A.; Robb, M. J.; Fujimoto, R. M.: *Modular VME Rollback Hardware for Time Warp*, Proceedings of the SCS Multiconference on Distributed Simulation, San Diego, CA, USA, January, 17-19, 1990, pp. 153–156.

[BS93] Bauer, H.; Sporrer, C.: *Reducing Rollback Overhead in Time-Warp Based Distributed Simulation with Optimized Incremental State Saving*, Proceedings of the 26th Annual Simulation Symposium (ASS '93), Washington, D. C., USA, March 29 - April 1, 1993, pp. 12–20.

[CEGL82] Christopher, T.; Evens, M.; Gargeya, R. R.; Leonhardt, T.: *Structure of a distributed simulation system*, Proceedings of the 3rd IEEE International Conference on Distributed Computing Systems, Miami, Fort Landerdale, October, 18-22, 1982, pp. 584–589.

[CFE94] Carothers, C. D.; Fujimoto, R. M.; England, P.: *Effect of Communication Overheads on Time Warp Performance: An Experimental Study*, Proceedings of the 8th Workshop on Parallel and Distributed Simulation (PADS '94), Edinburgh, Scotland, GB, July, 6-8, 1994, pp. 118-125.

[CJB97] Chen, Y.; Jha, V.; Bagrodia, R.: *A Multidimensional Study on the Feasibility of Parallel Switch-Level Circuit Simulation*, Proceedings of the 11th Workshop on Parallel and Distributed Simulation (PADS ’97), Lockenhaus, A, June 10-13, 1997, pp. 46–54.

[CM79] Chandy, K. M.; Misra, J.: *Distributed simulation: a case study in design and verification of distributed programs*, IEEE Transactions on Software Engineering, Vol. 5, No. 5, USA, September 1979, pp. 440–452.

[CMH83] Chandy, K. M.; Misra, J.; Haas, L. M.: *Distributed Deadlock Detection*, ACM Transaction on Computer Systems, Vol. 1, No. 2, USA, May 1983, pp. 144–156.

[Con85] Concepcion, A. I.: *Mapping distributed simulators onto the hierarchical multi-bus multiprocessor architecture*, Distributed Simulation, Vol. 15, No. 2, San Diego, CA, USA, January 1985, pp. 8–13.

[Con89] Concepcion, A. I.: *A hierarchical computer architecture for distributed simulation*, IEEE Transactions on Computers, Vol. 38, No. 1, USA, 1989, pp. 311–319.

[Cou85] Courtois, P. J.: *On time and space decomposition of complex structures*, Communications of the ACM, Vol. 28, No. 6, USA, 1985, pp. 590–603.

[CPSV91] Ciccarella, G.; Pignatelli, R.; Susanna, L.; Valent, F.: *Parallel Simulation of DQDB MAN Protocol*, Proceedings of the EFOC/LAN ’91, London, UK, June 1991, pp. 23–27.

[CT97] Cleary, J. G.; Tsai, J. J.: *Performance of a Conservative Simulator of ATM Networks*, Proceedings of the 11th Workshop on Parallel and Distributed Simulation (PADS’97), Lockenhaus, A, June 10-13, 1997, pp. 142-145.

[Cyp93] Cypress: *High Performance Data Book*, CA, USA, August 1993.

[Dav86] Davis, B. R.: *An Improved Importance Sampling Method for Digital Communication System Simulations*, IEEE Transactions on Communications, Vol. 34, No. 4, USA, 1986, pp. 715–719.

[DFW94] D’Souza, L. M.; Fan, X.; Wilsey, P. A.: *pGVT: An Algorithm for Accurate GVT Estimation*, Proceedings of the 8th Workshop on Parallel and Distributed Simulation (PADS ’94), Edinburgh, Scotland, GB, July, 6-8, 1994, pp. 102–109.

[DSYB90] Dupuy, A.; Schwartz, J.; Yemini, Y.; Bacon, D.: *NEST: a network simulation and prototyping testbed*, Communications of the ACM, Vol. 33, No. 10, USA, 1990, pp. 63–74.

[DT93] Devetsikiotis, M.; Townsend, J. K.: *An algorithmic Approach to the Optimization of Importance Sampling Parameters in Digital Communication System Simulation*, IEEE Transactions on Communications, Vol. 41, No. 6, USA, June 1993, pp. 1464–1473.

[Dud93] Dudenverlag: *DUDEN Informatik (2. Auflage)*, Dudenverlag, Mannheim, D, 1993.

[Ens98] Enssle, J.: *Modellierung und Leistungsuntersuchung eines verteilten Video-On-Demand-Systems für MPEG-codierte Videodatenströme mit variabler Bitrate*, 68. Bericht über verkehrstheoretische Arbeiten, Institut für Nachrichtenvermittlung und Datenverarbeitung, Universität Stuttgart, März 1998.

[Ern99] Erne, H.: *Entwurf und Implementierung eines Gerätetreibers für eine Hochgeschwindigkeitsschnittstellenkarte unter Mircrosoft Windows NT 4.0*, Diplomarbeit, Institut für Nachrichtenvermittlung und Datenverarbeitung, Universität Stuttgart, März 1999.

[ES90] Ellis, M. A.; Stroustrup, B.: *The annotated C++ Reference Manual*, Addison-Wesley Pubishing Company, Reading, USA, 1990.

[ETSI90] ETSI/TC: *GSM Recommendation 06.31: Discontinuous Transmission (DTx) for full-rate speech traffic channels*, January 1990

[FBKA89] Frater, M. R.; Bitmead, R. R.; Kennedy, R. A.; Anderson, B. D. O.: *Fast Simulation of Rare Events Using Reverse-Time Models*, ITC Specialists Seminar, Adelaide, AUS, September 1989.

[Fly66] Flynn, M. J.: *Very High-Speed Computing Systems*, Proceedings of the IEEE, Vol. 54, No. 12, USA, December 1966, pp. 1901–1909.

[FM93] Frost, V. S.; Melamed, B.: *Traffic Modeling for Telecommunications Networks*, IEEE Communications Magazine, New York, USA, March 1994, ISSN 0163-6804, pp. 70–81.

[FN92] Fujimoto, R.; Nicol, D.: *State of the Art in Parallel Simulation*, Proceedings of the Winter Simulation Conference 1992, Arlington, Virg., USA, December 1992, pp. 246–254.

[Frö94] Fröhlings, U.: *Adaptive Verfahren in der optimistischen verteilten Simulation*, Workshop über parallele und verteile Simulation - Bericht Nr. AIS 21, Fachbereich Informatik, Universität Oldenburg, D, Dezember 1995, S. 26–31.

[FTG88a] Fujimoto, R. M.; Tsai, J. J.; Gopalakrishnan, G.: *The roll back chip: hardware support for distributed simulation using time warp*, Distributed Simulation, Vol. 19, No. 3, San Diego, CA, USA, 1988, pp. 81–86.

[FTG88b] Fujimoto, R. M.; Tsai, J. J.; Gopalakrishnan, G.: *Design and performance of special purpose hardware for time warp*, Proceedings of 15th annual international symposium on computer architecture, Honolulu, Hawaii, USA, May-June, 1988, pp. 401–408.

[Fuj88] Fujimoto, R. M.: *Lookahead in Parallel Discrete Event Simulation*, Proceedings of the International Conference on Parallel Processing, 1988, pp. 34–41.

[Fuj89] Fujimoto, R. M.: *Performance Measurements of Distributed Simulation Strategies*, Transactions of the Society for Computer Simulation, Vol. 6, No. 2, 1989, ISSN 0740-6797, pp. 89–132.

[Fuj90a] Fujimoto, R. M.: *Parallel discrete event simulation*, Communications of the ACM, Vol. 33, No. 10, USA, 1990, pp. 30–53.

[Fuj90b] Fujimoto, R. M.: *Performance of Time Warp under Synthetic Workloads*, Proceedings of the SCS Multiconference on Distributed Simulation, San Diego, CA, USA, January 17-19, 1990, pp. 23–28.

[GF98] Görg, C.; Fuß, O.: *Comparison and Optimization of RESTART Run Time Strategies*, Archiv für Elektronik und Übertragungstechnik, Band 52, Nr. 3, Hirzel-Verlag, D, Mai 1998, Seiten 197–204.

[GH81] Gross, D.; Harris, C. M.: *Fundamentals of Queueing Theory*, John Wiley and Sons, New York, USA, 1981.

[GH85] Gross, D.; Harris, C. M.: *Fundamentals of Queueing Theory: Second Edition*, Wiley Series in Probability and Mathematical Statistics, John Wiley & Sons, New York, USA, 1985.

[GM98] Gerlich, N.; Menth, M.: *The Performance of AAL-2 carrying CDMA Voice Traffic*, Research Report No. 199, Institute of Computer Science, Universität Würzburg, D, April 1998.

[Gör98] Görg, C.: *Verkehrstheoretische Modelle und stochastische Simulationstechniken zur Leistungsanalyse von Kommunikationsnetzen*, Aachener Beiträge zur Mobil- und Telekommunikation (Band 13), RWTH Aachen, D, März 1998.

[GPM81] Georgiadis, P. I.; Papazoglou, M. P.; Maritsas, D. G.: *Towards a parallel SIMULA machine*, Proceedings of the 8th annual symposium on computer architecture, Minneapolis, USA, May, 12-14, 1981, pp. 263–277.

[Gru94] Gruschwitz, R.: *Einfluß der Kommunikationszeit auf den Speedup der parallelen Logiksimulation für Transputernetze und Workstation-Cluster*, Workshop über parallele und verteile Simulation - Bericht Nr. AIS 21, Fachbereich Informatik, Universität Oldenburg, D, Dezember 1995, S. 14–19.

[GT91] Groselj, B.; Tropper, C.: *The distributed simulation of clustered processes*, Distributed Computing, Vol. 4, No. 4, 1991, pp. 111–121.

[GT93] Glazer, D. W.; Tropper, C.: *On process migration and load balancing in Time Warp*, IEEE Transactions on Parallel and Distributed Systems, Vol. 4, No. 3, USA, March 1993, pp. 318–327.

[Hab92] Habermann, R.: *Die Simulation von Vermittlungssystemen unter Anwendung von Techniken der parallelen Datenverarbeitung*, Dissertationsschrift, Fakultät für Maschinenwesen, Universität Hannover, D, 1992.

[Hau98] Hauser, C.: *Parallele und verteilte Simulation von TCP über ATM-ABR*, Semesterarbeit, Institut für Nachrichtenvermittlung und Datenverarbeitung, Universität Stuttgart, D, September 1998.

[Hee97a] Heegaard, P. E.: *A bibliography on speedup simulation techniques*, Proceedings of the Workshop on Rare Event Simulation, Aachen, D, August, 28-29, 1997, pp. 92–99.

[Hee97b] Heegaard, P. E.: *Speedup simulation techniques (survey)*, Proceedings of the Workshop on Rare Event Simulation, Aachen, D, August, 28-29, 1997, pp. 7.1–7.16.

[Hee98] Heegaard, P. E.: *A Scheme for Adaptive Biasing in Importance Sampling*, Archiv für Elektronik und Übertragungstechnik, Band 52, Nr. 3, Hirzel-Verlag, D, Mai 1998, Seiten 172–182.

[HOMM96] Hasegawa, G.; Ohsaki, H.; Murata, M.; Miyahara, H.: *Performance Evaluation and Parameter Tuning of TCP over ABR Service in ATM Networks*, IEICE Transactions on Communications, Vol. E79-B, No. 5, J, May 1996, pp. 668–683.

[Hos85] Hoshino, T.: *PAX Computer High-Speed Parallel Processing and Scientific Computing*, Series in Electrical and Computer Engineering, Addison-Wesley Publishing Company, Reading, USA, 1985.

[INM88] INMOS: *The Transputer Databook*, INMOS Databook Series, INMOS, Bath, GB, November 1988.

[ISO94] ISO: *ISO/IEC 7498-1 - Information Processing Systems - OSI Reference Model - The Basic Model*, International Standardization Organisation (ISO), 1994.

[ISO95] ISO: *ISO/IEC DIS 10746 RM-ODP (Reference Model - Open Distributed Processing)*, 1995.

[IT91] ITU-T: *ITU-T Recommendation I.321 (4/91) - B-ISDN Protocol Reference Model and its Application*, International Telecommunication Union, Geneva, CH, April 1991.

[IT93a] ITU-T: *ITU-T Recommendation I.150 (3/93) - B-ISDN Asynchronous Transfer Mode Functional Characteristics*, International Telecommunication Union, Geneva, CH, March 1993.

[IT93b] ITU-T: *ITU-T Recommendation I.432 (3/93) - B-ISDN User-Network Interface - Physical Layer Specification*, International Telecommunication Union, Geneva, CH, March 1993.

[IT95a] ITU-T: *ITU-T Recommendation I.361 (11/95) - B-ISDN ATM Layer Specification*, International Telecommunication Union, Geneva, CH, November 1995.

[IT95b] ITU-T: *ITU-T Recommendation Q.2931 (2/95) - Broadband integrated services digital network (B-ISDN) - Digital subscriber signalling system No. 2 (DSS-2) - User-Network interface (UNI) layer 3 specification*, International Telecommunication Union, Geneva, CH, February 1995.

[IT96a] ITU-T: *ITU-T Recommendation I.363.1 (8/96) - B-ISDN Adaptation Layer specification: Type 1 AAL*, International Telecommunication Union, Geneva, CH, August 1996.

[IT96b] ITU-T: *ITU-T Recommendation I.363.3 (8/96) - B-ISDN ATM Adaptation Layer specification: Type 3/4 AAL*, International Telecommunication Union, Geneva, CH, August 1996.

[IT96c] ITU-T: *ITU-T Recommendation I.363.5 (8/96) - B-ISDN ATM Adaptation Layer specification: Type 5 AAL*, International Telecommunication Union, Geneva, CH, August 1996.

[IT96d] ITU-T: *ITU-T Recommendation I.371 (8/96) - Traffic control and congestion control in B-ISDN*, International Telecommunication Union, Geneva, CH, August 1996.

[IT97] ITU-T: *ITU-T Recommendation I.363.2 (9/97)- B-ISDN ATM Adaptation layer specification: Type 2 AAL*, International Telecommunication Union, Geneva, CH, September 1997.

[IT98] ITU-T: *ITU-T Recommendation G.707 (3/96) - Digital transmission systems - Terminal equipments - General - Network node interface for the synchronous digital hierarchy (SDH)*, Transmission Systems and Media, International Telecommunication Union, Geneva, CH, March 1998.

[ITG97] ITG: *Entwurf für die ITG-Empfehlung 5.2-03 „Begriffe der Nachrichtenverkehrstheorie“*, Begriffswerk des ITG-Fachausschusses 5.2, ITG, D, Oktober 1997.

[JaSi88] Jaffe, J. M.; Sidi, M.: *Distributed Deadlock Resolution*, Proceedings of the International Conference on Computer Communication (ICCC), Tel Aviv, IL, 1988, pp. 434–438.

[Jef83] Jefferson, D.: *Virtual Time*, Report, University of Southern California, May 1983.

[JS82] Jefferson, D. R.; Sowizral, H. A.: *Fast Concurrent Simulation Using the Time Warp Mechanism, Part I: Local Control*, The Rand Corporation, Santa Monica, California, USA, December 1982.

[JS83] Jefferson, D. R.; Sowizral, H. A.: *Fast Concurrent Simulation Using the Time Warp Mechanism, Part II*, The Rand Corporation, Santa Monica, California, USA, August 1983.

[JS85] Jefferson, D. R.; Sowizral, H.: *Fast concurrent simulation using the time warp mechanism*, Distributed Simulation, Vol. 15, No. 2, San Diego, CA, USA, January 1985, pp. 63–69.

[Jur96] Jurczyk, M.: *Modellierung und Parallele Simulation von Verbindungsnetzen*, 70. Bericht über verkehrstheoretische Arbeiten, Institut für Nachrichtenvermittlung und Datenverarbeitung, Universität Stuttgart, D, 1996.

[Kle75] Kleinrock, L.: *Queueing Systems: Theory*, John Wiley and Sons, New York, USA, 1975.

[Klu90] Kluth, B.: *Multiprozessorarchitekturen mit funktionsorientierter Parallelisierung für die stochastische Simulation*, Dissertation, RWTH Aachen, D, 1990.

[KM53] Kahn, H.; Marshall, A. W.: *Methods of Reducing Sample Size in Monte Carlo Simulations*, Journal of the Operations Research Society of America, Vol. 1, USA, 1953, ISSN 0096-3984, pp. 263–278.

[Ko94] Kocher, H.: *Entwurf und Implementierung einer Simulationsbibliothek unter Anwendung objektorientierter Methoden*, 59. Bericht über verkehrstheoretische Arbeiten, Institut für Nachrichtenvermittlung und Datenverarbeitung, Universität Stuttgart, D, 1994.

[Köh94] Köhrer, M.: *Simulation seltener Ereignisse unter Verwendung des Restart-Verfahrens*, Semesterarbeit, Institut für Nachrichtenvermittlung und Datenverarbeitung, Universität Stuttgart, D, April 1994.

[KSGL95] Kühn, P. J.; Seibold, W.; Götzer, M.; Lemppenau, W.: *Abschlußbericht des DFG-Forschungsvorhabens Parallele und verteilte Simulation von Hochgeschwindigkeits-LANs*, Institut für Nachrichtenvermittlung und Datenverarbeitung, Universität Stuttgart, D, 1995.

[Küh77] Kühn, P. J.: *Approximate analysis of general queueing networks by decomposition*, IEEE Transactions on Communications, Vol. COM-27, No. 1, USA, January 1979, pp. 113–126.

[Küh95] Kühn, P. J.: *Unterlagen zur Lehrveranstaltung "Technische Informatik III"*, Institut für Nachrichtenvermittlung und Datenverarbeitung, Universität Stuttgart, D, 1995.

[KWWR93] Kalantery, N.; Winter, S. C.; Wilson, D. R.; Redfern, A. P.: *Fast Parallel Simulation of SS7 Telecommunication Networks*, Proceedings of the MASCOTS ’93, San Diego, CA, USA, January, 17-20, 1993, pp. 171–175.

[L’E88] L’Ecuyer, P.: *Efficient and portable combined random number generators*, Communications of the ACM, Vol. 31, No. 6, USA, June 1988, pp. 742–749/ 774.

[L’E96] L’Ecuyer, P.: *Combined Multiple Recursive Random Number Generators*, Operations Research, Vol. 44, No. 5, USA, September 1996, pp. 816–822.

[Läu98] Läuger, A.: *Aufbau, Inbetriebnahme und Test einer PCI-Schnittstellenkarte für die Datenübertragung zwischen einem PC und einem Parallelrechner*, Institut für Nachrichtenvermittlung und Datenverarbeitung, Universität Stuttgart, D, November 1998.

[Leh79a] Lehnert, R.: *Ein modularer Prozessor zur Simulation von Zufallsnetzwerken*, Dissertation, RWTH Aachen, D, 1979.

[Leh79b] Lehnert, R.: *A special processor for fast simulation of queueing networks*, Proceedings of the 9th International Teletraffic Congress (ITC), Torremolinos, ES, October, 16-24, 1979, pp. 1–7.

[Leh80] Lehnert, R.: *Ein Spezialprozessor zur schnellen Simulation von Zufallsnetzwerken*, Elektrische Rechenanlagen, Band 22, D, 1980, pp. 74–82.

[Lei94] Leitner, T.: *Entwicklung und Implementierung von Bibliotheksfunktionen für die Darstellung von Simulationsdaten unter Microsoft Windows*, Institut für Nachrichtenvermittlung und Datenverarbeitung, Universität Stuttgart, D, Juli 1994.

[Lem89] Lemppenau, W.: *Umweltsimulator für den Funktions- und Lasttest von Nachrichtenvermittlungssystemen*, 47. Bericht über verkehrstheoretische Arbeiten, Institut für Nachrichtenvermittlung und Datenverarbeitung, Universität Stuttgart, D, 1989.

[LGS93] Lemppenau, W. W.; Götzer, M.; Seibold, W.: *Access protocols for high-speed LANs and MANs*, Proceedings of the EFOC&N ’93, The Hague, NL, June 30 - July 2, 1993, pp. 253–259.

[Lin90] Lin, Y. B.: *Understanding the Limits of Optimistic and Conservative Parallel Simulation*, Technical Report 90-08-02, Department of Computer Science and Engineering, University of Washington, USA, August 1990.

[Liv85] Livny, M.: *A study of parallelism in distributed simulation*, Distributed Simulation, Vol. 15, No. 2, San Diego, CA, USA, January 1985, pp. 94–98.

[LK91] Law, A. L.; Kelton, W. D.: *Simulation modelling and analysis*, 2nd Edition, McGraw-Hill, New York, USA, 1991.

[LM90] Lipton, R. J.; Mizell, D. W.: *Time Warp vs. Chandy-Misra: A Worst-Case Comparison*, Proceedings of the SCS Multiconference on Distributed Simulation, San Diego, CA, USA, January, 17-19, 1990, pp. 137–143.

[LSTea96] Li, H.; Siu, K. Y.; Tzeng, H. Y.; Ikeda, C.; Suzuki, H.: *A Simulation Study of TCP Performance over ABR and UBR Services in ATM LANs*, IEICE Transactions on Communications, Vol. E79-B, No. 5, J, May 1996, pp. 658–667.

[Lub89] Lubachevsky, B. D.: *Efficient Distributed Event-Driven Simulations of Multiple-Loop Networks*, Communications of the ACM, Vol. 32, No. 1, USA, January 1989, pp. 111–123.

[LvAS93] Lemppenau, W. W.; van As, H. R.; Schindler, H. R.: *A 2.4 Gbit/s ATM implementation of the CRMA-II dual-ring LAN and MAN*, Proceedings of the EFOC&N ’93, The Hague, NL, June 30 - July 2, 1993, pp. 274–281.

[LY88] Lu, D.; Yao, K.: *Improved importance sampling technique for efficient simulation of digital communication systems*, IEEE Journal on Selected Areas in Communications, Vol. 6, No. 1, USA, 1988, pp. 67–75.

[MACea94] Marsan, M. A.; Albertengo, G.; Casetti, C.; Neri, F.; Panizzardi, G.: *On the performance of topologies and access protocols for high-speed LANs and MANs*, Computer Networks and ISDN Systems, No. 26, North-Holland, NL, 1994, pp. 873–893.

[Man98] Mandjes, M.: *Asymptotically Optimal Importance Sampling for Tandem Queues with Markov Fluid Input*, Archiv für Elektronik und Übertragungstechnik, Band 52, Nr. 3, Hirzel-Verlag, D, Mai 1998, Seiten 152–161.

[Mat93] Mattern, F.: *Efficient Algorithms for Distributed Snapshots and Global Virtual Time Approximation*, Journal of Parallel and Distributed Computing, Vol. 18, USA, 1993, pp. 423–434.

[MB97] Mauger, R.; Brueckheimer, S.: *ATM Adaptation Layer Switching*, Proceedings of the ISS ’97: World Telecommunications Congress; Toronto, CDN, October, 21-26, 1997, pp. 207–214.

[MBCea98] Marsan, A.; Bianco, A.; Casetti, C.; Chiasserini, C. F.; Francini, A.; Lo Cigno, R.; Munafo, M.: *An integrated simulation environment for the analysis of ATM networks at multiple time scales*, Computer Networks and ISDN Systems, Vol. 29, No. 17+18, North-Holland, NL, February 1998, pp. 2165–2185.

[Mis83] Misra, J.: *Detecting Termination of Distributed Computations using Markers*, Proceedings of the ACM Principles of Distributed Computing Conference, Montreal, CDN, August 1983.

[Mis86] Misra, J.: *Distributed discrete event simulation*, ACM Computing Surveys, Vol. 18, No. 1, USA, 1986, pp. 39–65.

[MM93] Mallet, L.; Mussi, P.: *Object Oriented Parallel Discrete Event Simulation: The PROSIT Approach*, Proceedings of the European Simulation Multiconference, Lyon, F, June, 7-9, 1993, pp. 603–607.

[MP92] Mouly, M.; Pautet, M. B.: *The GSM System for Mobile Communications*, published by the authors, Palaiseau, F, 1992.

[MR93] Mussi, P.; Rakotoarisoa, H.: *Parseval: A Workbench for Queueing Networks Parallel Simulation*, Proceedings of the European Simulation Multiconference, Lyon, F, June, 7-9, 1993, pp. 89–94.

[Ne94] Necker, T.: *Verteilte Simulation auf der Basis einer objektorientierten Bibliothek*, Workshop über parallele und verteile Simulation - Bericht Nr. AIS 21, Universität Oldenburg, D, Dezember 1994, Seiten 8–10.

[Ne98] Necker, T.: *Entwicklung eines objektorientierten Werkzeugs für verschiedene Verfahren der parallelen ereignisgesteuerten Simulation*, 69. Bericht über verkehrtheoretische Arbeiten, Institut für Nachrichtenvermittlung und Datenverarbeitung, Universität Stuttgart, D, 1998.

[Nec98] Necker, F.: *Entwurf der Anbindung einer Hochgeschwindigkeitsschnittstelle an einen MIMD-Rechner*, Semesterarbeit, Institut für Nachrichtenvermittlung und Datenverarbeitung, Universität Stuttgart, D, Juli 1998.

[NF98] Nagata, K.; Fujiya, H.: *Evaluation of AAL-2 for Low-bit-rate ATM Voice Communications*, NTT Review, Vol. 10, No. 1, J, January 1998, pp. 72–80.

[NR85] Nicol, D. M.; Reynolds Jr., P. F.: *A statistical approach to dynamic partitioning*, Distributed Simulation, Vol. 15, No. 2, San Diego, CA, USA, January 1985, pp. 53–56.

[OS94] Obal II, W. D.; Sanders, W. H.: *Importance Sampling Simulation in UltraSAN*, Simulation, Vol. 62, No. 1, USA, 1994, pp. 98–111.

[Pfe92] Pfeffer, D.: *Beschreibung, Klassifikation und Bewertung paralleler Algorithmen für die verteilte, zeitdiskrete Logiksimulation*, Semesterarbeit, Institut für Parallele und Verteilte Höchstleistungsrechner, Universität Stuttgart, Januar 1992.

[PGM83] Papazoglou M. P.; Georgiadis, P. I.; Maritsas, D. G.: *Designing a parallel SIMULA machine*, Computer Design, Vol. 22, No. 11, USA, October 1983, pp. 125–132.

[Por97] Porbadnigk, D.: *Erstellung einer Klassenbibliothek zur Anzeige von Simulationsdaten*, Diplomarbeit, Institut für Nachrichtenvermittlung und Datenverarbeitung, Universität Stuttgart, D, Dezember 1997.

[Pos81] Postel, J.: *Transmission Control Protocol*, RFC 793, September 1981.

[Pro98] Proißl, B.: *Spezifikation und Implementierung eines portablen grafischen Netzeditors mit Hilfe der Klassenbibliothek ZINC*, Diplomarbeit, Institut für Nachrichtenvermittlung und Datenverarbeitung, Universität Stuttgart, D, April 1998.

[RA94] Rönngren, R.; Ayani, R.: *Adaptive Checkpointing in Time Warp*, Proceedings of the 8th Workshop on Parallel and Distributed Simulation (PADS '94), Edinburgh, Scotland, GB, July, 6-8, 1994, pp. 110–117.

[Ree85] Reese, R. M.: *A software development environment for distributed simulation*, Distributed Simulation, Vol. 15, No. 2, San Diego, CA, USA, January 1985, pp. 37–40.

[Rey91] Reynolds Jr., P. F.: *An Efficient Framework for Parallel Simulations*, Proccedings of the SCS Multiconference on Advances in Parallel and Distributed Simulation, Anaheim, USA, January 1991, pp. 167–174.

[RMM88] Reed, D. A.; Malony, A. D.; McCredie, B. D.: *Parallel Discrete Event Simulation Using Shared Memory*, IEEE Transactions on Software Engineering, Vol. 14, No. 4, USA, April 1988, pp. 541–553.

[Rös93] Rössig, K.: *Eine Übersicht über aktuelle Ansätze zur verteilten optimistischen Simulation*, Arbeitsgruppe Informatik-Systeme, Fachbereich Informatik, Universität Oldenburg, Band 11, Oldenburg, D, August 1993.

[Rößler95] Rößler, G.: *Einfluß des Netzmanagements auf die Struktur von OSI-Systemen und eine Methode für die erstmalige Konfiguration*, 60. Bericht über verkehrstheoretische Arbeiten, Institut für Nachrichtenvermittlung und Datenverarbeitung, Universität Stuttgart, D, 1995.

[Roz85] Rozenblit, J. W.: *Experimental frames for distributed simulation architectures*, Distributed Simulation, Vol. 15, No. 2, San Diego, CA, USA, January 1985, pp. 14–20.

[RP92] Reynolds Jr., P. F.; Pancerella, C. M.: *Hardware Support for Parallel Discrete Event Simulation*, Computer Science Report No. TR-92-08, Charlottesville, USA, April 1992.

[RPS92] Reynolds Jr., P. F.; Pancerella, C. M.; Srinivasan, S.: Design and Performance Analysis of Hardware Support for Parallel Simulations, Computer Science Report No. CS-92-20, Charlottesville, USA, June 1992.

[RPS93] Reynolds Jr., P. F.; Pancerella, C. M.; Srinivasan, S.: *Design and Performance Analysis of Hardware Support for Parallel Simulations*, Journal of Parallel and Distributed Computing, Vol. 18, USA, 1993, pp. 435–453.

[RRRea94] Rönngren, R.; Rajaei, H.; Ropescu, A.; Liljenstam, M. et al.: *Parallel Simulation of a High Speed LAN*, Proceedings of the 8th Workshop on Parallel and Distributed Simulation (PADS ’94), Edinburgh, Scotland, GB, July, 6-8, 1994, pp. 132–138.

[RW97] Rathgeb, E.; Wallmeier, E.: *ATM-Infrastruktur für die Hochleistungskommunikation*, Springer, Berlin, D, 1997.

[Säg85] Sägebarth, J.: *Über Eigenschaften und Dimensionierung von rekursiven Zufallszahlengeneratoren und ihre Anwendung in der Verkehrssimulation*, Fortschritt-Berichte VDI, Reihe 8, Dortmund, D, 1985.

[Sch90] Schildt, H.: *The annotated ANSI C Standard: American National Standard for programming languages C*, McGraw-Hill, USA, 1990.

[Sch94] Schmauch, C. H.: *ISO 9000 for Software Developers*, ASQC Quality Press, Wisconsin, USA, 1994.

[SCM85] Sheppard, S.; Chandrasekaran, U.; Murray, K.: *Distributed simulation using Ada*, Distributed Simulation, Vol. 15, No. 2, San Diego, CA, USA, January 1985, pp. 27–31.

[Sei92] Seibold, W.: *Beschreibung von parametrisierbaren Funktionsblöcken zur Systemsimulation mit der Hardware-Beschreibungssprache VHDL*, Institut für Parallele und Verteilte Höchstleistungsrechner, Universität Stuttgart, D, Mai 1992.

[Sim95] Simonovich, D.: *Entwurf und Bewertung konservativer Verfahren in der verteilten Simulation*, Berichte des Fachbereichs Informatik, Universität Hamburg, D, Mai 1995.

[SJ96] Schwederski, T.; Jurczyk, M.: *Verbindungsnetze: Strukturen und Eigenschaften*, Leitfäden der Informatik, Teubner, Stuttgart, D, 1996.

[SMB87] Smith, S. P.; Mercer, M. R.; Brock, B.: *Demand Driven Simulation: BACKSIM*, Proceedings of the 24. International Conference on Computer Aided Design, Miami Beach 1987, USA, ACM/IEEE, October 1987, pp. 181–187.

[SS97] Smith, P. J.; Shafi, M.: *Quick Simulation: A Review of Importance Sampling Techniques in Communications Systems*, IEEE Journal on Selected Areas in Communications, Vol. 15, No. 4, USA, May 1997, pp. 597–613.

[Sta98] Stallings, W.: *High-speed networks: TCP/IP and ATM design principles*, Prentice-Hall, New Jersey, USA, 1998.

[Ste97] Stevens, W.: *TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms*, RFC 2001, January 1997.

[Str91] Stroustrup, B.: *The C++ Programming Language: 2nd edition*, Addison-Wesley, Reading, USA, 1991.

[tB95] ten Brink, S.: *Untersuchungen zur optimistischen parallelen Simulation von Warteschlangensystemen auf einem Parallelrechner*, Diplomarbeit, Institut für Nachrichtenvermittlung und Datenverarbeitung, Universität Stuttgart, November 1995.

[THH80] Takenouchi, H.; Hatada, M.; Hiyama, K.: *Parallel Processing Simulator for Network Systems using Multi-Microcomputer*, Proceedings of COMPCON ’80, September, 23-25, 1980, pp. 55–62.

[TI91] Texas Instruments: *TMS320C4x User’s Guide*, Texas Instruments, USA, May 1991.

[ULA85] Unger, B.; Lomow, G.; Andrews, K.: *A process oriented distributed simulation package*, Distributed Simulation, Vol. 15, No. 2, San Diego, CA, USA, January 1985, pp. 76–81.

[vA94] van As, H. R.: *Media access techniques: The evolution towards terabit/s LANs and MANs*, Computer Networks and ISDN Systems, Vol. 26, North-Holland, NL, 1994, pp. 603–656.

[vALZ92a] van As, H. R.; Lemppenau, W. W.; Zafiropulo, P.: *Performance of CRMA-II: A Reservation-Based Fair Media Access Protocol for Gbit/s LANs and MANs with Buffer Insertion*, Proceedings of the 5th IEEE workshop on MAN, Taomina, I, May 1992, p. 8.

[vALZ92b] van As, H. R.; Lemppenau, W. W.; Zafiropulo, P.: *Performance of CRMA-II: A Reservation-Based Fair Media Access Protocol for Gbit/s LANs and MANs with Buffer Insertion*, Proceedings of the EFOC/LAN ’92, Paris, F, June 1992, pp. 162–169.

[vALZZ91] van As, H. R.; Lemppenau, W. W.; Zafiropulo, P.; Zurfluh, E. A.: *CRMA-II: A Gbit/s MAC Protocol for Ring and Bus Networks with Immediate Access Capability*, Proceedings of the EFOC/LAN ’91, London, UK, June 1991, pp. 56–71.

[VAVA91] Villen-Altamirano, M.; Villen-Altamirano, J.: *RESTART: A Methode for accelerating Rare Event Simulations*, Proceedings of the 13th International Teletraffic Congress, Copenhagen, DK, 1991, pp. 71–76.

[VT93] Verboven, F.; Tallieu, F.: *Automated Generation of Discrete Event-Driven Simulation Code from Computer Network specifications*, Proceedings of the MASCOTS ’93, San Diego, CA, USA, January, 17-20, 1993, pp. 191–194.

[Wag98] Wagner, S.: *Entwurf und Implementierung einer Simulationssteuerung für einen Parallelrechner*, Diplomarbeit, Institut für Nachrichtenvermittlung und Datenverarbeitung, Universität Stuttgart, D, Juni 1998.

[Weg96] Wegel, D.: *Simulation von CRMA-II mit einem optimistischen Simulationsverfahren auf einem Multiprozessorrechner*, Semesterarbeit, Institut für Nachrichtenvermittlung und Datenverarbeitung, Universität Stuttgart, D, August 1996.

[Wit94] Wittmershaus, D.: *Simulation des Medienzugriffsprotokolls CRMA-II mittels über Shared Memory kommunizierender UNIX-Prozesse*, Diplomarbeit, Institut für Nachrichtenvermittlung und Datenverarbeitung, Universität Stuttgart, D, März 1994.

[WLB89] Wagner, D. B.; Lazowska, E. D.; Bershad, B. N.: *Techniques for efficient shared-memory parallel simulation*, Distributed Simulation, San Diego, CA, USA, 1989, pp. 29–37.

[Wol99] Wolfer, R.: *Entwicklung, Aufbau und Inbetriebnahme einer Hardware zur Erzeugung von Zufallszahlen*, Semesterarbeit, Institut für Nachrichtenvermittlung und Datenverarbeitung, Universität Stuttgart, D, Februar 1999.

[Zbo95] Zborschil, B.: *Entwicklung einer Bibliothek für optimistische verteilte Simulation von Warteschlangensystemen*, Diplomarbeit, Institut für Nachrichtenvermittlung und Datenverarbeitung, Universität Stuttgart, D, Januar 1995.

[Zinc94a] Zinc Software Incorporated.: *Zinc Application Framework Programmer's Reference Volume One: Support Objects*, USA, 1994.

[Zinc94b] Zinc Software Incorporated.: *Zinc Application Framework Programmer's Reference Volume Two: Window Objects*, USA, 1994.

Anhang

A Begriffserläuterungen

Assertion proofs	Assertion proofs sind Prüfungen auf Einhaltung von Zusicherungen in (formal beschriebenen) Systemen. Sie werden z.B. zur Erkennung illegaler Systemzustände verwendet.
Dienstgüte	Die Dienstgüte gibt die Gesamtheit der Qualitätsmerkmale eines Kommunikationsnetzes aus der Sicht der Benutzer eines betrachteten Dienstes an [ITG97].
Effizienz	In der Informatik wird ein Algorithmus als effizient bezeichnet, wenn er ein vorgegebenes Problem in kürzest möglicher Zeit und/ oder mit möglichst geringem Aufwand an Betriebsmitteln löst [Dud93]. Im Zusammenhang mit der Parallelverarbeitung bedeutet dies, daß die Geschwindigkeitssteigerung in Relation zu den eingesetzten Recheneinheiten gesetzt wird (vgl. Gleichung 2-2, Seite 14).
Emulation	Unter einer Emulation versteht man die Nachbildung eines Systems unter Verwendung eines anderen Systems, wobei das Augenmerk normalerweise auf einer hohen Systemleistung liegt. Im Gegensatz zur Simulation erbringt die Emulation eines technischen Systems dessen Funktion und kann im Wirkbetrieb eingesetzt werden.
Ereignis	In der ereignisgesteuerten Simulation versteht man unter einem Ereignis das Abbild eines Vorkommnisses (z.B. Beginn und Ende eines Vorgangs), welches das Modell des Systems beeinflußt. Das Ereignis findet zu einem bestimmten Zeitpunkt statt und erhält daher in der Simulation einen Eintrittszeitpunkt zugewiesen.
Ereignislokalität	Die Ereignislokalität beschreibt den Wirkungsbereich von Ereignissen eines logischen Prozesses. Sie unterteilt sich in eine räumliche und zeitliche Lokalität (vgl. Kapitel 2.4.4)
Erreichbarkeits-analyse	Die Erreichbarkeitsanalyse ist eine analytische Methode zur Ermittlung aller erreichbaren Systemzustände.
Hamilton-Pfad	Ein Hamilton-Pfad ist ein geschlossener Pfad in einem Graphen, der genau einmal durch alle Knoten des Graphen führt [Dud93].
funktionale Simulation	Siehe „Simulation“.
instationäre Simulation	Im Gegensatz zur stationären Simulation liegt bei der instationären Simulation kein eingeschwungenes System vor. Gegenstand der Untersuchung ist in der Regel das Systemverhalten nach einem äußeren Ereignis (z.B. Einschalten, Fehler), bis wieder ein stationärer Zustand erreicht wird.

Kanal	Ein Kanal ist ein Kommunikationsweg zum Transport von Nachrichten zwischen zwei Knoten des Nachrichtennetzes. Ein physikalischer Kanal kann durch Multiplexen Nachrichten mehrerer logischer Kanäle transportieren.
Kanalzeit	Werden über einen Kanal Nachrichten mit monoton wachsendem Zeitstempeln übertragen, so ist die Kanalzeit die Zeit der zuletzt übertragenen Nachricht.
konservative Simulation	Parallele und verteilte Simulationen werden als konservativ bezeichnet, wenn ein Synchronisationsverfahren verwendet wird, das Kausalitätsverletzungen dadurch vermeidet, daß lokale Ereignisse erst dann bearbeitet werden, wenn sie sicher sind (vgl. „sicheres Ereignis“).
Korrektheit	Ein technisches System ist korrekt, wenn es vollständig mit seiner Spezifikation übereinstimmt. Damit wird nichts über die Sinnhaftigkeit des Systems ausgesagt.
Logischer Prozeß	Bei einer parallelen und verteilten Simulation wird das Simulationsmodell auf logische Prozesse verteilt. Jeder logische Prozeß stellt ein Modul dar, welches von einer Recheneinheit getrennt, aber nicht notwendigerweise unabhängig bearbeitet werden kann. Jeder logische Prozeß besitzt eine eigene Simulationszeitverwaltung und eine eigene Simulationszeit. Die Simulationszeiten der logischen Prozesse sind daher nicht identisch.
Monte-Carlo-Simulation	Simulationsprinzip zur Untersuchung von Systemen mit Hilfe von Zufallszahlen, wobei die Zeit dabei keine substantielle Rolle spielt, d.h. es werden ohne zeitliche Komponente Zufallsgrößen an ein Systemmodell angelegt bzw. in ihm angenommen und die Reaktion darauf protokolliert und ausgewertet [LK91]. Dies gilt für viele funktionale Simulationen. Teilweise wird in der Literatur unter Monte-Carlo-Simulation auch jegliche Simulation mit Zufallszahlen verstanden.
optimistische Simulation	Parallele und verteilte Simulationen werden als optimistisch bezeichnet, wenn ein Synchronisationsverfahren verwendet wird, bei welchem Kausalitätsverletzungen auftreten können. Tritt eine Kausalitätsverletzung auf, wird die Simulation zurückgesetzt und der unter falschen Voraussetzungen durchgeführte Teil der Simulation verworfen.
Parallele und verteilte Ausführung	Wird eine Applikation auf einem MIMD-Rechner (Multiple Instruction Stream, Multiple Data Stream) unter Ausnutzung mehrerer unabhängiger Recheneinheiten ausgeführt, so liegt eine parallele und verteilte Ausführung der Applikation vor.
Prototyp	Ein Prototyp ist die Realisierung eines technischen Systems, wobei die wesentlichen, aber nicht notwendigerweise alle Vorgaben der Spezifikation eingehalten werden.

Pseudozufallszahlenfolge	Nach [Dud93] ist eine Zufallszahl eine Zahl, die rein statistisch („zufällig“) aus einer Menge von Zahlen herausgegriffen wird. Eine (unendliche) Folge von Zahlen ohne (algorithmisches) Bildungsgesetz heißt Zufallszahlenfolge. Die Nachbildung einer „echten“ Zufallszahlenfolge auf einem Rechner wird nach [GH85] als Pseudozufallszahlenfolge bezeichnet, da die Pseudozufallszahlen durch einen deterministischen und vollständig spezifizierten Algorithmus auseinander hervorgehen. Aus diesem Grunde sind Pseudozufallszahlenfolgen insbesondere identisch reproduzierbar. Gute Pseudozufallszahlenfolgen gehorchen den gleichen statistischen Gesetzmäßigkeiten wie Zufallszahlenfolgen.
Referenzsystem	Ein Referenzsystem ist die Realisierung eines technischen Systems, dessen relevante Eigenschaften als Spezifikation aufgefaßt werden, d.h. ein technisches System ist konform zur Spezifikation, wenn es sich gleich wie das Referenzsystem verhält.
Rollback	Bewahrheitet sich eine getroffene Annahme bei einer optimistischen Simulation nicht, dann muß sie auf den letzten gesicherten Zustand zurückgesetzt werden. Dieser Vorgang wird als Rollback bezeichnet.
sicheres Ereignis	Ein Ereignis gilt bei der parallelen und verteilten Simulation zum Zeitpunkt t_0 als sicher, wenn durch dessen Bearbeitung zum Zeitpunkt t_0 keine Kausalitätsverletzung entstehen kann.
Simulation	Nachbildung von Vorgängen auf einer Rechenanlage auf der Basis von Modellen. Sie wird meist zur Untersuchung von Abläufen eingesetzt, die man in der Wirklichkeit aus Zeit, Kosten-, Gefahren- oder anderen Gründen nicht durchführen kann [Dud93]. Hintergrund einer funktionalen Simulation ist in der Regel die Validierung oder Veranschaulichung eines Simulationsmodells. Dies geschieht dadurch, daß die Reaktion des Modells auf unterschiedliche äußere Einflüsse gezeigt wird. Leistungsuntersuchungen werde mit Hilfe der statistischen Simulation durchgeführt. Dazu werden interessante Systemgrößen während der Simulation oder im Anschluß daran mit Hilfe statistischer Methoden ausgewertet.
Simulationszeitgewinn	Siehe „Speedup“.
Speedup	Speedup ist der Beschleunigungsfaktor der sich aus der Parallelisierung einer Problemlösung ergibt. Der Speedup berechnet sich als Quotient aus den Ausführungszeiten bei sequentieller und bei paralleler Bearbeitung (vgl. Gleichung 2-1, Seite 14).
stationäre Simulation	Die stationäre Simulation ist eine statistische Simulation eines eingeschwungenen (und damit stationären) Systems. Die Zeit bis zum Erreichen des stationären Zustands wird bei der statistischen Auswertung nicht berücksichtigt (vgl. „instationäre Simulation“).

statistische Simulation	Siehe „Simulation“.
Validierung	Unter Validierung wird im allgemeinen ein „für gültig Erklären“ verstanden. Im technischen Umfeld bedeutet dies, daß das zu validierende Objekt durch Plausibilitätsüberlegungen oder Tests geprüft wird. Dadurch kann in der Regel keine formale Korrektheit nachgewiesen werden (Verifikation) sondern es können allenfalls vorhandene Fehler bzw. Unstimmigkeiten aufgedeckt werden.
Verifikation	Der formale Nachweis der Korrektheit wird als Verifikation bezeichnet. Für technische Systeme bedeutet Korrektheit die vollständige Übereinstimmung des Systems mit seiner Spezifikation. Dadurch ist nichts über die Sinnhaftigkeit des Systems ausgesagt.
verkehrstheoretische Analyse	Unter einer verkehrstheoretischen Analyse eines Systems versteht man die auf Modellen basierende analytische Untersuchung eines Systems bzw. seines Nachrichtenflusses mit Hilfe der Methoden der Mathematik unter Zugrundelegung stochastischer Prozesse.
Vorausschau (Lookahead)	Im Zusammenhang mit der konservativen, parallelen und verteilten Simulation wird unter der Vorausschau eines logischen Prozesses LP_1 bezüglich eines anderen logischen Prozesses LP_2 die Simulationszeit t verstanden, die vergeht, bevor sich ein Ereignis in LP_2 frühestens in LP_1 auswirkt.
Zeitstempel	Unter dem Zeitstempel einer Nachricht oder eines Ereignisses wird im Simulationskontext die Zeit verstanden, zu der die Nachricht im simulierten System eintrifft bzw. zu der sich das Ereignis zuträgt.
Zusicherung	Im Kontext der formalen Verifikation technischer Systeme versteht man unter einer Zusicherung eine nachprüfbare Aussage über das System bzw. einen Teil des Systems. Dies kann z.B. der zulässige Wertebereich einer Variablen sein. Im Zuge der Verifikation wird geprüft, ob die Zusicherung eingehalten wird.
Zustandssicherung	Unter der Zustandssicherung wird das Sichern des Systemzustands bei optimistischen Simulationsverfahren verstanden. Dadurch kann bei falschen Annahmen die Simulation auf einen gültigen Systemzustand zurückgesetzt werden (Rollback). Die Zustandssicherung kann periodisch oder in Abhängigkeit von den bearbeiteten Ereignissen durchgeführt werden.

B Simulationsparameter

In den Tabellen B-1 bis B-3 sind die Parameter der Simulationen aufgelistet, die in Kapitel 5 für die Bewertung der Simulatorarchitektur und der prototypischen Realisierung herangezogen wurden.

Parameter	Wert
Mediumbandbreite	2,4 Gbit/s
Anzahl Bits je Zeitschlitz	544
Anzahl Bits je ADU (Atomic Data Unit)	32
Schwellwertberechnung	dynamisch nach [LvAS93]
Zeit für die Schwellwertberechnung	0 s
Anzahl der CRMA-II-Stationen	3
Entfernung zwischen den Stationen	1 m .. 1 km
Überholen freier Zeitschlitze im Einfügepuffer	ja
Maximaler Füllstand des Einfügepuffers beim Beginn einer Übertragung	0
Einfügepuffergröße	500 Zellen
Sendepuffergröße	500 Zellen
Verzögerung der Nachrichten beim Durchlaufen einer Station ohne die Verzögerung durch den Einfügepuffer	0 Bit
Verteilungsfunktion für die Zwischenankunftszeit von Nachrichten	negativ-exponentiell Mittelwert: 49,48 µs
Verteilungsfunktion für die Nachrichtenlänge	verschoben geometrisch Mittelwert: 1920 Oktett Obergrenze: 3840 Oktett
Verteilungsfunktion für die Nachrichtenzieladressen	gleichverteilt ohne eigene Station

Tabelle B-1: Parameter der CRMA-II-Simulation

Parameter	Wert
Anzahl der TCP-Generatoren	4
Anzahl der TCP-Senken	4
Anzahl der Vermittlungsknoten	4
Abstand zwischen den Vermittlungsknoten	10 m .. 200 km
Kanalbandbreite zwischen den Vermittlungsknoten	155,52 Mbit/s
ABR-Parameter: PCR (Peak Cell Rate) ICR (Initial Cell Rate) MCR (Minimum Cell Rate) RDF (Rate Decrease Factor) CDF (Cutoff Decrease Factor) RIF (Rate Increase Factor) Trm (max time between RM-cell generation) ADTF (Actual Cell Rate Decrease Time Factor) Crm (max. number of outstanding RM-cells) Mrm (min. number of cells between RM-cell generation) Nrm (max. number of cells between RM-cell generation) Steueralgorithmus Warteschlangengröße Nutzungsgrad des Ausgangskanals max. Dauer eines ERICA-Meßintervalls in Zeitschlitzen max. Dauer eines ERICA-Meßintervalls in Sekunden	 155,52 Mbit/s 20 Mbit/s 1 Mbit/s 0,0625 0,0625 0,0625 0,1 s 0,5 2 2 32 ERICA 1000 0,95 100 0,001
Allgemeiner Nachrichtengenerator (generiert Nachrichten für die TCP-Schicht)	Greedy, 2 der 4 Generatoren werden in 200 ms-Intervallen ein- bzw. ausgeschaltet
Puffergröße der Reassembly-Schicht	200 Zellen

Tabelle B-2: Simulationsparameter für die TCP über ATM/ABR-Simulation

Parameter	Wert
Anzahl der GSM-Verkehrsgeneratoren je Vermittlungsknoten	8..64
Anzahl der Vermittlungsknoten	3
Abstand zwischen den Vermittlungsknoten	10 m .. 200 km
Bandbreite zwischen den Vermittlungsknoten	2 MBit/sec
Paketgröße	36 Oktett (nach GSM)
Verteilungsfunktion der Sprecheraktivität	negativ exponentiell Mittelwert 1,4 s
Verteilungsfunktion der Sprecherinaktivität (Sprechpause)	negativ exponentiell Mittelwert 1,721 s
Verteilungsfunktion für Interpaketabstand bei aktivem Sprecher	konstant 20 ms
Verteilungsfunktion für Interpaketabstand bei inaktivem Sprecher	konstant 480 ms

Tabelle B-3: Simulationsparameter für die GSM über ATM/AAL Typ 2-Simulation