ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

TIK Institut für Technische Informatik und Kommunikationsnetze

Christian Zimmermann

Simulative Leistungsbewertung und Entwicklung rekonfigurationsfester Routingverfahren für mehrstufige Verbindungsnetzwerke

Master-Arbeit (MA-2006-16)
6. März bis 6. September 2006

Gutachter: Prof. Dr. B. Plattner
Betreuer: Dr. Lukas Ruf

Zusammenfassung

Der Simulation von Verbindungsnetzen kommt eine grosse Bedeutung zu, da diese im Trend zu Multiprozessorsystemen immer vielfältiger verwendet werden. Der Forschungsbereich von Networks on Chip erlaubt dabei nicht nur statische Topologien, sondern auch dynamisches Rekonfigurieren der Netzwerkstruktur. Solche Rekonfigurationen zu simulieren, blieb das bisher unerreichte Ziel der Entwicklung des komponentenbasierten Verbindungsnetz-Simulators CINSim. Die vorliegende Arbeit fasst den Entwicklungsstand inklusive der statischen Rekonfiguration zusammen und dokumentiert die Fortschritte, die CINSim zu einem Simulator dynamischer Rekonfigurationen gemacht hat. Ein Theoriekapitel bespricht das entworfene Modell der Analyse und Auflösung von Deadlocks, das in Software umgesetzt worden ist. Aufgetretene Probleme, allen voran die Herausforderung, eine neue Flusskontrolle zu implementieren, werden in einem nächsten Kapitel beschrieben. Das entstandene auf dynamische Rekonfiguration erweiterte Simulationstool CINSim konnte validiert werden und zeigt den erwarteten Performance-Vorsprung zum bisherigen System mit statischer Rekonfiguration. Die bleibende Aufgabe, das System auf andere Routingverfahren als Shortest Path zu verallgemeinern, bildet unter anderem den Ausblick auf weitere Entwicklungsarbeit.

Abstract

Simulation of interconnection networks is of big interest, because interconnection networks are used in multiprocessors and multicomputer systems more and more. Networks on Chip not only perform static topologies, but dynamic reconfiguration of the network structure as well. Simulating these is the unreached aim of the development of CINSim, the component-based interconnection network simulator. The thesis on hand recapitulates the state of development, including static reconfiguration and documents the proceedings of CINSim becoming a simulator of dynamic reconfiguration. The theory part discusses the design of an analysis and resolution model for deadlocks, that is implemented in software. Occurred problems, besides the challenge to find a new method of flow control, are specified in an implementation chapter. This way, CINSim was enhanced and validated. It performs an improvement to the previous system being able to simulate static reconfiguration only. The demand for other routing schemes, than shortest path, affects the further prospects of development. A more detailed executive summary in english is given in section 7.2, on page 115.

Danksagung

Mein Dank gilt den Professoren, Assistenten und Mitarbeitern der ETH-Zürich und der TU Berlin, die mir diese Master-Arbeit ermöglicht haben. Als Erstes möchte ich Herrn Prof. Bernhard Plattner danken, der mich unterstützt hat bei dem Unterfangen, eine Master-Arbeit im Ausland zu schreiben, und nötige organisatorische Vorarbeit für mich geleistet hat. Ich danke Dr. Lukas Ruf, der sich bereit erklärt hat, die Assistenz der Ausland-Arbeit zu übernehmen, und dessen Unterstützung für das Gelingen entscheidend gewesen ist. Ich möchte in gleichem Mass auch dem Prof. Dr.-Ing. Dr. h.c. Günter Hommel für die Überlassung des Themas danken, und ebenso den Betreuern der Arbeit Dr.-Ing. habil. Dietmar Tutsch, Herrn Dipl.-Ing. Daniel Lüdtke und Herrn Dipl.-Ing. Marcus Brenner. Ich danke ebenfalls den Mitarbeitern des Departements-Sekretariats des D-ITET der ETH Zürich, Herrn Marcel Kreuzer und Frau Doris Döbeli, die mir zu den häufigen Fragen zum Thema Auslandsemester Auskunft gegeben haben.

Berlin, September 2006

II

Inhaltsverzeichnis

Kapitel 1

Einführung

1.1 Motivation für dynamische Rekonfiguration

Im Informationszeitalter hat der Computer eine zentrale Rolle in der Gesellschaft eingenommen. Die Defence Advanced Research Projects Agency (DARPA) [1], zu Deutsch die Agentur für fortgeschrittene Forschungsprojekte des Militärs, und die nationale Forschungsagentur der USA haben 20 Forschungsbereiche aufgelistet, die alle mit dem Thema der Informationsverarbeitung zu tun haben. Man kann sagen, dass sich um dieses Thema alles dreht, was Forschung im Bereich Computertechnik anbelangt. Wenn Endgeräte mit einander kommunizieren, dann muss Information ausgetauscht werden. Damit dies geschehen kann, sind Verbindungsnetze notwendig, welche die Informationen vom einen zum anderen Gerät vermitteln. Doch nicht nur zwischen Endgeräten, die meistens Computersysteme enthalten, sondern auch innerhalb eines Computersystems befinden sich Verbindungsnetze, um die einzelnen Elemente des Systems mit einander zu verbinden. Es gibt eine grosse Nachfrage nach immer leistungsfähigeren Verbindungsnetzen. Die Anforderungen beschränken sich dabei nicht nur auf die Geschwindigkeit der Informationsübertragung, sondern betreffen beispielsweise auch die Flexibilität der Topologie. Einer der wichtigsten Gründe, weshalb an dieser Forschung ein grosser Bedarf besteht, ist die Tendenz weg von Einzel- und hin zu Multiprozessor- oder Multiuser-Systemen. Sobald mehrere Prozessoren oder Teilsysteme vorhanden sind, die zu einem Computer-Gesamtsystem verbunden werden, kommt den Verbindungsnetzwerken dieser Systeme eine grosse Bedeutung zu.

Die vorliegende Arbeit beschäftigt sich damit, ein Routing-Konzept für dynamische Rekonfigurationen zu entwickeln und in das bestehende Simulationstool CINSim (A Component Based Interconnection Network Simulator [2]) zu implementieren. Mit

dynamischer Rekonfiguration wird es möglich sein, je nach Kommunikationsmuster im Netzwerk die Geschwindigkeit und Bandbreite der Übertragung zu erhöhen. Dabei werden die Meldungen, die das Netz passieren, gestoppt, und die Verbindungen innerhalb des Netzwerks werden neu angelegt, so dass häufige Kommunikationswege kürzer werden. Wenn das Netzwerk rekonfiguriert ist, soll ein spezielles Routingverfahren für Pakete, die im Netz verblieben sind, die neuen Wege zu ihren Zielen ermitteln. Eine solches dynamisch rekonfigurierbares Netz könnte als Network on Chip auf einem FPGA (engl. field programmable gate array) realisiert werden. FPGAs haben die Eigenschaft, dass sowohl ihre inneren Gatterstrukturen als auch die Verbindungsleitungen durch digitale Befehle, also ohne mechanischen Eingriff, von aussen rekonfigurierbar sind.

Das zweite Unterkapitel gibt eine Einführung in CINSim und in Rekonfiguration. Das dritte Unterkapitel zeigt anhand dessen auf, worin die genaue Aufgabenstellung dieser Arbeit liegt. Das folgende Kapitel 2 führt dann in die verwendeten Begriffe der Netzwerktehorie ein und zeigt auf, welche Schwerpunkte eine zu entwickelnde Theorie der dynamischen Rekonfiguration haben muss. Ein 3. Kapitel wird die entwickelte Theorie der dynamischen Rekonfiguration und das allgemeine Routingverfahren für dynamisch rekonfigurierte Netze erläutern. Kapitel 4 beschreibt Einzelheiten der Implementation des entwickelten Verfahrens in CINSim, das die Simulation allgemeiner Verbindungsnetze und bis anhin nur statische Rekonfigurationen bewältigt. Das 5. Kapitel wird die Funktionstüchtigkeit und Leistungsverbesserung des Konzeptes anhand von Testläufen und verschiedener Netzwerksimulationen belegen und Kapitel 6 präsentiert das Ergebnis des Leistungsvergleich zwischen statischer und dynamischer Rekonfiguration. Ein letztes Kapitel fasst die Arbeit zusammen und diskutiert mögliche weitere Forschungsschwerpunkte.

1.2 CINSim und statische Rekonfiguration

Das Komponenten-basierte Verbindungsnetz-Simulationssystem CINSim ist bisher so weit entwickelt worden, dass damit statische Rekonfigurationen modelliert und simuliert werden können. Dies dokumentieren die Arbeiten [2], [3] und [4]. Der Simulator benutzt das stochastische Simulationswerkzeug Akaroa-2 und die damit zusammenhängenden Simulationsprinzipien, die [5] zusammenfasst. Hier wird ein Überblick gegeben, welchen Entwicklungsstand CINSim zu Beginn dieser Master-Arbeit hatte. Aus diesem wird die Aufgabenstellung und der Entwicklungsbedarf ersichtlich, der mit dem Ziel, dynamische Rekonfigurationen mit CINSim simulieren zu können, verbunden ist.

1.2.1 Variablen der Netzwerkkonfiguration

In diesem Abschnitt werden die Themenbereiche von Verbindungsnetzen, die in den Abschnitten 2.2 bis 2.7 besprochen werden, vorgegriffen. Es sind diese Topologie, Switching, Routing, Scheduling, Multicast und Flusskontrolle. Dieses Kapitel wird CINSim und die damit zusammenhängende Aufgabenstellung klar abstecken. Für die Definitionen und theoretischen Details der verwendeten Begriffe wird jedoch auf das Kapitel 3 verwiesen. CINSim wurde entwickelt, um Verbindungsnetze jeglicher Variationen dieser Bereiche simulieren zu können. Eine graphische Benutzeroberfläche, genannt CINSim-GUI (von engl. graphical user interface), steht zur Verfügung, um die verschiedenen Topologien und Verfahren zu modellieren. Die Abbildung 1.1 zeigt zwei Ansichten der CINSim-GUI.

Topologie

In der CINSim-GUI können Netzwerke mit beliebiger Topologie modelliert und als Datei abgespeichert werden, die vom Simulationstool CINSim geladen und simuliert werden kann. Es werden nur unidirektionale Verbindungen verwendet, woraus aber auch bidirektionale Kanäle zusammengesetzt sind.

Switching

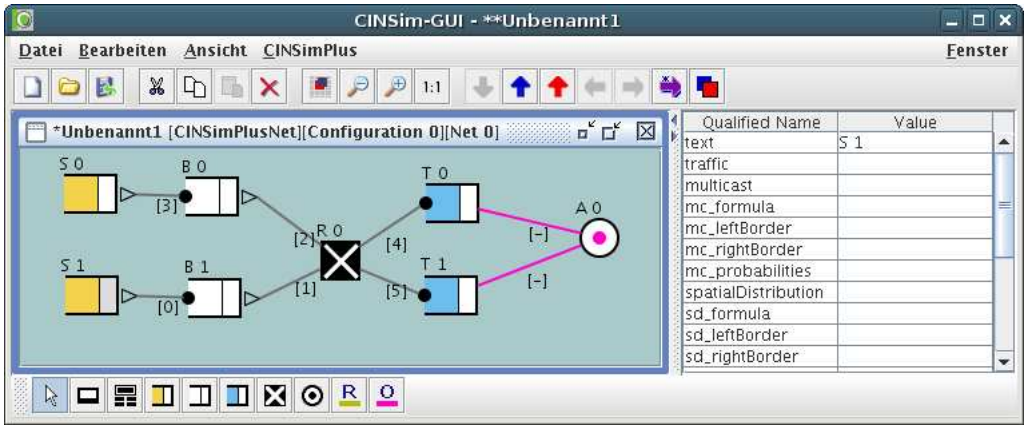
Das Switchingverfahren einer Simulation, die CINSim durchführt, ist über ein Menu der CINSim-GUI einstellbar. Dabei können die vier Switchingverfahren „Store And Forward“, „Wormhole“, „Virtual Cut Through“ und „Partial Cut Through“ ausgewählt werden.

Routing

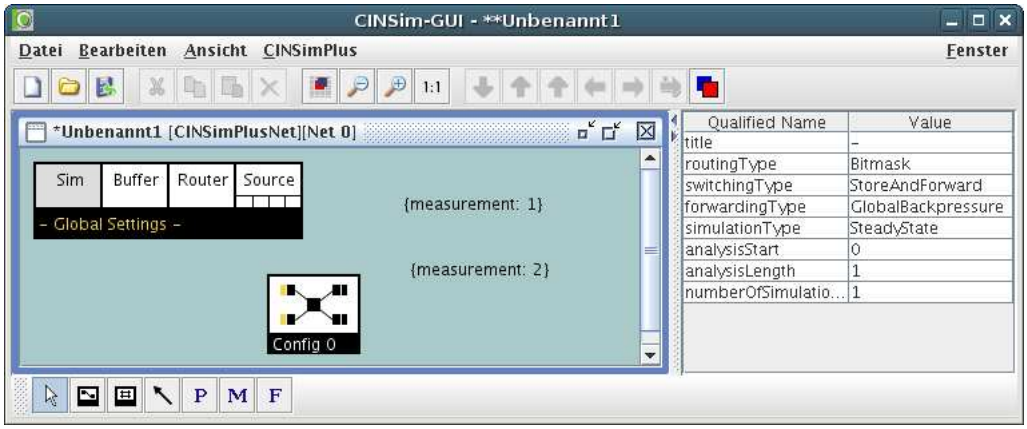
CINSim bietet nur drei verschiedene Routingverfahren an. Das erste und wichtigste ist Shortest Path. Weil 2D-Meshes mit diesem Verfahren nicht korrekt funktionieren, sind zwei weitere verwendbar: XY-Routing und West First, die speziell für 2D-Meshes zur Verfügung stehen.

Scheduling

Insgesamt sind zwölf verschiedene Scheduling-Verfahren verfügbar. Allen ist gemeinsam, dass das erste Flit vermittelt wird, das mit einem Scheduling-Verfahren als vermittelbar erkannt wurde. Schedulingverfahren, die aus verschiedenen Sende-Kombinationen die optimale erkennen und auswählen, werden in CINSim bisher nicht unterstützt.



(a) Netz-Zusammenbau



(b) Übersicht

Abbildung 1.1: Zwei Ansichten der CINSim-GUI. Das Bild (a) zeigt ein modelliertes Netzwerk und Einstellmöglichkeiten für die angewählte Quelle S1. A0 zeigt ein Messgerät an. Bild (b) zeigt die Übersichtsdarstellung und allgemeine Simulations-Einstellmöglichkeiten. Vor dem grünen Hintergrund sind zwei Messvariablen symbolisiert.

Multicast und Paketgenerierung

Die CINSim-GUI bietet die Möglichkeit, diverse Einstellungen der Quellen vorzunehmen. Multicasts sind zu beliebigen Zielmengen möglich, werden allerdings in dieser Arbeit nicht verwendet. Anders ist es mit den Einstellmöglichkeiten der Paketgenerierung, die in CINSim sehr vielfältig sind. Paketlänge (PL), Wahrscheinlichkeitsverteilung der Anzahl der Ziele pro Paket, Wahrscheinlichkeitsverteilung der Ziele und Prioritätsverteilung können für das gesamte Netz eingestellt werden, aber auch für jede Quelle separat. Die zeitliche Verteilung des Paketverkehrs zu variieren, ist eine weitere Fähigkeit von CINSim.

Flusskontrolle

CINSim unterstützt nur zwei Verfahren der Flusskontrolle. Es sind die Verfahren Local und Global Backpressure.

1.2.2 Simulation von statischer Rekonfiguration in CINSim

Die Rekonfiguration von Verbindungsnetzen zu simulieren, ist das eigentliche Entwicklungsziel von CINSim. Li schreibt über CINSim ([3], Seite 50):

> *Ein universales Werkzeug für die Bewertung der rekonfigurierbaren Verbindungsnetze gibt es bis jetzt nicht. CINSim soll diese Lücke schliessen.*

Ein ehrgeiziges Ziel, das noch unerreicht geblieben ist. Das sich entwickelnde Forschungsfeld der Rekonfiguration von Verbindungsnetzen zeigt, dass Modelle, die sie beschreiben, bisher auf spezielle Topologien oder Routingverfahren beschränkt blieben. Ein Beispiel dafür ist [6], das dynamische Rekonfiguration von 2D-Meshes bespricht. Li erwähnt [7] von Sanchez, der sich darin nicht auf eine Topologie, sondern auf das Switchingverfahren Wormhole beschränkt. Um in der vorliegenden Arbeit den Entwicklungsstand von den Zielen abgrenzen zu können, wird eine CINSim-spezifische Definition von statischer und dynamischer Rekonfiguration gegeben.

Definition 1 *Die beliebige Neuordnung von Komponenten eines Netzes, das keine Pakete in Puffern enthält, wird* statische Rekonfiguration *genannt. Wird ein Netz auf diese Weise verändert, das einen Knoten enthält, der einen nicht-leeren Puffer hat, heisst die Veränderung* dynamische Rekonfiguration.

Die Veränderung betrifft weder Routing, Switching, Scheduling noch Flusskontrolle und ist vollkommen beliebig. Diese Freiheit ist beispielsweise bei [6] nicht gegeben, wo die Rekonfiguration spezifisch für 2D-Meshes detailierter definiert worden ist. Definition 1 kennt keine veränderbare Puffergrösse, weil sie in CINSim nicht veränderbar ist während einer Rekonfiguration. Mögliche andere Verfahren für die neue Konfiguration werden aus demselben Grund ausgeschlossen.

CINSim war zu Beginn der vorliegenden Arbeit im Stande, statische Rekonfigurationen der in CINSim simulierbaren Netze durchzuführen. Dazu ist es nötig gewesen, eine Leerlauf-Phase zu modellieren, die genügend lange dauert, um alle sich im Netz befindlichen Pakete zu ihren Zielen zu vermitteln. Während dieser Leerlauf-Phase werden keine neuen Pakete generiert. Die Verzögerung zwischen dem Initiationszeitpunkt einer Rekonfiguration und der Durchführung muss gross sein, um keine Paketverluste zu riskieren, und die Auslastung des Netzes bleibt während der Leerlauf-Phase weit unter der maximalen Kapazität, eine Einbusse, die mit der dynamischen Rekonfiguration umgangen werden soll, die bisher in CINSim noch nicht simulierbar war. Abbildung 1.2 zeigt das Schema einer Rekonfiguration, wie es in der CINSim-GUI aufgebaut werden kann.

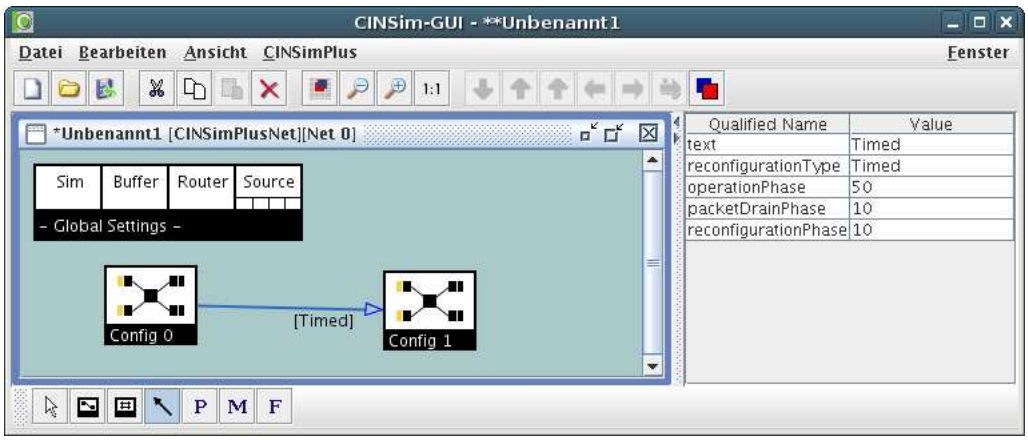


Abbildung 1.2: CINSim-GUI-modellierte Rekonfiguration eines Netzes: Das Bild zeigt zwei Netz-Symbole, die mit einem Pfeil verbunden sind. Der Pfeil symbolisiert die Rekonfiguration, deren Initiationszeitpunkt (Operation Phase), Länge der Leerlauf-Phase (Packet Drain Phase) und Länge der Rekonfiguration (Reconfiguration Phase) angegeben werden kann.

1.3 Aufgabenstellung der vorliegenden Arbeit

Die Aufgabe dieser Master-Arbeit besteht darin, CINSim für die Simulation von dynamischen Rekonfigurationen zu erweitern. Dabei soll die Funktionalität, die CINSim bisher für statische Rekonfiguration bereitstellt und die in diesem Kapitel beschrieben wurde, nicht eingeschränkt werden. [8] und [9] erwähnen, dass bei der dynamischen Rekonfiguration Deadlocks auftreten können, und diskutieren Deadlock-Vorbeugung und Deadlock-Auflösung als mögliche Lösungsansätze. Ein rekonfigurationsfestes Routing, das für die dynamische Rekonfiguration für CINSim neu entwickelt und implementiert wird, muss Teilaspekte dieser Ansätze enthalten. Da die Ansätze in diesen Literaturangaben leicht andere Definitionen für dynamische Rekonfiguration verwendet haben, ist es nötig, die Theorie der dynamischen Rekonfiguration für die gegebene Definition neu aufzubauen, deren Ziel es ist, einen Deadlock-Analyse- und -Auflösungs-Algorithmus anzugeben. Was unter einem Deadlock im Kontext einer Rekonfiguration zu verstehen ist, ist dazu vorgängig zu definieren (Definition 12 auf Seite 36). Eine grundsätzlich andere Herausforderung bedeutet die Implementation der erarbeiteten Konzepte, die vor allem zusätzliche Flusskontrolle-Mechanismen erfordert. Die neuen Mechanismen sollen es ermöglichen, jedes Paket, das sich zum Zeitpunkt der Rekonfiguration im Netz befand, an seine angepeilten Ziele zu vermitteln, auch wenn Pakete auf Irrwegen sind und Rückwege nehmen müssen. Die unbedingte Forderung, dass kein Paket während der Vermittlung verloren geht, bleibt auch für dynamische Rekonfiguration bestehen. Eine Vorhersage darüber, ob eine dynamische Rekonfiguration ohne Paketverluste durchgeführt werden kann, soll anhand von formalen Prüfregeln gemacht werden können. Diese Regeln aufzustellen, ist eine zusätzliche theoretische und implementatorische Aufgabe. Eine zufriedenstellende Implementierung lässt das Simulationstool CINSim Varianten von dynamischen Rekonfigurationen korrekt simulieren oder Warnungen ausgeben, die ein Fehlverhalten voraussagen. Die Kapitel 3 und 4 beschreiben die entwickelte Theorie und Implementation der dynamischen Rekonfiguration in CINSim und Kapitel 5 untermauert die Korrektheit der Konzepte durch eine Validierung. Kapitel 6 kann schliesslich die Leistungssteigerung der dynamischen zur statischen Rekonfiguration mit dem erweiterten Simulationswerkzeug erstmals aufzeigen.

Kapitel 2

Grundlagen der Netzwerktheorie

2.1 Begriffsdefinitionen

Am Anfang aller Theorie der Verbindungsnetze steht deren Definition, wie sie in [8] auf Seite 2 gegeben wird. In diesen wichtigsten Begriff der Arbeit soll das erste Unterkapitel einführen.

Definition 2 *Ein* Verbindungsnetz *ist ein programmierbares System, das Daten von einem zum anderen Endgerät transportiert.*

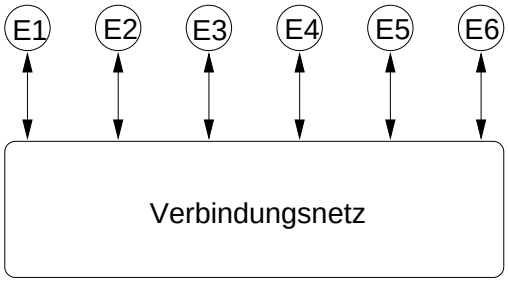


Abbildung 2.1: Funktionale Darstellung eines Verbindungsnetzes: Endgeräte (Endknoten, bezeichnet mit E1 bis E6) sind über Verbindungen mit dem Verbindungsnetz verbunden. Die Doppelpfeile signalisieren bidirektionale Verbindungen, die sowohl Daten in das Netz wie aus dem Netz übertragen.

Abbildung 2.1 zeigt ein Verbindungsnetz, an dem 6 Endgeräte angeschlossen sind. Endgeräte werden im Folgenden Endknoten genannt. Wenn der Endknoten E3 Daten an E5 vermitteln möchte, so sendet er eine *Nachricht* an das Verbindungsnetz und dieses vermittelt die Nachricht an E5 weiter. Das Verbindungsnetz kann je nach erhaltener Nachricht verschiedene Verbindungen aufbauen und ist in diesem Sinn programmierbar. E3 kann deshalb zu einem späteren Zeitpunkt auch zu anderen Endknoten als zu E5 senden. Die Funktionalität des Verbindungsnetzes wird dadurch erreicht, dass es aus verschiedenen Komponenten aufgebaut ist. Es enthält Puffer, Verbindungen, Kreuzschienenverteiler (auch Switches genannt) und Kontroller, die alle mit einander verknüpft sind, um die Datenvermittlung zu gewährleisten. Das Verbindungsnetz wird aus diesem Grund ein System genannt.

Ein Verbindungsnetz besteht aus *Vermittlungsknoten* (auch Knoten oder Router genannt) und *Verbindungen* zwischen ihnen. Bei der Paketvermittlung wird zwischen zwei Endknoten nicht eine feste Verbindung hergestellt wie beispielsweise bei einem Telefongespräch, sondern es werden einzelne Informationseinheiten, songenannte *Pakete*, jeweils über mehrere Stationen von Knoten zu Knoten weitervermittelt. Dabei generieren *Quellen* Pakete, die zu einem oder mehreren *Zielen* des Netzwerks versendet werden. Eine Quelle kann nur Pakete in das Netz einspeisen und ein Ziel kann nur Pakete empfangen. Daraus wird klar, dass das Endgerät der Definition 2 aus einer Quelle und einem Ziel besteht. Jedes Paket trägt die Informationen, zu welchen Zielen es vermittelt werden soll. In den Knoten wird jeweils gemäss dem *Routing* eine Entscheidung gefällt, in welche Richtung das Paket weitervermittelt wird, die davon abhängt, welche Ziele das Paket hat. Das Routing geschieht im Kontroller eines Knotens. Abbildung 2.2 zeigt ein Verbindungsnetz, aufgebaut aus diesen Komponenten. Quellen und Ziele sind ebenfalls abgebildet, die streng genommen nicht zum Verbindungsnetz gehören, doch der Einfachheit halber im Rahmen dieser Master-Arbeit als Komponenten betrachtet werden.

Bei der festen Verbindung, die bei einem Telefongespräch aufgebaut wird, muss die Information nicht zwischengespeichert werden, denn sie geht direkt vom Sender zum Empfänger. Da aber die Informationsübertragung bei der Paketvermittlung schrittweise von Knoten zu Knoten verläuft, wobei in jedem Netzzyklus ein Paket oder ein Paketteil nur eine Station weiter vermittelt wird, müssen die Pakete vor jedem Router zwischengespeichert werden. Dies geschieht in den *Puffern* der Knoten. Puffer haben eine bestimmte Speichergrösse, so dass sie auch besetzt sein können und keine weiteren Pakete mehr aufnehmen. Puffer befinden sich für die Modellvorstellung, die im Rahmen dieser Arbeit verwendet wird, direkt vor den Eingängen der Kreuzschienenverteiler. Diese Positionierung der Puffer führt zu einer Eingangspufferung. Die Kreuzschienenverteiler werden $m \times n$-*Switches* genannt. m ist dabei die Anzahl der Eingänge und n die Anzahl der Ausgänge eines Switches. Jeder Ausgang kann unabhängig von den ande-

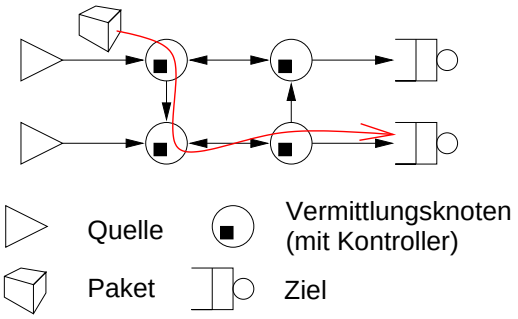


Abbildung 2.2: Verbindungsnetz, aufgebaut aus Komponenten

ren auf jeden beliebigen Eingang des Switches geschaltet werden, was vom Routing des Knotens gesteuert wird. Sowohl ein Eingang als auch ein Ausgang wird als I/O (engl. Input/Output) bezeichnet.

Lange Nachrichten können aus mehreren Paketteilen bestehen, die *Flits* genannt werden (Abkürzung für engl. flow control units). Alle Flits eines Paketes gehen den selben Weg durch das Netzwerk und können höchstens den Abstand von je einer Verbindung zueinander haben. Der Nachteil, dass ein vermittelter Nachrichtenteil warten muss, bis das ganze Paket den Knoten erreicht hat, wird so beseitigt. Denn sobald ein Flit des Pakets einen Knoten erreicht hat, kann es je nach Vermittlungsverfahren zum nächsten weitervermittelt werden. Die Übermittlungszeit des Paketes von einer Quelle zu einem Ziel kann dadurch verringert werden. Unter welchen Umständen eine solche Durchschaltung eines Paketes erlaubt wird, bespricht das Scheduling in Abschnitt 2.5. Das vorderste Flit eines Pakets wird als *Kopf* bezeichnet, das auch eine oder mehrere Zieladressen enthält. Das letzte Flit eines Pakets hat die Bezeichnung *Schwanz*. Alle anderen Flits heissen *Datenflits*, was nicht bedeutet, dass der Kopf und der Schwanz nicht auch Daten tragen können. Abbildung 2.3 zeigt das Innere eines Knotens mit seinen Bestandteilen und ein Paket, dessen Flits teilweise in diesem Knoten zwischengespeichert sind. Der Speicherplatz eines Puffers ist als maximale Anzahl von aufzunehmenden Flits definiert.

Mit der Einführung der Flits sind die grundlegenden Begriffsdefinitionen, die im Rahmen dieser Arbeit verwendet werden, abgeschlossen. In den folgenden Abschnitten werden detailiertere Verfahren beschrieben, die in CINSim Verwendung finden. Für die Beschreibeung des Simulationstools muss geklärt werden,

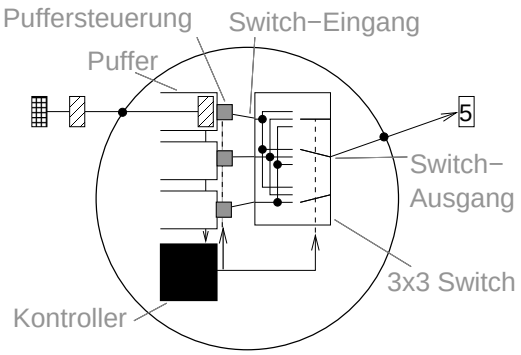


Abbildung 2.3: Netzknoten und seine Bestandteile. Am detailiertesten ist der 3x3-Switch dargestellt mit einem Schalter pro Ausgang, die alle vom Kontroller angesteuert werden. Die Ansteuerung ist durch die Pfeile dargestellt. Der Kontroller liest die Typen und Zieladressen der vordersten Flits in den Puffern und berechnet daraus die Schalterstellungen des Switches und steuert das Sendeverhalten der Puffer. Das gezeigte Paket hat einen Kopf, der die Zieladresse 5 hat, und ist 4 Flits lang, von denen das letzte mit einer speziellen Schraffierung das Schwanz-Flit ist.

- wie Kontroller für Kopf-Flits den richtigen Switch-Ausgang wählen
- wie verhindert wird, dass mehrere Flits gleichzeitig denselben Switch-Ausgang benutzen
- wie ein Paket mit mehreren Zielen verzweigt werden kann und
- wie erkannt wird, ob der nachfolgende Knoten das zu versendende Flit aufnehmen kann.

Die weiteren Abschnitte des Unterkapitels sind folgendermassen gegliedert: Der Abschnitt Topologie bespricht die Möglichkeiten, wie Knoten verbunden werden können. Switching erläutert die Charakterisierung von verschiedenen Vermittlungsverfahren von Paketen. Die Abschnitte Routing, Scheduling, Multicast und Flusskontrolle besprechen schliesslich der Reihe nach die formulierten Fragen.

2.2 Topologie

Die Art, wie Knoten verbunden sind, ist für die Kommunikationsleistung des Verbindungsnetzes entscheidend. Dieses Kapitel wird einige Topologien von Verbindungsnetzen einführen und deren Vor- und Nachteile diskutieren. Dazu müssen folgende Eigenschaften definiert werden:

Definition 3 *Wird ein Netzwerk in zwei etwa gleich grosse Teile getrennt, so ist die* Halbierungsbandbreite *die minimale Anzahl an Verbindungsleitungen, die dabei durchtrennt werden müssen.*

Definition 4 *Wenn ohne sprunghaft ansteigende Hardwareinvestitionen eine Topologie auf eine grössere Anzahl von Knoten ausgeweitet werden kann, so wird die Topologie* skalierbar *genannt.*

Definition 5 *Der* Durchmesser *eines Netzwerks ist die Anzahl Verbindungen, die minimal benutzt werden, um eine Nachricht zwischen zwei Knoten zu versenden, die am weitesten von einander entfernt sind.*

Die Halbierungsbandbreite ist aus [8], Seite 48, entnommen und die anderen Definitionen wurden in [10] gegeben, sind aber auch in [8] mit anderen Begriffen beschrieben (network implementation cost und largest minimal hop count H_{max}). Ein fairer Leistungsvergleich verschiedener Verbindungsnetze ist mit den Leistungsgrössen durchschnittliche Verzgerung und durchschnittlicher Datendurchsatz pro Endknoten und der Pfaddiversität möglich. In [8] sind diese Grössen folgendermassen definiert:

Definition 6 *Die* Verzögerung *ist die Zeit, die benötigt wird, um ein Paket durch das Netz zu befördern.*

Die durchschnittliche Verzögerung pro Endknoten definiert sich als statistischer Mittelwert aller Paketverzögerungen, gemittelt über alle Endknoten. [8] unterscheidet dabei die Anteile Kopf-Latenz T_h und Serialisierungslatenz T_s. T_h ist die Zeit, die das Kopf-Flit des Paketes hat, um von der Quelle zum Ziel zu gelangen und T_s ist die Zeitdifferenz der Ankunft von Kopf und Schwanz eines Pakets. Unter der Verzögerung wird die Kopf-Latenz verstanden.

Definition 7 *Der* Durchsatz *eines Netzwerks ist die Datenmenge pro Zeit, die ein Netzwerk pro Endknoten verarbeiten kann.*

Definition 8 *Die* Pfaddiversität *ist die Anzahl kürzester Pfade zwischen zwei Endknoten.*

Die durchschnittliche Pfaddiversität ist ein Wert der Robustheit und Flexibilität eines Netzes, die verhindert, dass Engpässe von Datenströmen auftreten, während andere Teile des Netzes ungenutzt bleiben. Die Leistungsgrössen sind durch die Eigenschaften der Topologie bestimmt. Deshalb muss beim Entwurf leistungsfähiger Topologien darauf geachtet werden, dass sie möglichst gute Eigenschaften besitzen. Erwünscht ist für n kombinierte Knoten (Verbindungsknoten, mit denen Endknoten verbunden sind, Abbildung 2.4) eine Halbierungsbandbreite und eine Pfaddiversität der Ordnung $O(n)$ und ein Durchmesser der Ordnung $O(\log(n))$. Die Skalierbarkeit einer Topologie, die in Hardware umgesetzt werden soll, muss gegeben sein.

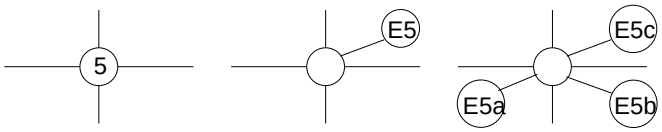


Abbildung 2.4: Ein kombinierter Knoten (links) besteht aus einem Router und einem oder mehreren Endknoten (mitte, rechts).

Die Tabelle 2.1 zeigt fünf einfache Topologien, besprochen in [10], und vergleicht deren Eigenschaften. Unter diesen Topologien findet sich keine, die den genannten Leistungsanforderungen entspricht. 1D-Mesh, Ring und Baum sind skalierbare Netzwerke, doch haben entweder eine zu kleine Halbierungsbandbreite oder einen zu grossen

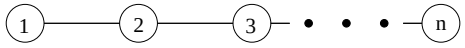
Durchmesser. Die vollständige Vernetzung und der Stern hingegen sind zwar sehr leistungsfähige Topologien, doch haben eine schlechte Pfaddiversität und sind nicht skalierbar. Beispielsweise hat ein Stern mit 1024 Endknoten einen Zentralknoten mit 1024 Ein- und Ausgängen. Ein solcher 1024×1024-Kreuzschienenverteiler stellt einen immensen Hardwareaufwand dar. Werden hingegen 1024 Knoten jeder mit jedem verbunden, so würde dies $\frac{1024 \cdot 1023}{2} = 523776$ Leitungen benötigen.

Kompliziertere Netzwerktopologien verbinden die Vorteile der einfachen Topologien. Zusätzliche Leitungen und Knoten haben das Ziel, Mischformen der verschiedenen einfachen Topologien zu erzeugen und ihre Nachteile zu elliminieren. Es ergeben sich dadurch Topologien, die interessante Verzögerungs- und Durchsatzeigenschaften haben und dabei skalierbar bleiben und eine gute Pfaddiversität aufweisen. Vier solche Topologien werden in der Tabelle 2.2 vorgestellt, die in [10] behandelt werden.

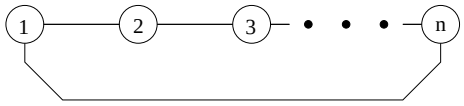
Unter den komplizierteren Topologien ist der Fat-Tree eine der leistungsstärksten, weil er eine grosse Halbierungsbandbreite hat bei kleinem Durchmesser und guter Skalierbarkeit und Pfaddiversität. Mehrstufige Verbindungsnetzwerke, auf Englisch Multistage Interconnection Networks (MINs), sind dem Fat Tree ähnlich und sollen hier in Kürze gemäss [11] beschrieben werden. MINs werden unterteilt in unidirektionale und bidirektionale Netzwerke und topologisch unterschieden nach dem Verbindungs-Schema (Cube, Butterfly) ihrer Verdrahtung. Abbildung 2.5 zeigt drei verschiedene Verbindungsnetze, die durch diese Merkmale zu unterscheiden sind. Von diesen Netzen wird im Rahmen der Master-Arbeit nur das Butterfly BMIN (bidirektionales MIN) verwendet. Die Bezeichnung UMIN bedeutet „unidirektionales MIN“.

2.3 Switching

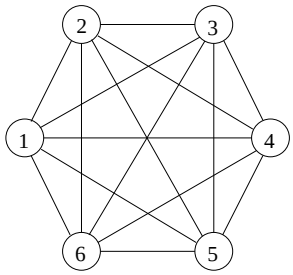
Die Darstellung der verwendeten Switching-Verfahren orientieren sich an [3], einer Diplomarbeit, in der das komponenten-basierte Verbindungsnetz-Simulationssystem CINSim verwendet wurde, das auch in dieser Master-Arbeit zur Anwendung gekommen ist. Die Verfahren, die in CINSim verwendet werden, sind Store-And-Forward (SAF), Wormhole, Virtual-Cut-Through (VCT) und Partial-Cut-Through (PCT) Switching. Ausser für Wormhole werden im folgenden die Abkürzungen der Bezeichnungen verwendet. Jedes dieser Verfahren ist Takt-gesteuert. In jedem Taktzyklus werden alle anstehenden Sendungen durchgeführt, die gleichzeitig erfolgen können. Ein Puffer kann demnach nie zweimal senden oder empfangen und ein Flit kann nie weiter als bis zum nächsten Knoten weitergesendet werden.



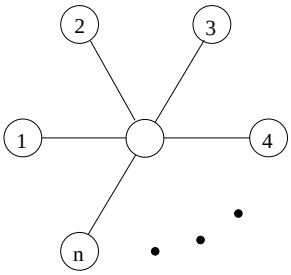
(a) 1D-Mesh (Kette)



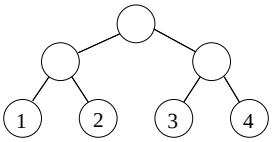
(b) Ring



(c) vollständig vernetzt



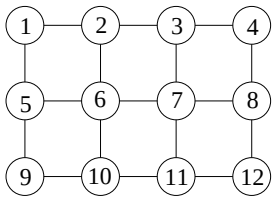
(d) Stern



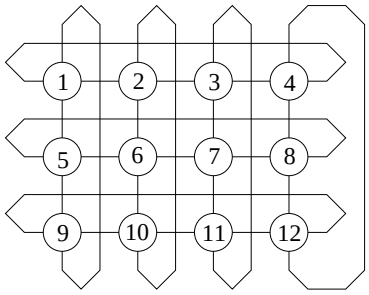
(e) Baum

Topologie	**Halbierungs-bandbreite**	**Durch-messer**	**maximale Pfad-diversität**	**skalierbar**
1D-Mesh (Kette)	1	n	1	Ja, nur eine Verbindung pro neuem Knoten.
Ring	2	$\frac{n}{2}$	2	Ja, aus gleichem Grund wie 1D-Mesh.
vollständig vernetzt	$\frac{n^2}{4}$	1	1	Nein. Neuer Knoten benötigt zusätzliche n Leitungen und pro Router einen zusätzlichen I/O.
Stern	$\frac{n}{2}$	2	1	Nein. Zentralknoten benötigt n I/Os.
Baum	1	$2\log(n)$	1	Ja. Konstante Router-I/O-Anzahl und Verbindungs-Zahl steigt nur logarithmisch mit n.

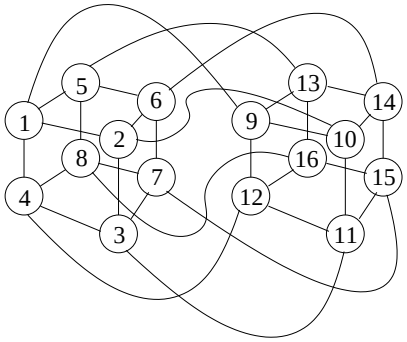
Tabelle 2.1: Diagramme und Vergleiche einfacher Netzwerktopologien. Knoten mit Nummern sind kombinierte Knoten, die aus einem Router und einem oder mehreren Endknoten bestehen, wie Abbildung 2.4 zeigt. Innere Netzknoten enthalten keine Nummer, weil sie an keinen Endknoten angeschlossen sind. Die Verbindungen sind jeweils bidirektional.



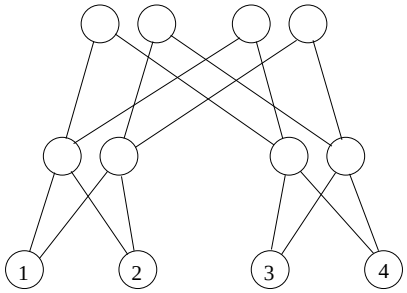
(a) 2D-Mesh



(b) 2D-Torus



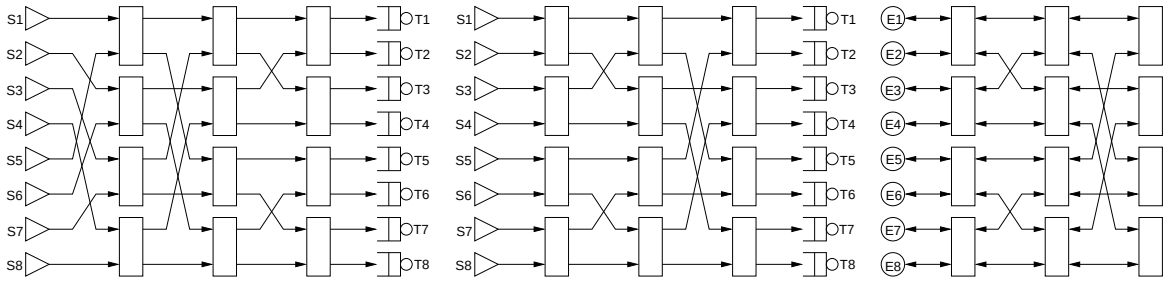
(c) Hyperkubus



(d) Fat Tree

Topologie	**Halbierungs-bandbreite**	**Durch-messer**	**maximale Pfaddi-versität ungefähr**	**skalierbar**
kD-Mesh	$\frac{n}{l+1}$	kl	$\frac{(kl)!}{l!^k}$	Ja.
kD-Torus	$\frac{2n}{l+1}$	$\left\lceil \frac{kl}{2} \right\rceil$	$\frac{(k\frac{l}{2})!}{(\frac{l}{2})!^k}$	Ja.
kD-Hyperkubus	$\frac{n}{2}$	$\log(n) = k$	$k!$	Ja.
Fat-Tree	n	$2\log(n)$	n	Ja.

Tabelle 2.2: Diagramme, Leistungsgrössen und Kommentare zu komplizierteren Topologien. kD-Mesh und kD-Torus sollten in allen Dimensionen etwa gleich viele Schichten haben, damit die formalen Ausdrücke den tatsächlichen Werten nahe kommen, wobei $l = \sqrt[k]{n} - 1$ die Seitenlänge pro Dimension ist. Sie sind exakt für gleich viele Schichten in allen Dimensionen.



(a) Cube UMIN (b) Butterfly UMIN (c) Butterfly BMIN

Abbildung 2.5: Drei verschiedene mehrstufige Verbindungsnetzwerke: Cube UMIN, Butterfly UMIN und Butterfly BMIN. Das nicht aufgeführte Netzwerk „Cube BMIN“ hat dieselbe Verdrahtung wie das Cube UMIN und dieselben bidirektionalen Verbindungen und Endknoten wie das Butterfly BMIN.

2.3.1 SAF

Beim SAF-Switchingverfahren wird jedes Paket in jedem Puffer, den es passiert, vollständig zwischengespeichert. Das bedeutet, dass eine Sendeaktion von einem Knoten erst dann begonnen wird, wenn die vorangehende Sendeaktion abgeschlossen ist, das heisst, wenn der Schwanz des Pakets angekommen ist. Pakete, die n Flits enthalten, benötigen deshalb im SAF-Verfahren genau n Zyklen, um von einem Knoten vollständig zum nächsten versendet zu werden. Ein Weg über m Knoten benötigt mindestens $m{\cdot}n$ Zyklen. Dabei muss in jedem Puffer des Netzes Platz für ein ganzes Paket sein, bestehend aus n Flits.

2.3.2 Wormhole

Wormhole ist ein Verfahren, das Speicherplatz in den Puffern der Knoten einspart, so dass nur noch ein Flit Kapazität pro Puffer ausreicht. Diese Einsparung wird möglich, indem Wormhole jeden Knoten seine Flits unabhängig davon, ob der Schwanz des Pakets schon eingetroffen ist, weitervermitteln lässt. Im Fall der Blockierung eines Paket-Kopfes, die dadurch entstehen kann, dass ein anderes Paket vor ihm den Ausgang gewählt hat, den er passieren will, kann nun aber nicht das gesamte Paket in einen Puffer gesammelt werden. Dadurch werden alle hinteren Flits ebenfalls blockiert. Das Paket bewegt sich

erst dann fort, wenn der Kopf an seinen nächsten Knoten gesendet werden kann. Mit Wormhole braucht ein Paket mit n Flits über m Stationen im Idealfall nur $m + n - 1$ Zyklen.

2.3.3 VCT

Die Kombination der Vorteile von SAF und Wormhole wird im VCT-Verfahren erreicht, indem die Einsparung des Speicherplatzes weggelassen wird. Die schnelle Paketvermittlung von Wormhole über mehrere Stationen kann mit der Rückstau-Freiheit von SAF verbunden werden. Bei VCT hat jeder Puffer wie bei SAF die Speicherkapazität für mindestens ein ganzes Paket. Gleich wie bei Wormhole lässt VCT Flits unabhängig davon weitervermitteln, ob der Paket-Schwanz schon eingetroffen ist. Die Ausnahme bildet eine Blockierung des Kopfes. Wird er gestoppt, so wartet der Puffer, in dem der Kopf sich befindet, bis er den Schwanz des Paketes empfangen hat, wie es auch in SAF geschieht.

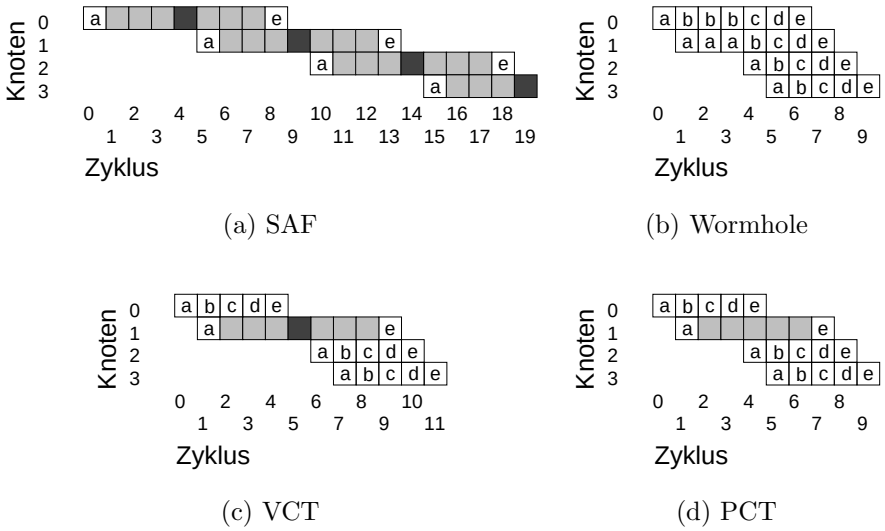
2.3.4 PCT

Eine Verbesserung des VCT erreicht das PCT-Verfahren, indem es bei einer Blockierung des Kopfes eines Pakets hintere Paketteile nachrücken und sobald möglich den Kopf weitersenden lässt. Im Übrigen entspricht es dem VCT-Verfahren. Auch bei PCT muss jeder Puffer mindestens ein ganzes Paket zwischenspeichern können. Die vier Switchingverfahren sind in Abbildung 2.6 als Zeitdiagramme einander gegenübergestellt.

2.4 Routing

Routing heisst auf Deutsch „Wegfindung“ oder „Routenplanung“ und beschreibt die Methode, wie ein Paket seinen Weg durch das Netzwerk findet. Dafür müssen die Kontroller der Knoten die Paketkopf-Informationen auswerten, um das Paket durch den richtigen Ausgang des Knotens weiter zu senden.

[8] bespricht eine grosse Anzahl von verschiedenen Routingverfahren. Hier soll nur die Klassifizierung in *deterministische*, *gedächtnislose* und *adaptive* Routingverfahren erwähnt werden. Deterministisch ist ein Routingverfahren, das eine gegebene Paketkopf-Information für einen bestimmten Knoten immer an den gleichen Ausgang weitervermittelt. Ein Routingverfahren gilt dann als gedächtnislos, wenn es nur den aktuellen Netzzustand zur Wegfindung benutzt. Adaptive Routingverfahren sind im Gegensatz dazu fähig, frühere Netzzustände und Sendevorgänge zur Ermittlung der besten Wegfindung zu nutzen. Jeder frühere Netzzustand, der für eine spätere Wegfindung genutzt werden



(a) SAF

(b) Wormhole

(c) VCT

(d) PCT

Abbildung 2.6: Zeitdiagramme der Vermittlung eines Pakets mit 5 Flits, das über drei Knoten versendet wird. Bei den Cut-Through-Verfahren Wormhole, VCT und PCT wird im Knoten 1 das Paket zwei Zyklen lang blockiert. Wormhole und PCT verlieren dadurch beide nur zwei Zyklen, VCT jedoch vier, weil es auf den Schwanz wartet, bevor es weitersenden lässt. Ein hellgraues Feld bedeutet, dass mehrere Flits in einem Puffer liegen und ein dunkelgraues, dass ein Puffer das ganze Paket enthält.

soll, muss dazu im Router gespeichert werden, was einen Nachteil von adaptiven Routingverfahren darstellt. Jedes deterministische Verfahren ist gedächtnislos, jedoch gibt es auch nicht-deterministisches gedächtnisloses Routing, das einen Teil der Vermittlungs-Entscheidung dem Zufall überlässt, so dass beispielsweise eine gleichmässige Ausnutzung mehrerer möglicher Pfade erzielt wird. Es gibt auch minimales gedächtnisloses Routing, das keine Umwege durch das Netz zulässt, wie es sonst bei zufälligen Verfahren geschehen kann.

Das einzige in dieser Master-Arbeit diskutierte Routingverfahren wird *Shortest-Path* (SP) genannt, und ist ein nicht-deterministisches, minimales gedächtnisloses Routing. Es ist das optimale Routing für BMINs. Dieses Routingverfahren ist aber nicht für alle Topologien geeignet. Bei Meshes kann es mit SP zu einer Verklemmung der Paketbewegungen kommen. Ist für eine Topologie nicht SP Routing verwendbar, so wird im Folgenden angenommen, dass ein anderes Routingverfahren existiert und verwendet wird, das Verklemmungen vermeidet. CINSim bietet für Meshes zwei spezielle Routingverfahren an, wie in 1.2.1 erläutert wurde. Abbildung 2.7 demonstriert eine Konfliktsituation eines 2D-Meshes mit Shortest Path Routing.

2.5 Scheduling

Sind zwei Pakete gleichzeitig dazu bestimmt, über denselben Ausgang eines Knotens versendet zu werden, so sorgt das Scheduling dafür, dass eine Entscheidung gefällt wird. Eine faire und gleichzeitig einfach zu implementierende Methode ist das Zufallsprinzip. Das zufällige Scheduling ist das einzige in dieser Arbeit verwendete Scheduling-Verfahren.

2.6 Multicast

Jedes Paket kann eines oder mehrere Ziele haben. Im zweiten Fall wird es Multicast-Paket genannt. Im Knoten, wo sich die Wege für die verschiedenen Ziele verzweigen, wird ein Multicast-Paket vervielfältigt und je eine Kopie über jeden benutzten Ausgang des Knotens geschickt. Die Zielmenge aller vervielfältigten Pakete muss aber dieselbe bleiben. Mit dieser Methode können Multicasts mit beliebig vielen Zielen durchgeführt werden. Einen Multicast illustriert Abbildung 2.8. Ein Spezialfall ist der Broadcast, bei dem ein Paket jedes Ziel des Netzes ansteuert. Paketsendungen, die nur ein Ziel haben, werden Unicasts genannt.

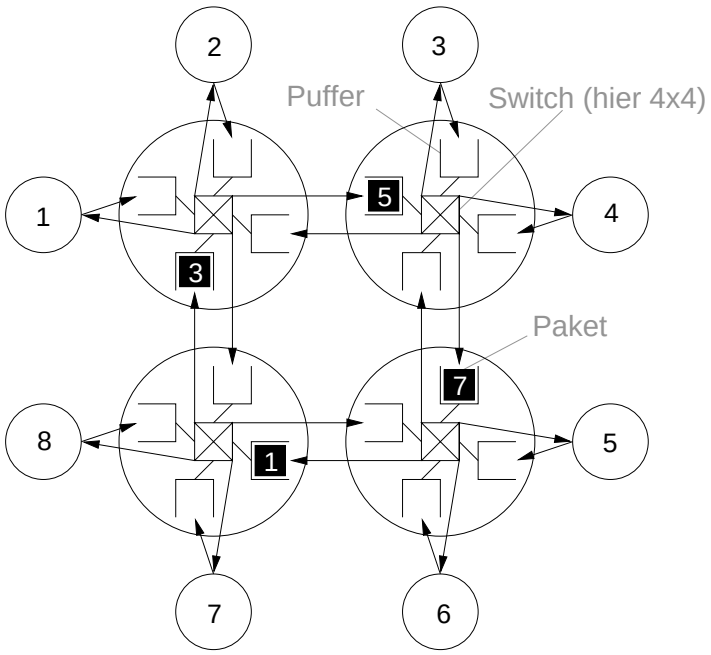


Abbildung 2.7: 2D-Mesh mit Shortest Path Routing (Paketlänge 1): Es entsteht eine Situation, wo alle Pakete stillstehen, denn jedes Paket muss an einen Puffer weitervermittelt werden, der schon voll ist. Eine Lösung des Problems ist XY- oder West-First-Routing, die eine solche Situation verhindern. Hier sind die von nun an gängigen Symbole für die Komponenten bezeichnet.

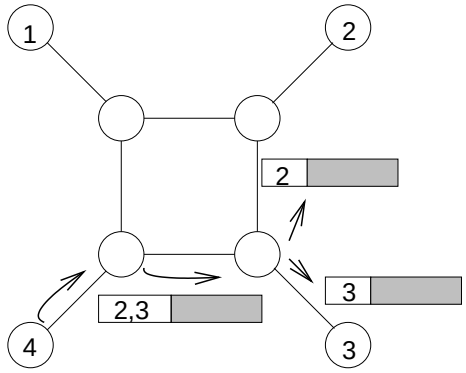


Abbildung 2.8: Multicast

2.7 Flusskontrolle

[8] kennt verschiedene Verfahren der Flusskontrolle, die Paketbewegungen verhindern oder zulassen: Credit-based, on/off und ack/nack. Die drei Verfahren sollen kurz beschrieben werden.

Credit-based: Bei dieser Flusskontrolle benachrichtigt der zu empfangende Router (Netzknoten) alle möglichen Sender, ob der jeweilige Eingang des Routers empfangsbereit ist oder nicht. Die Empfangsbereitschaft ergibt sich aus dem Puffer-Füllstand des Eingangs und der Möglichkeit des Routers, ein Flit aus diesem Puffer weiter zu vermitteln.

on/off: Dieses Verfahren der Flusskontrolle optimiert das Credit-based, indem anstelle der Empfangsbereitschaft der Wechsel derselben signalisiert wird. Empfängt ein Router von seinem Nachfolger ein on-Signal, so sendet er ununterbrochen Flits an diesen, bis er von ihm ein off-Signal erhalten hat.

ack/nack: Dies ist eine Flusskontrolle, bei der jedes einzelne Flit bestätigt wird, wenn es fehlerfrei empfangen wurde („ack“ von engl. „acknowledge“). Wird eine Unstimmigkeit beim Empfangen bemerkt, so wird die Meldung „nack“ (engl. „not acknowledge“) an den Sender zurückgegeben, worauf dieser das Flit erneut sendet. Die Redundanz in der Bandbreite der Übertragung wird dadurch grösser. Dies wird bei Netzen in Kauf genommen, deren Paket-Vermittlungen nicht sicher sind, das

heisst, bei denen Pakete fehlerhaft beim Empfänger ankommen können. Die Fehlerhäufigkeit der Pakete kann bei solchen Netzen mit der ack/nack-Flusskontrolle auf Null reduziert werden.

On/off bringt das Risiko des Paketverlustes mit sich und ack/nack hat mit den Rückmeldungen des Empfängers eine Redundanz der Kommunikationsbandbreite. Es wird aber eine verlustfreie Übertragung vorausgesetzt, weshalb einerseits Rückmeldungen des Empfängers über allfällige Misserfolge der Übertragung unnötig sind und andererseits das Risiko eines Paketverlustes nicht eingegangen werden darf. Deshalb kommt nur Credit-based Flusskontrolle in Frage.

Zwei verschiedene Verfahren sind anwendbar, Local und Global Backpressure (LBP/GBP). Bei LBP sendet ein Knoten nur dann ein Flit an einen anderen, wenn dieser in dem entsprechenden Puffer noch freien Speicherplatz hat. GBP erlaubt auch das Versenden an volle nächste Knoten, wenn diese im gleichen Zyklus ebenfalls senden. In einer Kette von sich gegenseitig sendenden Knoten muss bei GBP nur der vorderste einen freien Speicherplatz haben.

2.8 Paketgenerierung

Die beinahe unbegrenzten Einstellmöglichkeiten, die CINSim zur Paketgenerierung in den Quellen bietet, wurden in Abschnitt 1.2.1 kurz zusammengefasst. Zwei von ihnen spielen im Bezug auf die Bewertung von dynamischen Rekonfigurationen im Kapitel 5 eine wichtige Rolle. Es sind Verkehrscharakteristik und angebotene Last, die beide die zeitliche Verteilung des Paketverkehrs betreffen.

Definition 9 *Die* Verkehrscharakteristik *ist die Beschreibung des zeitdiskreten stochastischen Prozesses $P[k]$ der Paketgenerierung in den Quellen. Für jedes k ist dabei $P[k]$ eine Zufallsvariable mit den möglichen Werten Eins oder Null (Bernoulli-Variable). Bei Eins wird ein Paket generiert, bei Null nicht.*

Stochastische Prozesse sind in [12] beschrieben. CINSim bietet vier verschiedene Verkehrcharakteristiken an, geometrische, periodische, pareto-verteilte und „Random Burst“. Bei der *geometrischen* Verkehrscharakteristik sind alle Zufallsvariablen $P[k]$ von einander unabhängig. Das bedeutet, dass die Quelle kein Gedächtnis hat. Die Entscheidung, ob zu einem Zeitpunkt k ein Paket generiert wird, ist unabhängig davon, was zu früheren Zeitpunkten gesendet wurde. Die Wahrscheinlichkeit, dass zwischen der Generierung zweier Pakete genau t Zyklen gewartet wird, hat eine geometrische Verteilung, daher der Name dieser Verkehrscharakteristik. Abbildung 2.9 zeigt diese Verteilung für die angebotene Last von 0,5. Bei der *periodischen* Verkehrscharakteristik ist die

Wartezeit zwischen den generierten Paketen konstant. *Pareto-verteilte* Verkehrscharakteristiken zeichnen sich durch starke Gegenahängigkeiten der Zufallsvariablen $P[k]$ aus. Sie haben in dieser Arbeit keine Relevanz, deshalb wird für Details auf [13] verwiesen. *Random Burst* ist eine Verkehrscharakteristik, die kurzzeitiges hohes Verkehrsaufkommen, sogenannte Bursts, und zwischenzeitlichen geringen Paketverkehr modelliert. Sie ist für diese Master-Arbeit ebenfalls nicht relevant. Ihre genauere Beschreibung findet sich in [3].

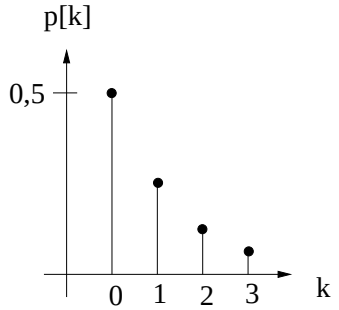


Abbildung 2.9: Geometrische Wahrscheinlichkeitsverteilung $p[k] = q\ (1-q)^{k-1}$ für $q = 0,5$.

Definition 10 *Die* angebotene Last *(OL, von engl. „offered load“) ist der Erwartungswert der Verkehrsverteilung $P[k]$ gemittelt über alle Zeiteinheiten k:*

$$OL = \lim_{n\to\infty} \frac{1}{n} \sum_{k=0}^{n} E(P[k]) \tag{2.1}$$

Eine Quelle generiert im Mittel mit der Wahrscheinlichkeit OL ein Paket, doch muss es im Fall einer blockierten Netzsituation wieder löschen. Das Paket wird dem Netz angeboten, doch kann nicht in jedem Fall versendet werden.

Kapitel 3

Theorie der Dynamischen Rekonfiguration

Die Theorie der dynamischen Rekonfiguration wird passend zur Definition 1 aufgebaut. Da keine Literatur vorhanden war, die eine solche vorzeichnet, sind die Konzepte eigens erarbeitet. Ähnliche Abhandlungen sind in [8] vorhanden, doch sind nicht daraus entnommen.

3.1 Kriterien einer dynamischen Rekonfiguration

Theoretische Modelle für die Konzepte, die im Kapitel 2 vorgestellt wurden, sind bekannt und bis in kleinste Details erarbeitet. Für die dynamische Rekonfiguration, die in Definition 1 gegeben wurde, ist hingegen noch kein Routingverfahren gegeben worden. Es muss erst noch entworfen werden. Es gibt zwar Teilbereiche, zum Beispiel für Gitterstrukturen [6], die in der Vergangenheit erschlossen worden sind, doch für die dynamische Rekonfiguration von BMINs muss erst noch ein Modell entworfen werden. Es wird sich zeigen, dass dieses beliebige Topologien erfasst und verschiedene Routingverfahren zulässt. In diesem Kapitel wird dieses Modell vorgestellt, das im Rahmen der Master-Arbeit entwickelt wurde.

Ein erster Abschnitt wird Routing-Information pro Router formal beschreiben. Ein zweiter wird die Bedingung für Paket-verlustfreie dynamische Rekonfiguration aufstellen und begründen, dass diese mit der Paketverteilung im Netz vor der Rekonfiguration zusammenhängt. Zum Schluss dieses Unterkapitels wird die Bedingung so verallgemeinert, dass sie auch anwendbar ist ohne den Bezug auf die Paketverteilung.

3.1.1 Vermittlungs-Informationen in einem Router

Jeder Router hat gewisse Vermittlungs-Informationen gespeichert, die jedem Ziel einen oder mehrere Ausgänge zuweisen. Für Pakete, die ein bestimmtes Ziel ansteuern, darf der Router nur die angegebenen Ausgänge verwenden, um die Pakete weiter zu vermitteln. Wichtig ist im Kontext der Theorie dynamischer Rekonfigurationen, dass es Ziele geben kann, für die ein Router keinen Ausgang hat. Für nicht rekonfigurierte Netze oder für statische Rekonfigurationen, die mit Paket-leeren Netzen erfolgen, existiert dieses Problem nicht. Der Grund ist, dass ein Paket, das von einem Router nicht mehr weitervermittelt werden könnte, dort auch nie hingesendet wird. Die Routing-Information ist in einem Netzwerk gerade so angelegt, dass jeder erlaubte Ausgang eines Routers wieder zu einem Router führt, für den ein weiterer Ausgang für dasselbe Ziel existiert oder das Ziel selbst erreicht wurde. Die Vermittlungs-Informationen in einem Router bestehen insgesamt aus einer Menge von vermittelbaren Zielen für jeden Ausgang des Routers.

Definition 11 *Die* Vermittlungs-Information *jedes Routers besteht aus Mengen $M_k \subset T$ von Zielen. Jedem Ausgang k des Routers ist eine Menge zugeordnet. Ein Ziel $t \in T$ ist genau dann in der Menge M_k, wenn über den Ausgang k ein Paket zum Ziel t vermittelt werden kann.*

3.1.2 Paketverlustfreie dynamische Rekonfiguration

Wurde für Netze ohne Rekonfiguration und für statisch rekonfigurierte Netze gezeigt, dass nie ein Paket in einem Puffer liegen kann, von dem aus keine Weitervermittlung möglich ist, so gilt dies bei der dynamischen Rekonfiguration nicht mehr. Ein Fall kann auftreten, bei dem sich das Netz rund um einen Puffer, in dem ein Paket liegt, so verändert, dass sich das Paket nicht mehr zu seinem Ziel vermitteln lässt. Wird ein solcher Fall nicht verhindert, muss an dieser Stelle das Paket gelöscht werden, um den weiteren Informationsfluss nicht zu behindern. Ganz allgemein ergibt sich dadurch für Paket-verlustfreie dynamische Rekonfiguration eine notwendige Bedingung: Für eine verlustfreie dynamische Rekonfiguration muss jedes Paket, das vor der Rekonfiguration von einem Puffer aus (über den dazugehörigen Switch) vermittelbar war, auch nachher wieder vermittelbar sein. Dies bedeutet, dass für jedes Paket, das sich während der Rekonfiguration im Netz befindet, folgende Bedingung gelten muss:

Bedingung 1 *Die Menge von Zielen M_x sei dem Paket als seine Ziele zugeordnet. Seien weiter die Mengen M_k den Ausgängen des Routers zugeordnet, in dem sich jetzt das Paket befindet. Es muss nun gelten, dass die Vereinigung aller M_k die Menge M_x*

von angepeilten Zielen als Teilmenge enthält:

$$M_x \subseteq \bigcup_{k=1}^{\#Ausgänge} M_k \tag{3.1}$$

Diese Bedingung impliziert eine Regel an eine dynamische Rekonfiguration: Ziele, die von einem Paket im Moment der Rekonfiguration noch angesteuert werden, dürfen nicht entfernt werden. Eine andere triviale Regel für verlustfreie Rekonfiguration ist das Verbot, nicht-leere Puffer zu entfernen.

3.1.3 Bedingung unabhängig von der Paketverteilung

Wird die Bedingung 1 auf ein Netzwerk angewendet, so kann bestimmt werden, ob eine Rekonfiguration zum betrachteten Zeitpunkt möglich ist oder nicht. Die Bedingung ist situationsabhängig, denn sie ist je nach Paketverteilung eingehalten oder verletzt. Soll ein Netz unabhängig von der Paketverteilung als verlustfrei dynamisch rekonfigurierbar erkannt werden, so reicht die Bedingung 1 nicht aus. Es muss eine strengere Bedingung aus ihr abgeleitet werden, die garantiert, dass bei jeder möglichen Paketverteilung die Rekonfiguration verlustfrei ablaufen wird. Dafür muss für die Ziel-Menge M_x die Gesamtmenge aller möglichen Ziele genommen werden. Dies sind nicht etwa alle bestehenden Ziele, sondern genau die, zu denen der vorangehende Router über den betreffenden Ausgang vermitteln konnte. Es muss für jeden Puffer gelten:

Bedingung 2 *Es bezeichne $M_{y,alt}$ die Menge, die demjenigen Ausgang des vorangehenden Routers zugeordnet ist, der zum Puffer führt. Dabei ist der Router gemeint, der vor der Rekonfiguration auf diese Beschreibung zutraf. Die Menge $M_{y,alt}$ muss nun eine Teilmenge der Vereinigung aller Ausgangsmengen $M_{k,neu}$ sein, die dem Router zugeordnet sind, in dem sich der Puffer in der neuen Konfiguration befindet:*

$$M_{y,alt} \subseteq \bigcup_{k=1}^{\#Ausgänge} M_{k,neu} \tag{3.2}$$

Dieselbe Implikation wie für Bedingung 1 lautet für die Bedingung 2 strenger: Es dürfen keine Ziele entfernt werden. Strenger ist auch die Einschränkung an die Router-Entfernung: Generell dürfen keine Puffer entfernt werden.

Die Bedingung 2 ist die allgemeinste notwendige Bedingung für verlustfreie dynamische Rekonfiguration, wenn man die Paketverteilung nicht im speziellen betrachten will. Es wurde angenommen, dass nur dort während der Rekonfiguration Pakete für

ein gewisses Ziel liegen können, wohin sie auch vor der Rekonfiguration hin vermittelt werden durften. Notwendig ist die Bedingung in dem Sinne, dass wenn sie nicht eingehalten wird, Pakete gelöscht werden müssen. Sie garantiert, dass keine Verluste auftreten, wenn auftretende Paketvermittlungs-Verklemmungen im neuen Netz gelöst werden können (im folgenden nur Paketverklemmungen genannt). Diese Verklemmungen treten tatsächlich auf. Deshalb kann eine allgemeine hinreichende Bedingung für verlustfreie Rekonfiguration nur im Zusammenhang mit einer Auflösung der verklemmten Paketsituation formuliert werden.

3.1.4 Beispiele zur Veranschaulichung

Es wurden zwei Bedingungen formuliert, um die verlustfreie Rekonfiguration zu prognostizieren. Die erste betrachtet ein Netz und ihre momentane Paketverteilung, die zweite nur das Netz. Wenn die zweite eingehalten ist, so ist die erste immer auch eingehalten. Die Umkehrung gilt nicht. Zur Veranschaulichung dieser Bedingungen sollen drei Beispiele von Rekonfigurationen gezeigt werden:

1. Ein Netz A_0, das momentan nicht verlustfrei in ein Netz A_1 rekonfiguriert werden kann, denn weder Bedingung 1 noch Bedingung 2 ist erfüllt (Abbildung 3.1).

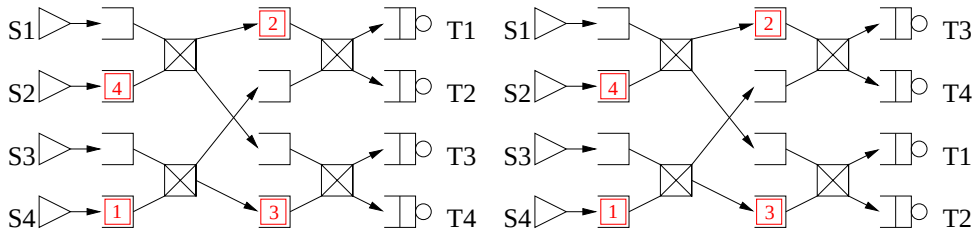
2. Dasselbe Netz A'_0, das wegen einer anderen Paketverteilung verlustfrei in A'_1 rekonfigurierbar wurde. Bedingung 1 ist erfüllt, nicht aber Bedingung 2 (Abbildung 3.2).

3. Eine andere Rekonfiguration $A_0 \rightsquigarrow A_2$, die im Vergleich zur Rekonfiguration $A_0 \rightsquigarrow A_1$ jederzeit verlustfrei durchführbar ist gemäss der erfüllten Bedingung 2 (Abbildung 3.3).

Wie es auch diese Beispiele zeigen, kann die Bedingung 2 vor jeglichen Paketbewegungen geprüft werden. Wenn sie gilt, dann für jeden Zyklus der Paketbewegung. Bedingung 1 hängt von der Paketverteilung im Netz ab und kann (wenn Bedingung 2 nicht erfüllt ist) je nach Situation eingehalten sein oder nicht.

3.2 Handhabung einer Paketverklemmung

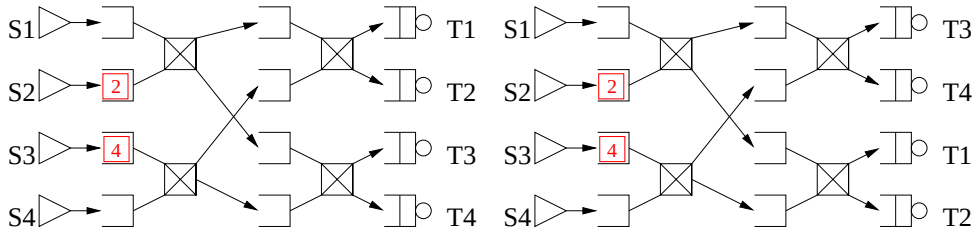
Im vorangehenden Abschnitt wurde die Vielfalt der möglichen dynamischen Rekonfigurationen eingeschränkt. Erfüllt eine Rekonfiguration $M \rightsquigarrow N$ (Überführung des Netzes M durch Modifikationen in das Netz N) die Bedingung 2, so sind alle Probleme ausgeschlossen, die zu einem direkten Paketverlust führen infolge von Paketen in Sackgassen.



(a) Netz A_0

(b) rekonfiguriertes Netz A_1 (vertauschte Ziele)

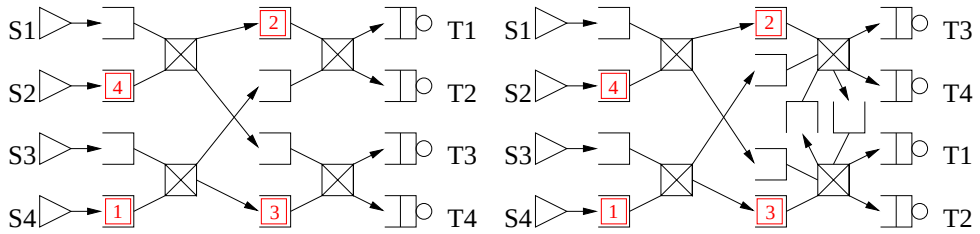
Abbildung 3.1: Bedingungs-Beispiel 1: Die Bedingung 2 ist nicht erfüllt. Da die weiter fortgeschrittenen Pakete nach der Rekonfiguration ihre Ziele nicht mehr erreichen, ist auch die Bedingung 1 nicht erfüllt.



(a) Netz A'_0 (wenige Pakete im Netz)

(b) rekonfiguriertes Netz A'_1 (vertauschte Ziele)

Abbildung 3.2: Bedingungs-Beispiel 2: Die Bedingung 2 ist für diese Rekonfiguration nicht erfüllt. Nur wenige Pakete sind im Netz, insbesondere ist die kritische zweite Puffer-Stufe gänzlich leer. Alle Pakete erreichen ihre Ziele auch nach der Rekonfiguration, also ist die Bedingung 1 erfüllt.



(a) Netz A_0

(b) rekonfiguriertes Netz A_2 (vertauschte Ziele und Umwegmöglichkeit)

Abbildung 3.3: Bedingungs-Beispiel 3. Nun ist die Bedingung 2 erfüllt: Alle Pakete erreichen ihre Ziele auch nach der Rekonfiguration, egal in welchem Puffer sie liegen. Dies garantieren die rückführenden Wege in der zweiten Stufe.

Die Garantie, dass eine solche Rekonfiguration verlustlos durchgeführt werden kann, ist dadurch noch nicht gegeben. Es können nach einer dynamischen Rekonfiguration Paketverklemmungen auftreten. Diese Möglichkeit zu begründen, die Verklemmungen zu analysieren, eine Auflösungsmethode anzugeben und die Leistungsfähigket der Methode zu diskutieren ist der Inhalt dieses Unterkapitels.

3.2.1 Grund des Auftretens von Paketverklemmungen

Die Netze M und N werden zusammen mit ihren Routing- und Switchingverfahren als Verklemmungsfrei angenommen. Ein Netz kommt niemals in einen Zustand, in dem Paketbewegungen dauerhaft blockiert werden. Dass dies bei ungeschickter Wahl der Topologie, des Routing- und des Switchingverfahrens passieren kann, verdeutlicht Abbildung 3.4. Hier wurde Wormhole als Switchingverfahren gewählt. Der Schwanz eines Paketes blockiert jeweils den Kopf des anderen.

Die Annahme besagt, dass das Problem weder für M noch für N auftritt, wenn die Netze separat arbeiten, das heisst, wenn alle Pakete im Netz tatsächlich von Quellen der momentanen Konfiguration generiert wurden. Diese Einschränkung wird bei einer dynamischen Rekonfiguration offensichtlich verletzt. Es gibt Pakete in der neuen Konfiguration, die noch vor der Rekonfiguration generiert wurden. Sie wurden von einer Quelle von M generiert, aber kommen in einem Puffer von N zu liegen. Deshalb ist die An-

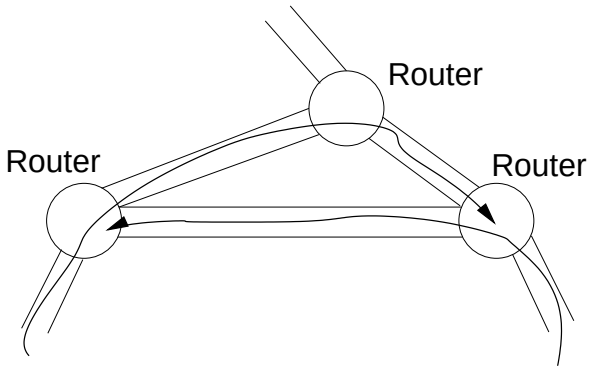


Abbildung 3.4: Beispiel einer Paketverklemmung

nahme von verklemmungsfreien Einzelnetzen keine Garantie für eine nicht verklemmte Übergangssituation. Ein Beispiel soll den Sachverhalt belegen. Eine Verklemmung wird dann verursacht, wenn viele Pakete im Netz sind, doch eine Rekonfiguration sie zwingt, von ihren schon zurückgelegten Wegen umzukehren. Dann müssen die Pakete Rückwege nehmen, die aber schon von anderen Paketen besetzt sind, was zu einem Paket-Stau führt. Eine solche Situation wird beispielsweise dadurch erreicht, dass man die Verteilung der Ziele um das ungefähre Zentrum des Netzes punktspiegelt. Abbildung 3.5 zeigt eine solche Situation. Es handelt sich dabei um eine Rekonfiguration, die die Bedingung 2 erfüllt. Das Netzwerk wird Bogen-Netz genannt, das auch in der Validierung wieder Verwendung findet.

Das Gesagte lässt darauf schliessen, dass nur dann eine Verklemmung auftreten kann, wenn ein Paket aus der Sicht des Routing-Verfahrens der neuen Konfiguration auf einem Irrweg ist. Dies ist auch beim gegebenen Beispiel in Abbildung 3.5 der Fall: beide Pakete wären gemäss dem Routing-Verfahren der neuen Konfiguration nie an ihre momentanen Positionen, sondern direkt an ihre Ziele vermittelt worden. Die folgende Erkennungsregel für Pakete auf Irrwegen basiert auf dem Prinzip, dass ein Paket genau dann auf einem Irrweg ist, wenn der neue vorangehende Router es nicht in den Puffer hätte versenden dürfen, wo es jetzt liegt. Dieser Puffer wird *Irrwegpuffer* genannt.

Regel 1 *Ein Paket, deren Menge von Zielen M_x ist, befindet sich in seinem Puffer auf einem Irrweg, wenn für die Zielmenge M_a des Router-Ausgangs zu diesem Puffer gilt:*

$$M_x \nsubseteq M_a \tag{3.3}$$

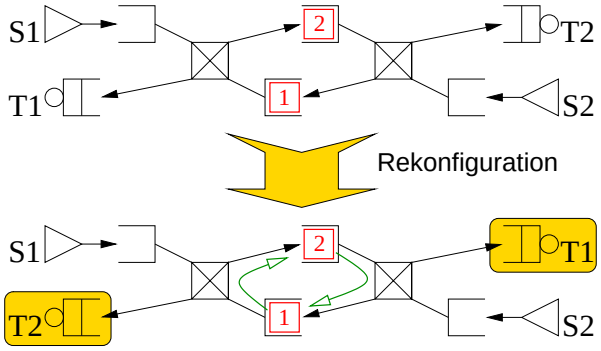


Abbildung 3.5: Beispiel einer Paketverklemmung, die nach einer Rekonfiguration auftritt: Die Rekonfiguration bestand darin, die Ziele um das Zentrum des Netzes punkt zu spiegeln. Nun müssen die Pakete Rückwege nehmen, was mit den Pfeilen angedeutet wird, doch verhindern gegenseitig ihre Weitervermittlung.

3.2.2 Identifikation von Paketverklemmungen

Zur eindeutigen Identifikation von Paketverklemmungen werden die in den Grundlagen eingeführten Flits betrachtet, die Grundeinheiten eines Pakets. Eine Verklemmung ist eine Gegenabhängigkeit von Flits und ihren Pufferpositionen. Die Pakete als Einheiten anzuschauen, reicht nicht aus, denn es können auch mehrere Flits desselben Pakets von einer Paketverklemmung betroffen sein. Um eine solche zu identifizieren, muss für jeden Puffer b_i eine Folgepuffermenge B_i ermittelt werden, und diese ist eindeutig durch das vorderste Flit dieses Puffers bestimmt. Bevor aber weitere Begriffe gebildet werden, soll ein Überblick über die Schritte einer Identifikation gegeben werden.

Ermitteln der Folgepuffermengen: Diese Mengen beinhalten für jeden Puffer des gesamten Netzwerks die Folgepuffer. Das vorderste Flit des Puffers kann im nächsten Schritt gemäss dem Routingverfahren in einen dieser Folgepuffer weitervermittelt werden.

Ermitteln von Abhängigkeiten: Sind alle Puffer einer Folgepuffermenge voll, so ist der Puffer von seinen Folgepuffern abhängig und kann nur dann senden, wenn auch ein Nachfolger sendet. Ein solcher Puffer wird als blockiert vorgemerkt.

Aufstellen der Abhängigkeitsbäume: Wenn Folgepuffer eines blockierten Puffers ebenfalls blockiert sind, wird dies graphisch dargestellt, indem jedem blockierten

Irrwegpuffer ein Abhängigkeitsbaum zugeordnet wird.

Analyse der Abhängigkeitsbäume: Anhand der aufgestellten Baumstrukturen können Paketverklemmungen identifiziert werden. Wenn sich Puffer in einem Abwärtspfad eines Baumes wiederholen, liegt eine Verklemmung vor.

Ermitteln der Folgepuffermengen

Die Folgepuffermenge eines Puffers wird durch sein vorderstes Flit bestimmt. Ist das Flit ein Paket-Kopf, so ist jeder Puffer, zu dem der nächste Router vermitteln darf, ein Folgepuffer. Wenn das Flit hingegen nicht ein Paket-Kopf ist, dann sind nur Puffer, zu denen der Kopf des Pakets schon hin vermittelt wurde, Folgepuffer. Abbildung 3.6 soll diese Zuordnungen illustrieren. Formal wird die Bildung der Folgepuffermenge für einen Puffer gemäss Regel 2 beschrieben:

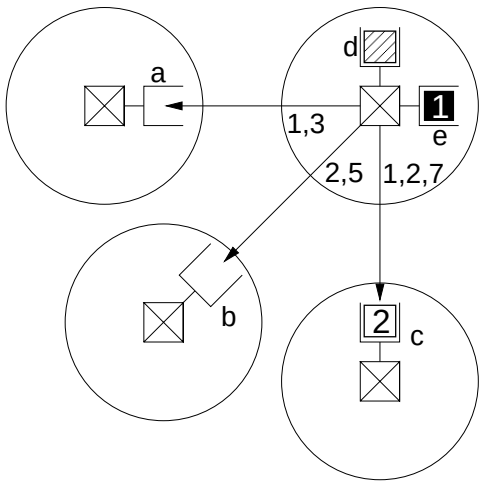


Abbildung 3.6: Netz-Situation: Puffernamen in Kleinbuchstaben, Nummern beschreiben Ziele. Das schraffierte Flit ist ein Datenflit des Pakets mit Ziel 2. Puffer d hat deshalb als Folgepuffer nur Puffer c. Puffer e hat als Folgepuffer c und a, da das Flit das erste eines Pakets ist.

Regel 2 *Sei M_k die Ausgangsmenge des Routers für den Ausgang k und a_k der Puffer, zu dem der Ausgang führt. Ein Puffer b des Routers hat die Folgepuffermenge B. Puffer*

b's vorderstes Flit hat die Ziele M_x.

$$\textit{Kopf-Flit} \;:\; a_k \in B \Leftrightarrow M_x \cap M_k \neq \varnothing \tag{3.4}$$

$$\textit{sonst} \;:\; B = \{a_p\};\ a_p \textit{ enthält einen vorderen Paketteil} \tag{3.5}$$

Ermitteln von Abhängigkeiten, Deadlock-Definition

Wenn alle Puffer einer Folgepuffermenge voll sind, dann ist der Puffer, zu dem die Folgepuffermenge gehört, blockiert und ist von seinen Nachfolgern abhängig. Er muss warten mit Senden, bis einer der Nachfolger nicht mehr voll ist. Blockierte Puffer sind für eine Analyse von eventuellen Verklemmungen vorzumerken.

An dieser Stelle ist es möglich und auch notwendig, eine wohldefinierte Art von Paketverklemmungen als *Deadlock* zu bezeichnen. Dazu muss eine Zusatzüberlegung angestellt werden. Ein blockierter Puffer a, sei es ein Irrwegpuffer oder nicht, habe eine Anzahl von Nachfolgern $b_1 \ldots b_n$, von denen er abhängig ist. Ist einer von ihnen (b_i) nicht blockiert, so kann dieser senden, woraufhin auch a senden kann. Dies gilt auch für alle Nachfolger, wenn sie alle blockiert sind: Hat einer (b_j) einen Nachfolger (c_i), der nicht blockiert ist, dann kann c_i senden sowie auch b_j nach ihm, und so auch a. Finden wir also in einer noch so langen Kette von Nachfolgern einen Puffer, der nicht blockiert ist, dann kann letztlich auch a senden, und es handelt sich bei seiner Situation nicht um eine Verklemmung. Eine solche kann nur dann auftreten, wenn sich in den Ketten der Nachfolger Wiederholungen ergeben, beispielsweise wenn a nur nach b senden kann, b aber auch nur nach a. Ketten von Nachfolgern sollen nach der ersten Wiederholung nicht weiterverfolgt werden, so dass die Kette in diesem Fall mit $\{a, b, a\}$ beschrieben ist. Wenn aber in jeder Kette von Nachfolgern, die von einem gewissen Puffer ausgeht, Wiederholungen auftreten, dann liegt ein Deadlock vor.

Definition 12 *Ein* Deadlock *wird als die Situation definiert, in der ein blockierter Puffer in allen seinen Nachfolgeketten Wiederholungen aufweist.*

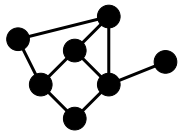
Wenn im Folgenden von mehreren Typen von Abhängigkeitsbäumen gesprochen wird, so sind dies die verschiedenen Perspektiven (der beteiligten Puffer) auf die Deadlocks im Netzwerk. Eine Klassifizierung in verschiedene Deadlock-Typen macht aber keinen Sinn. Die Definition schliesst alle möglichen gegenseitigen Abhängigkeiten von Puffern mit ein, und da eine Verklemmung ohne eine gegenseitige Abhängigkeit nicht denkbar ist, gibt es keine Paketverklemmungen, die nicht Deadlocks sind.

Aufstellen der Abhängigkeitsbäume

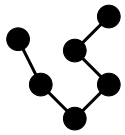
Die Gesamtstruktur der Abhängigkeiten, die mit einem Puffer zusammenhängen, ist das Objekt der Untersuchung. Sie ist verantwortlich dafür, dass Deadlocks entstehen. Wenn sie analysiert wird, können die auftretenden Deadlocks identifiziert werden. Es wurde gesagt, dass nur dann eine Paketverklemmung auftritt, wenn mindestens ein Paket sich verirrt hat. Das bedeutet, dass bei jedem Deadlock mindestens ein Irrwegpuffer involviert ist. Deshalb macht es Sinn, die Untersuchung auf die Irrwegpuffer zu beschränken, obwohl Abhängigkeitsbaüme auch für andere blockierte Puffer aufgestellt werden können. Nicht blockierte Puffer sind hingegen auch nicht von anderen abhängig und können deshalb nicht mit einem Abhängigkeitsbaum in Verbindung gebracht werden. Der Grund dafür ist, dass ihr vorderstes Flit an einen Folgepuffer, der noch freien Platz hat, weitervermittelt werden kann.

Eine formale Definition von Abhängigkeitsbäumen stützt sich auf die Graphentheorie, einem Spezialgebiet der Mathematik (entnommen aus [14]). Ein zusammenhängender Graph G besteht aus Knoten und Verbindungen (auch Kanten genannt), so dass jeder Knoten über einen Pfad jeden anderen Knoten erreichen kann. Ein *Pfad* eines Graphen ist eine Teilmenge von Verbindungen und Knoten des Graphen, also auch wieder ein Graph. Ein Pfad der Länge n hat genau zwei Knoten mit nur einer Verbindung und $n-2$ Knoten mit genau zwei Verbindungen. Man nennt einen Graphen, dessen Elemente Teilmengen der Elemente eines anderen Graphen G sind, einen Teilgraphen von G. Ein *Zyklus* der Grösse n ist ein Teilgraph von G, der genau n Verbindungen enthält und n Knoten, die je genau zwei Verbindungen haben. *Bäume* sind Graphen, die keine Zyklen als Teilbäume haben. Aus dieser Definition wird klar, dass jeder Pfad auch ein Baum ist. Die besprochenen Abhängigkeitsbäume, die auch [8] kennt, gehören einer speziellen Art an, den *Bäumen mit Wurzel*. Ein Baum mit Wurzel hat einen Knoten als Wurzel gekennzeichnet. Die Knoten eines Baumes mit Wurzel haben verschiedene Bezeichnungen. Er enthält die Wurzel, die nur durch die Kennzeichnung erkannt wird, Astgabeln, die mehr als zwei Verbindungen haben, Äste, die genau zwei Verbindungen haben und Blätter mit nur einer Verbindung. Als *Baum-Pfade* sollen nur diejenigen Pfade eines Baumes mit Wurzel bezeichnet werden, die die Wurzel als einen der beiden Endknoten enthalten. Die Abbildungen 3.7 und 3.8 geben Beispiele der definierten Begriffe. Es sind: Graph, Pfad, Zyklus, Baum und Baum mit Wurzel.

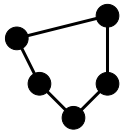
Definition 13 *Ein* Abhängigkeitsbaum *ist ein Baum mit Wurzel, dessen Knoten Puffernamen enthalten. Ein Abhängigkeitsbaum hat einen Bezugspuffer, dessen Name in der Wurzel enthalten ist. Jegliche Pfade des Abhängigkeitsbaums sind Nachfolgeketten des Bezugspuffers.*



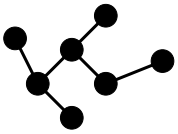
(a) *Graph* mit Knoten und Verbindungen



(b) *Pfad* der Länge sechs mit zwei Endknoten und vier Zwischenknoten



(c) *Zyklus* aus Knoten mit je zwei Verbindungen



(d) *Baum*, definitionsgemäss ohne Zyklen

Abbildung 3.7: Verschiedene Typen von Graphen

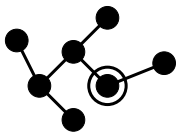
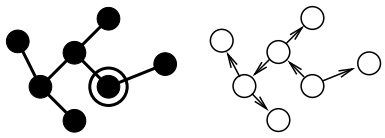


Abbildung 3.8: Ein *Baum mit Wurzel*. Die Wurzel hat keine speziellen Eigenschaften, ausser dass sie als solche markiert ist.

Die Regel 3 gibt an, wie Bäume aufgestellt werden, die dieser Definition entsprechen. Es gilt dabei zu beachten, dass darin Puffernamen mehrmals vorkommen können (höchstens 2 mal), was bedeutet, dass Abhängigkeitsbäume keine Teilbäume der Pufferanordnung im Netzwerk sind. Zur besseren Verständlichkeit werden die Abhängigkeitsbäume etwas anders gezeichnet als die mathematischen Bäume mit Wurzel. Die Wurzel wird nicht markiert, aber Pfeile zeigen jeweils von der Wurzel weg, so dass die Wurzel der Knoten ist, auf den als einzigen kein Pfeil zeigt. Pfeile bedeuten die Sende-Abhängigkeit der Puffer. Abbildung 3.9 zeigt den Vergleich dieser beiden Darstellungen.

Regel 3 *Der Abhängigkeitsbaum eines blockierten Irrwegpuffers wird unter Beachtung der folgenden Punkte aufgestellt:*

1. *Die Wurzel enthält den Puffernamen des betreffenden Puffers.*
2. *Jeder Knoten hat Nachfolger, wenn der entsprechende Puffer des Netzes blockiert ist.*
3. *Die Nachfolger eines Knotens sind die Folgepuffer B_i.*
4. *Tritt auf einem Pfad von der Wurzel bis zu einem Blatt ein Puffername mehrmals auf, so wird diesem Blatt kein weiterer Nachfolger hinzugefügt.*



(a) (b)

Abbildung 3.9: Ein Baum mit Wurzel (a), und derselbe Baum als Abhängigkeitsbaum mit Pfeilen dargestellt (b), hier ohne Puffernamen in den Knoten.

Abbildung 3.10 führt die Abhängigkeitsbäume des schon auf Seite 34 gezeigten Deadlocks für beide Irrwegpuffer auf. Bei diesem kleinen Netzwerk handelt es sich um simple Strukturen. Es wurde schon betont, dass nur blockierte Puffer Nachfolger im Baum haben. Hier ist jeder volle Puffer blockiert, so dass im Baum jeder Knoten seinen anderen

Puffer als Nachfolger hat. Man beginnt bei einem Puffernamen als Wurzel des Baumes, setzt ihm den anderen Puffernamen als Nachfolger hinzu, und so weiter. Die erste Wiederholung eines Puffers bildet das Ende einer Kette.

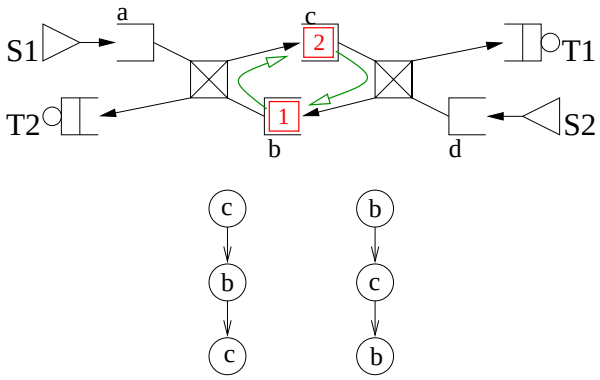


Abbildung 3.10: Bildung von einfachen Abhängigkeitsbäumen

3.2.3 Klassifizierung von Abhängigkeitsbäumen

Mit der Aufstellung der Abhängigkeitsbäume ist die Identifikation der Netzsituation abgeschlossen. Insbesondere sind so alle Deadlocks identifiziert. In den Bäumen steckt nun die gesamte Information, die zur Klassifizierung der Bäume und der schlussendlichen Auflösung der Deadlocks benötigt wird.

Formale Typen-Definition

Bevor einige von ihnen dargestellt werden, sollen Pfadeigenschaften der Abhängigkeitsbäume definiert werden, die es ermöglichen, einem Baum in eine von vier Klassen einzuteilen.

Definition 14 *Ein Pfad wird* passierbar *genannt, wenn die Wurzel und sein Blatt den gleichen Puffernamen haben, aber im Pfad der Name sonst nicht vorkommt.*

Definition 15 *Ein Pfad wird* problematisch *genannt, wenn dessen Blatt den gleichen Puffernamen hat wie ein Zwischenknoten, der aber nicht die Wurzel ist.*

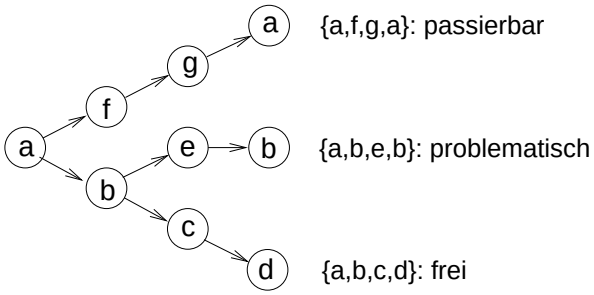


Abbildung 3.11: Illustration zu den Baum-Eigenschaftsdefintionen: Dieser Baum hat je einen passierbaren, einen problematischen und einen freien Pfad.

Definition 16 *Ist ein Pfad weder problematisch noch passierbar, so ist er* frei.

Die Abbildung 3.11 stellt die Definitionen graphisch dar. Die drei Definitionen reichen aus, um alle benötigten Baum-Klassifizierungen vorzunehmen. Es sind deren vier:

Definition 17 *Die vier* Klassen*, in die Abhängigkeitsbäume eingeteilt werden, sind aufgrund ihrer Pfade definiert.*

Normale Bäume *haben mindestens einen freien Pfad.*

Lösbare Bäume *bestehen ausschliesslich aus passierbaren Pfaden.*

Unlösbare Bäume *enthalten nur problematische Pfade.*

Teilbare Bäume *haben sowohl passierbare als auch problematische Pfade, aber keinen freien Pfad.*

Deadlocks äussern sich gemäss Definition 12 dadurch, dass nicht normale Abhängigkeitsbäume auftreten. Hier sind die Bäume nun noch in drei Klassen eingeteilt: in die lösbaren, die unlösbaren und die teilbaren. Ein Abhängigkeitsbaum repräsentiert die Netzsituation aus der Sicht eines bestimmten Puffers. Von einem Puffer her kann ein Deadlock also lösbar sein und gleichzeitig von einem anderen her nicht. Wieder sei betont, dass nicht die Deadlocks klassifiziert werden, sondern nur die zu Puffern gehörigen Abhängigkeitsbäume. Jene, die teilbar sind, sind eine Mischform von lösbaren und unlösbaren. Dementsprechend muss bei der Auflösung darauf geachtet werden, dass nur der lösbare Anteil als Vermittlungsvorschrift in Betracht gezogen wird. Bei den lösbaren Bäumen hingegen führt jeder gegangene Pfad als Abfolge von Vermittlungen zu einer korrekten Weitervermittlung der betreffenden Flits.

Beispiele von Baum-Bildungen in Netzwerk-Situationen

Bevor die Auflösung im nächsten Unterkapitel detailierter besprochen wird, soll die Aufstellung von Abhangigkeitsbäumen anhand einer Reihe von Beispielen (Abbildungen 3.12 bis 3.15) demonstriert werden. Was ein Abhängigkeitsbaum repräsentiert und was nach seiner Auflösung unter der lokal verklemmungsfreien Situation zu verstehen ist, soll zusammenfassend am Schluss dieses Unterkapitels rekapituliert werden.

Abbildung 3.12 zeigt dieselbe Situation wie Abbildung 3.10, die schon im Unterkapitel 3.2.2 besprochen wurde. Nach der Diskussion der Klassifizierung von Abhängigkeitsbäumen können die Bäume nun in eine bestimmte Klasse eingeteilt werden. Sowohl der Baum von b als auch derjenige von c hat nur einen Pfad. Dieser Pfad ist jeweils passierbar, also bestehen die Bäume ausschliesslich aus passierbaren Pfaden und gehören laut der Definition 17 der Klasse von lösbaren Bäumen an. Es wird klar, weshalb diese Bezeichnung gewählt wurde: Wenn ein Pfad als eine Abfolge von Vermittlungen angewendet wird, so bewegen sich die Pakete jeweils einen Schritt weiter, obwohl die Situation für das normale Routingverfahren eine Paketverklemmung darstellt.

Dieselbe Situation zeigt Abbildung 3.13, mit dem Unterschied, dass nun in Puffer d ein Paket liegt, während d in Abbildung 3.12 leer war. Dieser Puffer ist nicht ein Irrwegpuffer, denn er erfüllt die Regel 1 auf Seite 33 nicht. Trotzdem ist er von der Verklemmung mitbetroffen, denn er kann nicht weitervermittelt werden. Der gezeigte Abhängigkeitsbaum von Puffer d hat nur einen Pfad, der problematisch ist. Bäume mit ausschliesslich problematischen Pfaden werden gemäss der Definition 17 „unlösbar“ genannt. Das Fehlen eines Pfades, der als mögliche Vermittlungsabfolge verwendet werden kann, erklärt die Bezeichnung dieses Baum-Typs: Die Paketverklemmung seines Bezugspuffers ist für das normale Routingverfahren unbeweglich und auch mit dem Analyseverfahren ist keine Bewegung der Verklemmung möglich.

Die Situation von Abbildung 3.14 ist aus der Sicht von Puffer 5 kein Deadlock, denn sein Abhängigkeitsbaum ist vom Typ „normal“, weil er einen freien Pfad enthält. Der Baum kam zustande, indem als Erstes die Wurzel mit dem Namen 5 gebildet wurde und dann seine weiteren Knoten hinzugefügt wurden. Der Puffer 5 hat nur volle Nachfolger, ist also blockiert. Deshalb hat der Baum seine Nachfolgepuffer als mit sich verbundene Knoten. Puffer 11 und 9 sind beide auch blockiert, deshalb gilt dasselbe für diese Knoten. 9 hat 5 als Nachfolger und wird als Blatt ohne Nachfolger im Baum an das Blatt 9 angehängt. Der Puffer 11 hat mit 8 einen unblockierten Nachfolger, der ungehindert an Puffer 10 senden kann. Entsprechend wird ein Blatt mit Beschriftung 8 an den Knoten 11 des Baumes angefügt. Es sei bemerkt, dass „blockiert“ in diesem Zusammenhang nicht mit „verklemmt“ verwechselt werden darf. Jeder Puffer wird als blockiert bezeichnet, der nur volle Nachfolgepuffer besitzt. Nachfolgepuffer sind solche,

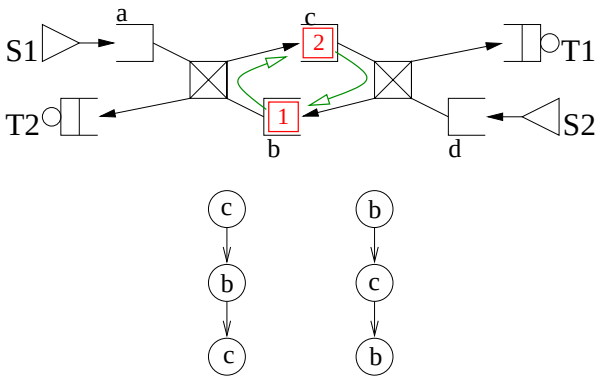


Abbildung 3.12: Dieses schon verwendete Beispiel zeigt nach der nun definierten Klassifizierung zwei *lösbare* Bäume.

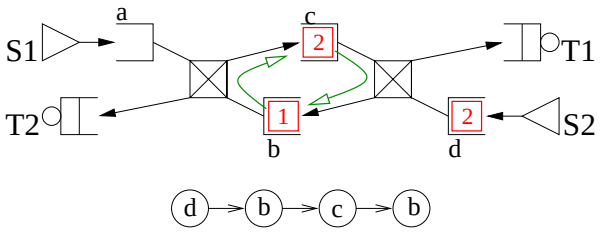


Abbildung 3.13: Ein Beispiel, in dem ein *unlösbarer* Abhängigkeitsbaum auftritt.

zu denen er sein vorderstes Flit senden kann. Als verklemmt wird nur derjenige Anteil der blockierten Puffer bezeichnet, die von einer Deadlock-Situation betroffen sind und mit dem normalen Routingverfahren nicht weitervermittelt werden können.

Ein zusätzliches Paket in Puffer 10 im Netz von Abbildung 3.14 macht die Situation komplizierter. Sie ist in Abbildung 3.15 gezeigt. Es entstand eine Situation nach der Rekonfiguration, die für Puffer 5 einen teilbaren Abhängigkeitsbaum erzeugen lässt.

Informationsgehalt der Abhängigkeitsbäume

Es ist falsch, einen Baum-Typ dem Charakter des vorliegenden Deadlocks gleich zu setzen. Dafür gibt es zwei Gründe: Einerseits gilt die Definition 12, die Deadlocks genau beschreibt. In dieser Definition kommt keine Klassifizierung in mehrere Deadlock-Typen vor. Das bedeutet, dass es nur eine Art von Deadlocks gibt. Andererseits sind Abhängigkeitsbäume subjektiv für den Puffer, auf den er bezogen ist. Es sind Netzsituationen denkbar, bei denen verschiedene Baum-Typen bei Puffern vorkommen, die von demselben Deadlock betroffen sind. Ein Beispiel zeigt Abbildung 3.13, deren Puffer b und c die gleichen Abhängigkeitsbäume wie in Abbildung 3.12 haben. Also treten in der Situation von Abbildung 3.13 zwei verschiedene Baum-Typen auf.

Wenn ein passierbarer Pfad eines Baumes als Vermittlungsabfolge auf seinen Bezugspuffer angewendet wird, so entsteht aus der Sicht dieses Puffers entweder eine verklemmungsfreie Situation oder ein neuer Deadlock, nachdem der bestehende gelöst wurde. Deshalb werden beide Fälle als Auflösung bezeichnet. Wenn kein neuer Deadlock entsteht, ist von einer lokal verklemmungsfreien Situation die Rede. Diese Beschreibung erklärt sich dadurch, dass dies nur aus der Sicht des Bezugspuffers gesagt wird. Aus der Sicht eines anderen Puffers entstand ein neuer Deadlock, wenn sein neuer Abhängigkeitsbaum nicht normal ist.

3.2.4 Deadlock-Auflösung

Eine Kurzzusammenfassung dessen, was die besprochene Analyse nun vermag, macht ihren Nutzen klar. Stellt man die Abhängigkeitsbäume der Irrwegpuffer auf und klassifiziert sie, so kann man einerseits die Puffer, die von einem Deadlock betroffen sind, von den unbetroffenen unterscheiden, und andererseits ist es möglich, anhand der Klassifizierung des Baumes zu entscheiden, ob die Situation von diesem Puffer her ganz, nur teilweise oder gar nicht lösbar ist. Wie dies geschieht, beantwortet nun die vorgestellte Auflösungsmethode. Zuerst muss gesagt werden, welche Puffer an einem Deadlock beteiligt sind und welche nicht. Nicht alle betroffenen Puffer sind beteiligt. Die Beteiligung eines Puffers wird dadurch bestimmt, dass er in der Ring-Abhängigkeit eines

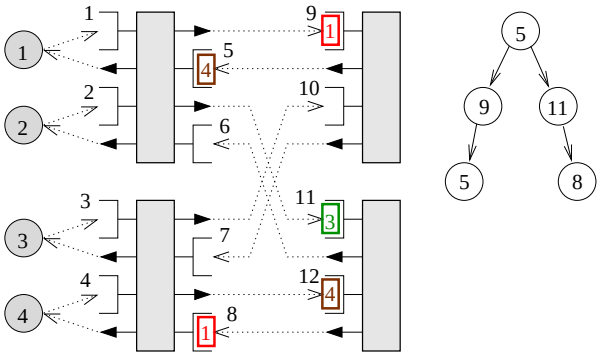


Abbildung 3.14: Eine Situation mit einem *normalen* Abhängigkeitsbaum des Puffers 5. Aus seiner Sicht entstand hier kein Deadlock. Diese Netzwerkkonfiguration entstand, indem die Endknoten 1 und 4 vertauscht wurden.

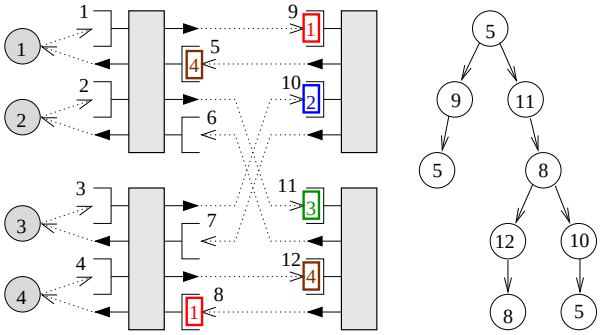


Abbildung 3.15: Deadlock und dazugehöriger *teilbarer* Abhängigkeitsbaum von Puffer 5.

Deadlocks vorkommt. Diese Ring-Abhängigkeit wird durch die passierbaren Pfade eines nicht normalen Baumes repräsentiert.

Definition 18 *Ein Puffer ist an einem vorhandenen Deadlock einer Netzsituation* beteiligt*, wenn er in einem passierbaren Pfad eines Abhängigkeitsbaumes vorkommt.*

Es wurde festgestellt, dass passierbare Pfade eine unkritische gegenseitige Vermittlung von beteiligten Puffern darstellen. Das heisst, dass die Abfolge der Vermittlung, wie sie ein passierbarer Pfad beschreibt, möglich ist, auch wenn alle Puffer blockiert sind. Ein Ignorieren der Blockierung führt hier also zu einer Auflösung. Es sei bemerkt, dass bei lösbaren Bäumen jeder der Baum-Pfade verwendet werden darf und somit bei dieser Auflösung keine Einschränkung des Vermittlungsverfahrens nötig ist. Bei teilbaren Bäumen hingegen dürfen nur die passierbaren Pfade verwendet werden, und bei unlösbaren Bäumen gar keine. Puffer, die unlösbare Abhängigkeitsbäume haben, bleiben einen Netz-Zyklus lang inaktiv und können frühstens im nächsten wieder senden.

Regel 4 *Bei einer Deadlock-Auflösung dürfen nur passierbare Pfade verwendet werden.*

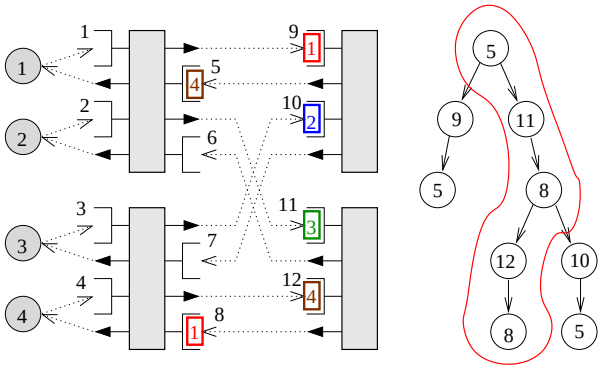


Abbildung 3.16: Illustration Regel 4: Gleicher Deadlock und dazugehöriger teilbarer Abhängigkeitsbaum wie auf Seite 45. Werden die Vermittlungen des umrahmten komplexen Pfades ausgeführt, so empfängt Knoten 8 zwei Mal. Dies ist nicht zulässig.

Die Abbildung 3.16 macht deutlich, weshalb Regel 4 eingehalten werden muss. Die Analyse hat gezeigt, wo beim Versenden Verklemmungen auftreten, und diese sind nun ausgeschlossen durch die Regel 4. Deshalb kann ein *bedingungsloses Empfangen* von Flits aus beteiligten Puffern die Auflösung herbeiführen.

Definition 19 *Durch das* bedingungslose Empfangen *von einem Puffer werden ihm Flits auch dann gesendet, wenn er voll ist.*

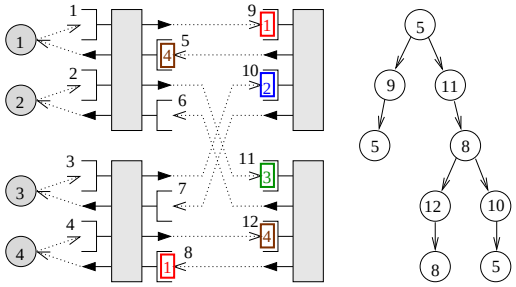
Dies führt temporär bei dem Empfänger zu einer offensichtlichen Überlastung. Da aber auch sein Nachfolger bedingungslos empfängt, weil er am Deadlock beteiligt ist, wird die Last weitergegeben, bis sie schliesslich dort aufgenommen wird, wo sie begonnen hat: beim ersten Puffer, der gesendet hat, denn es wurde ein passierbarer Pfad zur Vermittlung verwendet. Ein Deadlock-Auflösungsbeispiel soll die Funktionsweise einer solchen Vermittlung verständlich machen (Abbildung 3.17).

3.2.5 Diskussion des Verfahrens

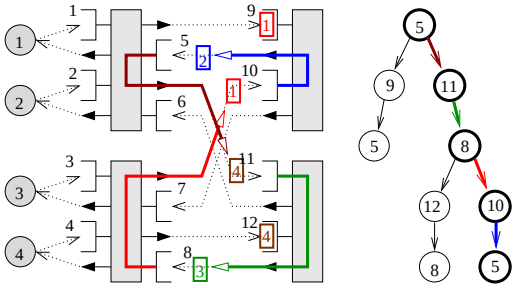
In der Praxis erweist sich das vorgestellte Verfahren als funktionstüchtig. Es funktioniert, weil es eine Netzwerksituation darauf untersucht, welche Puffer an einer Deadlock-Situation beteiligt sind, wobei Puffer mit unlösbaren Bäumen als nicht beteiligt angesehen werden. Zudem gibt sie an, wie eine solche Situation aufgelöst wird. In einem nächsten Netzzyklus können (beispielsweise durch zuerst unlösbare Anteile) weitere Deadlocks entstehen, die wiederum durch Analyse und Auflösung behandelt werden. Die Theorie weist nicht darauf hin, dass Deadlocks entstehen können, die immer einen neuen Deadlock im nächsten Zyklus erzeugen. Es ist anzunehmen, dass es solche Situationen nicht gibt, weil jedes Paket immer nur endlich weit von seinem Ziel entfernt ist, also nach einer endlichen Anzahl von Auflösungen nicht mehr in einem Deadlock beteiligt ist. Da Deadlocks nur durch beteiligte Irrwegpakete auftreten, muss das Netz nach einer endlichen Zeit unkritisch werden. Folglich wird die Bedingung 1 und ihre allgemeine Form unabhängig von der Paketverteilung (Bedingung 2) nicht nur als notwendig, sondern auch als hinreichend betrachtet. Diese Schlussfolgerung ist in Satz 1 zusammengefasst.

Satz 1 *Wird ein Netzwerk gemäss der Bedingung 1 rekonfiguriert, so können in der neuen Konfiguration auftretende Verklemmungen immer aufgelöst werden und es entstehen infolge der Rekonfiguration keine Paketverluste, wenn Regel 4 eingehalten wird.*

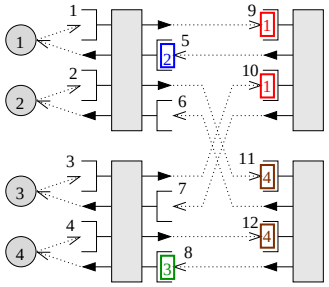
Es ist zu beachten, dass dieser Satz für gewisse Spezialfälle der dynamischen Rekonfiguration nicht gilt. Es sind dies Netzwerke mit Wormhole-Switchingverfahren und solche, die Multicasts verwenden. Ein letztes Unterkapitel muss abschliessend betonen, dass dynamische Rekonfigurationen von solchen Netzen grundsätzlich nicht möglich sind. Für die anderen in dieser Arbeit verwendeten Switchingverfahren SAF, PCT und VCT und für jegliche Unicast-Sendungen ist der Satz gültig, wenn für die Netztopologie ein



(a) Situation 1: Deadlock aus Abbildung 3.15



(b) Übergang: Auflösungs-Abfolge gemäss dem passierbaren Pfad



(c) Situation 2: Verklemmungsfreie Situation im nächsten Zyklus

Abbildung 3.17: Schrittweises Auflösen eines Deadlocks.

Routingverfahren gewählt wurde, das ohne dynamische Rekonfiguration deadlockfrei ist. Dass es von dieser Regel auch Ausnahmen gibt, zeigte das Beispiel eines 2D-Mesh mit SP-Routing (Abbildung 2.7 auf Seite 22). Es ist zu erkennen, das der aufgetretene Deadlock die gleiche Form hat wie diejenigen, die in diesem Kapitel besprochen wurden. Daraus ergibt sich die Möglichkeit, auch die Deadlocks, die ohne Rekonfiguration entstehen, mit dem angegebenen Auflösungs-Algorithmus zu beheben. Das Ziel, bidirektionale Netze beliebiger (auch irregulärer bidirektionaler) Topologien verlustfrei dynamisch rekonfigurieren zu können, ist erreichbar, wenn SP-Routing angewendet wird und in jedem Zyklus auftretende Deadlocks aufgelöst werden. Dabei darf die Suche nach Deadlocks nicht nur auf die Irrwegpuffer beschränkt werden. Damit wurde ein deadlockfreies Routingverfahren für beliebige Topologien definiert und dynamische Rekonfigurationen können für beliebige bidirektionale Netze verlustfrei durchgeführt werden. Im Rahmen dieser Master-Arbeit hat die Zeit nicht ausgereicht, es zu implementieren. Folglich können mit dem implementierten Algorithmus Netze nur dann dynamisch rekonfiguriert werden, wenn für deren Topologie ein deadlockfreies Routingverfahren verwendet wird.

3.3 Grenzen der dynamischen Rekonfiguration

Grundsätzlich unlösbar sind Deadlocks, die mit Wormhole-Switching entstehen können. Die Möglichkeit eines Multicast wurde bisher noch nicht diskutiert, doch erweist sich ebenso als verhängnisvoll. Deadlock-Situationen, die mit Multicast-Paketen entstehen, können unlösbar sein.

3.3.1 Wormhole Switching

In Abbildung 3.18 sind zwei Deadlock-Situationen gezeigt, die mit Wormhole-Switching entstanden und auch mit einer Analyse nicht auflösbar sind. Der Grund ist, dass sich bei Wormhole-Switching die Pakete bei einer Blockierung nicht in einem Puffer sammeln können und dadurch Schwänze von Paketen die Ausgänge von Switches blockieren. In beiden gezeigten Fällen sind alle Pakete auf Irrwegen und müssen Router-Ausgänge wählen, die in die Richtung führen, von der die Pakete herkommen. Die Pakete müssen umkehren, da ihre Ziele (als Nummern in den Paketköpfen) ihren Platz gewechselt haben. In beiden Fällen ist jeder Ausgang in diese Richtung schon von einem Schwanz eines Pakets blockiert und kein Paket kann sich weiter bewegen. Die grundsätzliche Unlösbarkeit solcher Deadlocks verbietet das Wormhole-Verfahren für dynamische Rekonfiguration.

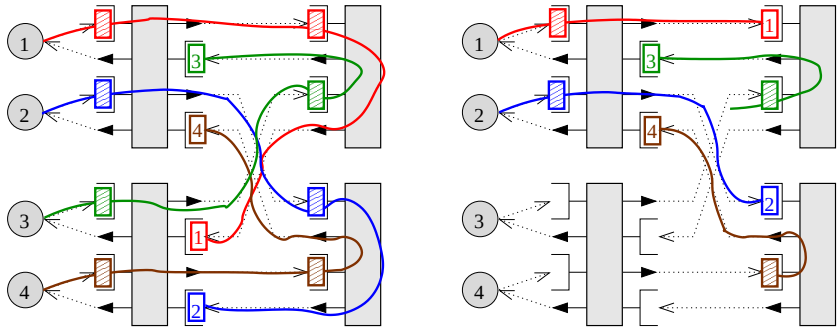


Abbildung 3.18: Deadlocks mit Wormhole-Switching: Die Paketlänge ist grösser als 1 und die hinteren Paketteile blockieren sich gegenseitig. Ohne dass sich diese in die Puffer der Kopf-Flits sammeln dürfen, wie es in PCT der Fall ist, besteht keine Möglichkeit, diesen Deadlock aufzulösen.

3.3.2 Multicast

Die Abbildung 3.19 zeigt eine typische Multicast-Deadlock-Situation. In Tabelle 3.1 sind die entsprechenden Gegenabhängigkeiten der Puffer gezeigt. Ein Versuch, die Abhängigkeiten graphisch darzustellen, zeigt Abbildung 3.20. Es handelt sich dabei nicht um einen Baum, der der graphentheoretischen Definition genügt. Die Grafik zeigt dennoch die Komplexität eines solchen Deadlocks auf. Jeder „Pfad“ löst mehrere andere mit aus. Das bedeutet, dass die Paketmenge im Netzwerk steigt. Da aber das Netzwerk schon an der Grenze seiner Paket-Aufnahmekapazität ist, würde jedes bedingungslose Empfangen den betreffenden Puffer überfüllen. Aufgrund dieses Problems wird Multicast bei dynamischen Rekonfigurationen verboten.

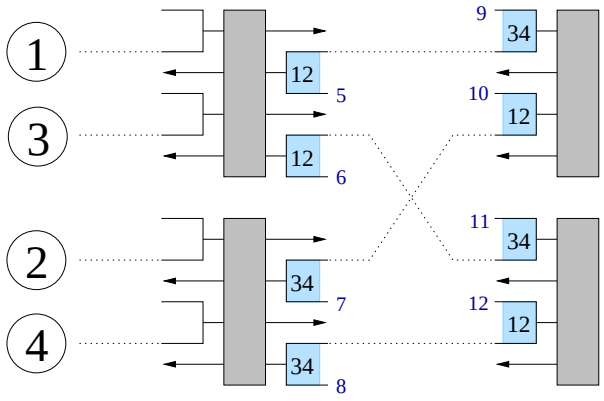


Abbildung 3.19: Deadlock, der von Multicast-Paketen erzeugt wurde. Jedes Paket hat zwei Ziele.

Puffer	Abhängigkeiten
5	9,11
6	9,11
7	10,12
8	10,12
9	{5,7}
10	{5,7}
11	{6,8}
12	{6,8}

Tabelle 3.1: Abhängigkeitstabelle des Multicast-Deadlocks: Manche Abhängigkeiten der Puffer sind als Mengen von Puffern dargestellt. Dies bedeutet, dass zu allen Puffern in der Menge gleichzeitig gesendet wird.

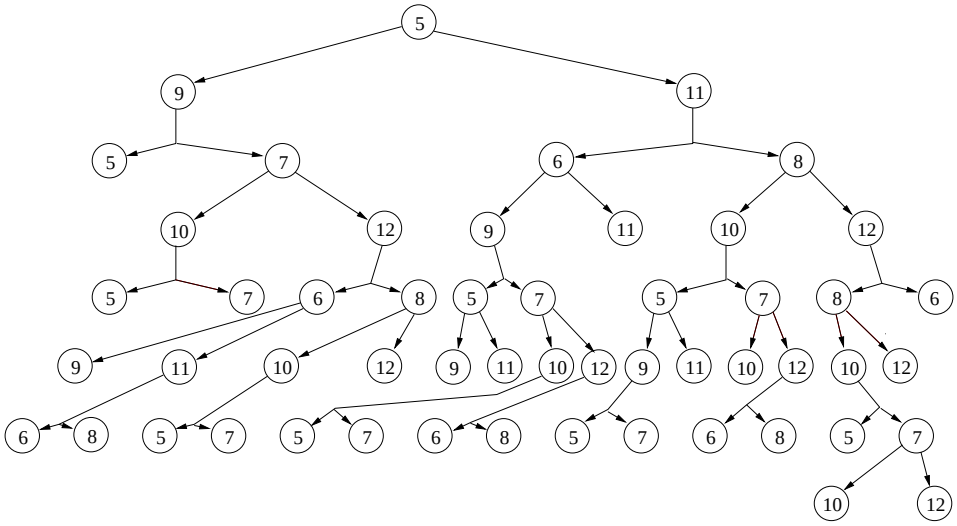


Abbildung 3.20: Abhängigkeits-„Baum“ des Multicast-Deadlocks: Es ist derjenige von Puffer 5, der von der Abhängigkeitstabelle 3.1 abgeleitet ist. Er lässt sich mit den gegebenen Kriterien nicht klassifizieren, denn es treten Verzweigungen auf, die Gleichzeitigkeit ausdrücken. Es gibt keine Sendeabfolge, die für keinen Puffer ein zweimaliges Empfangen bedeutet.

Kapitel 4

Implementierung

4.1 Einleitung

Der bisherige Stand der Entwicklung von CINSim legt [2], [3], [4] und zuletzt die Diplomarbeit des Vorgängers Matthias Kühm [15] fest. Auf diese Arbeiten aufbauend, werden die vorgestellten Analyseverfahren in das Simulationswerkzeug CINSim implementiert, wobei verschiedene Varianten denkbar sind. Die im Kapitel 2 beschriebenen Verfahren der Flusskontrolle LBP und GBP stellen zwei grundsätzlich unterschiedliche Herausforderungen an die Implementierung dar. Während bei LBP die Reihenfolge der Puffer-Sendungen zufällig ausgewählt werden, geschieht dies bei GBP streng regelmässig. Der Grund ist leicht einzusehen: Wenn bei GBP in einer Kette von hinter einander liegenden Puffern jeder senden soll, doch keine Sendereihenfolge eingehalten wird, so gibt es zeitweise überfüllte Puffer. In LBP kann dies nicht geschehen, denn es darf nur senden, wer einen nicht-vollen Puffer vor sich hat.

Es liegt nahe, mit einer LBP-Implementierung zu beginnen. Obwohl zuerst für möglich gehalten wurde, dass bei dieser Implementierung keine Deadlocks entstehen können, ist dem nicht so. Deshalb wird in dieser einfacheren Implementierung, die im zweiten Unterkapitel beschrieben wird, die Deadlock-Handhabung schon hinzugefügt neben verschiedenen kleinen Erweiterungen der statischen Rekonfiguration. Die GBP-Implementierung bringt eine weitere Herausforderung mit sich. Im Normalfall wird die erwähnte Kette von Puffern von den Zielen her ermittelt. Ein Ziel empfängt ein Flit von einem Puffer, der dann empfangsbereit wird, auch wenn er zuvor voll gewesen ist. Im gleichen Schritt werden alle leeren Puffer, die zum selben Router gehören, empfangsbereit. Diese empfangen wiederum ein Flit von einem anderen Puffer, und so weiter. Im Deadlock-Fall kann dieses Verfahren nicht mehr funktionieren, weil an Deadlocks keine

Ziele beteiligt sind. Es muss also eine grundsätzlich neue Reihenfolge der Sendungen ermittelt werden. Diese Aufgabe wird in einem weiteren Unterkapitel gelöst, das die GBP-Implementierung behandelt. Rekonfigurationen nicht nur mit SAF, sondern auch mit PCT simulierbar zu machen, ist das Thema des letzten Unterkapitels.

4.1.1 Vorbemerkungen

Es ist wichtig, sich bewusst zu sein, dass von der Theorie zur Simulation ein Paradigmenwechsel stattfindet (Paradigma = Denkmuster). Theorie, Simulation und Praxis sind drei verschiedene Forschungsbereiche, die in einander übergreifen. Was in der Theorie auf eine gewisse Art und Weise beschrieben wurde, kann bei der Simulation etwas anders durchgeführt werden, und in der Praxis geschieht wiederum etwas anderes. Wichtig ist, die Kernpunkte der Theorie auf die Simulation und die Praxis zu übertragen. Was sich bei der Simulation einer dynamischen Rekonfiguration ändert, ist die Sichtweise. Während in der Theorie von ein und demselben Netz gesprochen wurde, das eine Änderung erfährt, so sind es bei der Simulation in CINSim zwei verschiedene Netze, die einzeln zu betrachten sind. Die dynamische Rekonfiguration besteht in CINSim nicht darin, ein Netz zu verändern, sondern ein erstes laufendes Netz zu stoppen und ein neues zu starten. Sollen dabei Pakete die Rekonfigurationsphase überdauern, so müssen sie in das neue Netz kopiert werden, und zwar in den Puffer hinein, der dem Puffer im alten Netz entspricht. Dies bedarf einer genauen Definition. Der Paradigmenwechsel bedeutet in CINSim für die Rekonfiguration eine Umkehrung der Sichtweise: In der Theorie wird das Netz verändert und die Pakete bleiben unberührt, wie es der Praxis entspricht, und in der Simulation gibt es zwei Netze, die unverändert bleiben, aber die Pakete werden vom einen Netz ins andere kopiert.

Eine weitere Änderung der Sichtweise ist notwendig: In der Theorie haben die Ziele Namen, und die Pakete haben eine Anzahl von Zielen, die ihnen zugeordnet sind und auf die sie zusteuern. In der Simulation und der Praxis hingegen müssen die Ziele Nummern haben und die Ziele der Pakete als eine Art Adressierung codiert werden. Dasselbe gilt für die Ausgänge der Router. Wenn in der Theorie ein Ausgang eine Menge von zu erreichenden Zielen hatte, so muss nun auch dies codiert werden. Diese Codierung wird wichtig, wenn bei der Rekonfiguration Pakete in das neue Netz kopiert werden. Die neuen Ziele haben nicht unbedingt die gleichen Nummern wie die alten, also muss eine Umadressierung stattfinden. Ob die Bedingung für eine verlustfreie Rekonfiguration erfüllt ist, muss ebenfalls anhand der Adressen der Pakete ermittelt werden. Diese Begriffe, die für alle Implementationen wichtig sind, werden in den folgenden Unterabschnitten zur Sprache kommen.

4.1.2 Adressierungs-Konzept

Zuerst soll das wichtige Konzept der *Bitmasken* eingeführt werden. Jede Menge von Zielen M wird durch eine Binärzahl z_M dargestellt. Die Anzahl Bits ist die Anzahl n der Ziele in einem Netzwerk. Die Ziele des Netzwerks sind von 1 bis n nummeriert und Bit k stellt die Anwesenheit des k-ten Zieles in der Menge M dar. Verschiedene Mengenoperationen entsprechen dabei den Binärzahl-Operationen. Beispielsweise entspricht die Schnittmenge einer bitweisen Und-Funktion von zwei Binärzahlen, und die Vereinigungsmenge entspricht der bitweisen Oder-Funktion.

$$\begin{aligned} z_{M\cap N} &:= z_M \oplus z_N \\ z_{M\cup N} &:= z_M \otimes z_N \end{aligned}$$

Genauso werden auch die mengentheoretischen Wahrheitsfunktionen als binäre Wahrheitsfunktionen interpretiert. $M \subset N$ ist eine Wahrheitsfunktion mit dem Wert 1, wahr, oder 0, falsch. Dieselbe Wahrheitsfunktion stellt $z_M \oplus z_n = z_n$ dar.

$$M \subseteq N \quad \Leftrightarrow \quad z_M \subseteq z_N := (z_M \oplus z_N = z_N)$$

$\oplus$ bezeichnet hier die bitweise Oder-Funktion und $\otimes$ die bitweise Und-Funktion. Zielmengen von Paketen werden nun *Adressen-Masken* genannt und Ausgangsmengen von Puffer-Ausgängen dementsprechend *Ausgangsmasken*. Die Bedingungen und Regeln aus dem Theorie-Kapitel sind also ganz einfach in die Bitmasken-Schreibweise umsetzbar. Der Vollständigkeit halber sollen sie die noch einmal aufgelistet werden.

Bedingung 1 *Die Paketverteilungs-abhängige Rekonfigurationsbedingung lautet*

$$z_{M_x} \subseteq \bigotimes_{k=1}^{\#Ausgänge} z_{M_k}$$

für die Adressen-Maske z_{M_x} des Pakets und die z_{M_k} Ausgangsmasken der Ausgänge des Routers (gemäss der Deklaration auf Seite 28).

Bedingung 2 *Die allgemeine, Paketverteilungs-unabhängige Rekonfigurationsbedingung lautet*

$$z_{M_{y,alt}} \subseteq \bigotimes_{k=1}^{\#Ausgänge} z_{M_{k,neu}}$$

wobei $z_{M_{y,alt}}$ die Ausgangsmaske des alten Vorgängerrouters ist und $z_{M_{k,neu}}$ die Adressen-Masken vom neuen Nachfolge-Router eines Puffers (gemäss der Deklaration auf Seite 29).

Regel 1 *Die Regel, wann sich ein Paket auf einem Irrweg befindet, lautet*

$$z_{M_x} \not\subseteq z_{M_a}$$

z_{M_a} ist die Ausgangsmaske des neuen Vorgänger-Puffers, z_{M_x} die Adressen-Maske des Pakets (gemäss der Deklaration auf Seite 33).

Die Bitmasken und die dazugeörigen Bedingungen und Regeln stellen eine 1:1-Übertragung der theoretisch besprochenen Konzepte dar und sind damit uneingeschränkt auf diese anzuwenden und mit ihnen gleichzusetzen.

4.2 Implementierung von LBP-Flusskontrolle

4.2.1 Übersetzung der Zieladressen der Pakete

Die Anzahl der Ziele muss nicht in beiden Netzen gleich sein und die Zuordnung von alten zu neuen Zielen ist ebenfalls beliebig bei der Rekonfiguration. Wenn Pakete vom alten Netz in das neue übertragen und weitervermittelt werden sollen, so müssen diese beiden Aspekte berücksichtigt werden. Die Tabelle 4.1 zeigt eine mögliche Situation. Dabei werden von fünf bisherigen Zielen die Ziele 1, 3 und 4 entfernt. 2 wird zu 1 und 5 wird zu 3. Das Ziel 2 der zweiten Konfiguration ist neu dazugekommen. Die Frage ist, wie die Adressen-Maske eines Pakets übersetzt werden muss, bevor es in das neue Netz kopiert wird. Dies muss analog zur Definition der Netz-Rekonfiguration aus der Theorie geschehen. Wenn vorher Ziel 2 angepeilt wurde, so muss es im neuen Netz Ziel 1 sein und das früher angepeilte Ziel 5 wird zu Ziel 3. Die übrigen alten Ziele des Pakets werden ignoriert. Das neue Ziel 2 wird nie von einem alten Paket angesteuert.

Die komplette Übersetzungsvorschrift für alle möglichen alten Adressen-Masken kann jeweils mit einer Zusammensetzungs-Formel beschrieben werden. Ist im Beispiel Ziel $t_{2,alt}$ mit $t_{1,neu}$ und $t_{5,alt}$ mit $t_{3,neu}$ äquivalent, so ergibt sich für eine alte Adressen-Maske $z_{M_{alt}} = [b_1, b_2, b_3, b_4, b_5]$ (Bits b_k jeweils 1 oder 0) die neue Maske durch

$$z_{M_{neu}} = [b_2, 0, b_5]$$

Eine solche Übersetzungsvorschrift ist jeweils aus der Zuordnungstabelle ableitbar. Diese lässt sich aus den Modifikationen einer gegebenen Rekonfiguration ermitteln. Wenn

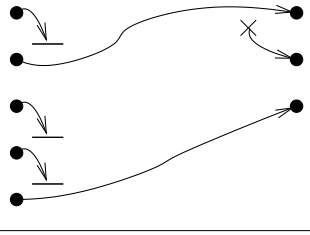
Vor	während Rekonfigurtion	nach
Zielnr.	Zuordnung	Zielnr.
1		1
2		2
3		3
4		
5		

Tabelle 4.1: Zuordnungstabelle von Zielen für eine Rekonfiguration: Einige Ziele wurden entfernt, andere wurden beibehalten und eines wurde neu dazugefügt.

ein Paket nach der Übersetzung eine Adressen-Maske hat, für die Bedingung 1 nicht gilt, so wird das Paket gelöscht. In diesem Fall soll eine Warnung ausgegeben werden. Erfüllt hingegen jedes Paket diese Bedingung, und war nie ein Bit 1, dessen Ziel entfernt wurde, dann war die Übersetzung erfolgreich. Der Spezialfall, dass eine Adressen-Maske nach der Übersetzung nur Nullen enthält, ist durch die Annahme ausgeschlossen, dass jede Maske vorher mindestens einen Eintrag 1 enthielt.

4.2.2 Deadlock-Handhabung

Wie die Deadlock-Handhabung theoretisch funktioniert, ist im vorangehenden Kapitel erläutert worden. Die Umsetzung bei der Simulation soll nun beschrieben werden. Jeder Punkt der behandelten Vermittlungskonflikt-Handhabung in Unterkapitel 3.2 muss simulativ realisiert werden. Die folgenden Unterabschnitte erläutern diese jeweiligen Teillösungen für die Implementation, die sich auf das Verhalten des Netzes nach der Rekonfiguration beschränken. Es wird zudem nicht mehr von allgemeinen Konflikten gesprochen, sondern nur noch von wohldefinierten Deadlocks, da davon ausgegangen wird, dass alle möglichen Vermittlungskonflikte Deadlocks sind.

Deadlock-Identifikation

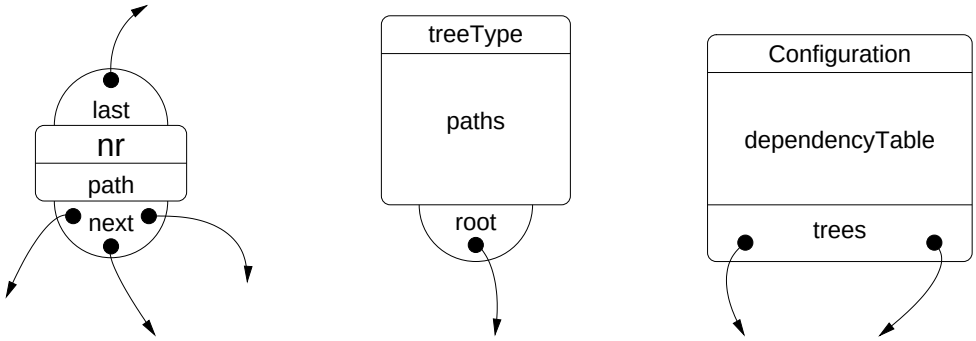
Da das Aufstellen der Abhängigkeitsbäume bei grossen Netzen entsprechend viel Laufzeit in Anspruch nimmt, ist es wichtig, dass die Deadlock-Identifikation so selten wie möglich durchgeführt wird. Die Theorie besagt, dass Deadlocks nur dann auftreten, wenn

mindestens ein Irrweg-Puffer existiert. Die Deadlock-Identifikation muss demnach nur bei Irrweg-Puffern durchgeführt werden, die durch die effizient prüfbare Regel 1 erkannt werden. Wird ein solcher gefunden, so wird das Netz *kritisch* genannt und es werden für alle Irrweg-Puffer die Abhängigkeitsbäume aufgestellt. In der Theorie ist gezeigt, dass dies ausreicht, denn jeder Deadlock, der auftreten kann, muss mindestens einen mitbeteiligten Irrweg-Puffer haben. Durch diese Eingrenzungen wird benötigte Laufzeit für die Deadlock-Identifikation eingespart.

Das Aufstellen der Bäume stellt einen der interessantesten Aspekte der Implementation dar. Ein Objekt der Klasse `DLHandler` verwaltet die Bäume, die als Objekte der Klasse `SwitchingTree` im Speicher abgelegt werden. Ein Baum ist eine verkettete Struktur von `TreeNode`s, die gewisse Informationen tragen. Kurze Ausführungszeit erhielt den Vorzug gegenüber kleinem Speicherbedarf, deshalb sind die Informationen redundant, die jeder Knoten trägt, aber dafür schnell abrufbar. Jeder Knoten trägt folgende Informationen:

- `unsigned int nr`: Die entsprechende Puffernummer
- `vector<TreeNode*> next`: Eine Liste von Zeigern auf Nachfolger
- `TreeNode* last`: Ein Zeiger auf den Vorgänger; `NULL`, wenn es die Wurzel ist.
- `vector<unsigned int> path`: Eine geordnete Liste von Vorgänger-Nummern über alle Baumebenen bis zur Wurzel. Diese Information ist redundant, denn sie könnte auch rekursiv aus den Vorgängern ermittelt werden.

Die Klassen sind in Abbildung 4.1 dargestellt und eine Objektstruktur der gesamten Deadlock-Handhabung zeigt Abbildung 4.2. Ihre Aufstellung ist ein wichtiger Anfangspunkt und soll kurz erklärt werden. Sie wird vom übergeordneten `DLHandler`-Objekt übernommen, das zuerst eine Tabelle der blockierten Puffer und ihren Nachfolgepuffern erstellt. Die Konfiguration, die es dafür beachten soll, wurde mit `void setConfig (Configuration* conf)` angegeben. Danach wird für jeden Irrwegpuffer ein `Switching Tree`-Objekt instanziert, das als Konstruktor-Argument einen Zeiger auf einen `TreeNode` enthält. Ruft nun der `DLHandler` seine Funktion `void DLHandler::addNodes (TreeNode* where)` auf mit demselben Zeiger als Argument, so baut diese Funktion rekursiv den ganzen Baum auf. `addNodes(TreeNode* where)` fügt alle Nachfolger von `where` an und fordert diese auf, dasselbe zu tun, indem sie auch für diese wiederum `addNodes(TreeNode* where)` aufruft. Der Funktionsaufruf belässt die Baumstruktur, wie sie ist, wenn `where` eine Wiederholung eines Puffernamens ist oder nicht blockiert ist.



(a) TreeNode (b) SwitchingTree (c) DLHandler

Abbildung 4.1: Klassen der Deadlock-Handhabung

Die Baumstruktur in Abbildung 4.2 besteht aus Objekten der in Abbildung 4.1 gezeigten Klassen. `TreeNode`s (Abbildung 4.1(a)) sind Baumknoten und enthalten eine Puffernummer, dazu den Pfad bis zu ihrer Wurzel, der hier als Zahlenmenge in geschweiften Klammern dargestellt ist. Jeder Knoten hat einen Zeiger `last` auf seinen Vorgänger, auch die Wurzel, dessen `last`-Zeiger jedoch auf `NULL` zeigt, was mit einem Erdungssymbol dargestellt ist. Im unteren Halbkreis des `TreeNode`-Symbols sind Zeiger auf Nachfolger symbolisiert. Die Blätter haben keine Nachfolger. Der `root`-Zeiger jedes `SwitchingTree`-Objekts zeigt auf die Wurzel eines Baumes. Die Pfade, die im `SwitchingTree`-Objekt gespeichert sind, zeigt die eckige Klammer im quadratischen mittleren Feld des Objekt-Symbols. Das oberste Feld enthält die Bezeichnung des Baum-Typs. Der linke Baum von Abbildung 4.2 speichert einen Pfad, der passierbar ist, der andere Baum hat keinen Pfad gespeichert. Wie erwähnt worden ist, werden nur die passierbaren Pfade gespeichert. Die einzige Instanz eines `DLHandler`s hat Zeiger auf `SwitchingTree`s, enthählt die Abhängigkeitstabelle und ist mit der Bezeichnung der zweiten Konfiguration betitelt. Die Abhängigkeitstabelle ist in Pfeil-Darstellung gezeigt. Jeder blockierte Puffer ist aufgelistet. `a->{b,c}` bedeutet, dass Puffer a die Nachfolger b und c hat und von ihnen abhängig ist. Dies macht die grobe Objektstruktur des Analysealgorithmus verständlich. Details können anhand der Kommentare im Programmcode studiert werden.

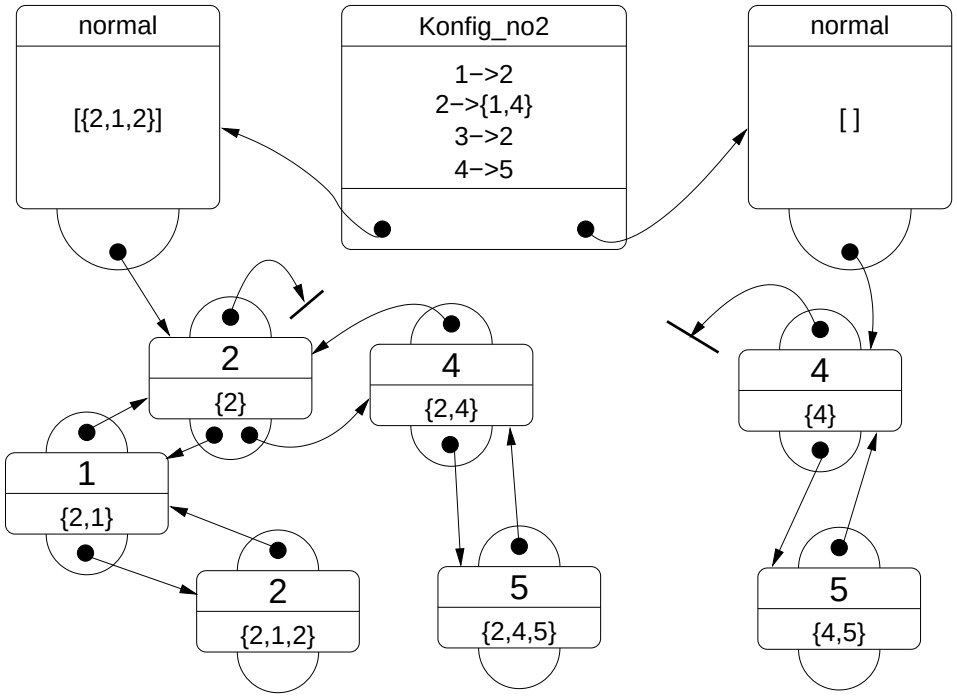


Abbildung 4.2: mögliche Objektstruktur: Gemäss der Abbildung 4.1 (a) bis (c) ist eine denkbare Objektstruktur gezeigt, wie sie anhand einer Netzsituation generiert werden könnte. SwitchingTrees speichern nur passierbare Pfade, was in Unterabschnitt 4.2.2 begründet wird.

Klassifizierung von Abhängigkeitsbäumen

Als Grundbausteine der Baumstruktur sind auch die Methoden der Knoten von spezieller Wichtigkeit:

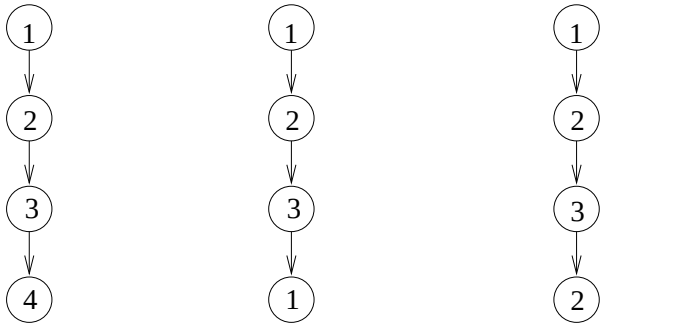
- `bool isleaf()`: Information darüber, ob der Knoten ein Blatt ist.
- `downProtocol downAnalyze()`: Diese Methode teilt mit, welchen Typ Baum der Knoten inklusive aller seiner Nachfolger darstellt und liefert in der Struktur `down Protocol` nebst dieser Antwort auch eine Liste von Zeigern, die auf alle Blätter des Baumes verweisen. Sie funktioniert rekursiv. Blätter melden, dass sie normal sind und nur ein Blatt haben, nämlich sich selbst. Alle anderen Knoten fragen ihre Nachfolger und vereinigen die Antworten. Die Vereinigung der Typ-Antwort geschieht nach gewissen *Vereinigungsregeln*, die in diesem Abschnitt näher erläutert werden.

Die Vereinigungsregeln stellen nichts anderes als eine Laufzeit-Optimierung dar. Es bestünde auch die Möglichkeit, alle Blätter nach ihren Pfaden zu fragen und die vorhandenen Pfad-Typen zu bestimmen. Dies würde ausreichen, um den Typ des Baumes zu bestimmen. Die Vereinigungsregeln sind effizienter als diese Methode, da sie für nicht normale Unterbäume direkt auf problematische Pfade schliessen. Sie werden in Tabelle 4.2 zusammengefasst. Sie bestimmen, unter welchen Bedingungen freie, passierbare und problematische Pfade im vereinigten Baum auftreten. Die Regeln werden durch die Klassifizierung der Baum-Typen aufgestellt, die in der Theorie definiert wurde (Definition 17), und aus derselben Klassifizierung wird auch der Typ des vereinigten Baumes bestimmt. Eine Wurzel ruft die Baum-Typen t_k und die Blättermengen L_k ihrer n Nachfolger $k = 1 \ldots n$ ab. Der Pfad jedes Blattes $l \in L_k$ ist auch verfügbar als $p_{k,l}$. `root->nr` ist die Nummer der Wurzel des Baumes. Aus diesen Informationen wird mit den Vereinigungsregeln bestimmt, welche Pfad-Typen im vereinigten Baum vorkommen. Daraus lässt sich der Typ des vereinigten Baumes gemäss Definition 17 bestimmen.

Diese rekursive Typen-Erkennung stellt das Kernstück der Implementierung dar. Deshalb kommt den drei Vereinigungsregeln eine grosse Bedeutung zu. Jede soll anschliessend hergeleitet werden, was jeweils nur wenige Schritte benötigt, wenn man beachtet, dass der zusammengesetzte Baum mit Sicherheit nichts anderes als freie, passierbare und problematische Pfade enthält. Pfade mit mehr als zweimaligem Auftreten desselben Puffernamens sind beispielsweise nicht erlaubt. Abbildungen 4.3 und 4.4 zeigen die möglichen Baum-Pfade und einen zusammengesetztern Baum mit Unterbäumen und Wurzel.

Pfad-Typ	**Bedingung des Auftretens**
frei	Es existiert ein normaler Unterbaum t_k und ein Blatt aus L_k mit freiem Pfad $p_{k,l}$, das *ungleich* `root->nr` ist.
passierbar	Es existiert ein normaler Unterbaum t_k und ein Blatt aus L_k mit freiem Pfad $p_{k,l}$, das *gleich* `root->nr` ist.
problematisch	Es existiert ein nicht normaler Unterbaum t_k oder ein Pfad $p_{k,l}$, der nicht frei ist.

Tabelle 4.2: Vereinigungsregeln für Bäume: Gegeben sind die Baum-Typen t_k und die Blätter-Listen L_k für alle Nachfolger k der Wurzel. Der Pfad-Typ kommt im Baum der Wurzel vor, wenn die Bedingung des Auftretens erfüllt ist.



(a) frei (b) passierbar (c) problematisch

Abbildung 4.3: Mögliche Abhängigkeitsbaum-Pfade

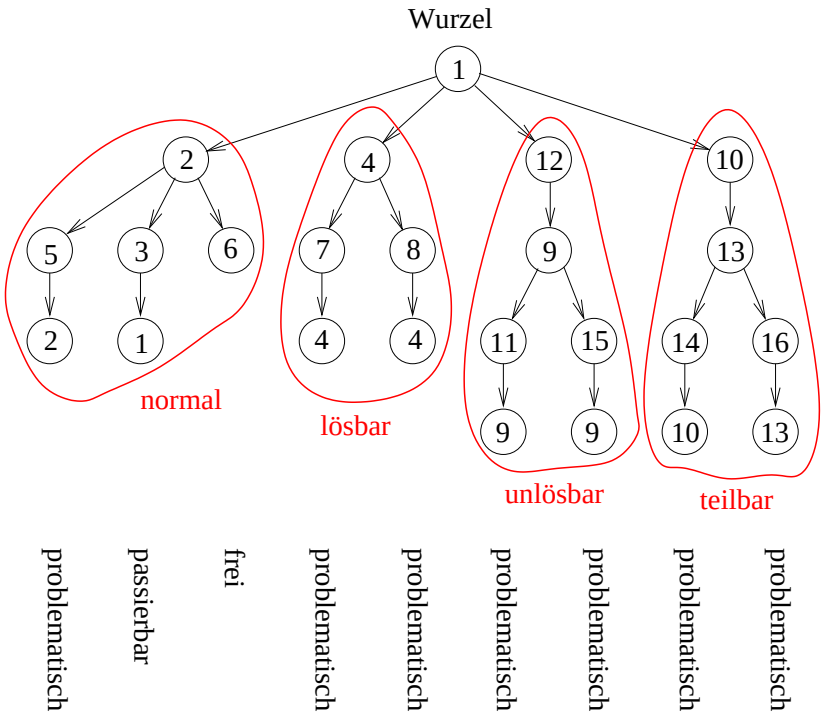


Abbildung 4.4: Mögliche Unterbäume und Pfade: Nicht normale Unterbäume erzeugen nur problematische Pfade. Normale Unterbäume können alle drei Pfad-Typen des Baumes erzeugen.

Frei kann ein Pfad nur dann sein, wenn auch ein Unterbaum einen freien Pfad enthält. Solche Unterbäume sind definitionsgemäss normal. Die Bedingung, dass durch die Wurzel ein freier Pfad des normalen Unterbaumens nicht eine Wiederholung erfährt, ist die Ungleichheit mindestens eines freien Pfad-Blattes mit der Wurzel.

Passierbar kann ein Pfad nur sein, wenn sein Teilpfad in einem Unterbaum frei, also der Unterbaum normal ist. Zusätzlich muss die Wurzel die gleiche Bezeichnung haben wie das Blatt dieses Teilpfades. Dann enthält der Pfad am Anfang und am Ende zwei gleichnamige Knoten.

Problematisch werden jegliche nicht-freien Pfade, wenn am Anfang ein Knoten als Wurzel hinzugefügt wird, weil dann die Wiederholung des Blatt-Namens nicht in der Wurzel, sondern bei einem Ast auftritt.

Die rekursive Typen-Erkennung wird nach der Aufstellung des Baumes einmal durchgeführt und das Resultat in der Variable `treeType` des entsprechenden `SwitchingTree`-Objektes gespeichert, damit die Rekursion nicht bei jeder Typen-Abfrage wiederholt werden muss. Wichtiger noch als der Typ sind aber die Pfade, die ein Baum hat. Sie müssen ebenfalls abrufbar sein und werden deshalb im gleichen `SwitchingTree`-Objekt gespeichert. Dabei wäre es nicht effizient, noch ein zweites Mal rekursiv durch den Baum zu gehen. Als Nebenprodukt ohne zuätzlichen Aufwand nennt die rekursive Typen-Erkennung die nötigen Daten, um die Pfade zu ermitteln. Bei der Typen-Erkennung wurde `downProtocol downAnalyze()` des `TreeNode* root` aufgerufen, der im `SwitchingTree`-Objekt referenziert ist. Für die Rekursion muss `downAnalyze()` jeweils in der Struktur `downProtocol` den Typ und die Blätterliste zurückgeben. Die Blätterliste von `root` ist irrelevant für dessen Baum-Typ. Trotzdem kann sie dazu benutzt werden, die Pfade aller Blätter abzurufen. Diese kennen alle Knoten bezüglich ihrer Position im Baum. Es muss lediglich die Blätterliste durchgegangen werden und jedes Blatt aufgefordert werden, seinen Pfad zu nennen. So ermittelt das `SwitchingTree`-Objekt die Pfade seines Baumes. Um etwas Speicherplatz zu sparen, werden nur die passierbaren Pfade abgespeichert, denn es werden nur diese zur Deadlock-Auflösung verwendet.

Deadlock-Auflösung

Die Theorie zeigte, dass es eine wichtige Regel gibt, die bei der Deadlock-Auflösung zu beachten ist. Es ist die **Regel 4**: *Bei einer Deadlock-Auflösung dürfen nur passierbare Pfade verwendet werden.* Wenn sie erfüllt ist, kann durch *bedingungsloses Empfangen* (Definition 19 auf Seite 47) die Paketbewegung erfolgen, ohne dass die Pufferfüllstände

berücksichtigt werden, wodurch der Deadlock aufgelöst wird. Wie nun Regel 4 und das bedingungslose Empfangen umgesetzt werden, ist eine Implementierungsfrage. Der grundsätzlich freie Vermittlungs-Algorithmus von CINSim soll durch einen Mechanismus in die Schranken (der passierbaren Pfade) gelenkt werden. Es ist wünschenswert, einen Mechanismus anzuwenden, der die Vermittlungs-Freiheit des Routers nicht antastet. Eine Möglichkeit besteht darin, als anliegendes Flit dem Router mitzuteilen, dass es gar nicht versendet werden will. Der Router lässt das Flit liegen, als wenn sein Nachfolgepuffer nicht empfangsbereit wäre. Es bedeutet also keine Einschränkung der Freiheit des Vermittlungsverfahrens. Dieses Verweigern soll dann geschehen, wenn ein am Deadlock beteiligter Puffer gerade im Begriff ist, über eine Leitung zu senden, die zu einem nicht-passierbaren Pfad gehört.

Es gilt jedoch zuvor noch zu bestimmen, welcher Anteil der beteiligten Puffer überhaupt bedingungslos empfangen soll. Im Allgemeinen wird nur ein Teil der Deadlockbeteiligten Puffer aufgelöst. Eine Markierung *auflösend* kennzeichnet solche Puffer gemäss einer Auswahlregel (Regel 5). Im Code wird diese Markierung `infinite` genannt, weil diesen Puffern kurzzeitig sehr viel Speicherplatz (`intMax`/2 Registerplätze) zuerkannt wird, damit sie ein gesendetes Paket auch dann aufnehmen können, wenn sie voll sind. Zusammengefasst: Ein Teil der Puffer eines Netzwerks ist von einem Deadlock betroffen, doch nur eine Teilmenge von den Betroffenen ist auch beteiligt. Weiter wird nur eine Teilmenge der Beteiligten in diesem Zyklus aufgelöst. Zum Beispiel sind zwei verschiedene Pfade, die nicht disjunkt sind, also gleiche Nummern enthalten, nicht beide im gleichen Zyklus auflösend. Diesen Sachverhalt demonstriert Abbildung 4.5. Es lässt sich daraus Regel 5 ableiten.

Regel 5 *Die Puffer einer Menge von passierbaren Pfaden,die an einer Auflösung teilnehmen, müssen disjunkt sein.*

Im Simulationsprogramm CINSim wird die Menge der auflösbaren Puffer folgendermassen bestimmt: Der erste passierbare Pfad in der Liste `SwitchingTree::paths` wird ausgewählt und alle seine Puffer als teilnehmend markiert. Alle weiteren Pfade werden durchgegangen und deren Puffer markiert, wenn der Pfad keine bisher markierten Puffer enthält und somit zu allen bisherigen Pfaden disjunkt ist. Diese Methode ist nicht optimal, doch lässt mindestens einen passierbaren Pfad immer ausgeführen, und so kann ein Teil des Deadlocks gelöst werden.

Damit solche Auflösungsprozesse nun nicht von anderen Puffern gestört werden, die ihnen allenfalls senden wollen, muss eine Einschränkung der Vermittlungsmöglichkeiten gemacht werden. Ein Flit verweigert das Senden, wenn Regel 6 nicht erfüllt ist.

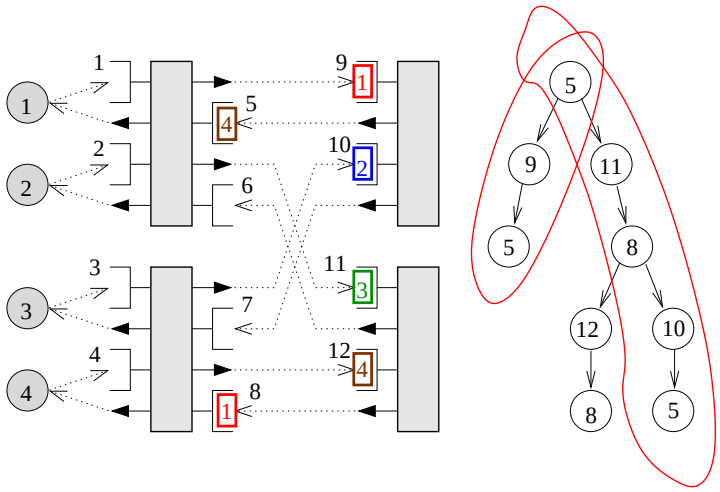


Abbildung 4.5: Zwei nicht disjunkte Pfade desselben Deadlocks, die nicht gleichzeitig ausgelöst werden können, weil Puffer 5 zweimal senden würde, was keinen Sinn ergibt.

Regel 6 *Als teilnehmend markierte Puffer dürfen nur von Puffern Flits empfangen, die ebenfalls an der Auflösung teilnehmen.*

Wenn ein teilnehmender Puffer von einem nicht teilnehmenden ein Flit empfangen würde, dann immer auch von einem teilnehmenden, der ja in einer Ring-Abhängigkeit steht. Das heisst, er würde zweimal empfangen, was nicht erlaubt ist. Wird aber die Regel 6 erfüllt, dann sind alle Vermittlungsmöglichkeiten so eingeschränkt, dass keine Auflösungskonflikte entstehen. Der teilnehmende Anteil der Puffer wird mit dem normalen Vermittlungsverfahren aufgelöst und danach werden die Markierungen aufgehoben. Eine ähnliche Bedingung wie Regel 6 für nicht teilnehmende Puffer ist unnötig, denn sie empfangen nicht bedingungslos. Wenn sie voll sind, blockieren sie automatisch jegliche Vermittlungen. Nicht teilnehmende Puffer senden und empfangen normal ohne spezielle Empfangsbedingungen.

Beispiel 1

An einem Beispiel soll gezeigt werden, wie die Markierungs-Methode funktioniert. Es wird in Abbildung 4.6 gegeben. Durch Anwendung der Regel 6 wird verhindert, dass

der problematische Pfad des Abhängigkeitsbaumes ausgelöst wird.

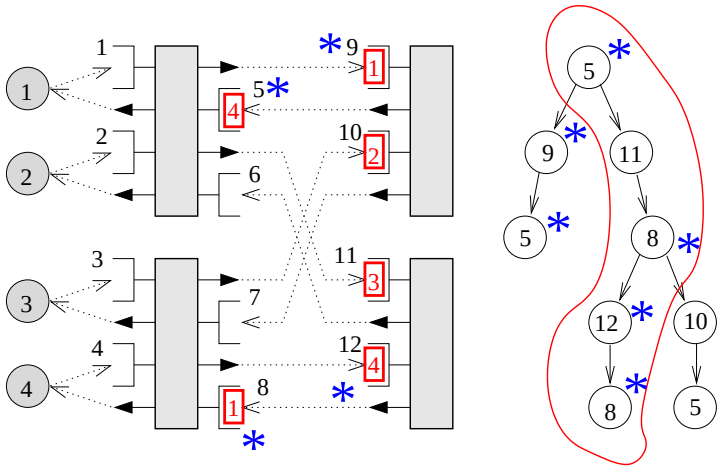


Abbildung 4.6: Beispiel einer Auflösung durch Markierung der Teilnehmer: Der umzeichnete, problematische Pfad ist nicht vermittelbar, da der Puffer 8 von Puffer 11 nicht empfangen darf. Der Pfad {5,11,8,10,5} ist nicht auflösend markiert, weil er mit dem ersten markierten {5,9,5} nicht disjunkt ist.

Beispiel 2

Ein zweites Beispiel zeigt, dass die Methode der Markierung von teilnehmenden Puffern nicht abschliessend ist. Ihr Ziel ist das Erfüllen der Regel 4, ohne dass die freie Wahl des Vermittlungsverfahrens eingeschränkt wird. Es gibt aber Netze, bei denen trotz dieser Markierungsmethode immer noch problematische Pfade vermittelt werden können. Es sind Netze, bei denen in Abhängigkeitsbäumen disjunkte passierbare Pfade direkt aneinander angerenzen. Auch ein problematischer Pfad enthält dann nur markierte Puffer, wird daher von der Methode als passierbar erkannt. Ein solches Beispiel zeigt Abbildung 4.7. Solche Situationen werden *unsimulierbar* genannt. Es ist wichtig, hier zwischen Theorie und Simulation zu unterscheiden. Die Theorie liefert auch für unsimulierbare Rekonfigurationen die richtige Vermittlungs-Einschränkung, Regel 4. Die Regel-Übertretung ist jedoch bei unsimulierbaren Situationen mit der besprochenen Implementationsmethode nicht erkennbar. Der Aufwand, sie erkennbar zu machen, wäre extrem hoch und wurde daher vermieden.

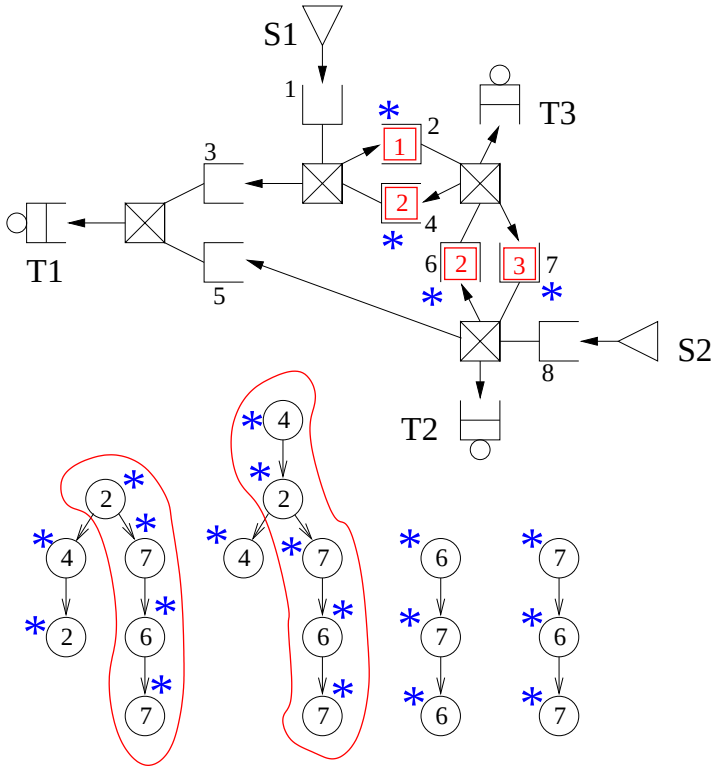
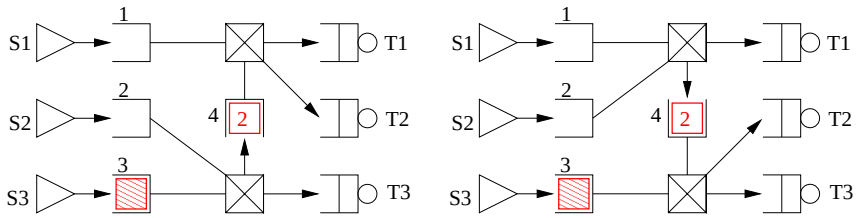


Abbildung 4.7: In CINSim nicht simulierbare Netzsituation: Die Ziele wurden jeweils um eine Position im Gegenuhrzeigersinn verschoben. Die beiden Pfade $\{2, 4, 2\}$ und $\{6, 7, 6\}$ wurden als disjunkt erkannt und ihre Puffer wurden mit einem Stern im Netz markiert. Ohne dass ein unmarkierter einem markierten Puffer senden muss, sind die problematischen Pfade der Bäume ausführbar. Würde diese Rekonfiguration tatsächlich simuliert, so ist die Wahrscheinlichkeit sehr gering, dass gerade diese Paketsituation zum Rekonfigurationszeitpunkt eintritt.

4.2.3 Handhabung von Paketlängen grösser als 1

Es wurde von der Paketziel-Übersetzung und der Deadlock-Auflösung gesprochen. Die Paketziel-Übersetzung betrifft nur die Kopf-Flits eines Pakets. Die Deadlock-Auflösung betrachtet nur Flits und nicht zusammenhängende Pakete. Deshalb kam bei diesen beiden Themen die Paketlänge nicht zur Sprache. Ihre Implementationsmethoden gelten für beliebige Paketlängen. Vor der Rekonfiguration tritt jedoch ein Problem auf, das mit Paketlängen zu tun hat. Es betrifft Paketlängen grösser 1. Abbildung 4.8 zeigt eine Rekonfiguration, die nicht durchgeführt werden darf, weil ein Paket dadurch in zwei Teile getrennt würde. Die Regel 1 ist erfüllt, doch ein hinterer Paketteil (schraffiert) hängt mit seinem Kopf nicht mehr zusammen.



(a) vor Rekonfiguration (b) nach Rekonfiguration

Abbildung 4.8: Rekonfiguration, die ein Paket in zwei Teile trennt

Damit dies nicht geschehen kann, muss eine Flit-Sammelphase vor der Rekonfiguration stattfinden, in der weder Puffer noch Ziele den Kopf eines neuen Pakets versenden dürfen, sondern nur noch hintere Paketteile. Die Betrachtung ist hier auf konstante Paketlängen l beschränkt. Nach spätestens $l-1$ Zyklen dieser Sammelphase befinden sich alle Pakete des Netzes in einzelnen Puffern und sind nicht mehr über mehrere Stationen verteilt. Rekonfigurationen werden nun genauso durchgeführt wie bei Paketlänge 1, denn es können keine Pakete mehr durchtrennt werden.

4.3 Implementierung von GBP-Flusskontrolle

4.3.1 Alternatives Paketverschiebungsverfahren

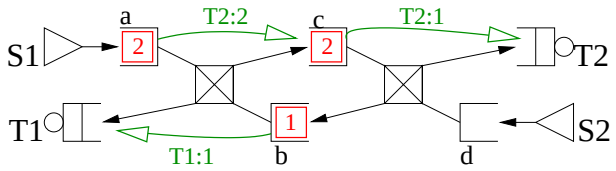
In der Einleitung dieses Kapitels wurde schon erwähnt, dass eine grundsätzlich neue Sende-Reihenfolge der Pakete gefunden werden muss, wenn bei GBP ein Deadlock auftritt. Ohne Deadlocks kann sich eine solche Reihenfolge immer an Zielen orientieren, die als erste empfangen. Im Fall eines Deadlocks sind aber alle beteiligten Puffer aneinander orientiert. Keiner von den beteilitgten Puffern kann in einer Kette von blockierten Puffern sein, deren vorderster Puffer einem Ziel senden kann. Diese Neuartigkeit der Paketverschiebung, die für Deadlocks mit GBP nötig wird, ist in Abbildung 4.9 veranschaulicht. Bei LBP trat dieses Problem nicht auf, weil sich jeder Puffer nur an seinem direkten Vorgänger orientierte. Wenn jener nicht voll war, konnte er senden. Bei Deadlocks wurden die Puffer mit zusätzlichem Speicherplatz versehen, so dass jeder Puffer seinen Vorgänger als empfangsbereit ansah. Diese Deadlock-Analyse und -Auflösung inklusive der Speicherplatz-Vergrösserung wird auch bei GBP verwendet. Als Lösungsalgorithmus reicht sie aber nicht mehr aus.

Wie Abbildung 4.9 demonstriert, müssen im Deadlock-Fall Irrwegpuffer dazu aufgefordert werden, Bezüge für neue Vermittlungs-Ketten zu finden. Die Herausforderung der Implementation von GBP ist das korrekte Entwerfen eines solchen Bezugsänderungs-Algorithmus, der folgende Regel erfüllt:

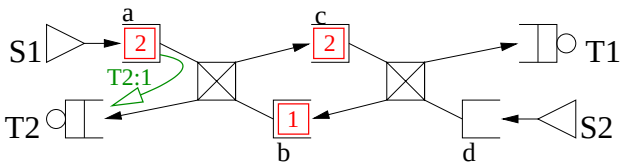
Regel 7 *Ein möglicher neuer Bezugspuffer ist auflösend (`infinite`) oder nicht voll. In jedem Fall ist er in einer Nachfolgekette desjenigen Irrwegpuffers, der einen neuen Bezugspuffer sucht.*

Lösung mit bisher benutzten Strukturen

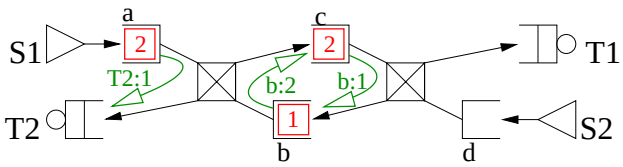
Eine theoretische Lösung des Bezugsänderungs-Algorithmus besteht darin, den Abhängigkeitsbaum des Puffers aufzubauen, der einen neuen Bezug benötigt. Ist es ein normaler Baum, so hat er einen oder mehrere freie Pfade. Von diesen freien Pfaden muss bei jedem Blatt ermittelt werden, welchen Nachfolgern der betreffende Puffer senden kann, der nicht blockiert ist. Diese möglichen Empfänger jedes Blattes von freien Pfaden sind die möglichen neuen Bezugspuffer des Irrwegpuffers, zu dem der Abhängigkeitsbaum gehört. Abbildung 4.10 zeigt eine Situation, wo der neue Bezugspuffer mithilfe des Abhängigkeitsbaums gefunden wird. Im Falle eines Deadlocks muss die Suche auf *auflösbar markierte, passierbare* statt freie Pfade beschränkt werden. Folgepuffer der Wurzel, die auf solchen Pfaden sind, kommen als Bezugspuffer in Frage.



(a) Normale GBP-Vermittlung



(b) Normale GBP-Vermittlung mit Deadlock



(c) Neue GBP-Vermittlung

Abbildung 4.9: Illustration der Vermittlungsproblematik von GBP: (a) Die normale GBP-Vermittlung ohne Deadlock geht der Reihe nach für jedes Ziel. T2 empfängt ein Flit von c, dann c eines von a. T1 empfängt bezüglich des Algorithmus dasjenige von b. (b) Will bei einem Deadlock T1 etwas empfangen, so bleibt es erfolglos. T2 empfügt ein Flit von a. Die Pakete in b und c bleiben liegen, obwohl das bedingungslose Empfangen aktiv ist. Die Puffer sind in keiner Sende-Reihenfolge eingeplant. (c) Dies ändert die neue GBP-Vermittlung, wo c als Vorgänger von T1 aufgefordert wird, einen neuen Bezug herzustellen. Diesen findet c in b, sendet an b, worauf b nach c sendet.

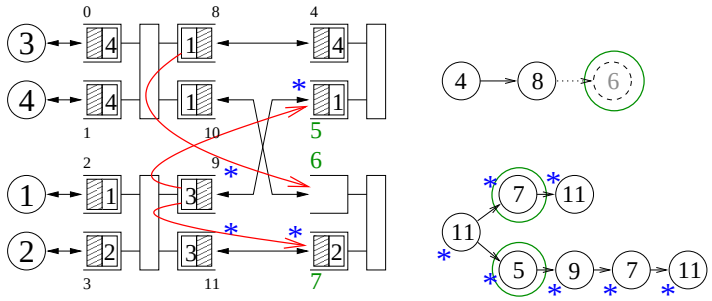


Abbildung 4.10: Veranschaulichung der theoretischen Lösung: Puffer 4 hat einen normalen Abhängigkeitsbaum. Der Puffer 6, an den sein Blatt senden kann, wird der neue Bezugspuffer. Die Pfeile zeigen den möglichen ersten Vermittlungsweg. Dasselbe gilt für auflösbar markierte Puffer, wobei aber die ersten Nachfolger der Wurzel des Baumes in Frage kommen.

Die Entdeckung, dass eine solche Implementation möglich wäre, ist erst kurz vor dem Abgabetermin der Master-Arbeit gemacht worden und konnte nicht mehr umgesetzt werden. Sie würde den letzten Teil eines insgesamt eleganteren Algorithmus der Deadlock-Auflösung darstellen als die implementierte Methode, da er die Analyse der Netzsituation einmal durchführt und dann für die Markierung der auflösbaren Pfade und die Suche nach neuen Bezugspuffern verwendet. Die implementierte Methode führt die zweite Aufgabe separat durch, was eine Redundanz dieser Berechnungen bedeutet.

Implementierung einer speziellen Suchfunktion

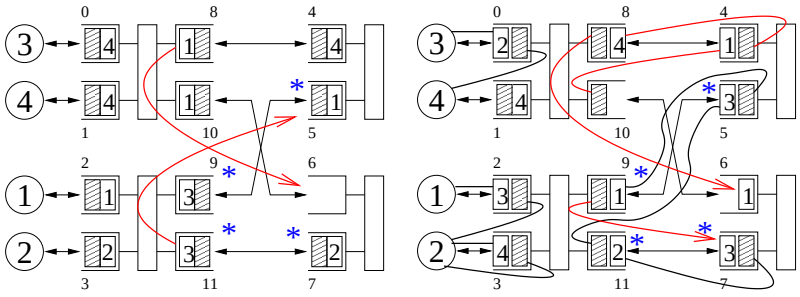
Der implementierte Deadlock-Auflösungs-Algorithmus enthält für GBP nebst dem Analyseverfahren der Abhängigkeitsbäume auch eine spezielle Suchfunktion für neue Bezugspuffer, die anstelle der Ziele zum Empfangen bereit werden sollen. Sie wird in `bool GBForwarding::activateChildren()` aufgerufen und heisst `SpaceSuccessors`. Sie Funktion führt eine Tiefensuche nach möglichen Puffern durch, die der Methode mit Abhängigkeitsbäumen gleichgesetzt ist. Die Suchfunktion gibt die Antwort auf die Frage, von welchen Puffern her eine Kette von Sendungen den Irrwegpuffer im Argument der Funktion betreffen könnte. Es sollen drei wichtige Aspekte der Suche besprochen werden, aus denen die Bildung des implementierten Algorithmus hervor geht:

Vormerken jedes Puffers, dessen Eignung für eine Bezugsänderung geprüft wurde.

Dem Kopf folgen. Puffer, die nicht Paketköpfe an erster Stelle haben, suchen als Bezüge nur in die Richtung, wohin der Kopf schon versendet wurde.

Iterativ suchen. Puffer suchen iterativ nach neuen möglichen Bezügen.

Die Abbildung 4.11 zeigt die Auflösung eines Deadlocks in den ersten zwei Taktzyklen nach der Rekonfiguration. Die drei Konzepte sind in dieser Auflösung zu erkennen. Der Puffer 8 findet 6 als seinen neuen Bezugspuffer. Er könnte aber auch 4 erwogen haben. 4 hingegen ist voll, und es gilt die Iterationsregel. D.h. 4 sucht weiter und erwägt 8. Nur weil 8 vorgemerkt ist, kann 4 die Suche abbrechen, sonst würde 4 wieder 8 erwägen und die Iteration würde nicht abbrechen. Der Puffer 10 sucht im 2. Taktzyklus einen neuen Bezugspuffer und muss seinem Kopf nachfolgen. Der so gefundene Puffer muss dies mit seinem vordersten Flit iterativ ebenso tun, bis er einen nichtvollen Puffer vorfindet. Puffer 11 findet als auflösender Puffer den neuen Bezugspuffer 5.



(a) 1. Taktzyklus (b) 2. Taktzyklus

Abbildung 4.11: Auflösung eines Deadlocks im GBP-Fall. Die zweite Stufe (Puffer 4-11) des BMIN bleibt gänzlich inaktiv mit der normalen Flusskontrolle. Grund: Puffer 8 bis 11 senden nicht in Richtung der Ziele (gleiche Situation wie in Abbildung 4.9). Die Puffer 8 und 11 suchen nach neuen Bezügen und finden sie in 6 und 5 (Pfeile). Im zweiten Takt suchen 9 und 10 neue Bezüge. 9 muss 7 wählen und 10 über drei Stationen 6. Die Paketzusammenhänge sind hier mit Linien dargestellt.

Jeder Puffer, der erwogen wird, wird zuerst auf seine Vormerkung geprüft. Wenn er schon vorgemerkt ist, dann wird die Suche bei diesem Puffer abgebrochen. Andernfalls wird er vorgemerkt und die Suche wird von diesem Puffer aus weitergeführt. Die Suchfunktion beginnt folgendermassen:

```
00: vector<Buffer*> SpaceSuccessors(Buffer* b)
01: {
02:   vector<Buffer*> rtn_vec; // erzeugt leeren Vektor
03:   if(checked[b->nr]) return rtn_vec; // wenn b schon markiert return
04:   checked[b->nr] = true; // sonst markiere b und fahre fort
05:   ...
xx: }
```

Die Klassenvariable `vector<bool> checked` enthält für jeden Puffer des Netzes einen Eintrag, der zu Beginn `false` ist. Nach jedem Suchdurchlauf müssen alle Einträge wieder zurückgesetzt werden. Dies passiert am Ende von jedem Taktzyklus. In Worten bedeutet der Anfang der Funktion folgendes:

> *SpaceSuccessors ist eine Funktion, dessen Argument ein Zeiger auf einen Puffer ist. Das Resultat der Funktion wird eine Liste von Zeigern auf Puffer sein (die als Bezugspuffer in Frage kommen). Eine leere Liste von Puffer-Zeigern wird erzeugt. Wenn das Funktionsargument auf einen geprüften Puffer zeigt, dann soll die leere Liste zurückgegeben werden, ansonsten wird der Puffer als geprüft markiert, auf den das Argument zeigt.*

Wenn der Puffer x, der einen neuen Bezug finden soll und die Funktion `SpaceSuccessors(x)` aufruft, als vorderstes Flit nicht einen Paketkopf hat, dann gestaltet sich die Suche einfach. Es gibt nur einen Puffer, zu dem x senden kann. Es ist der Puffer y, dessen hinterstes Flit zu demselben Paket gehört wie das vorderste Flit von x. Ist nun das vorderste Flit von y wiederum kein Kopf, so geht die Suche dementsprechend weiter. Andernfalls wurde der einzige mögliche neue Bezugspuffer gefunden. Wenn y als auflösend markiert ist, dann wird die Suche abgebrochen und y als neuer Bezugspuffer gewählt. Situationen mit hinteren Paketteilen als forderste Flits von Puffern zeigte Abbildung 4.11 (b).

Die iterative Suche nach Köpfen von Schwanz-Flits an erster Puffer-Position ist je Schritt eindeutig und kommt daher ohne Rekursion aus. Das iterative Suchen bei Kopf-Flits an erster Puffer-Position ist jedoch rekursiv realisiert, da es im Gegensatz zu der ersten Situation echte Tiefensuche eines Baumes darstellt mit mehreren Suchmöglichkeiten pro Zwischenschritt. Als Abschluss dieses Unterabschnitts wird der vereinfachte Code der gesamten Suchfunktion gezeigt und die rekursive Tiefensuche beschrieben.

```
00: vector<Buffer*> SpaceSuccessors(Buffer* b)
01: {
02:   vector<Buffer*> rtn_vec; // erzeugt leeren Vektor
03:   if(checked[b->nr]) return rtn_vec; // wenn b schon markiert return
04:   checked[b->nr] = true; // sonst markiere b und fahre fort
05:   vector<Buffer*> b_next = b->next(); // lade Nachfolger
06:   if(b_next.empty()) return rtn_vec; // wenn leer return
07:   bool head = b->front()->head(); // hat b Kopf an 1. Stelle?
08:   if(!head) // wenn nein
09:   { // suche den Kopf, speichere einen Zeiger davon in
10:     rtn_vec.push_back(findHead(b)); // rtn_vec
11:     return rtn_vec; // return 1-elementige Liste rtn_vec
12:   }
13:   else // wenn ja, vorderstes Flit ist Kopf
14:   { // suche in allen Nachfolgern b_next nach Bezugspuffern
15:     for(unsigned int k = 0; k<b_next.size(); k++)
16:       if(!b_next[k]->full() && b_next[k]->i() == b->i())
17:         rtn_vec.push_back(b_next[k]);
18:     if(rtn_vec.empty()) // Wenn nicht Erfolg, dann beauftrage die Nachfolgepuffer mit der Suche und sammle ihre Antworten in rtn_vec auf
19:       for(unsigned int k = 0; k<b_next.size(); k++)
20:         rtn_vec.insert(SpaceSuccessors(b_next[k]));
21:     return rtn_vec; // return gefundene Antworten
22:   }
23: }
```

Folgende spezielle Schreibweisen sind zu beachten:

- Die Funktion `b->next()` ermittelt die Nachfolgepuffer von `b` (Zeile 05).
- `b->front()->head()` ruft die `bool head()`-Funktion des vordersten Flits von `b` auf, die wahr ist, wenn das Flit der Kopf eines Pakets ist (Zeile 06).
- `findHead(b)` stellt eine Funktion dar, die einen Puffer sucht in der Kette von Nachfolgern von `b`, der als vorderstes Flit einen Kopf hat (Zeile 10).
- `b->i()` führt für jeden Pufferzeiger b den Lesezugriff auf die boolsche `infinite`-Variable aus, die angibt, ob der Puffer auflösend markiert ist (Zeile 16).

- `b->full()` gibt eine boolsche Variable zurück, die falsch ist, falls im Puffer b noch freier Platz vorhanden ist, und sonst wahr.

- `rtn_vec.insert(SpaceSuccessors(b_next[k]))` stellt die Vereinigung von `rtn_vec` mit der Antwort von `SpaceSuccessors(s_next[k])` dar (Zeile 20). Diese Syntax existiert in C++ genau genommen nicht, wird aber hier der Einfachheit halber so verwendet.

Die rekursive Tiefensuche betrifft den Programmteil ab Zeile 15. Im Abschnitt von Zeile 15 bis 17 sucht die Funktion Nachfolger von `b`, die freien Platz haben, und sammelt diese im Antwortvektor `rtn_vec`. Hat sie welche gefunden, so springt sie zu Zeile 21, wo der Antwortvektor zurückgegeben wird. Andernfalls lässt sie die Nachfolger von `b` in Zeile 20 je ihre eigenen möglichen Bezugspuffer ermitteln, sammelt die Antworten in `rtn_vec` und gibt diesen schlussendlich in Zeile 21 wiederum zurück. Die Rekursion findet in Zeile 20 statt. Das Beispiel einer Netz-Situation, für die sie nötig wird, zeigt Abbildung 4.12. Es ist eine vereinfachte Situation, die das Paket in Puffer 1 mit den Zielen 1 und 6 als Bezugszielen nicht bewegen kann. Durch die Flusskontrolle von den Zielen 2, 4, 5 und 7 her kommt sie jedoch ohne die Suchfunktion aus. Eine Situation, die von keinem Ziel her beweglich ist, ist analog zu Abbildung 4.12 konstruierbar, doch verliert einiges an Überschaubarkeit. Der Abhängigkeitsbaum, mit dem die Bezugspuffer ebenfalls gefunden werden können, zeigt Figur 4.13.

4.3.2 Setzen und Aufheben von Puffersignaturen

Für die GBP-Implementierung wurden drei verschiedene Puffermarkierungen eingeführt.

- `bool Buffer::maeander` zeigt an, ob das vorderste Flit eines Puffers zu einem Irrwegpaket gehört, d.h. ob der Puffer ein Irrwegpuffer ist. Die Markierung wird gesetzt, wenn beim Aufruf der Funktion `Critial()` von `DLHandler` erkannt wird, dass ein Puffer auf diese Beschreibung zutrifft. Die `Critial()`-Funktion prüft, ob das Netz kritisch ist.

- `bool BufferQueue::infinite` ist wahr, wenn der Speicher des Puffers „unendlich“, d.h. der Puffer aufzulösend ist. Diese Signatur wird bei der Warteschlange `BufferQueue` des Puffers gesetzt, wenn die Analyse von kritischen Netzen beendet wurde und der Puffer als aufzulösend erkannt wurde.

- `vector<bool> GBForwarding::checked` ist eine Liste von boolschen Variablen mit einem Eintrag pro Puffer im Netzwerk. Das GBP-Packetforwarding merkt sich für jeden Puffer, ob er als Bezugspuffer schon erwogen wurde.

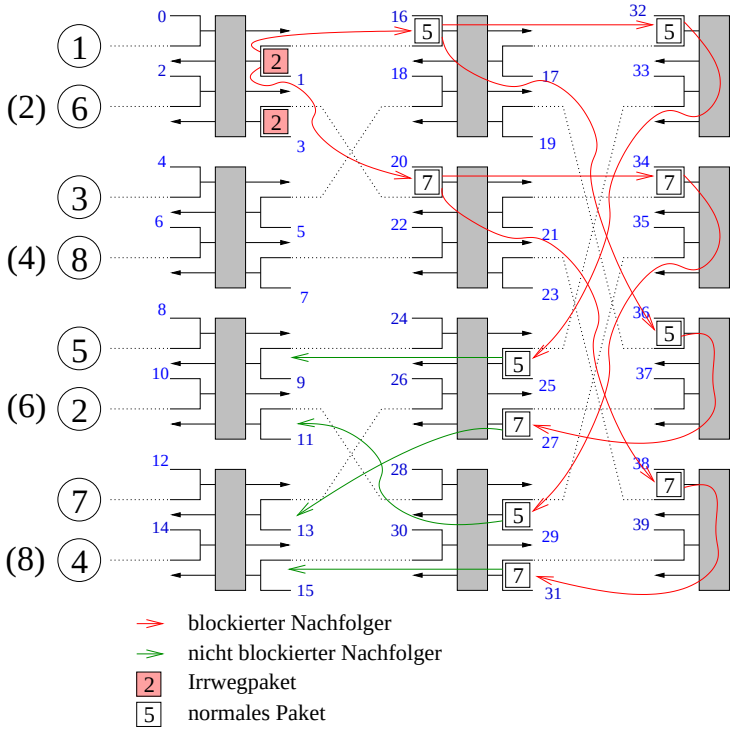


Abbildung 4.12: Netzsituation mit notwendiger Tiefensuche der Bezugspuffer: Die Situation entstand, indem die alten in Klammern stehenden Zielnamen zu den angegebenen geändert wurden. Irrwegpuffer 1 sucht in seinen Nachfolgepuffern 16 und 20 nach möglichen Bezugspuffern. Diese sind beide blockiert und es wird rekursiv tiefer gesucht, bis schliesslich die Puffer 9, 11, 13 und 15 gefunden werden. Diese vier hat nun 1 zur Auswahl als seine neuen Bezugspuffer.

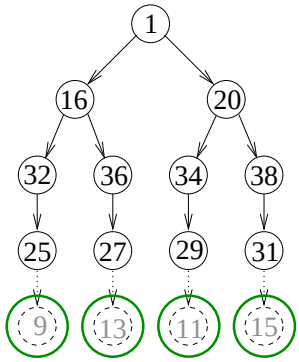


Abbildung 4.13: Abhängigkeitsbaum der Beispiel-Situation für Tiefensuche. Kreise zeigen die gefundenen möglichen Bezugspuffer.

Es ist entscheidend, die jeweiligen Markierungen zur rechten Zeit zu entfernen. Diese Zeitpunkte sind durch die Definitionen der Markierungen gegeben:

- `maeander` wird dann zurückgesetzt, wenn der Puffer nicht mehr ein Irrwegpuffer ist, das heisst genau dann, wenn das letzte Flit des Irrwegpakets versendet wurde.
- Die `infinite`-Markierung darf ein Puffer nicht schon nach dem ersten Zyklus der Auflösung verlieren. Solange der Deadlock noch besteht, das heisst solange die betreffenden Pakete nicht vollständig versendet sind, muss sie beibehalten werden. Also wird auch diese zurückgesetzt, wenn die aufzulösenden Puffer ihre letzten Flits der entsprechenden Pakete gesendet haben.
- `checked` wird am Ende eines Taktzyklusses für jeden Puffer zurückgesetzt

4.4 Implementierung von Cut-Through Switching-verfahren

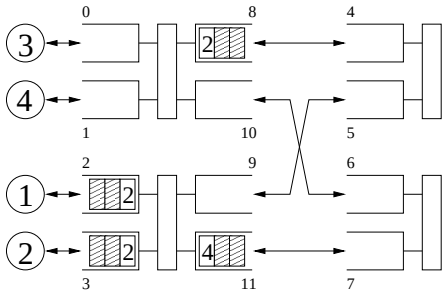
LBP und GBP wurden als zwei verschiedene Verfahren der Flusskontrolle für die dynamische Rekonfiguration erweitert, was einen erheblichen Entwicklungs- und Implementierungsaufwand bedeutete. Diese funktionieren ohne weitere Ergänzungen nur mit dem Switchingverfahren „Store And Forward“ (SAF). Die Erweiterung anderer Switchingverfahren wie „Partial Cut Through“ (PCT) und „Virtual Cut Through“ (VCT) für

dynamische Rekonfiguration hat sich als weniger aufwändig erwiesen als diese der Flusskontrolle. Es treten dabei nur zwei kleinere Implementierungsprobleme auf, die durch unscheinbare Eingriffe in die bisherige Implementation gelöst werden können. Das eine Problem ist das mögliche Auftreten einer Deadlock-Situation für PCT und VCT, die mit der bisherigen und neuen Flusskontrolle von GBP nicht lösbar ist. Ein kleiner Eingriff löst das Problem gänzlich mit nicht viel mehr als einer Handvoll zusätzlicher Programmzeilen. Ein eher unschöner Eingriff in die bisherige Programmstruktur ist nötig, um das zweite Problem zu lösen. Die Durchbrechung der herkömmlichen Vermittlungsreihenfolge für GBP, wie es im letzten Unterkapitel demonstriert wurde, muss für PCT und VCT so erweitert werden, dass wiederum SAF zu viele Sendemöglichkeiten erhält. Eine prinzipielle Schwierigkeit, dieses Problem zu lösen, zwingt den Entwickler, bei der Paketbewegung das verwendete Switchingverfahren zu erfragen, um dann je nach Verfahren eine Bewegung zu erlauben oder zu verbieten. Dies ist eine Unschönheit, deren Auflösung weitere Gedanken wert ist, die im Rahmen dieser Master-Arbeit nicht im Bereich des Möglichen gelegen sind. „Wormhole“ wurde als viertes mögliches Switchingverfahren einer dynamischen Rekonfiguration verworfen, da es unlösbare Deadlocks erzeugen kann. Die Validierung hat zudem gezeigt, dass auch PCT ein inkorrektes Verhalten zeigt. Unbekannte Probleme verursachen zeitweise überfüllte Puffer.

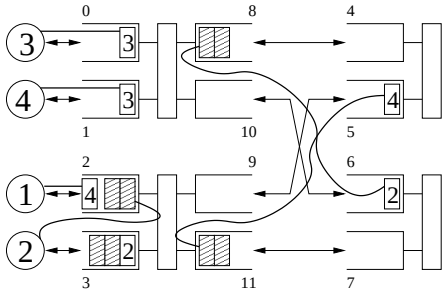
4.4.1 Behandlung von kritischen Deadlock-Fällen

PCT und VCT erlauben einem vorderen Flit eines Paketes, sich weiter zu bewegen, auch wenn der hintere Teil des Paketes noch nicht im Puffer angekommen ist, in dem das Flit sich befindet. Kommen nun Pakete in jedem Zyklus weiter aus ihren Puffern heraus und verbreiten sich im Netzwerk, so kann es geschehen, dass sich Pakete erst nach einer gewissen Zeit gegenseitig blockieren. Eine solche Situation zeigt Abbildung 4.14.

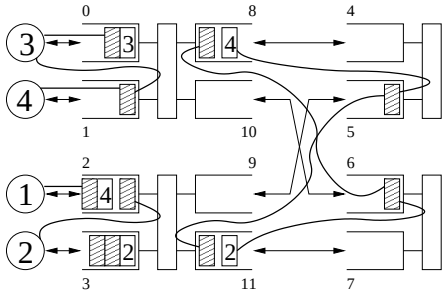
Es handelt sich dabei nicht um einen prinzipiell neuen Deadlock-Charakter. Das Problem besteht lediglich darin, dass der Deadlock nicht als solcher erkannt wird, weil die Puffer 8 und 11 nicht als blockiert angesehen werden. Dies benötigt eine Änderung in der Abhängigkeitstabelle dieser Puffer. Sie sind als blockiert zu vermerken, obwohl ihre Nachfolger nicht voll sind. Es ist nicht von Vornherein klar, weshalb diese Puffer keine weiteren Flits aufnehmen können. Die theoretisch definierten Switchingverfahren PCT und VCT definieren diesen Fall genau genommen nicht, denn es ist nicht spezifiziert worden, ob blockierte Schwanz-Flits weitere Flits desselben Pakets nachziehen dürfen oder nicht. Dies wurde nur für die Kopf-Flits als gültig erklärt. In CINSim wurde die Annahme gemacht, dass Schwanz-Flits dies nicht tun dürfen. Deshalb können die Puffer 5 und 6, in denen Schwanz-Flits liegen, keine weiteren Flits aufnehmen, obwohl sie nicht voll sind.



(a) Ausgangssituation



(b) Nach 1 Zyklus



(c) Nach 2 Zyklen

Abbildung 4.14: Spezieller Deadlock-Fall für PCT und VCT: Die Ausgangssituation (a) wird von der Analyse richtig behandelt. Irrwegpakete gehen ihre Rückwege. Nach zwei Taktzyklen hat sich ein neuer Deadlock gebildet, der bisher vom Analyseverfahren unerkennbar ist, weil die Puffer 5 und 6 nicht voll sind. Die Analyse hat 8 und 11 bisher nicht als blockiert erkannt.

Regel 8 *Sei n die Nummer des Pakets, zu dem das vorderste Flit eines Puffers b gehört. Ist in einem Nachfolgepuffer ein Flit an letzter Stelle, das zu demselben Paket der Nummer n gehört, so ist der Puffer b von dem Nachfolger abhängig, auch wenn dieser nicht voll ist.*

Diese Zusatzregel wurde mit nur 8 zusätzlichen Codezeilen in das bisherige System eingefügt. Es sind die letzten Codezeilen der Funktion `void DLHandler::setTables()` die in der Datei `dlhandler.cpp` spezifiziert ist. Mit dieser Ergänzung kann das System den speziellen Deadlock-Fall lösen.

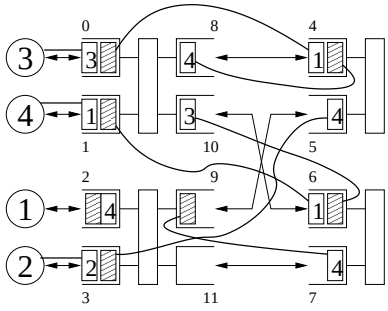
4.4.2 Sendemöglichkeiten von SAF, PCT und VCT

Eine andere Situation, für die das System bisher mit PCT und VCT ein Fehlverhalten simuliert, zeigt Abbildung 4.15. Die korrekte Paketbewegung für beide Switchingverfahren kann nur dann erreicht werden, wenn bei der GBP-Flusskontrolle der Typ des Switchingverfahrens bekannt ist. Ein Programmswitch in `void GBForwarding::receivePackets()` löst das Problem, doch das Ideal des inerten Implementierens der Algorithmen musste damit aufgegeben werden.

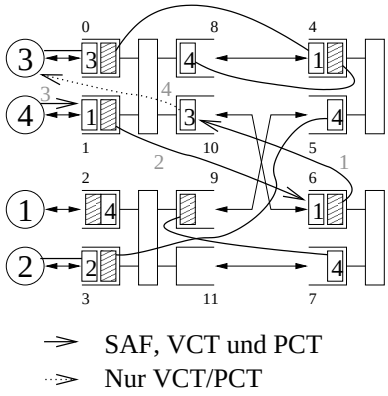
Während für SAF und PCT diese Programmswitch-Methode einfach zu implementieren ist, ist dies bei VCT nicht der Fall. Als Mischverfahren zwischen SAF und PCT kennt VCT sowohl Situationen, in denen es sich als SAF verhält, als auch solche, in denen es sich als PCT verhält. Einen nötigen tieferen Eingriff in den Paketverschiebungsalgorithmus für VCT konnte im Rahmen dieser Master-Arbeit aus Zeitgründen nicht vollendet werden. CINSim simuliert nach der Implementierung des Programmswitch für VCT nicht alle Paketbewegungen richtig. Das heisst, dass VCT mit der aktuellen Implementierung nicht verwendet werden darf.

4.5 Zusammenfassung der Implementierungen

Die Deadlock-Analyse und -Auflösung in das bestehende System CINSim zu implementieren, machte neben dem Entwurf der Deadlock-Handhabungs-Klasse `DLHandler`, die Abhängigkeitsbäume aufstellt und analysiert, eine relativ auwfändige Bezugsänderung des Paketverschiebungsalgorithmus nötig. Paketlängen grösser 1 und die dadurch neu dazugekommene Switchingverfahren haben ernsthafte Probleme gestellt, so dass die nun vorliegende Implementierung in CINSim dynamische Rekonfigurationen nur allgemein für LBP oder für GBP mit Paketlänge 1 korrekt simulieren kann. Rekonfigurationen mit VCT und PCT und Paketlängen grösser 1 weisen bei der Simulation durch CINSim trotz den Implementierungs-Bemühungen Mängel auf, wie die Validierung zeigen wird.



(a) Ausgangssituation



(b) Paketbewegung

Abbildung 4.15: Netzituation mit speziellem VCT/PCT-Sendeverhalten: Die Situation (a) kann für alle drei verwendeten Switchingverfahren eintreten (jeweils mit GBP). Bei der Paketbewegung (b) wird Ziel 1 aktiv und macht Puffer 11 empfangsbereit. 11 versucht, von 6 zu empfangen. Da aber 6 ein Irrwegpuffer ist, sucht er sich einen neuen Bezugspuffer, den er in 10 findet. 10 wird von 6 schliesslich das zweite und letzte Flit seines Pakets empfangen. Kommt nun Ziel 3 an die Reihe, so darf bei VCT/PCT 10 an Ziel 3 senden, da diese Verfahren ausgedehnte Pakete über mehrere Puffer erlauben. SAF erlaubt dasselbe nicht, obwohl Puffer 10 das Paket vollständig enthält, weil dies zu Beginn des Zyklusses nicht der Fall war und SAF demnach das Versenden von 10 an Ziel 3 verbietet.

Kapitel 5

Validierung

5.1 Vorüberlegungen

Die vorliegende Implementation benötigt einen Nachweis, dass die Simulationsresultate der Modellvorstellung der Verbindungsnetze entsprechen. Wie viel von einem solchen Nachweis zu erwarten ist und was für Nachweise grundsätzlich im Bereich des Möglichen liegen, soll im Voraus kurz diskutiert werden.

5.1.1 Motivation einer Validierung

Das erweiterte CINSim kann dynamische Rekonfigurationen simulieren und Messwerte wie Verzögerung und Durchsatz pro Takt berechnen. Diese Resultate sind vorerst mit Vorsicht zu geniessen, denn ihre Korrektheit ist noch nicht bestätigt. Wird beispielsweise die Rekonfiguration eines Netzes mit dem VCT-Switchingverfahren simuliert, so liefert CINSim falsche Resultate, wie es Abschnitt 4.4.2 erklärt. Es stellt sich heraus, dass fehlerhafte Resultate viel leichter zu erkennen sind als korrekte. Der Wissenschaftstheoretiker Karl Popper, der Physiker Albert Einstein und andere mit ihnen haben generell festgestellt, dass die Bestätigung von wissenschaftlichen Thesen einen negativen Charakter hat: Sie können durch einen aufgezeigten Widerspruch zu anderen Themenbereichen leicht widerlegt werden. Die Bestätigung einer These muss jedoch darauf beruhen, dass eine möglichst intelligente Suche nach Widersprüchen bisher auf keine Widerlegung geführt hat. Wenn bei aller Anstrengung keine fehlerhaften Resultate gefunden wurden, so kann eine These (vorläufig) als wahr gelten. Diese wissenschaftstheoretische Auffassung ist in der „kleinen Weltgeschichte der Philosophie“ von Hans Joachim Störig [16] erläutert. Abbildung 5.1 zeigt eine mögliche Validierung von CINSim, die auf dieses Prinzip aufbaut.

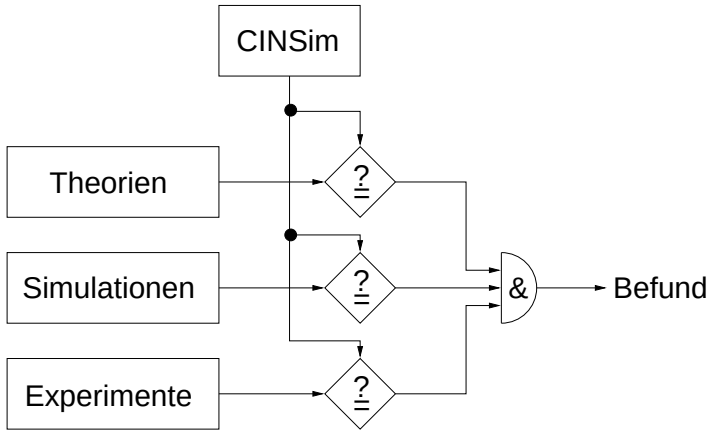


Abbildung 5.1: Validierungsprinzip von CINSim: Die Implementation von CINSim kann dann als (vorläufig) korrekt angesehen werden, wenn eine intelligent angelegte Suche nach Versuchen gescheitert ist, widersprüchliche Resultate verglichen mit anderen Paradigmen zu erzeugen. Bei der Validierung von CINSim spielen die drei Paradigmen Theorie, Simulation und Experiment eine Rolle.

Bei der Validierung des erweiterten Simulationstools CINSim für dynamische Rekonfigurationen werden die durch Simulation gewonnenen Resultate mit denen desselben oder anderer Paradigmen (Theorie, Experiment) verglichen. Liefert beispielsweise die Theorie einen oberen Grenzwert für den Durchsatz eines rekonfigurierten Netzes, so wird die Implementation von CINSim dann falsifiziert, wenn der Grenzwert bei der Simulation überschritten wird. Die Implementation von CINSim wird ebenfalls falsifiziert, wenn eine in CINSim eingebaute Warnung wegen verlorenen Paketen mindestens einmal erscheint. Dies ist eine Falsifizierung von CINSim durch Simulation. Ein Experiment, das heisst ein in Hardware aufgebautes Netzwerk, das CINSim simuliert hat, kann auf ähnliche Weise das Simulationstool bestätigen oder widerlegen. Die Bedingung, dass Bestätigung oder Widerlegung gültig sind, ist die vollständige Abdeckung des in CINSim verwendeten Modells im entsprechenden Paradigma. Das heisst, dass bei der Abbildung des Modells in das Paradigma keine relevanten Modell-Aspekte weggelassen oder verändert wurden. Sind die Möglichkeiten des Vergleichs von Resultaten auf diese Weise ausgeschöpft, kann die Funktionalität von CINSim als belegt gelten.

5.1.2 Möglichkeiten und Grenzen der Validierung

Es gibt eine Reihe von Möglichkeiten, die Funktionalität des erweiterten CINSim zu prüfen. Zwei wichtige Prüfverfahren müssen aber von vornherein im Rahmen dieser Master-Arbeit ausgeschlossen werden:

Experiment: Die benötigten Hardwarekomponenten stehen für eine experimentelle Reproduktion der simulierten Resultate nicht zur Verfügung.

Theoretische Verifizierung: Die analytische Verifizierung der Simulationsresultate ist für kein Netz möglich, das Deadlocks bei der Rekonfiguration hervorruft. Die Arbeiten [17], [18] und [19] weisen darauf hin, dass eine solche schon für kleinstmögliche rekonfigurierbare Netze zu aufwändig ist, um im Rahmen dieser Arbeit durchgeführt zu werden.

Es bleibt nur die simulative Möglichkeit der Validierung bestehen. Doch auch diese kann nicht vorbehaltlos sein, denn sie kann nicht sicherstellen, dass alle möglichen Simulationsvarianten und Netzsituationen (inklusive aller möglichen Topologien) getestet werden. Warnmeldungen zeigen dabei unerlaubte Netzsituationen an. Eine Warnmeldung wird während der Simulation ausgegeben, wenn:

- Ein Paket während der Rekonfiguration durch die Paket-Adressen-Übersetzung alle Ziele verlor und gelöscht werden musste.

- Ein Paket nach einem Simulationslauf im Netz liegen geblieben ist, also nicht zu einem Ziel vermittelt werden konnte.
- Eine Netzsituation auftrat, die nur unlösbare Abhängigkeitsbäume erzeugte.
- Ein Puffer kurzzeitig überfüllt wurde.

Alle diese Warnungen weisen darauf hin, dass CINSim eine falsche Berechnung durchgeführt hat, denn es wurde angenommen, dass eine dynamische Rekonfiguration (mit der in 4.2.3 erwähnten Sammelphase) ohne Paketverlust ist, und auftretende Deadlocks immer lösbar sind. Paketbewegung ist zusätzlich nie so angelegt, dass Puffer kurzzeitig mehr Flits tragen als Speicherplatz für sie bereitgestellt wurde. Eine gegenteilige Warnmeldung würde bedeuten, dass die Paketbewegung inkorrekt funktioniert. Das Ausbleiben jeglicher Meldungen lässt darauf schliessen, dass in den Simulationsläufen keine Fehler aufgetreten sind. Der Vorbehalt, dass Fehler auftreten könnten, auf die keine Warnmeldung anspricht, wie es beispielsweise bei der falschen Vermittlung von VCT der Fall ist, bleibt bestehen. Er bedeutet, dass eine breiter angelegte Validierung von CINSim eine grössere und intelligentere Menge von Fehlerquellen prüfen muss, wozu die verbleibende Zeit für diese Master-Arbeit nicht ausgereicht hat. Das nächste Unterkapitel bespricht die simulative Validierung mit den ausgewählten Fehlerquellen.

Um den positiven Befund der simulativen Validierung zu illustrieren, diskutiert ein weiteres Unterkapitel Resultate von Simulationen aus theoretischer Sicht und zeigt auf, wie theoretisch zu erwartende Verhaltensweisen bei der dynamischen Rekonfiguration verschiedener Netze tatsächlich zu beobachten sind. Im weiteren Sinn stellt auch eine solche Kurven-Interpretation eine Validierung dar, die aber nicht mit einer theoretischen Verifizierung zu verwechseln ist. Während die theoretische Verifizierung exakte Werte voraussagt, was für rekonfigurierbare Netze nicht möglich ist, so nimmt eine Interpretation simulierte Mittelwerte von Messgrössen pro Zyklus und begründet deren Verlauf. Diese Begründung kann nur in Teilaspekten erfolgen, was die Aussagekraft einer Kurven-Interpretation im Verhältnis zu einer theoretischen Verifizierung schmälert.

5.1.3 Verwendete Netze und Rekonfigurationen

In Abbildung 5.2 ist die Topologie und Rekonfiguration des Bogen-Netzwerkes gezeigt. Das Bogen-Netzwerk ist das simple Netz, das in den Beispielen der Theorie und der Implementierung benutzt wurde, gezeigt in der Abbildung 3.5 auf Seite 34. Die angewendete Rekonfiguration ist die Punktspiegelung der Ziele um das Zentrum des Netzes, wie dort im Theorieteil beschrieben ist. Zeitpunkt und Dauer der Rekonfiguration sowie Dauer der vorangehenden Sammelphase sind speziell zu beachten. Die Rekonfiguration

des Bogen-Netzes dauert hier jeweils einen Taktzyklus und je nach Paketlänge l sind $l-1$ Takte unmittelbar vor der Rekonfiguration Sammelphasen. Der Zeitpunkt der Rekonfiguration kann aus den Diagrammen anhand des Kurvenverlaufs des Durchsatzes abgelesen werden.

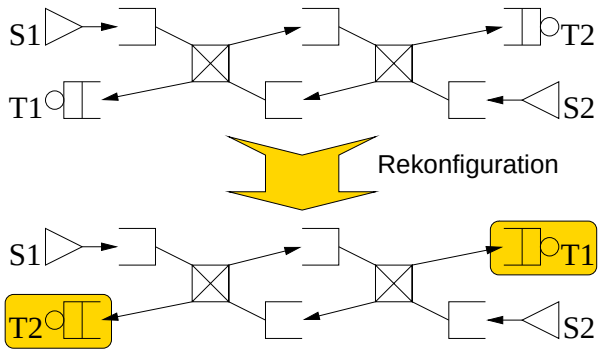


Abbildung 5.2: Topologie und Rekonfiguration des Bogen-Netzes

Durch die Rekonfiguration des Bogen-Netzwerkes wurde die grösstmögliche Anzahl von Irrwegpaketen erzeugt. Eine strapazierfähige Deadlock-Handhabung sollte solche Extremsituationen auch für grössere Netze bewältigen können. Das 8x8 BMIN und dessen Rekonfiguration, die schon in Abbildung 4.12 auf Seite 77 verwendet wurde, bietet sich dafür an. Abbildung 5.3 zeigt diese Rekonfiguration.

Die Aufgabe, CINSim auch für irreguläre Netze auf Rekonfiguration aufzurüsten, macht es notwendig, auch ein irreguläres Netz in der Validierung zu verwenden. Ein geeignetes Netz zu finden gestaltet sich schwierig. Dafür gibt es zwei Gründe:

1. Das verwendete Routingverfahren Shortest Path kann bidirektionale irreguläre Netze im allgemeinen nicht vermitteln, und zwar auch dann nicht, wenn keine Rekonfiguration stattfindet.

2. Eine sinnvolle Topologie eines Netzes, an dem die Veränderung des Durchsatzes und der Verzögerung nachvollziehbar ist, ist immer konstruiert und kann deshalb strenggenommen nicht irregulär genannt werden. Immer wenn eine bestimmte Überlegung beim Aufbau der Topologie mitgespielt hat, gibt diese auch die Regelmässigkeit der Topologie an.

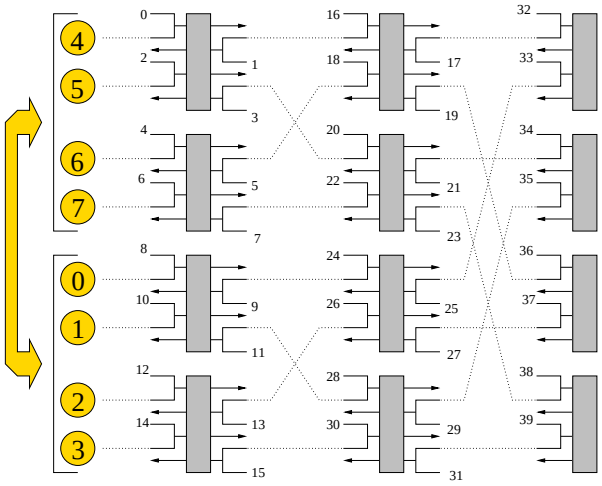


Abbildung 5.3: Topologie und Rekonfiguration des BMIN: Ziele 1 bis 4 werden mit 5 bis 8 vertauscht.

Schlichtes Fehlen von Symmetrieachsen oder Punktsymmetrien allein ist eine unzureichende Definition von Irregularität. Dennoch wird im Folgenden Irregularität als das Fehlen von Symmetrien in der Topologie definiert. Ein solches Netz entsteht, wenn in einem regulären Netz eine Leitung entfernt wird. Dabei muss entsprechend dem ersten Punkt ein unidirektionales Netz gewählt werden. Dieser als Rekonfiguration aufgefasste Ausfall einer Leitung wird in Abbildung 5.4 gezeigt und wird asymmetrische Rekonfiguration genannt.

Eine Schlussbemerkung: Das unsimulierbare Netz von Abbildung 4.7 ist ein irreguläres bidirektionales Netz. Aus diesem Grund kann es nicht zur Validierung verwendet werden. Insbesondere kann die Unmöglichkeit der korrekten Simulation des speziellen Rekonfigurationsfalls nich simulativ bestätigt werden.

5.2 Simulative Validierung

Mit den verschiedenen verwendbaren Verfahren ergibt sich ein vieldimensionaler Experiment-Raum, dessen Simulationsexmerimente alle auf ihre Korrektheit geprüft werden müssen. Flusskontrolle, Switching- und Routingverfahren, Paketlänge, angebotene Last und Verkehrscharakteristik sind variierbar für beliebige Topologien und Rekonfiguratio-

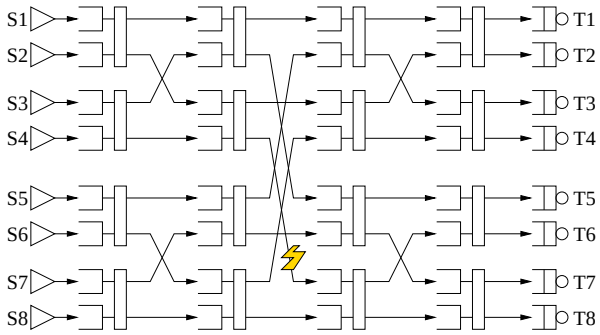


Abbildung 5.4: Topologie und Rekonfiguration eines unidirektionalen Netzes: Die asymmetrische Rekonfiguration besteht darin, dass eine Leitung getrennt wird, wodurch das das Netz irregulär wird.

nen. Offensichtlich ist es unmöglich, jede Möglichkeit einzeln zu testen und die korrekte Simulation durch CINSim festzustellen. Wenn jedoch eine repräsentative Auswahl von Experimenten getroffen wird, bei der jede Einstellmöglichkeit von CINSim mindestens einmal variiert und keine Fehler entdeckt werden, ist die Bedingung der Validierung erfüllt. Denn es wurde verlangt, dass eine möglichst intelligente Suche nach Experimente, die Fehler verursachen, erfolglos blieb. Werden zusätzlich Einstellmöglichkeiten vernachlässigt, die keinen Einfluss auf die Rekonfiguration haben können, wird die Menge der notwendigen Simulationen überschaubar. Tabelle 5.1 zeigt jede Einstellmöglichkeit von CINSim, bewertet deren Relevanz für die dynamische Rekonfiguration und gibt an, welche verschiedenen Einstellungen sinnvoll sind. Obwohl Routingverfahren eine grosse Bedeutung für die Rekonfiguration haben, sind davon nicht mehrere wählbar. Die anderen Parameter einer Simulation sind entweder nicht relevant für Rekonfigurationen oder werden variiert.

Es ergeben sich daraus zwölf Simulationen. Jede von ihnen wurde 10.000 mal durchgeführt. Welche Warnmeldungen bei jeder von ihnen aufgetreten sind, zeigt Tabelle 5.2. Die folgenden Unterkapitel diskutieren die Resultate für die beiden Netzwerktopologien.

5.2.1 8×8 BMIN-Rekonfiguration

Paketlänge 1 und LBP-Flusskontrolle ergeben jeweils in 10.000 Simulationen keine Fehlermeldung. Auch die Simulation der anderen Varianten erzeugt nie eine Fehlermeldung des Paketverlustes, aber auch nie die eines unvermittelbaren Paketes oder eines

Verfahren/ Variabilität	**Relevanz Rekonfiguration**	**sinnvolle Einstellungen**	**Variierungs-möglichkeiten**
Topologie	ja	8×8 BMIN, asymmetrisch	2
Flusskontrolle	ja	LBP, GBP	2
Switchingverfahren	ja	SAF, PCT	2
Routingverfahren	ja	Shortest Path	1
Scheduling	nein	zufällig	1
Paketlänge	ja	1 (nur SAF), 3 (SAF/PCT)	1,5
angebotene Last	nein	100%	1
Verkehrscharakteristik	nein	geometrisch	1

Tabelle 5.1: CINSim-Einstellmöglichkeiten für die simulative Validierung. Der spezielle Fall Routingverfahren hat zwar Rekonfigurations-Relevanz, doch kann nicht variiert werden, weil es nur ein Routingverfahren gibt. Alle anderen Verfahren sind entweder nicht Rekonfigurationsrelevant oder sind varrierbar.

Topologie	8×8 BMIN						irregulär					
Flusskontrolle	LBP			GBP			LBP			GBP		
Switchingverfahren	SAF		PCT	SAF		PCT	SAF		PCT	SAF		PCT
Paketlänge	1	3	3	1	3	3	1	3	3	1	3	3
Paketverlust	-	-	-	-	-	-	X	X	X	X	X	X
Unvermitteltes Paket							-	-	-	-	-	-
Unlösbarer Deadlock							-	-	-	-	-	-
Puffer-Überlauf					X	X						

Tabelle 5.2: CINSim-Simulationen und Fehlermeldungen: Ein „X“ bedeutet, dass die Fehlermeldung aufgetreten ist und ein „-“ kennzeichnet, dass eine Fehlermeldung theoretisch ausgeschlossen ist. Ein leeres Feld bedeutet, dass die mögliche Fehlermeldung bei 10.000 Simulationen ausgeblieben ist.

unlösbaren Deadlocks. Dies bestätigt für BMINs die Theorie der dynamischen Rekonfiguration, die unlösbare Deadlocks ausschliesst. Da jeder Puffer zu jedem Ziel senden kann, war die Fehlermeldung des Paketverlustes allerdings schon von Vornherein ausgeschlossen. Dass nie ein unvermitteltes Paket aus der ersten Konfiguration im rekonfigurierten Netz blieb, weist darauf hin, dass jeder auftretende Deadlock aufgelöst wird. Dass für GBP und Paketlänge grösser 1 Puffer-Überläufe auftreten, lässt auf Lücken in der Auflösungsmethode schliessen. Da GBP für Paketlänge 1 keine Fehler verursacht, liegt der Mangel des Auflösungsalgorithmus in der Behandlung von Paketlängen grösser 1. Der entstehende Simulationsfehler ist allerdings minimal, denn ein Flit, das von einem vollen Puffer empfangen wird, geht nicht verloren, wie das Ausbleiben der anderen Fehlermeldungen zeigt. In einem nächsten Zyklus wird der Füllstand des Puffers wieder auf seinen maximal erlaubten Wert zurückgehen, weil der Puffer keine weiteren Flits akzeptiert und vorderste Flits weitersendet. Dennoch ist die Korrektheit des Algorithmus für GBP und Paketlänge grösser 1 wiederlegt.

5.2.2 Asymmetrische Rekonfiguration

Es treten in allen Durchläufen Paketverluste auf. Dies ist für den Ausfall einer Leitung vorauszusehen, wie er in dieser Rekonfiguration simuliert wird. In der Tat ist es aber nur höchstens ein Paket pro Rekonfiguration, das verloren geht. Es ist anzunehmen, dass der Ausfall der einen Leitung kein Leck im Netz verursacht, es sind auch alle Sendemöglichkeiten noch vorhanden (jede Quelle kann noch zu jedem Ziel senden). Dies ist eine Bestätigung der richtigen Funktionsweise von CINSim bei Rekonfigurationen: Das eine Paket, das nach dem Ausfall der Leitung nicht mehr vermittelbar ist, wird gelöscht, so dass weiterer Paketverkehr ungehindert das Netz passieren kann. Die anderen drei Fehlermeldungen sind jeweils nicht aufgetreten. In dieser unidirektionalen Rekonfiguration ist dies auch nicht möglich, denn in unidirektionalen Netzen können Deadlocks nicht auftreten. Leider liefert CINSim für bidirektionale irreguläre Netze nicht die nötigen adaptiven Routingverfahren, bei denen Deadlocks zu erwarten sind. CINSim mit solchen Routinverfahren zu ergänzen und zu prüfen, ob die Rekonfiguration von bidirektionalen irregulären Netzen simulierbar wird, ist eine Aufgabe der weiteren Entwicklung. Belegt wurde im Rahmen dieser Master-Arbeit, dass CINSim Rekonfigurationen unidirektionaler Netze auch mit irregulären Topologien korrekt simuliert. Insbesondere ist für unidirektionale Netze auch PCT-Switching kombiniert mit GBP-Flusskontrolle simulierbar, was für BMINs widerlegt worden ist.

5.3 Qualitative Validierung

Für die qualitative Betrachtung muss aus der immensen Vielfalt von möglichen Simulationen wiederum eine Auswahl von sinnvollen Experimenten getroffen werden. Die Auswahl gliedert sich diesmal nach Merkmalen, die zu beobachten sind. Ein erster Abschnitt erklärt einführend Merkmale, die nicht mit der Rekonfiguration zusammenhängen. Weitere Abschnitte diskutieren verschiedene beobachtete Effekte, die nach den Rekonfigurationen auftreten und deren Begründungen im Rahmen des Möglichen liegen. Diese Validierung gibt keine zusätzliche Bestätigung der richtigen Funktionsweise von CINSim, sondern macht nur die Befunde von Unterkapitel 5.2 anschaulich. Die Simulationsresultate sind jeweils mit der Wahrscheinlichkeit von 0,99 höchstens 5% vom exakten Wert entfernt. Der Wert 0,99 wird Konfidenzintervall genannt und die 5% sind der relative Fehler. Ein simuliertes Resultat, das im Experiment eine Messung genannt wird, wird auch hier Messung genannt, um den Prozess des Simulierens vom Auswerten der Resultate zu unterscheiden.

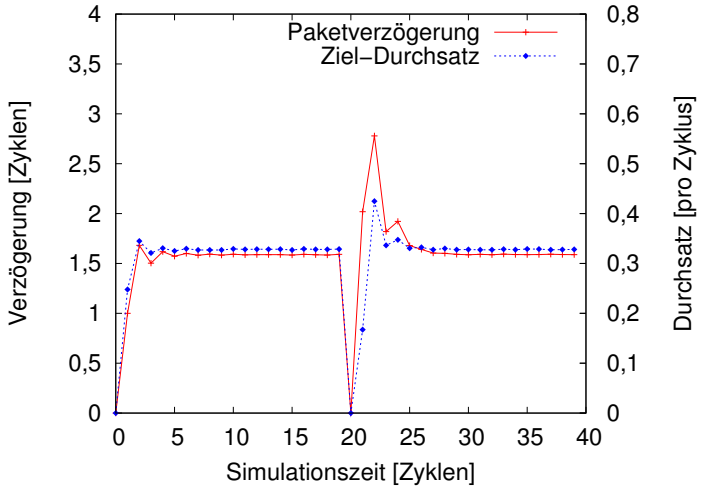
5.3.1 Nicht rekonfigurationsspezifische Merkmale

Um die rekonfigurationsspezifischen Merkmale einer Messreihe von Simulationen von den nicht rekonfigurationsspezifischen zu unterscheiden, muss voraus geschickt werden, welche Merkmale bei einer Messreihe nicht durch die Rekonfiguration bedingt sind. Abbildung 5.5 zeigt je zwei Messreihen von zwei Rekonfigurationen des Bogen-Netzwerks. Es sind Verzögerung und Ziel-Durchsatz für LBP und GBP mit Paketen der Länge 1, Switchingverfahren SAF und Quellenlast von 50%. Es gibt bei Paketlänge 1 keine Sammelphase und die Rekonfiguration im Takt 20 ist deutlich als Sendepause erkennbar. Die Rekonfigurations-unabhängigen Merkmale sollen nun diskutiert werden.

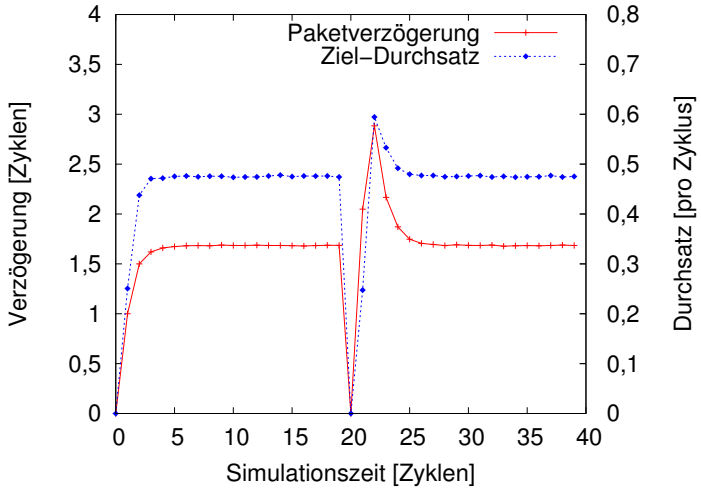
Glattere Kurvenformen von GBP verglichen mit LBP: Bei SAF verursachen Lücken in den Sendeketten Unregelmässigkeiten. Diese wirken sich nur bei Veränderungen des Verkehrs aus, also bei der Einschwingphase und der transienten Phase. GBP kennt keinen solchen Effekt bei Paketlänge 1.

Stationäre Verzögerung circa 1,5 Zyklen: Quellen können zu beiden Zielen versendet werden. Das eine Ziel geht über einen Puffer, das andere über zwei. Der Durchschnitt ist eineinhalb Puffer, der auch die stationäre Verzögerung erklärt.

GBP mit etwas hörerer stationärer Verzögerung: Da das Netz bei GBP etwas mehr gefüllt ist, kann es öfters passieren, dass Pakete das Scheduling für eine Übermittlung verlieren.



(a) LBP



(b) GBP

Abbildung 5.5: Bogen-Netz Resultate 1: SAF, Paketlänge 1, angebotene Last 50% für LBP und GBP. Anhand von diesen Simulationsvarianten können verschiedene Rekonfigurations-unabhängige und -spezifische Effekte gezeigt werden.

Stationärer Durchsatz ungefähr 0,33 bzw 0,5: Ein Ziel kann in jedem Taktzyklus ein Paket empfangen. Da bei GBP keine Lücken in den Paketketten vorhanden sind, ist der Durchsatz in diesem Fall ungefähr gleich der Last, 50%. Bei LBP muss beachtet werden, dass Ziele von beiden Quellen empfangen können. Die näher liegende Quelle ist nur einen Puffer entfernt, also entsteht hier nie eine Lücke. Die andere Quelle hingegen ist zwei Puffer entfernt und enthält Lücken, so dass von dort nur alle 2 Takte ein Paket herkommt. Das bedeutet, dass im Durchschnitt 1,5 mal weniger Pakete ankommen als bei GBP. Dies ergibt den Durchsatz von 0,33.

Vergleichbare Effekte treten auch für das 8x8 BMIN auf. Im Weiteren sind nur Effekte beschrieben, die bei der Rekonfiguration zu diesen und anderen Rekonfigurationsunabhängigen Merkmalen dazukommen.

5.3.2 Verzögerungs-Spitze in der transienten Phase

Dieser Effekt, der in beiden Diagrammen von Abbildung 5.5 zu sehen ist, lässt sich beim Bogen Netzwerk zahlenmässig abschätzen. Aus den Diagrammen ist ersichtlich, dass genau zwei Takte nach der Rekonfiguration die Verögerung einen Spitzenwert von etwa drei erreicht, und zwar bei beiden Simulationen (LBP und GBP). Die Anhebung um etwa 1,5 Zyklen ist dadurch zu erklären, dass einerseits Pakete einen Takt stillstanden und andererseits genau jedes zweite Paket einen Umweg nahm, was einen Zyklus mehr Zeit brauchte. Die durchschnittliche zusätzliche Verzögerung ist demnach 1,5 Zyklen. Im Diagramm ist ein etwas kleinerer Zahlenwert abzulesen. Der Grund sind Effekte des Scheduling und andere, die in diese Betrachtung nicht einbezogen werden können. Im Bogen-Netzwerk hat sich die qualitative Rechnung bestätigt, weil hier die Nebeneffekte klein sind.

5.3.3 Höherer Durchsatz kurz nach der Rekonfiguration

Es ist in Abbildung 5.5 zu beobachten, dass in der transienten Phase nach der Rekonfiguration ein erhöhter Durchsatz gemessen wird. Dieser Sachverhalt lässt sich mit einer einfachen Überlegung erklären. Die durchschnittliche Anzahl Pakete, die direkt nach der Rekonfiguration im Netz sind, ist dieselbe wie kurz vor der Rekonfiguration. Jedes dieser Pakete wird sein Ziel erreichen. Dass sich hingegen viele Pakete aufgrund von Umwegen verspäten, tut dabei nichts zur Sache. Das heisst, dass die Fläche der Durchsatz-Kurve nach der Rekonfiguration unabhängig von den Paketverspätung ist. Ein Stör-Effekt macht diese Regel ungenau: Wenn Pakete nach der Rekonfiguration Umwege gehen und deshalb länger im Netz bleiben, so wird auch das Einspeisen von

neuen Paketen in das Netz kurzzeitig mehr gehindert als kurz vor der Rekonfiguration. Dieser Effekt ist umso grösser, desto mehr Quellen-Last angelegt wurde. Zusammengefasst werden diese Sachverhalte in

Regel 9 *Der durchschnittliche Durchsatz ist in der transienten Phase kurz nach der Rekonfiguration ungefähr gleich dem Durchsatz kurz vor der Rekonfiguration. Diese Abschätzung stimmt umso genauer, desto kleiner die Quellen-Last ist.*

Wenn die transiente Phase den gleichen durchschnittlichen Durchsatz wie vor der Rekonfiguration haben soll und sich Pakete verspäten, dann treffen sie gehäuft ein paar (für Bogen-Netz zwei) Takte später ein, was den Durchsatz kurzzeitig erhöht. Dass bei kleiner Last dieser Sachverhalt als Flächen-Regel dargestellt werden kann, zeigt Abbildung 5.6.

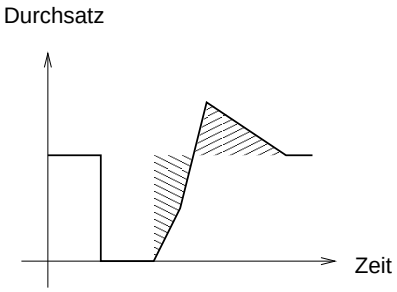
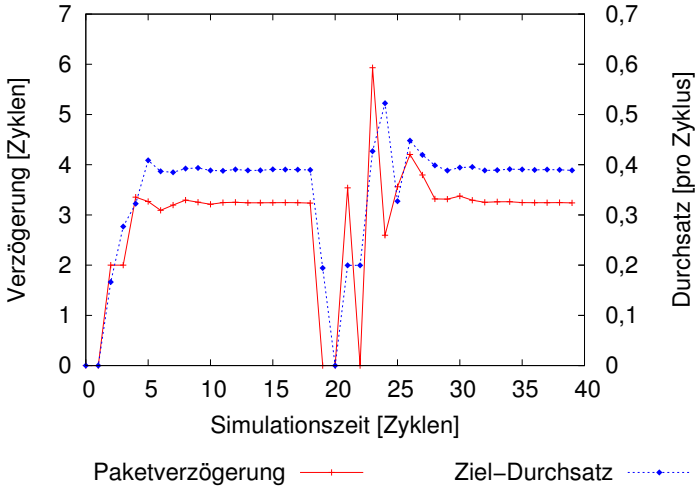
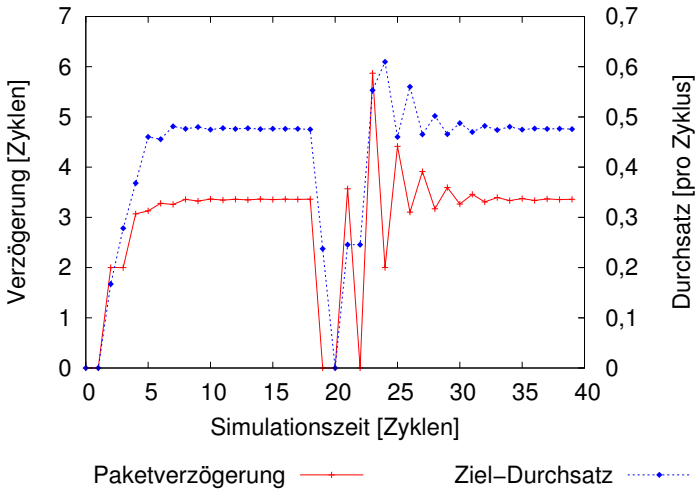


Abbildung 5.6: Flächenvergleich bei der Durchsatz-Kurve: Die dargestellten Flächen sind für kleine Quellen-Lasten gleich. Diese Bilanz betrifft eine transiente Phase nach einer Rekonfiguration.

Die Flächen-Regel gilt auch für grössere Paketlängen (Abbildung 5.7) und grössere Topologien (8x8 BMIN, Abbildung 5.9). Man beachte, dass bei höherer Quellen-Last die Flächenbilanz bei GBP negativ wird, doch bei LBP positiv, d.h. grössere Durchsatz-Fläche in der transienten Phase als vor der Rekonfiguration (Abbildung 5.8). Der Grund für die positive Flächenbilanz ist der, dass bei hoher Quellen-Last bei LBP nach der Rekonfiguration oft Lücken in Paketketten durch eingespeiste Pakete aufgefüllt werden und der Paketbestand dadurch kurzzeitig höher wird. Dieser Effekt kann den gegenteilige Effekt der Erschwernis, neue Pakete einzuspeisen, überwiegen. Bei GBP hingegen tritt er nicht auf, was die negative Flächenbilanz erklärt.

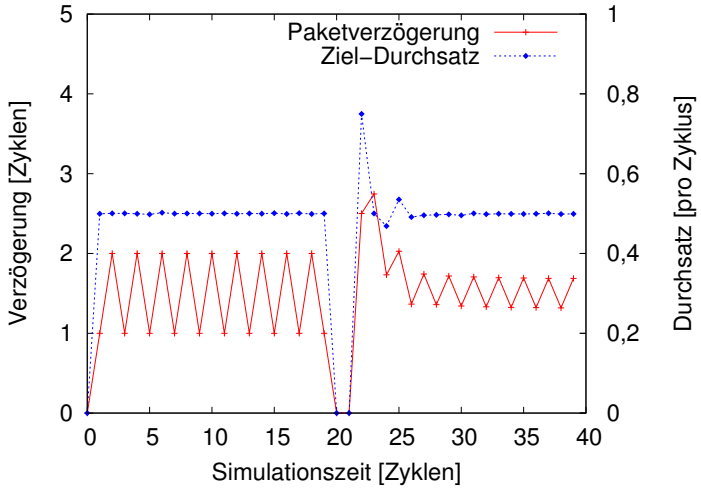


(a) LBP

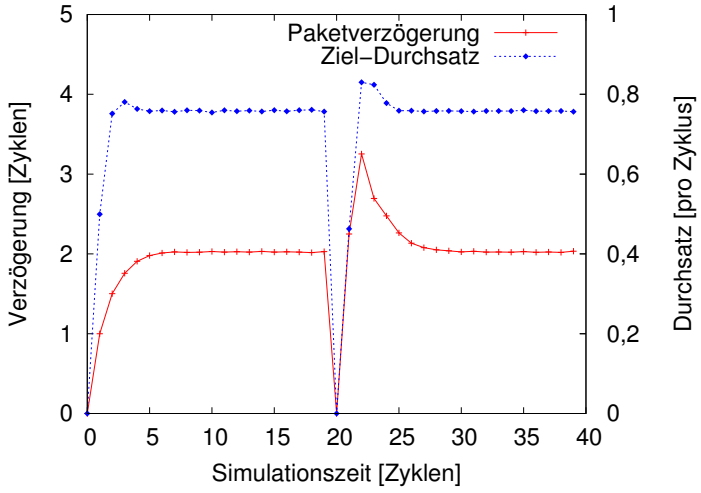


(b) GBP

Abbildung 5.7: Bogen-Netz Resultate 2: SAF, Paketlänge 2, angebotene Last 50%. Die Flächenbilanz des Durchsatzes nach der Rekonfiguration stimmt bei GBP udn LBP für diese Variante ungefähr.



(a) LBP



(b) GBP

Abbildung 5.8: Bogen-Netz Resultate 3: SAF, Paketlänge 1, angebotene Last 100%. Diese Diagramme illustrieren, dass bei Last 100% die Flächenbilanz nicht mehr aufgeht. GBP hat weniger durchschnittlichen Durchsatz in der transienten Phase als sonst, LBP mehr.

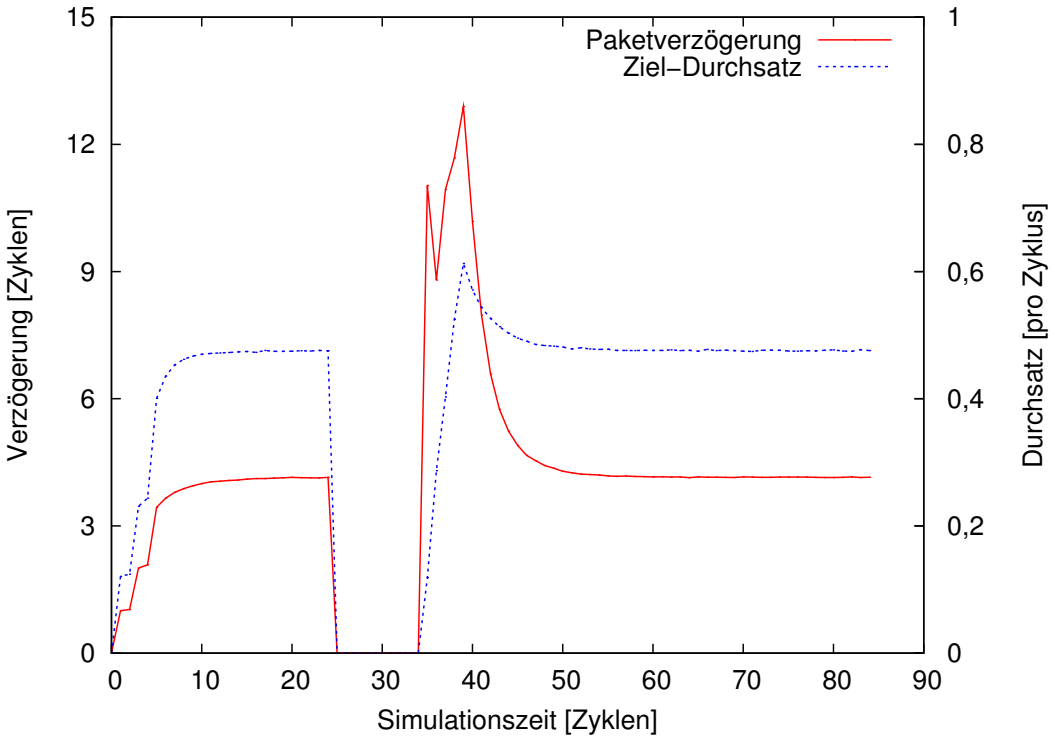


Abbildung 5.9: BMIN Resultate 1: SAF, Paketlänge 1, angebotene Last 50%. Auch beim 8x8 BMIN beobachtet man für diese Variante (GBP) die ungefähre Einhaltung der Flächenregel.

5.3.4 Spezialfall SAF, GBP und Quellen-Last 100%

Eine weiteres Beispiel, wo CINSim dynamische Rekonfigurationen nicht anders simuliert als es theoretisch verlangt wird, ist dieser Spezialfall. Zur Demonstration wurde hier Paketlänge 2 gewählt. Vor der Rekonfiguration ist dabei zwingend, dass nur in jedem zweiten Zyklus ein Paket ein Ziel erreicht. In der Verzögerungs-Messung ist dies daraus ersichtlich, dass in jedem anderen Zyklus kein Verzögerungs-Wert gemessen wurde. Gleichzeitig hat der Durchsatz immer zwei nacheinander folgende gleiche Werte, weil für jedes ankommende Kopf-Flit im nächsten Takt das Schwanz-Flit desselben Pakets ankommt. Da die Quellen eine Last von 100% anlegen, ergeben sich nie Lücken in den Sende-Ketten. Auch nach der Rekonfiguration muss dieser Effekt auftreten, und tut es auch. Er ist ein Hinweis darauf, dass Irrwegpakete und Deadlocks immer im ersten Takt nach der Rekonfiguration umgeleitet und aufgelöst wurden und dass nie ein Paket verloren ging. Andernfalls hätten sich Lücken in den Paketketten ergeben und der Effekt würde nach der Rekonfiguration nicht mehr auftreten.

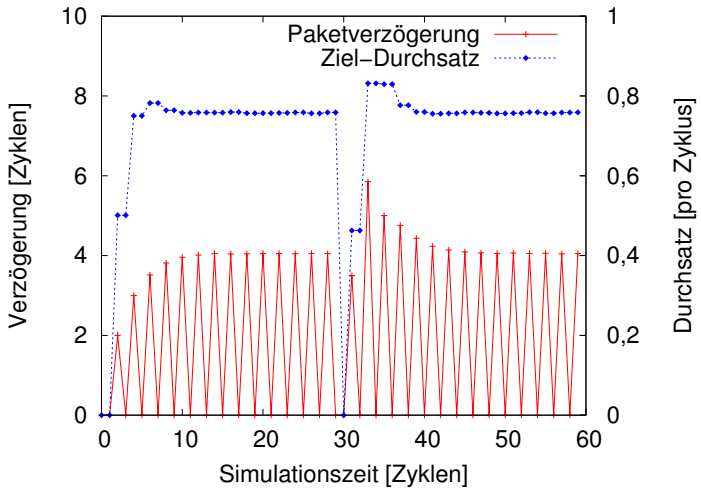


Abbildung 5.10: Bogen-Netz Resultate 4: SAF, Paketlänge 2, angebotene Last 100%. Diese GBP-Variante zeigt den Effekt, dass nur jeden zweiten Takt ein Paket eintrifft, auch nach der Rekonfiguration. Die volle Last bedingt, dass nie eine Lücke in einer Paketkette entsteht. Die Puffer müssen jedoch jeden zweiten Zyklus auf ihre hinteren Paketteile warten.

Derselbe Effekt tritt auch bei einem grösseren Netz auf. Hier wird er für das 8x8 BMIN gezeigt. Er ist ein Hinweis, dass die Algorithmen des rekonfigurationsfesten Routing hier auch in einem relativ grossen Netzwerk die Pakete korrekt vermitteln.

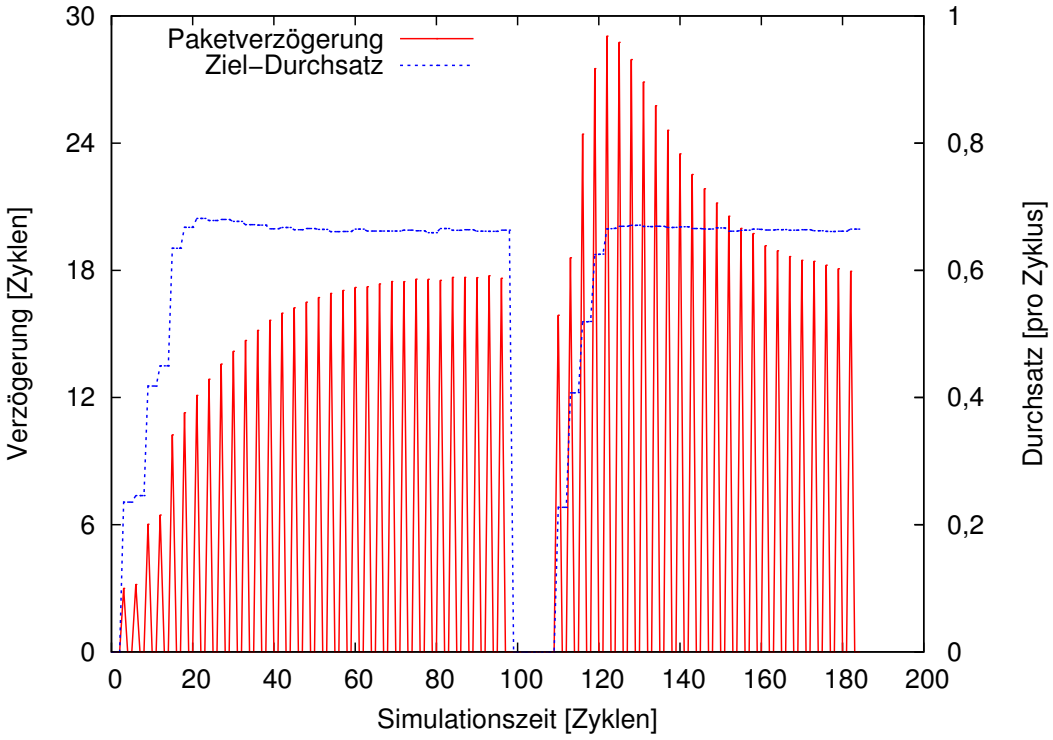


Abbildung 5.11: BMIN Resultate 2: SAF, Paketlänge 3, angebotene Last 100% (GBP). Pakete treffen, da sie Länge 3 haben, nur jeden dritten Takt ein, auch nach der Rekonfiguration, was dessen korrekte Funktion für dieses Beispiel belegt.

5.3.5 Zusätzliche nachvollziehbare Effekte

Abschliessend werden zwei weitere Effekte beschrieben und nachvollzogen, die in den gezeigten Diagrammen zu erkennen sind.

Den einen zeigt Abbildung 5.7 (b): Während bei der Startphase keine markanten Schwankungen der Verzögerung gemessen wurden, so schwingt sie nach der Rekonfiguration mit einer Auslenkung von bis zu sechs Zyklen. Diese Unregelmässigkeit kommt

daher, dass nur Pakete aus der ersten Konfiguration hohe Verzögerungen haben. Sie kommen aber für SAF und GBP nur jeden zweiten Zyklus bei Zielen an. Der Grund ist, dass bei GBP keine Lücken in den Sendeketten auftreten, doch bei SAF jedes Paket auf seinen Schwanz warten muss, bevor es weiter gesendet wird. Die Übertragung eines Pakets von einem zum nächsten Puffer dauert deshalb immer genau zwei Zyklen. Da mit Last 50% neue Pakete in jedem Takt generiert werden können, gleichen sie allmählich diese Unregelmässigkeit aus. In den geradzahligen Zyklen kommen ausschliesslich neue Pakete bei Zielen an und der wachsende Durchschnitt der Verzögerung wird nur von solchen gemessen. Dass der Anstieg in etwa gleich ist wie bei der Startphase, wird daraus klar. Insgesamt wurde die Verzögerung, die sich nach der Rekonfiguration wie eine gedämpfte Schwingung verhält, ausreichend erklärt und die richtige Simulation der Vermittlung in diesem Fall dadurch bestätigt.

In Abbildung 5.8 wird gezeigt, dass beim Bogen-Netz für Paketlänge 1, SAF und Quellen-Last 100% vor der Rekonfiguration die Verzögerungen in den ungeraden Zyklen eins sind und in den geraden zwei. Dies ist dadurch nachvollziehbar, dass stets die Quellen nur in den geraden Zyklen senden können, weil in den ungeraden Zyklen in den Puffern vor ihnen Pakete liegen. Das bedeutet aber auch, dass nur in den geraden Zyklen Pakete in den mittleren Puffern liegen können. Es gibt also nie Scheduling-Konflikte. Dadurch entsteht die gezeigte streng monotone Verzögerungs-Charakteristik. Eine Rekonfiguration kann nur Irrwegpakete erzeugen, wenn sie in einem geraden Zyklus stattfindet, wie es hier auch der Fall ist. Dann gehen die Pakete ihre Rückwege, es kann Scheduling-Konflikte geben und der streng monotone Wechsel von Paketen mit Verzögerung eins beziehungsweise zwei wird beeinträchtigt, so dass die Verzögerungs-Schwankung nach der transienten Phase der Rekonfiguration schwächer ist. Findet die Rekonfiguration in einem ungeraden Zyklus statt, kann sie keine Auswirkungen haben. Dies wird in Abbildung 5.12 simulativ bestätigt. Der Dämpfungs-Effekt tritt in der Simulation genau so auf, wie er theoretisch zu erwarten ist.

5.3.6 Asymmentrische Rekonfiguration

Abbildung 5.13 zeigt die Simulationsresultate der Rekonfiguration, die in Unterkapitel 5.1.3 definiert wurde. Eine Verbindungsleitung eines unidirektionales Netzes mit redundanten Pfaden für alle Quellen-Ziel-Kombinationen fällt aus, wodurch zwar immer noch alle Quellen allen Zielen senden können, aber mit einem durchschnittlich etwas kleineren Durchsatz. Da die Halbierungsbandbreite von acht auf sieben verringert wurde, ist eine Schätzung der Senkung um 12,5% gegeben. Der simulierte Wert ist 11,7%. Die Verzögerung, die mit dem Durchsatz gekoppelt ist, senkt sich ebenfalls leicht um den simulierten Betrag von 0,39 Zyklen. Der Ausfall der Verbindungsleitung wird hier

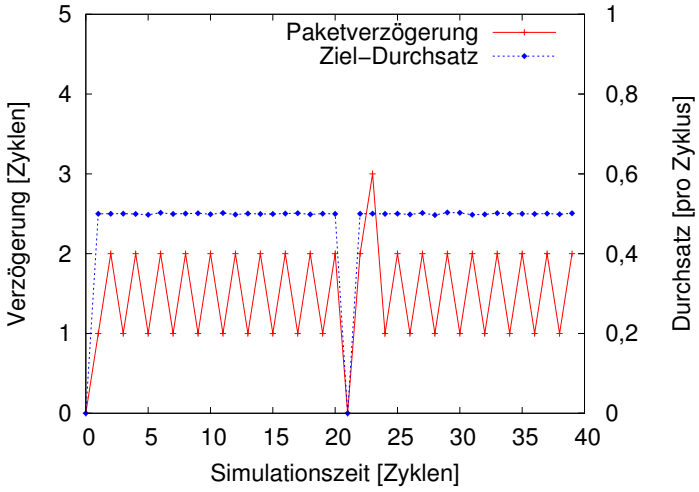


Abbildung 5.12: Alternatives Bogen-Resultat 3: SAF, Paketlänge 1, angebotene Last 100% (LBP). Der Effekt, der in Abbildung 5.8 (a) nach der Rekonfiguration eine Dämpfung der Verzögerungs-Schwankung verursachte, tritt hier nicht auf, wie es theoretisch vorausgesagt wurde. Grund: Es entstehen keine Irrwegpakete. Die Pakete, die zur Zeit der Rekonfiguration im Netz sind, haben zwar eine zusätzliche Verzögerung von eins, aber es sind keine Unregelmässigkeiten in der Paketbewegung aufgetreten, was daraus zu schliessen ist, dass der Durchsatz auch nach der Rekonfiguration konstant 0,5 beträgt.

als Korosionsprozess gedacht, der erst nach einer gewissen Zeit soweit fortgeschritten ist, dass keine Pakete mehr über die korrodierte Verbindung vermittelt werden können. Die Rekonfiguration stellt eine präventive Deaktivierung der Verbindung dar, wozu das Netz 10 Zyklen lang stillstehen muss. In dieser Grafik sind die wichtigsten Effekte der dynamischen Rekonfiguration mit PCT-Switching und GBP-Flusskontrolle zu beobachten. Die folgende Aufzählung erklärt Aspekte des Verhaltens von Durchsatz- und Verzögerung bei dieser Rekonfiguration.

Verzögerungsanstieg nach der Rekonfiguration Der Anstieg beträgt 10,1 Zyklen, wenn die Verzögerung der ersten sechs Zyklen nach der Rekonfiguration gemittelt und die Differenz zum stationären Wert vor der Rekonfiguration berechnet wird. Zehn Zyklen der Rekonfiguration und zwei Zyklen Sammelphase, die durchschnittlich jedes zweite Paket betreffen, ergeben einen geschätzten Wert von elf Zyklen. Die Schätzung ist relativ genau, weil Nebeneffekte von Irrwegpaketen oder Deadlocks hier nicht auftreten.

Rippel in beiden Kurven nach der Rekonfiguration Der Verzögerungs-Rippel in der Startphase der ersten Konfiguration ist dadurch zu erklären, dass im ersten Zyklus jede Quelle ein Paket zu senden beginnt. Diese Gleichzeitigkeit wird im Verlauf der Zeit in der ersten Phase wegen auftretenden Scheduling- und Switching-Konflikten verschwinden. Switching-Konflikte sind bisher bei SAF nicht aufgetreten, da bei SAF Paketköpfe immer synchron versendet werden, was bei PCT nicht der Fall ist. Ein Paketkopf wird bei PCT nur so lange gestoppt, bis einer der gewünschten Ausgänge wieder frei ist. Dies kann entweder einen, zwei oder drei Zyklen dauern bei Paketlänge 3. Die Sammelphase bewirkt, dass die Puffer wieder synchron Köpfe senden. Dies verursacht denselben Effekt wie in der Startphase. Allmählich wird der Effekt wegen Konfliktsituationen wieder abgedämpft.

5.4 Abschliessender Befund der Validierung

Am Anfang der Entwicklung von CINSim stand das Ziel, ein Simulationswerkzeug zu entwickeln, das dynamische Rekonfigurationen von verschiedene Topologien mit Routing-, Switching-, Scheduling-, Flusskontrolle-Verfahren und vielen Einstellmöglichkeiten simulieren kann. Hat die vorliegende Arbeit die Anforderungen des bis anhin unerreichten Ziels erfüllt? Diese Frage muss verneint werden. Die Validierung bestätigt zwar die Funktionsfähigkeit von CINSim, einige Spezialfälle von dynamischen Rekonfigurationen korrekt simulieren zu können. Für den allgemeinen Fall kann aber keine

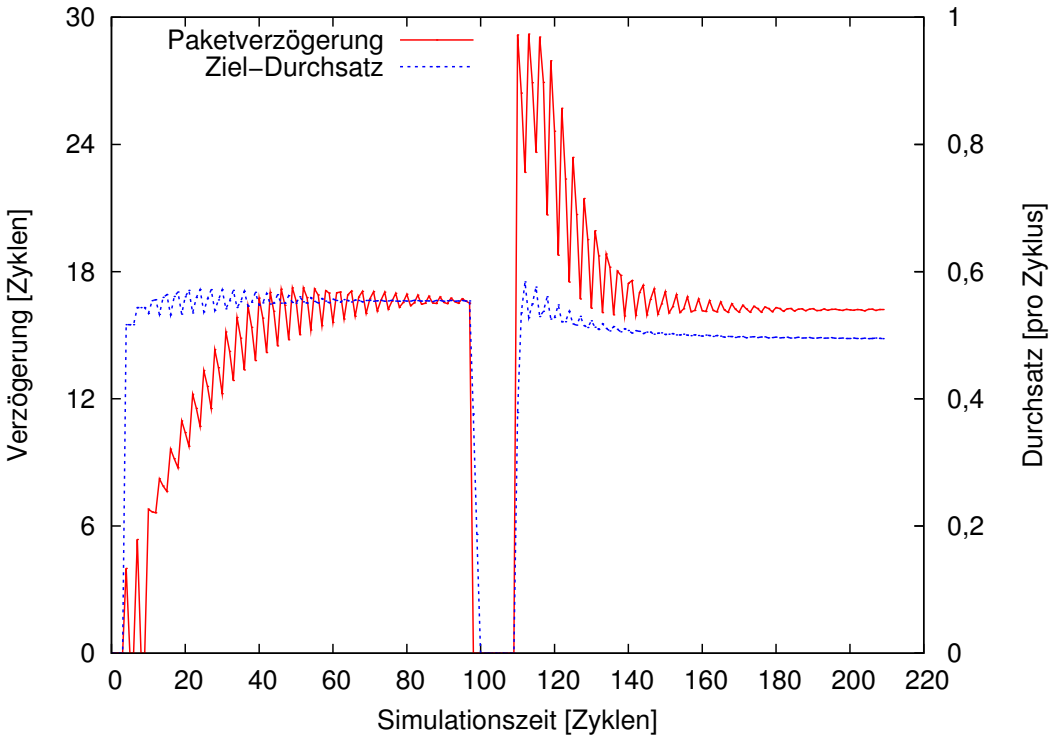


Abbildung 5.13: Rekonfiguration mit GBP und PCT-Switching mit Paketlänge 3 und angebotener Last 100%: Im unidirektionalen Netzwerk wird eine korrodierte Leitung entfernt. Diese Rekonfiguration bewirkt eine Einbusse des Durchsatzes von etwa $\frac{1}{8}$, wie es zu erwarten ist.

Bestätigung gegeben werden. Dazu ist die Anzahl verschiedener Netztopologien, die in der Validierung verwendet wurden, zu klein. Die bestätigten Spezialfälle sind unidirektionale Netze sowie bidirektionale MINs. Dass für die BMINs auch GBP mit Paketlänge 1 korrekt simuliert werden kann, ist eine Bestätigung, dass der Hauptteil der investierten Entwicklungsarbeit erfolgreich war. Dies betrifft die Deadlock-Analyse, Deadlock-Auflösung und Bezugspuffer-Suche für Netze mit Paketlänge 1. Es wurden Schwierigkeiten bei der Behebung von Deadlocks festgestellt, die mit Paketen der Länge grösser 1 entstanden sind. Weitere Arbeit muss diese Schwierigkeiten zu beheben suchen.

Kapitel 6

Ergebnisse

In diesem Kapitel wird die Leistungssteigerung von statischer zu dynamischer Rekonfiguration diskutiert, welche die Motivation für dynamische Rekonfiguration darstellt. Das im Unterkapitel 1.2.2 zitierte Entwicklungsziel von CINSim nicht vollständig erreicht. Dennoch zeigte schon die Validierung eine Menge von Beispielen von dynamischen Rekonfigurationen, die CINSim simulieren kann, ohne dass ein Fehler festgestellt werden kann. Ebenfalls ermöglicht die implementierte Erweiterung auf Simulationen dynamischer Rekonfigurationen erstmals, eine Abschätzung der Leistungssteigerung von statischer zu dynamischer Rekonfiguration zu machen, die seit Anbeginn der Forschungsarbeiten auf diesem Gebiet angestrebt worden ist. Eine Optimierung der Kommunikationsleistung kann dadurch erreicht werden, dass die Lokalität von intensiv kommunizierenden Endknoten eines Netzes erhöht wird und demgegenüber Endknoten, die selten oder nie mit einander kommunizieren, netzwerktopologisch gesehen weiter von einander entfernt werden. Die Optimierung konnte für statische Rekonfiguration mit CINSim schon nachgewiesen werden. [4] bespricht diese Leistungsverbesserung im Detail. Der letzte Abschnitt, „Further Work“ weist darauf hin, dass eine dynamische Rekonfiguration die gleiche Leistungssteigerung mit einer kürzeren Übergangsphase erreichen könnte. Ob dem so ist, kann nun überprüft werden. Dazu wird dasselbe Netz gewählt wie in [4]. Die Rekonfiguration ist jedoch etwas komplizierter, um die Effekte zu verstärken, die zu beobachten sind. Sie ist in Abbildung 6.1 zu sehen.

In Abbildung 6.2 sind durchschnittlicher Durchsatz und Paketverzögerung für die statische Rekonfiguration gezeigt. Nach 30 Zyklen wurde eine Leerlaufphase von 20 Zyklen, wie sie in der statischen Rekonfiguration nötig ist, eingeschaltet (Theoretisch müsste die Leerlaufphase sogar mindestens 42 Zyklen dauern, denn dies ist die benötigte Zeit, um das Netz mit der speziellen Kommunikation zu entleeren, in der alle Endknoten der Gruppe A stets dem gleichen Endknoten senden. Das Endresultat des Leistungsver-

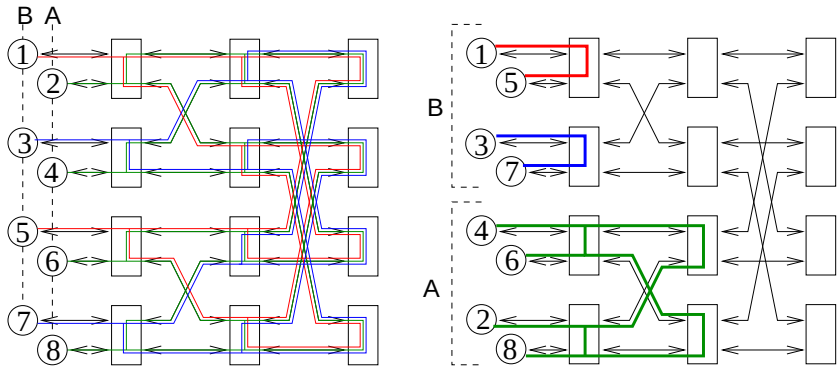


Abbildung 6.1: BMIN-Rekonfiguration mit spezieller Verkehrsverteilung: Die Endknoten 1 und 5 kommunizieren ausschliesslich unter einander, ebenfalls die Endknoten 3 und 7. Die Endknoten 2, 4, 6 und 8 kommunizieren auch nur untereinander. In der ersten Konfiguration entstehen lange Wege für die Pakete und auch Vermittlungskonflikte, die in der zweiten Konfiguration optimiert beziehungsweise beseitigt werden.

gleichs zwischen statischer und dynamischer Rekonfiguration würde also noch eindeutiger ausfallen). Nach der Leerlaufphase sind keine Pakete mehr im Netz. Es folgen 10 Zyklen Rekonfiguration, worauf die neue Konfiguration gestartet wird. Sehr bald haben sich die Niveaus für Durchsatz und Paketverzögerung verbessert gegenüber der ersten Konfiguration. Es wurde dabei die Paketlänge 1 verwendet und das Flusskontrolle-Verfahren GBP.

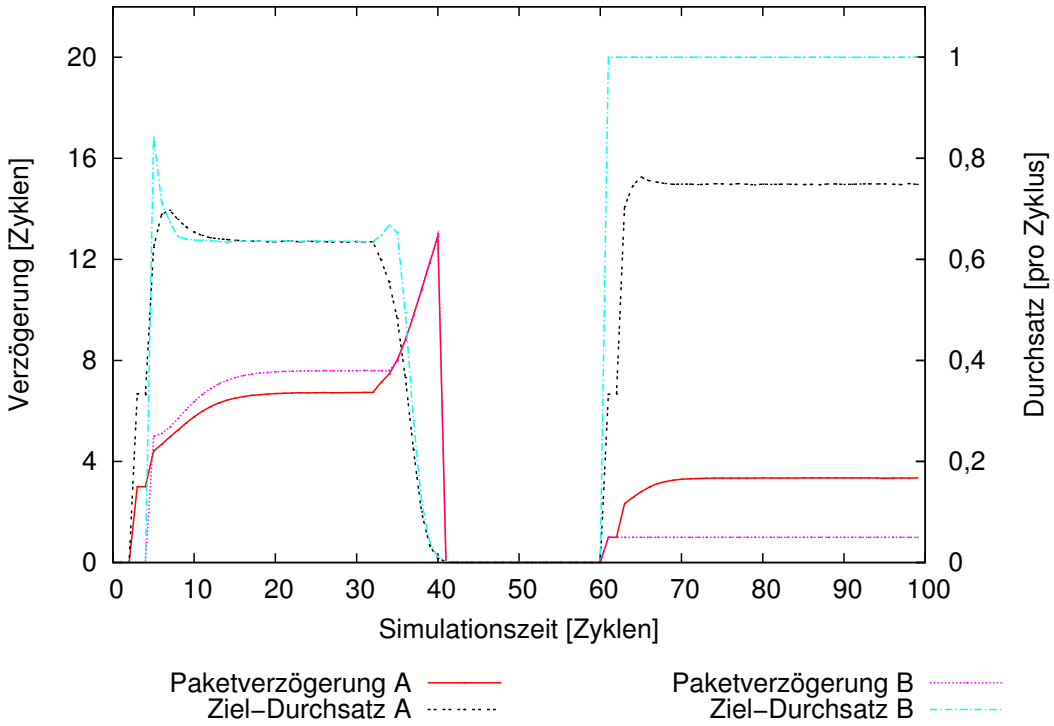


Abbildung 6.2: Leistungssteigerung mit *statischer* Rekonfiguration: SAF, Paketlänge 1, angebotene Last 100% mit GBP-Flusskontrolle. Die Charakteristiken A zeigen Durchsatz und Verzögerung für die Endknoten 2, 4, 6 und 8, die sich untereinander Pakete senden. Die übrigen Endknoten, bei denen je zwei mit einander kommunizieren, haben alle die mit B zusammengefassten Charakteristiken.

Bei der dynamischen Rekonfiguration derselben Situation kann die Leerlaufphase weggelassen werden und eine Sammelphase ist auch nicht nötig, da die Pakete die Länge 1 haben. Wiederum wird GBP verwendet. Die Simulationsresultate zeigt Abbildung 6.3

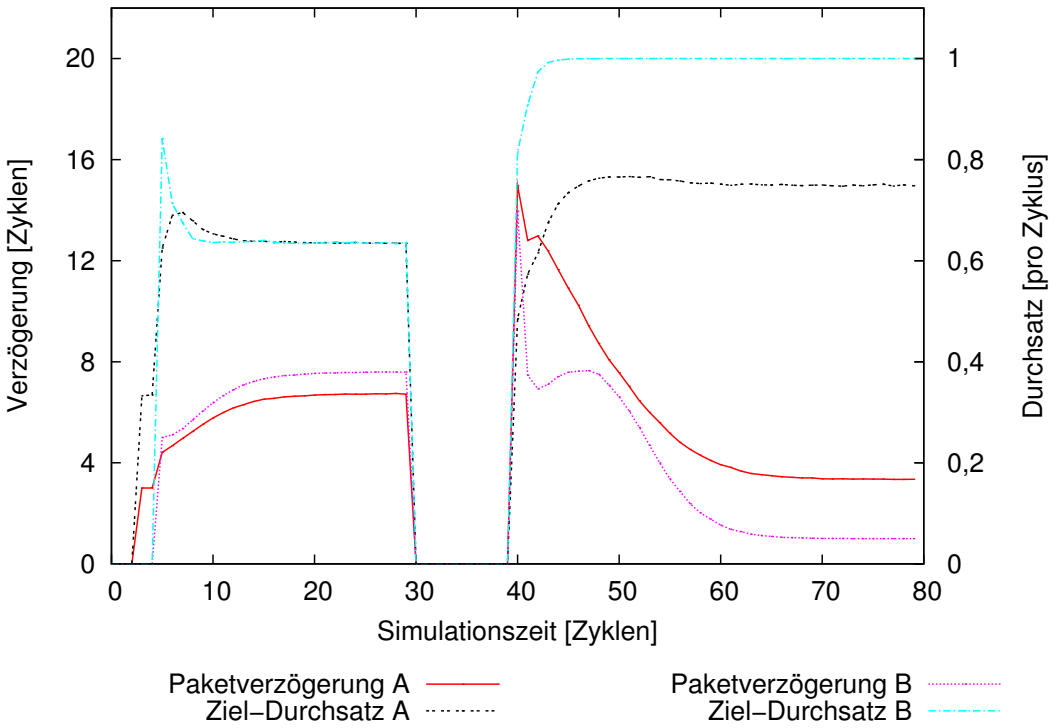


Abbildung 6.3: Leistungssteigerung mit *dynamischer* Rekonfiguration: gleiche Simulationsvariante wie in Abbildung 6.2. Im Vergleich zur statischen Rekonfiguration konnten 20 Zyklen Leerlaufphase weggelassen werden. Die besseren Durchsatz- und Verzögerungs-Niveaus werden aber nach der Rekonfiguration weniger schnell erreicht als bei der statischen Rekonfiguration. Der Grund sind viele Irrwegpakete, die zum Teil erst viele Takte nach der Rekonfiguration bei ihren Zielen eintreffen.

Um die Leistungssteigerung von statischer zu dynamischer Rekonfiguration bewerten zu können, müssen die Simulationen zuvor interpretiert werden. Es ist vor allem wichtig, die Simulationsresultate der dynamischen Rekonfiguration zu verstehen. Die Verzögerung wird erst zehn Zyklen nach der abgeschlossenen Rekonfiguration kürzer. Vor dem zehnten Zyklus jedoch bewirkt die dynamische Rekonfiguration eine Verschlechterung. Die Spitze kurz nach der Rekonfiguration verursachen die Pakete, deren Ziele noch am gleichen Ort sind wie vorher und die keine Irrwegpakete geworden sind durch die Rekonfiguration. Sie haben durchschnittlich genau zehn Takte mehr Verzögerung als die Pakete, die kurz vor der Rekonfiguration angekommen sind. Diese Pakete machen ungefähr die Hälfte des Paketverkehrs kurz nach der Rekonfiguration aus. Die andere Hälfte sind Irrwegpakete, die nach und nach bei ihren Ziele eintreffen, die nun über andere Wege zu erreichen sind. Dieser Anteil der Pakete verursacht die Anhebung der durchschnittlichen Verzögerung, die im Zyklus 48 für die unter B zusammengefassten Ziele am höchsten ist. Die Verzögerung seit dem Ende der ersten Konfiguration beträgt für diese Irrwegpakete 18 $(48-30)$. Diese haben am Ende der ersten Konfiguration schon eine durchschnittliche Verzögerung von 4 gehabt, der Hälfte der gemessenen Verzögerung bei den Zielen, woraus sich ihre gesamte Verzögerung von $18 + 4 = 22$ ergibt. Dass die durchschnittliche Verzögerung im Takt 48 trotzdem nur etwa Acht beträgt, lässt darauf schliessen, dass nur etwa ein Drittel der ankommenden Pakete noch Irrwegpakete sind. Die Mehrheit sind neue Pakete, die schon die optimalen Wege gegangen sind. Der Durchsatz pendelt sich innerhalb von ungefähr fünf Zyklen auf dem optimalen Niveau ein.

Verglichen mit den Simulationsresultaten der statischen Rekonfiguration zeigt die dynamische eine schnellere Leistungsverbesserung. Während die statische Rekonfiguration erst im Zyklus 60 die besseren Durchsatz- und Verzögerungsdurchschnitte aufweist, so erreicht die dynamische schon zehn Takte vorher eine Leistungsverbesserung, die sich bis zum Zyklus 60 ganz ausbaut. Wird beachtet, dass die Leerlaufphase bei statischer Rekonfiguration viel länger dauern würde, wenn Paketverluste garantiert vermieden werden sollen, so wird der Vorteil von dynamischer Rekonfiguration deutlich. Das letzte Kapitel wird die gesamte Master-Arbeit zusammenfassen und anschliessend auf weiterführende Arbeit hinweisen.

Kapitel 7

Zusammenfassung und Ausblick

In diesem Kapitel wird die Zusammenfassung der Master-Arbeit in zwei Sprachen gegeben, in Deutsch und in Englisch. Zum Schluss folgt der Ausblick auf weiterfürende Entwicklungen an CINSim.

7.1 Zusammenfassung

[3], [2] und [4] haben von der Absicht gesprochen, ein Simulationstool zu entwickeln, das dynamische Rekonfigurationen von beliebigen Verbindungsnetzen simulieren kann. CINSim wurde soweit entwickelt, dass es Netzwerke mit verschiedensten Verfahren simulieren konnte, dazu auch statische Rekonfigurationen. Nach einer kurzen Einführung (Kapitel 1) werden die Grundlagen der Netzwerktheorie mit den relevanten Bereichen Topologie, Switching-, Routing- und Schedulingverfahren sowie Multicast und Flusskontrolle im Kapitel 2 besprochen. Das Unterkapitel 1.2 bespricht anschliessend die speziellen Aspekte für statische Rekonfiguration. Es wird betont, dass aus der grossen Vielfalt von möglichen Verfahren CINSim jeweils nur eine kleine Auswahl unterstützt. Die vielfältigsten Auswahlmöglichkeiten bietet die Paketgenerierung mit detailierten Einstellmöglichkeiten der Verkehrscharakteristik, des Multicast, der Paketziel-Verteilung und weiteren. In diesem 2. Kapitel wird im dritten Unterkapitel die Definition von statischer und dynamischer Rekonfiguration gegeben, womit die Aufgabe der Arbeit klar abgesteckt wird, die darin besteht, CINSim für dynamische Rekonfigurationen zu erweitern.

Das Kapitel 3 enthält die theoretischen Grundlagen der dynamischen Rekonfiguration, die teils aus vorhandener Literatur genommen sind und teils neu entwickelt werden mussten. Ein erstes Unterkapitel leitet die Kriterien einer verlustfreien dynami-

schen Rekonfiguration her, die aus den Zielmengen der Pakete und der Vermittlungs-Informationen der Router abzuleiten sind. Die Frage, unter welchen Bedingungen eine dynamische Rekonfiguration stattfinden kann, ist damit beantwortet. Ein zweites Unterkapitel stellt anschliessend fest, dass nach einer dynamischen Rekonfiguration Deadlocks auftreten können, wie [8] erwähnte, und definiert diese eindeutig. Eine Regel wird aufgestellt, wie verirrte Pakete, die einen solchen auslösen, gefunden werden, und das Konzept der Abhängigkeitsbäume für Puffer von verirrten Paketen wird vorgestellt. Diese Strukturen erlauben es, die Deadlock-Situation formal zu erfassen und ein Routing anzugeben, das den Deadlock löst. Dazu müssen die Abhängigkeitsbäume in verschiedene Klassen eingeteilt werden, von denen nicht alle an der Lösung des Deadlocks teilnehmen können. Die Deadlock-Situation wird von einem Puffer her dann als auflösbar erkannt, wenn der dazugehörige Abhängigkeitsbaum einen Pfad enthält, der genau eine Puffernummer zweimal enthält, und zwar in der Wurzel und in einem Blatt. Durch dieses Analyse- und Auflösungsverfahren werden alle theoretisch lösbaren Deadlocks zusammengefasst. Für Multicast und Wormhole-Switching lässt sich zeigen, dass das Verfahren nicht auf eine Lösung führt. Deshalb werden diese Optionen für eine dynamische Rekonfiguration verboten.

Im Kapitel 4 wird erklärt, wie die theoretischen Konzepte in CINSim eingefügt worden sind. Als Erstes werden die Kriterien einer dynamischen Rekonfiguration auf das bestehende Bitmasken-Adressierungskonzept von CINSim angewendet. Wenn die Kriterien nicht erfüllt sind, werden Pakete gelöscht und eine Warnung wird ausgegeben. Die Simulation des Netzes kann aber in jedem Fall weiterlaufen. Ein zweites Unterkapitel bespricht die dabei erfolgende Paket-Adressen-Übersetzung und die Deadlock-Handhabung vorerst nur für die lokale-Rückstau-Flusskontrolle. Paketlängen grösser 1 benötigen zudem vor der Rekonfiguration eine Sammelphase der Pakete, die sie in einzelne Puffer packt, um das Auftrennen von Paketen, bestehend aus einzelnen Flits, zu verhindern. Die Simulation von Netzen mit globalem-Rückstau benötigt ein neues Paketbewegungsverfahren, das im dritten Unterkapitel eingeführt wird. Während ohne eine Rekonfiguration sich jedes Paket auf ein Ziel hin orientiert und somit Ziele erste Empfänger von Paketen pro Zyklus sein können, so muss im Deadlock-Fall die gegenseitige Orientierung der verklemmten Pakete zur Paketbewegung dienen. Die letzten beiden Unterkapitel der Implementierung geben eine Erweiterung für die Switchingverfahren Virtual- und Partial-Cut-Through und die Zusammenfassung der neuen Bestandteile von CINSim.

Inwiefern die Simulationsresultate des für dynamische Rekonfigurationen ausgerüsteten CINSim glaubwürdig und brauchbar sind, diskutiert Kapitel 5. Die Vergleichsdaten für die Validierung sind gering. Dennoch lässt sich belegen, dass die Rekonfiguration verlustfrei abläuft, wenn die aufgestellten Kriterien erfüllt sind. Die Rekonfiguration verläuft auch bei bidirektionalen Netzen mit Paketlänge grösser 1 und globalem

Rückstauverfahren verlustfrei, obwohl diese Varianten überfüllte Puffer verursachen, deren Füllstand sich erst in späteren Zyklen wieder normalisiert. Dieser Puffer-Überlauf stellt den grössten Mangel der vorliegenden CINSim-Version dar. Interpretationen von korrekten Simulationsresultaten runden das Kapitel ab.

Da die Validierung für die meisten Netzvarianten belegt hat, dass CINSim deren dynamische Rekonfiguration korrekt simuliert, wird im Kapitel 6 für eine solche Variante gezeigt, dass sie verglichen mit der statischen Rekonfiguration die Kommunikationsleistung der neuen Konfiguration trotz leistungsschwächerer Übergangsphase schneller erreicht. Die erwartete Verbesserung von statischer zu dynamischer Rekonfiguration konnte erstmals mit CINSim simulativ bestätigt werden.

7.2 Executive Summary

[3], [2] and [4] discussed the need to develop a simulation tool, being able to simulate dynamic reconfigurations of arbitrary network topologies. This aim was approached by CINSim, supporting a network, that runs with a big variety of techniques and can be statically reconfigured as well. After a short introduction to the topic (Chapter 1), the relevant basics of interconnection networks theory are presented in chapter 2: topology, switching, routing, scheduling, multicast and flow control. Further, section 1.2 discusses aspects of static reconfiguration. It is mentioned, that CINSim supports only a small set of adjustment possibilities among the huge amount of techniques available. The most various adjustments can be set up for packet generation, traffic trait, multicasts, probabilistic target distribution for packets and more. The third section of chapter 2 gives the definition of static and dynamic reconfiguration, that separates the state of the art from the given task, thereby. The enhancement of CINSim to dynamic reconfiguration is up to the developments of this thesis.

Chapter 3 introduces the basic theory of dynamic reconfiguration, being partly taken from literature or being newly developed. The first section derives the criteria of lossless dynamic reconfiguration from the packet target sets and the routing information. The conditions of successful packet-conservation at dynamic reconfiguration are given by task. Section two predicts the occurance of deadlocks, according to [8], after a dynamic reconfiguration and defines them unambiguously. The causes for deadlocks after reconfiguration are strayed packets. An identification rule for stray packets is demanded and evolved, as well as the concept of dependency trees of buffers, that store stray packets. These mechanisms afford to comprehend the deadlocks formally and to give a routing that resolves the situation. Since, not every dependency tree of an involved buffer depicts a solvable part of the deadlock situation, the trees must be classified.

The situation is demergeable, whenever one dependency tree contains a path from the root to one of the leaves, with equal buffer numbers in root and leaf. If this holds, the situation is solvable, respecting the dependencies of the corresponding buffer. This method of analysis and resolution integrates every possible deadlock situation within one theory. It is proven, that multicast and wormhole-switching are exceptions and their use is therefore prohibited.

Chapter 4, explains how the theoretical concepts have been implemented into CINSim. First, the criteria of dynamic reconfiguration are applied to the concept of bitmask adressing given by CINSim. If the conditions are not met, packets are deleted and a warning is displayed. The simulation continues to run after having removed all unroutable packets. Second, necessary bitmask translation and deadlock handling are introduced, for local backpressure flow control. Packet lengths greater 1 demand for a packet collection phase. Packets must be collected into single buffers, to be safe of being detached during reconfiguration. Deploying global backpressure simulation asks for a reorganisation of packet transmission, which is described in section 3. While a packet normally is oriented towards a target, that could be the first packet-receiver in a chain of transmissions, the deadlock-case, after reconfiguration, involves packets that are oriented towards each other. This mutual dependency must be demerged, without having a first receiver. The last section of this chapter discusses the implementation of the enhanced switching-techniques and summarizes the extensions of CINSim.

Whether the results of simulation, being produced by the enhanced CINSim are trustworthy is discussed in Chapter 5. Comparison data to validate CINSim's functionality is sparsely available. Nevertheless, the packet conservation of dynamic reconfiguration simulated by CINSim is proven whenever the corresponding criteria are met. Bidirectional networks with packet length greater one and global backpressure are reconfigured lossless, too, although buffer overflows are detected by CINSim for these variants. Buffer fill level normalize with time while no packet is lost. However, this effect is the worst deficiency of the enhanced CINSim version. Curve interpretation of correctly simulated reconfiguration results tops the Chapter off. The validation documents the correct simulation of dynamic reconfigurations for most of the network variants by CINSim. A correct working variant is taken and the improvement of dynamic reconfiguration, with respect to static reconfiguration, is demonstrated in chapter 6. Despite a lower communication performance in the transient phase of the reconfiguration the performance improvement of the new configuration is reached earlier than in static reconfiguration. This improvement is simulative validated for the first time in the development history of CINSim.

7.3 Ausblick auf weitere Arbeit

Trotz den Erfolgen hat die Implementierung der dynamischen Rekonfiguration Verbesserungspotential und wirft neue Fragen auf. Die Verbesserungsmöglichkeiten werden an den entsprechenden Stellen im Bericht erwähnt und werden hier zusammengetragen:

- Der Programmswitch, der zur Unterscheidung der Switchingverfahren SAF und PCT implementiert wurde, stellt als solcher eine Unschönheit dar. Es ist nach einer besseren Lösung zu suchen, die das Switchingverfahren vom Routingverfahren unabhängig macht. Eine solche Lösung wird eine grössere Umstrukturierung der Klassen von CINSim erfordern und konnte deshalb im Rahmen dieser Master-Arbeit nicht durchgeführt werden.

- Die Suche nach neuen Bezugspuffern für das GBP-Flusskontrolle-Verfahren ist bisher in einer speziellen Suchfunktion implementiert. Die in der Theorie vorgestellte elegantere Methode soll implementiert werden, um die Deadlock-Auflösung von CINSim abgerundeter und kompakter zu machen.

- Das Switching-Verfahren VCT hat erhebliche Probleme an die Implementierung des rekonfigurationsfesten Routing gestellt. Diese konnten im Rahmen dieser Master-Arbeit nicht gelöst werden. Das Routingverfahren so zu erweitern, dass auch VCT korrekt simuliert werden kann, ist ein wichtiger Schritt zur Erweiterung der Variationsmöglichkeiten von CINSim.

- Überfüllte Puffer bei GBP und Paketlänge grösser 3 verursachen zwar eine praktisch unmerkliche Veränderung der Simulationsresultate, doch sind nicht wünschenswert, wenn CINSim als universales Simulationswerkzeug von dynamischen Rekonfigurationen gelten soll. Der Fehler des Algorithmus, der die überfüllten Puffer erzeugt, soll behoben werden.

Die im Verlauf der Arbeit aufgeworfenen Fragen betreffen Multicast-Situationen und irreguläre bidirektionale Netze und solche mit beliebigen Topologien. Es ist nicht ausgeschlossen, dass eingeschränkte dynamische Multicast-Rekonfigurationen durchführbar sind. Der Ansatz, das Deadlock-Auflösungsverfahren nicht nur auf Situationen kurz nach der Rekonfiguration, sondern auf alle möglichen Deadlocks anzuwenden, verspricht ein deadlockfreies Routing auch für Topologien, für die gängige Routingverfahren nicht deadlockfrei sind. Dabei ist an die bidirektionalen irregulären Netze zu denken. Es bleibt offen, ob dieser Ansatz in CINSim realisierbar ist. Diese noch unbearbeiteten Themen zeigen, in welche Richtung weitere Forschung und Entwicklung rund um CINSim und die damit zusammenhängende Theorie der dynamischen Rekonfiguration gehen kann.

Anhang A

Aufgabenstellung

Technische Universität Berlin

FAKULTÄT IV
ELEKROTECHNIK
UND INFORMATIK
Institut für Technische Informatik und Mikroelektronik
Fachgebiet Prozeß-datenverarbeitung
Prof. Dr.-Ing. Dr. h.c.
Günter Hommel

Diplomarbeit

Simulative Leistungsbewertung und Entwicklung „rekonfigurationsfester“ Routingverfahren für mehrstufige Verbindungsnetzwerke

Bearbeitungs-zeitraum: xx. xx. 2006 – xx. xx. 2006
Bearbeiter: Christian Zimmermann (ETH Zürich)

Gutachter: Prof. Dr.-Ing. Dr. h.c. Günter Hommel
Gutachter: Dr.-Ing. Dietmar Tutsch
Betreuer: Dipl.-Ing. Daniel Lüdtke

Technische Universität Berlin
Fakultät IV – Elektrotechnik und Informatik
Institut für Technische Informatik und Mikroelektronik
Fachgebiet Prozeßdatenverarbeitung und Robotik
Sekretariat EN 10 – Einsteinufer 17 – D-10587 Berlin
Tel.: +49 (30) 314-73110, -73618, -25324, Fax: +49 (30) 314-21116
E-Mail: {hommel, DietmarT, dluedtke}@cs.tu–berlin.de

Themenbeschreibung

Am Fachgebiet für PDV und Robotik der Fakultät IV werden dynamische Kommunikationsnetze bezüglich ihrer Leistung durch Simulation und Analyse bewertet.

Diese Netze werden mit dem komponentenbasierten Simulationssystem CINSim [1] untersucht, das an diesem Fachgebiet entwickelt wird. CINSim bietet auch die Möglichkeit sich dynamisch rekonfigurierende Netze zu untersuchen [2]. Insbesondere Änderungen der Netzwerktopologie werden verstärkt untersucht. Rekonfigurierbare Netze können beispielsweise mit FPGAs realisiert werden.

Dynamische Rekonfiguration der Netzwerktopologie soll zur Leistungssteigerung eingesetzt werden, um beispielsweise Lokalitäten im Verkehrsprofil zu begünstigen [3]. Besonderes Interesse in diesem Themenkomplex liegt auf der Phase kurz vor und kurz nach der Rekonfiguration. Da die Netze mit dem Backpressure-Verfahren arbeiten, soll auch durch die Rekonfiguration kein Paketverlust auftreten. In der aktuellen Version von CINSim wird deshalb das Netz vor der Rekonfiguration geleert, da ansonsten die Zieladressen nach der Rekonfiguration nicht mehr aktuell sind.

Ziel dieser Arbeit soll die Entwicklung und Implementierung von Routingverfahren sein, die es erlauben, Pakete nach der Rekonfiguration zu ihren „neuen“ Zielen umzuleiten. Dadurch soll die Leerlaufzeit vor der Rekonfiguration eingespart werden.

Zunächst soll in dieser Arbeit das Routing für die Netzklasse der bidirektionalen mehrstufigen Verbindungsnetze (BMIN, [4]) erweitert werden um eine Rekonfiguration der Ein-/Ausgangsverdrahtung (siehe [3]) handhaben zu können. Hierbei können keine Situationen entstehen, die die erfolgreiche Vermittlung der bereits gesendeten Pakete verhindern könnten. Bei dem Entwurf des Routingverfahrens ist auch der zu erwartende Aufwand in Software (Simulator) und Hardware (zusätzlicher Speicher, Adressumsetzung) zu berücksichtigen.

Anschließend soll das Verfahren für irreguläre Netzstrukturen verallgemeinert werden. Natürlich lassen sich bei uneingeschränkter Rekonfigurationfreiheit Situationen erzeugen, die eine erfolgreiche Zustellung der Pakete nach der Rekonfiguration verhindern. Deshalb ist es Teil dieser Arbeit formale Kriterien herauszuarbeiten, mit denen entschieden werden kann, ob eine verlustfreie Rekonfiguration möglich ist.

Die erarbeiteten Verfahren sollen zum Abschluss mit der vorhandenen Implementierung (Leerlaufphase) bezüglich Durchsatz und Latenz verglichen werden. Die Auswertung der Simulationsergebnisse soll durch aussagekräftige statistische Verfahren [5, 6] erfolgen.

Die erzielten Ergebnisse sind in einer wissenschaftlichen Ausarbeitung zusammenzufassen.

Zu Beginn der Arbeit ist ein Meilensteinplan über die gesamte Bearbeitungszeit zu erstellen. Als Orientierung kann der beigefügt Arbeitsplan dienen.

Literatur

[1] TUTSCH, Dietmar ; LÜDTKE, Daniel ; WALTER, Arvid ; KÜHM, Matthias: CINSim – A Component-Based Interconnection Network Simulator for Modeling Dynamic Reconfiguration. In: *Proceedings of the 12th International Conference on Analytical and Stochastic Modelling Techniques and Applications (ASMTA 2005); Riga*, 2005, S. 132–137

[2] ZHOU, Li: *Simulation der Rekonfigurierung mehrstufiger Verbindungsnetze*. Fakultät für Elektrotechnik und Informatik, Technische Universität Berlin, Diplomarbeit, August 2005

[3] LÜDTKE, Daniel ; TUTSCH, Dietmar ; WALTER, Arvid ; HOMMEL, Günter: Improved Performance of Bidirectional Multistage Interconnection Networks by Reconfiguration. In: *Proceedings of 2005 Design, Analysis, and Simulation of Distributed Systems (DASD 2005)*. San Diego, CA, USA, April 2005, S. 21–27

[4] NI, Lionel M. ; GUI, Yadong ; MOORE, Sherry: Performance Evaluation of Switch-Based Wormhole Networks. In: *IEEE Transactions on Parallel and Distributed Systems* 8 (1997), Mai, Nr. 5, S. 462–474

[5] PAWLIKOWSKI, Krzysztof ; YAU, Victor W. C. ; MCNICKLE, Don: Distributed Stochastic Discrete-Event Simulation in Parallel Time Streams. In: *Proceedings of the 1994 Winter Simulation Conference; Lake Buena Vista*, 1994, S. 723–730

[6] EWING, Greg ; PAWLIKOWSKI, Krzysztof ; MCNICKLE, Donald: Akaroa2: Exploiting Network Computing by Distributing Stochastic Simulation. In: *Proceedings of the European Simulation Multiconference (ESM'99)*. Warsaw, Juni 1999, 175–181

[7] BULISCH, Peter: *Konzeption eines komponentenbasierten Simulationssystems für mehrstufige Verbindungsnetze variabler Struktur (in German)*. Germany, Technische Universität Berlin, Diplomarbeit, Februar 2003

[8] *Component-based Interconnection Network SIMulator*. `http://pdv.cs.tu-berlin.de/CINSim`,

[9] EWING, Gregory ; PAWLIKOWSKI, Krzysztof ; MCNICKLE, Donald: *Akaroa II. Version 2.7. User's Manual*. Christchurch, New Zealand: Simulation Research Group at the Department of Computer Science, University of Canterbury, Juli 2000

[10] WALTER, Arvid: *Simulative Leistungsbewertung bidirektionaler mehrstufiger Kommunikationsnetze (in German)*. Technische Universität Berlin, Fakultät für Elektrotechnik und Informatik, Technische Universität Berlin, Diplomarbeit, August 2003

Anhang B

Begriffe, Definitionen, Regeln, Abkürzungen

B.1 Begriffe

Die folgende Aufzählung erläutert häufig verwendete Begriffe.

Abhängigkeit, Abhängigkeitsbaum: Eine Abhängigkeit zwischen Puffern liegt dann vor, wenn gemäss dem Flusskontrolle-Verfahren ein Puffer nicht senden kann, weil seine Folgepuffer alle voll sind (LBP) oder voll sind und nicht senden (GBP). Ein Abhängigkeitsbaum stellt alle Abhängigkeiten eines Puffers und seiner Nachfolger dar.

Adaptive Routingverfahren: Sie verwenden Informationen über vergangene Transaktionen im Netz, um die optimale Wegfindung zu erreichen. Im Gegensatz dazu sind gedächtnislose Routingverfahren nicht fähig, die Vergangenheit auszuwerten. Deterministische Routingverfahren sind gedächtnislose Routingverfahren, die einen festen Algorithmus dazu verwenden, der keine Zufälligkeit enthält.

Auflösend: Dies ist die Bezeichnung eines Puffers, der für eine Deadlock-Auflösung markiert ist. Jeder auflösende Puffer muss demnach an einem Deadlock beteiligt sein.

Bedingungsloses Empfangen: Dies wird verwendet, um Puffern, die voll und von einander abhängig sind, das gegenseitige Senden zu ermöglichen. Jeder an einem Deadlock beteiligte Puffer muss Flits empfangen, obwohl er voll ist.

Beteiligt: So werden Puffer genannt, die von einem Deadlock betroffen sind und von denen sein Folgepuffer abhängig sind.

Betroffen: Ein Puffer ist von einem Deadlock betroffen, wenn er auf Grund einer Deadlock-Situation nicht mehr weitervermittelt werden kann.

Bezugsänderung, Bezugsziel, Bezugspuffer: Sind in der normalen GBP-Vermittlung Ziele die ersten Empfänger von Flits in Vermittlungsabläufen, so müssen es bei der Deadlock-Auflösung

Puffer sein. Diese Puffer werden Bezugspuffer genannt und werden durch die Bezugsänderung gefunden.

Daten-Flit: Sie sind weder vorderste noch hinterste Flits eines Pakets und modellieren im Fall der Simulation Daten-Tragende Flits ohne weitere Bedeutung.

Deadlock: Die wohldefinierte Verklemmung, die dadurch erkannt wird, dass sich die Nummer eines Puffers in jedem Pfad von dessen Abhängigkeitsbaum wiederholt. Dies ist der Fall, wenn jeder Folgepuffer von dem betreffenden Puffer abhängig ist.

Deterministische Routingverfahren: Siehe „Adaptive Routingverfahren“

Endknoten: Siehe Abbildung 2.1 auf Seite 9

Flit: Kleinste Informationseinheit, aus der Pakete gebildet werden. Es gibt Kopf-Flits, Daten-Flits und Schwanz-Flits. Flit = engl. Flow Control Unit.

Flusskontrolle: Das Verfahren, das angibt, ob ein Puffer sein Paket senden darf oder nicht. In der Arbeit wird nur das globale und das lokale Rückstauverfahren verwendet.

Folgepuffermenge: Die Menge der Puffer, zu denen ein Puffer sein vorderstes Flit gemäss dem Routingverfahren weitervermitteln kann.

Gedächtnislose Routingverfahren: Siehe „Adaptive Routingverfahren“

Irreguläres Netz: Eine Bezeichnung für Netze mit einer asymmetrischen Topologie.

Irrwegpaket: Siehe unter „Kritisch“

Knoten: Siehe unter „Vermittlungsknoten“

Kritisch: wird ein Netz dann bezeichnet, wenn es nach einer Rekonfiguration Pakete enthält, die gemäss dem verwendeten Routingverfahren nicht in den Puffern sein könnten, in denen sie liegen. Solche Pakete werden Irrwegpakete genannt.

Kopf, Kopf-Flit: Das vorderste Flit eines Pakets, das die Ziel-Information trägt.

Leerlauf-Phase: In der Leerlauf-Phase eines Netzes werden keine neuen Pakete generiert, sondern nur noch im Netz befindliche zu ihren Zielen vermittelt.

Mesh: Eine Rechteck- oder Quaderförmige Netztopologie, die beliebige Dimensionen haben kann. 1D-Mesh wird auch Kette genannt. Allgemein bezeichnet ein kD-Mesh eine k-dimensionale Struktur.

Multicast: Eine Nachricht, die mehrere Ziele als Adressaten hat.

Nachricht: Sie besteht aus mehreren Paketen und wird vom Verbindungsnetz von einem Endknoten zu einem anderen vermittelt.

Pakete: Informationseinheiten, aus denen Nachrichten aufgebaut sind, die durch ein Netz geschickt werden. Pakete können ihrerseits aus kleineren Einheiten, den Flits bestehen.

Pfad: Mathematisch definiert als eine Kette von Knoten. Knoten meinen hier die Elemente eines Abhängigkeitsbaumes. Der Begriff Pfad wird in der Arbeit generell als ein Weg von der Wurzel eines solchen zu einem seiner Blätter verwendet.

Programmswitch: Dieser Begriff wird verwendet, um eine Wenn-Dann-Anweisung in einem Programm zu bezeichnen. In der vorliegenden Arbeit ist die Abfrage des Switchingverfahrens im Paketbewegungsalgorithums gemeint, die idealerweise nicht nötig sein sollte.

Puffer: Sie befinden sich vor jedem Switch eines Routers, um eintreffende Flits zwischenzuspeichern. Dies wird Eingangspufferung genannt.

Quelle: Ein Endknoten, der ausschliesslich Pakete sendet und keine empfängt.

Router: Siehe unter „Vermittlungsknoten“

Routing: genauer Routingverfahren genannt. Es definiert, welche Wege Pakete nehmen, um das Verbindungsnetz zu traversieren. Routing = dt. „Wegfindung“ oder „Routenplanung“

Sammelphase: Das dynamische Gegenstück der Leerlauf-Phase. In der Sammelphase senden keine Quellen mehr neue Pakete und auch Puffer senden keine Paketköpfe mehr, damit nach $PL - 1$ Zyklen alle Pakete in einzelnen Puffern gespeichert sind.

Scheduling: Das Verfahren, das für Router, deren Switches auf gleiche Ausgänge schalten und deren Switch-Eingänge jeweils Köpfe sind, entscheidet, welcher Flit-Kopf von den konkurrenzierenden den Switch passieren darf.

Schwanz, Schwanz-Flit: Der Schwanz bezeichnet einen hinteren Paketteil, zu dem das Kopf-Flit nicht dazugehört. Das Schwanz-Flit hingegen bezeichnet das letzte Flit eines Pakets, das als solches für die Switchingverfahren eine Bedeutung hat.

Shortest Path: Das in der Master-Arbeit einzig verwendete Routingverfahren, das jedem Paket zufällig einen möglichen kürzesten Weg durch das Netz weist.

Switch, $m \times n$-Switch: Ein Switch ist ein Schalter, der m Eingänge und n Ausgänge hat. Jeder Ausgang kann unabhängig von den anderen auf jeden beliebigen Eingang des Switches geschaltet werden.

Topologie: Sie definiert, wie Knoten eines Verbindungsnetzes untereinander verbunden sind.

Verbindung: Vermittlungsknoten eines Netzes werden über Verbindungen vernetzt.

Vereinigungsregeln: Sie bestimmen den Typ eines Abhängigkeitsbaums anhand der Typen-Information seiner Unterbäume.

Verklemmung: Undefinierte Blockierung von Paketen in einem Netzwerk, so dass sie mit dem verwendeten Routingverfahren nicht weiter vermittelt werden können.

Vermittlungsknoten: Auch Knoten, Router genannt. Grundelement jedes Verbindungsnetzes, das Pakete, die es erreichen, mittels des Routingvefahrens an andere Knoten oder Ziele weitervermittelt.

Wormhole-Switching: Ein Switchingverfahren, bei dem die Puffer nie mehr als ein Flit eines Paketes aufnehmen können.

Ziel: Ein Endknoten, der ausschliesslich Pakete empfängt, jedoch keine sendet.

B.2 Definitionen

Definitionen, die in der Arbeit verwendet worden sind:

Abhängigkeitsbaum: Definition 13 auf Seite 37

Angebotene Last: Definition 10 auf Seite 25

Bedingungsloses Empfangen: Definition 19 auf Seite 47

Beteiligter Puffer: Definition 18 auf Seite 46

Deadlock: Definition 12 auf Seite 36

Durchmesser: Definition 5 auf Seite 13

Durchsatz: Definition 7 auf Seite 14

Freier Pfad: Definition 16 auf Seite 41

Halbierungsbandbreite: Definition 3 auf Seite 13

Klassen von Abhängigkeitsbäumen: Definition 17 auf Seite 41

Passierbarer Pfad: Definition 14 auf Seite 40

Pfaddiversität: Definition 8 auf Seite 14

Problematischer Pfad: Definition 15 auf Seite 40

Rekonfiguration: Definition 1 auf Seite 5

Skalierbarkeit: Definition 4 auf Seite 13

Verbindungsnetz: Definition 2 auf Seite 9

Verkehrscharakteristik: Definition 9 auf Seite 24

Vermittlungs-Information: Definition 11 auf Seite 28

Verzögerung: Definition 6 auf Seite 14

B.3 Regeln

Regeln, die in der Arbeit aufgestellt worden sind:

Erkennung von Irrwegpaketen: Regel 1 auf Seite 33

Bestimmen der Folgepuffermenge: Regel 2 auf Seite 35

Aufstellen eines Abhängigkeitsbaums: Regel 3 auf Seite 39

Pfade für eine Deadlock-Auflösung: Regel 4 auf Seite 46

Disjunkte Auflösende Pfade: Regel 5 auf Seite 65

Sendungen teilnehmender Puffer: Regel 6 auf Seite 65

Mögliche neue Bezugspuffer: Regel 7 auf Seite 70

Spezialregel für PCT-Deadlock-Auflösung: Regel 8 auf Seite 81

B.4 Abkürzungen

BMIN	Bidirektionales MIN
CINSim	Component-based Interconnection Network Simulator
CINSim-GUI	Die graphische Benutzeroberfläche von CINSim, die Netzdateien herstellen und modifizieren lässt.
Flit	engl. Flow Control Unit
GBP	Globales Rückstau Verfahren der Flusskontrolle (engl. global backpressure)
I/O	Input/Output
LBP	Lokales Rückstau Verfahren der Flusskontrolle (engl. local backpressure)
OL	angebotene Last (engl. Offered Load)
PCT	Partial-Cut-Through Switchingverfahren
SAF	Store-And-Forward Switchingverfahren
SP	Shortest-Path Routingverfahren
UMIN	Unidirektionales MIN
VCT	Virtual-Cut-Through Switchingverfahren

Tabelle B.1: Abkürzungstabelle

Literaturverzeichnis

[1] `http://www.darpa.mil/ipto/Programs/programs.htm`, *Information Processing Technology Office*, program list (5. 9. 2006)

[2] TUTSCH, Dietmar; LÜDTKE, Daniel; WALTER, Arvid; KÜHM, Matthias: „CINSim – A Component–Based Interdconnection Network Simulator For Modelling Dynamic Reconfiguration“. In: *Proceedings of the 12th International Conference on Analytical and Stochastic Modelling Techniques and Applications (ASMTA 2005), Riga* (2005), Seiten 132–137

[3] ZHOU, Li: *Simulation der Rekonfiguration mehrstufiger Verbindungsnetze* (2005). Fakultät für Elektrotechnik und Informatik, Technische Universität Berlin, Dimplomarbeit

[4] LÜDTKE, Daniel; TUTSCH, Dietmar; WALTER, Arvid; HOMMEL, Günter: „Improved Performance of Bidirectional Multistage Interconnection Networks by Reconfiguration“. In: *Proceedings of 2005 Design, Analysis and Simulation of Distributed Systems (DASD 2005)* (2005), April, San Diego, CA, USA, Seiten 21–27

[5] EWING, Greg; PAWLIKOWSKI, Krzysztof; McNICKLE, Donald: „Akaroa2: Exploiting Network Computing by Distributing Stochastic Simulation“. In: *Proceedings of the European Simulation Multiconference (ESM'99)* (1999) Warschau, Juni, Seiten 175–181

[6] VAIDYANATHAN, Ramachandran; TRAHAN, Jerry L. *Dynamic Reconfiguration Architectures and Algorithms* (2003), ISBN: 0–306–48189–8, Kluwer Academic/Plenum Publishers, New York

[7] SANCHEZ, José; GARCÍA, Francisco J. Alfaro: „Dynamic Reconfiguration of Node Location in Wormhole Networks“, In: *Journal of Systems Architecture* (2000), Nr. 10, Seiten 873–888

[8] DALLY, William James and TOWLES, Brian: *Principles and Practices of Interconnection Networks* (2004), ISBN: 0–12–200751–4, Morgan Kaufman Publishers

[9] SANCHEZ, José; GARCÍA, Francisco J. Alfaro: „Reconfigurable Wormhole Networks: A Realistic Approach“, In: *12. IPPS/9. SPDP Workshops* (1998), Seiten 428–437

[10] GUNZINGER, Anton, *Computer System Design I&II* (2005), jährliche Vorlesung am D–ITET ETH Zürich

[11] NI, Lionel M.; GUI, Yadong; MOORE, Sherry: „Performance Evaluation of Switch–Based Wormhole Networks“. In: *IEEE Transactions on Parallel and Distributed Systems* 8 (1997), Mai, Nr.5, Seiten 462–474

[12] PAPOULIS, Athanasios and PILLAI, S. Unnikrishna: *Probability, Random Variables and stochastic Processes* (2001), McGraw–Hill Science/Engineering/Math, ISBN 0–07–281725–9

[13] PAWLIKOWSKI, Krzysztof, *Simulation Modelling and Analysis with an emphasis on applications in performance evaluation of telecommunication networks* (2006), Lecture Notes, University of Canterbury, Christchurch, New Zealand

[14] `http://mathworld.wolfram.com/{Graph,Tree,PathGraph}.html`, *Wolfram MathWorld*, Definitionen der mathematischen Begriffe Graph, Baum und Pfad (6. 9. 2006)

[15] KÜHM, Matthias: *Stochastische Auswertung und Beschleunigung von Netzwerksimulationen* (2006), Fakultät für Elektrotechnik und Informatik, Technische Universität Berlin, Diplomarbeit

[16] STOERIG, Hans Joachim *Kleine Weltgeschichte der Philosophie* (1999), Fischer, Überarbeitete Neuausgabe

[17] JENQ, Yih–Chyun: „Performance Analysis of a Packet Switch Based on Single–Buffered Banyan Network“, In: *IEEE Jorunal on Selected Areas in Communications* (1983), Dezember, Nr. 6, , Seiten 1014–1021

[18] DIAS, Daniel M.; JUMP, J. Robert: „Analysis and Simulation of Buffered Delta Networks“, In: *IEEE Transactions on Computers* (1981), April, Nr. 4, Seiten 273–284

[19] PATEL, Janak H.: „Performance of Processor–Memory Interconnections for Multiprocessors“, In: *IEEE Transactions on Computers* (1981), Oktober, Nr. 10, Seiten 771–780