Quality of Service in Interconnection Networks

Doctoral Dissertation
by
Sven-Arne Reinemo

Submitted to
the Faculty of Mathematics and Natural Sciences
at the University of Oslo in partial fulfillment
of the requirements for the degree Doctor Scientiarum

11th September 2007

Series of dissertations submitted to the
Faculty of Mathematics and Natural Sciences, University of Oslo
Nr. 650

ISSN 1501-7710

Cover: Inger Sandved Anfinsen.
Printed in Norway: AiT e-dit AS, Oslo, 2007.

Produced in co-operation with Unipub AS.
The thesis is produced by Unipub AS merely in connection with the
thesis defence. Kindly direct all inquiries regarding the thesis to the copyright
holder or the unit which grants the doctorate.

Unipub AS is owned by
The University Foundation for Student Life (SiO)

"You may grow old and trembling in your anatomies, you may lie awake at night listening to the disorder of your veins, you may miss your only love, you may see the world about you devastated by evil lunatics, or know your honour trampled in the sewers of baser minds. There is only one thing for it then – to learn. Learn why the world wags and what wags it. That is the only thing which the mind can never exhaust, never alienate, never be tortured by, never fear or distrust, and never dream of regretting. Learning is the only thing for you. Look what a lot of things there are to learn."

T.H. White,
The Once and Future King
Putnam Adult, 1958

Abstract

Interconnection networks were traditionally confined to multiprocessors where low latency and high bandwidth were necessary for interprocessor communication. But in the last decade interconnection networks have become crucial in other application areas such as storage area networks and high performance computing clusters. This development has led to an increased interest in supporting quality of service in interconnection networks driven by the wish to converge cluster storage, cluster communication, and cluster management in one single network. In this thesis we propose and study a set of mechanisms to achieve this in existing interconnection technologies. We introduce two new topology agnostic routing algorithms, a service differentiation scheme for the InfiniBand Architecture, and several admission control schemes.

Acknowledgement

First and foremost I would like to thank my primary supervisor Tor Skeie for guidance and inspiration throughout the four years of thesis work. I would also like to thank my secondary supervisor Olav Lysne for valuable feedback and support during this period, and for granting me a Ph.D. scholarship at Simula Research Laboratory. Furthermore, I would like to thank the co-authors of the papers that this thesis is founded on: Jose Duato, Jose Flich, Andres Mejía, Frank Olaf Sem-Jacobsen, Thomas Sødring, Ingebjørg Thelin Theiss, and Ola Tørudbakken.

Many thanks also go to the people in the Condor project for making countless hours of simulations so much easier to handle, to the people in the J-Sim project for creating a great simulation platform, and to the Free Software Foundation for creating many of the tools I have used constantly for the last four years. I also wish to thank my former and current co-workers at the Department for Networks and Distributed System for a pleasant working environment and for some after work happy hours.

Finally, I would like to thank my family for their support during these years.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Interconnection networks have traditionally been used for multiprocessors where low latency and high bandwidth communication was necessary for interprocessor communication. This is still true, but in the last decade interconnection networks have become crucial in other application areas such as storage area networks, high performance computing clusters, embedded systems, switching fabrics inside switches and routers, and as on-chip networks connecting modules on a single chip.

Compared to conventional networks, interconnection networks can be characterised as short range, low latency, and high bandwidth networks. Moreover, they differ from other networks in that they are loss-less in the sense that packet loss never happens as a result of congestion, but only as a result of link transmission errors. Hence the available link bandwidth is used effectively as retransmissions are seldom necessary. Loss-less operation does, however, introduce other problems peculiar to interconnection networks, such as deadlock and back-pressure. These issues must be handled correctly in order to route packets and deliver *quality of service* (QoS), which is what we study in this thesis.

1.1 Motivation

The two most important performance metrics for interconnection networks are *throughput* and *latency*. High throughput, i.e. the number of bits per second that a network can handle, is necessary for quickly moving data in the network. While low latency, i.e. the difference in enter and exit time for a packet crossing the network, is necessary for fast access and messaging between nodes. Both these aspects have been studied for over two decades in the context of routing algorithms [1, 2]. Routing algorithms is also the first

of three topics that we will study in this thesis, since the routing algorithm is the most important function in any interconnection network. Routing algorithms come in many flavours with different requirements of the network architecture and with different restrictions on what topologies they can be used with. The routing algorithm is also important for QoS since it has a major impact on latency and throughput, and on how well we are able to utilise the resources in the network.

QoS has, until recent years, received little attention in the context of interconnection networks. But as the application domain for interconnection networks has grown in the last decade so has the need for QoS research. The concept of QoS strives to guarantee that the applications communication needs are fulfilled. One of the simplest forms of QoS found in most interconnection networks is the differentiation that happens between control packets and data packets. Technologies such as Ethernet [3], the InfiniBand Architecture (IBA) [4], and the Advanced Switching Interconnect (ASI) [5] all enforce the rule that control traffic *always* preempts data traffic. This strict priority of control frames is deemed necessary for correct operation of the network. Furthermore, as the amount of control traffic is negligible it has very little impact on the over all performance. The differentiation of control traffic and data traffic can be further generalised to other services and applications, such *service differentiation* is the second topic in this thesis.

Our third topic is *admission control*. Admission control is a general concept for controlling the operation point of a network in order to avoid saturation. The level of saturation that is acceptable depends on the service. Services only relaying on throughput will operate well even beyond the saturation point, while latency sensitive applications will suffer as the network reach and go beyond the saturation point.

The overall goal of QoS is to serve different applications according to their different needs. How, and at what granularity level QoS is used is important for how well the network is able to match the requirements of the application. The challenge is to balance the achievable QoS and the complexity of the network protocols and network hardware. Making the correct choice enables us to combine high volume data transfer and low latency messaging concurrently in the same network architecture.

1.2 Contributions

In this thesis we present a set of mechanisms to provide QoS in interconnection networks. The main goal of these mechanisms are to fulfil the vision of a converged interconnection network where I/O, message passing, and

management traffic can co-exist in a single network without one reducing the performance of the other. The primary contributions in this thesis are within the field of routing, service differentiation, and admission control in interconnection networks.

We suggest two new routing algorithms, Layered shortest-path routing [6] (LASH) and Segment-based routing [7] (SR). LASH is a deterministic and topology independent routing algorithm which requires virtual channels. The main advantage of LASH is that it guarantees *shortest path* routing while only using a modest number of virtual channels. Furthermore, it can be tweaked to be both source-adaptive and switch-adaptive. SR is also a deterministic and topology independent routing algorithm, but without the need for virtual channels. SR does not guarantee shortest path routing, but as it does not require virtual channels it is applicable to a wider range of network technologies. The main strengths of SR is its *locality independence* property that gives us a large degree of freedom when enforcing routing restrictions. This freedom also makes it possible to exploit the regularity present in regular and semi-regular topologies.

In service differentiation we propose how to use the *virtual channel* mechanism in IBA (or similar technologies) for QoS purposes. More specifically, we suggest how the *Differentiated services* [8] approach can be applied to IBA. We also show how relative differentiation can be used to give bandwidth guarantees and improve latency and *jitter*.

Finally, we propose three (and evaluate four) admission control schemes, which can be combined with service differentiation to give statistical guarantees. We show this for *centralised* and *distributed* admission control and evaluate these mechanisms at *class* and *flow* granularity.

1.3 Published Works

The core results of this thesis have been published or are under review for publication as listed in the bibliography. For convenience these papers are also listed separately in Appendix A, where a brief description of the authors contributions are given.

The results discussed in Chapter 3 are published in [6, 7, 9, 10, 11]. Chapter 4 is covered by [12, 13], and Chapter 5 is covered in [14, 15]. The work in [7, 10] is the result of a collaboration with researchers at the *Departamento de Informática de Sistemas y Computadores*[1] *(DISCA)* at the University of Valencia, Spain. The other papers have all been written in collaboration with

[1] http://www.disca.upv.es/

researchers in the ICON[2] project at Simula Research Laboratory as reflected by the author list.

1.4 Thesis Outline

Chapter 2 contains some background information about interconnection networks in general. This is recommended reading for those unfamiliar with interconnection networks, while other readers can safely skip this chapter. Chapter 3 describes, formalises, and evaluates the suggested routing algorithms LASH and SR. In Chapter 4 we describe a service differentiation scheme for IBA and similar technologies, which we evaluate in combination with LASH routing. In Chapter 5 we suggest and discuss several approaches to admission control, which we evaluate in combination with the mechanisms from Chapter 3 and 4. Finally, in Chapter 6 we conclude and give some suggestions for further work.

[2] http://www.simula.no/departments/networks/projects/ICONPRO

Chapter 2

Background

In this chapter we present some basic concepts in interconnection networks to make it easier to follow the discussion later. We introduce some general topics of interconnection networks in Section 2.1, which is followed by a presentation of Quality of Service (QoS) in Section 2.2. In Section 2.3 we discuss some of the research methods common in the field including the method used in this thesis.

2.1 Interconnection Networks

The building blocks of interconnection networks are hosts, switches, and links. *Hosts* produce and consume network traffic, while *switches* forward traffic from one point to another. Network *nodes* (i.e. switches and hosts) are interconnected through point-to-point *links*. This makes it possible for hosts to communicate with each other through one or more switches. Switched networks stand in contrast to bus-based networks where hosts are connected to each other through a shared link, referred to as a bus. Bus-based networks are inferior to switched networks because the bandwidth of a bus is inversely proportional to the number of hosts, which scales poorly when the number of hosts increase. In a switched network the scalability depends on how many hosts we connect to a switch and how switches are interconnected. We will only be concerned with switched networks in this thesis.

The fundamental part of any interconnect is the topology. A topology describes how nodes are connected together and is critical for performance. The fundamental part of any switch is its switching mechanism, that is, how are packets moved between ports through the switch. The main impact of the switching technique is on latency. Finally, in order to move packets around the network we need a routing algorithm. A routing algorithm describes

how we compute the path a packet travels to get from source to destination. The choice of routing algorithm has a major impact on both latency and throughput.

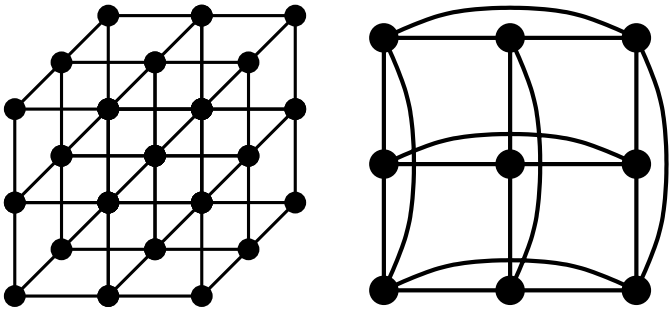
2.1.1 Topologies

A topology describes the conceptual layout of a network and how the nodes in the network are connected. It is the topology that ultimately determines the set of paths a packet can take between two nodes in a network, while the routing algorithm selects a subset of these paths according to certain criteria. The choice of topology depends on the application domain, cost, and packaging restrictions.

Topologies are divided into direct and indirect networks [1, 2]. *Direct networks* refers to topologies where every node in the network contains both a switch and a host. I.e. each node produces, consumes, and routes network traffic. This way each node can reach every other node in the network by going through one or more nodes. In an *indirect network* hosts and switches are separate units connected to each other through a link. A switch can be connected to other switches, other hosts, or a combination of hosts and switches. A direct network can always be converted to an indirect network by separating the host and the switch, and then connect them through a link. This makes the distinction somewhat mute [2]. It does, however, make more sense if we view a direct network as a network where every switch is associated with a host, and an indirect network as a network where only a subset of the switches is associated with a host. Using this definition, two of the most common direct networks are the *torus* and the *mesh*, more formally called the k-ary n-cube and the k-ary n-mesh. Both these belong to a class of topologies refereed to as *strictly orthogonal*. In a strictly orthogonal topology every node has at least one link in each dimension, a fact that makes routing in these networks simple [1].

The mesh is described by the number of dimensions and the number of nodes in each dimension. Figure 2.1(a) shows a 3-ary 3-mesh, formally we say that an n-dimensional mesh has $k_0 \times k_1 \times \ldots \times k_{n-2} \times k_{n-1}$ nodes with k_i nodes in dimension i, where $k_i \geq 2$ and $0 \leq i \leq n$ [1]. For a 3-dimensional mesh with a different number of nodes in each dimension we explicitly give the size of each dimension, e.g. a 2,3,4-ary 3-mesh is a 3-dimensional mesh with two nodes in dimension y, three nodes in dimension z, and four nodes in dimension x [2]. The mesh has been used in multi-computers such as the *MIT M-Machine*, which use a 3-dimensional mesh [16].

The torus, also referred to as a k-ary n-cube, is show in Figure 2.1(b). In the k-ary n-cube all nodes have the same number of connections and the

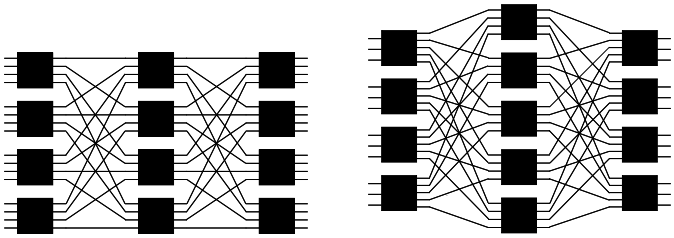


(a) A 3-ary 3-mesh. (b) A 3-ary 2-cube.

Figure 2.1: Direct network topologies.

number of nodes in each dimension are the same. To get the same number of connections to each node, wrap-over connections are added at the edge nodes. A k-ary n-cube has k^n nodes. Figure 2.1(b) shows a 3-ary 2-cube. The torus has been used in machines such as the *Cray T3D* [17] and the *Blue Gene/L* [18].

Indirect networks also comes in several forms, but the most popular category is *multistage interconnection networks* (MINs). MINs consists of a set of stages where hosts are only connected to the terminal stage, and connected to each other through a set of switching stages. The Clos network described by Charles Clos at Bell Labs in 1952 [19] is an example of a MIN. Figure 2.2(a) shows a 16-way Clos network, the number 16 refers to the number of hosts connected to the terminal stage. A Clos network consists of three or more stages where only the switches in the first and last stage are connected to hosts. The switches in the other stages are only connected to switches in the stages before and after itself. Clos networks are categorised as either *rearrangeable non-blocking* or *strictly non-blocking*. A rearrangeable non-blocking network is a network where it is possible to find a way from a free input to a free output by rearranging the existing connections, while in a strict non-blocking network it is possible to find a way from a free input to a free output without modifying the existing connections. Figure 2.2(a) shows a rearranging non-blocking Clos. A strict non-blocking switch requires more switches in the centre stage and fewer hosts connected to the terminal stage as shown i Figure 2.2(b). Clos networks have been used in



(a) A rearrangeable non-blocking 16 way Clos.

(b) A strict non-blocking 16 way Clos.

Figure 2.2: Multistage interconnection network topologies.

the *GF11 supercomputer* from IBM [20] and have also been widely proposed as an architecture for ATM switches [21, 22, 23]. Other types of MINs are the Butterfly, the Benes, the Omega, and the Fat Tree [1]. Depending on the topology, MINs can be simple to route as deadlock cannot occur.

Another class of indirect networks in widespread use is *irregular networks*. An irregular network is without any well defined structure, it consists of a set of randomly connected switches and hosts. Most local area networks are irregular networks, and so are most networks of workstations. Moreover, a regular network becomes irregular whenever a switch or link in the network fails. Irregular networks present the greatest challenge for routing as no assumptions can be made about the topology.

In the following chapters we will limit our discussion and performance evaluation to irregular and orthogonal networks. But, as all our work is topology agnostic it is compatible with any topology.

2.1.2 Switching

A switching technique describes how packets are moved from an input link, through a switch, and to an output link, how packets are buffered and what happens if the necessary resources are busy. We describe the three most common switching techniques below.

Store and forward (SAF) [1] switching is a traditional switching technique used in early multi-computer networks and local area networks, it is also used in most Ethernet switches. SAF switching works by first receiving a complete packet in a buffer, then examining the header, and finally forwarding the

packet out on the correct link. This happens for each switch a packet passes through and it makes it possible to check for and remove corrupt packets at every hop. The drawback is that it increases latency since each packet must be completely received by the switch before it can be routed and forwarded to its destination link. Thus, the latency is proportional to the number of hops in a path. SAF switching also puts a constraint on the maximum packet length since a complete packet must be able to fit in the buffer space of a switch. I.e. if we want to support larger packets we need larger buffers.

Virtual cut-through (VCT) [24] switching is a solution to the latency problem appearing in SAF switching. Instead of transmitting a complete packet, each packet is split into a number of smaller units called *flits*. The first flit contains the packet header and is used to make the routing decision, while the rest of the flits belonging to the packet follow the same path as the header flit through the network. If the header flit has to be buffered due to a busy output link the rest of the flits will be buffered behind it in the current buffer. In other words, the routing decision is made as soon as the header of the packet is received and if the necessary resources are available the rest of the packet is not buffered but forwarded directly to the destination link. If the necessary resources are busy the packet are buffered in the switch just as with SAF switching. Since we don't have to do any buffering when resources are available latency is reduced compared to SAF switching. But for the worst case where a packet is blocked at every switch VCT switching is equal to SAF switching with regards to latency. VCT switching still has the drawback that the maximum packet length depends on buffer size because we must be able to buffer a complete packet when a required resource is unavailable. Furthermore, per hop removal of corrupt packets is no longer possible because we do not receive the whole packet before it is forwarded. The InfiniBand Architecture [4] and the Advanced Switching Interconnect [5] use VCT switching.

Wormhole routing (WR) [25] removes the dependency between maximum packet length and buffer size found in SAF and VCT switching. As with VCT switching we use flits, but when the header flit is blocked it is buffered at the current switch together with *some* of the flits following it. When the buffer in the current switch is full the remaining flits are buffered in the switches along the established path all the way to the source. At the source the flow-control halts the transmission of data until the necessary resources are available. The buffered packet now spreads like a worm through the network thus the name Wormhole routing. WR yields the same resource gain as VCT, in addition it allows for unlimited packet size as we have removed the dependency between packet size and buffer size. Unfortunately, WR is more prone to deadlock than SAF and VCT switching as the distribution of a packet as a worm

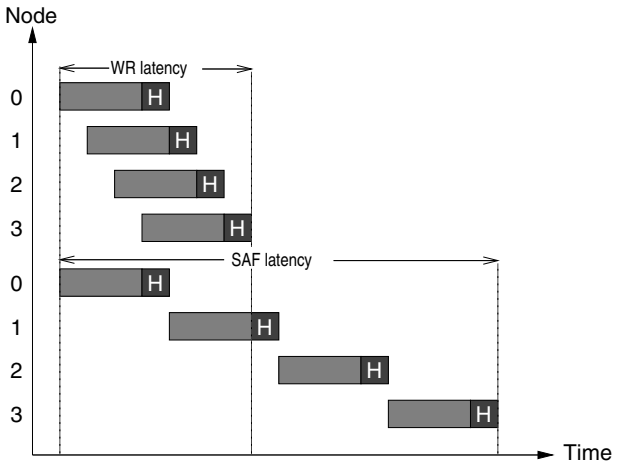


Figure 2.3: Latency for store and forward switching versus virtual cut-through switching and wormhole routing.

through the network grabs hold of more resources giving higher deadlock probability (see Section 2.1.3). Wormhole routing was used in the STC104 Asynchronous Packet Switch [26].

Figure 2.3 compares the latency of VCT/WR and SAF switching. We see that the start of a packet might be entering node 3 before the end of the packet has left node 0 when we use VCT/WR. While with SAF switching we have only reached node 1.

2.1.3 Routing

So far we have seen how the choice of topology puts the ultimate limits on network connectivity, while the choice of switching mechanism puts a lower bound on latency. With these limitations we must construct a routing algorithm that makes the best possible use of resources. The routing function decides which output link to send a packet to based on the information in the packet header. Obviously, we want the routing function to yield the highest possible throughput and lowest possible latency. Less obvious is the fact that our routing algorithm has to be *deadlock* free. Furthermore, in order to ensure efficient use of bandwidth and avoid retransmissions we want loss-less transmission of packets. We discuss the use of flow-control in order to support loss-less operation below, then we introduce the issue of deadlock,

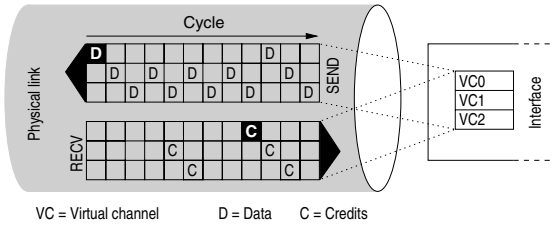


Figure 2.4: Virtual channel flow-control.

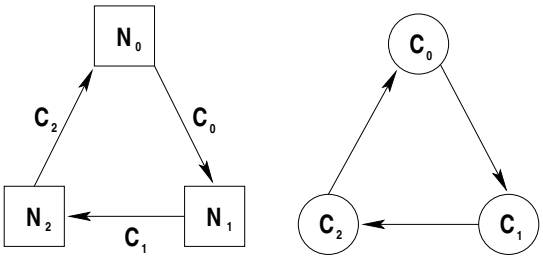
before we introduce some well know routing schemes. Finally, we consider some aspects important for low latency and high throughput routing.

Virtual Channel Flow-Control

Most interconnection technologies are loss-less networks, where packet loss only happens as a result of link transmission errors and hence the available link bandwidth is used effectively as retransmissions are seldom necessary. This does, however, introduce the problem of deadlock that we will discuss in the next section.

In order to avoid packet loss due to buffer overflows, most interconnection networks uses a point-to-point credit-based flow-control scheme. In such a scheme, the downstream side of a link keeps track of the available buffer resources (credits) by decreasing a credit counter whenever buffer space is allocated and increasing the credit counter whenever buffer space is de-allocated. Similarly, the upstream node keeps track of the available credits (i.e. the number of bytes it is allowed to send) and decreases this amount whenever it sends a packet. Whenever credits arrive from the downstream node it increases the amount of available credits. A packet is never sent downstream unless there is room for it. At regular intervals the downstream node details credit availability to the upstream node. The update interval depends on the load, high loads increase the frequency of updates, while low loads reduce the frequency.

As a further improvement flow-control can be performed per *virtual channel* (VC) [27]. The concept of VCs allows a physical link to be split into several virtual links, each with its own buffering and flow-control resources. Figure 2.4 shows an example of per VC credit-based flow-control where VC0 runs out of credits after cycle 1 (depicted by a bold D) and is unable to transmit until credit arrives in cycle 9 (depicted by a bold C). As the other lanes have sufficient credit they are unaffected and are able to use the slot



(a) A network with nodes N_i and links C_i.

(b) Dependency graph for the network in (a).

Figure 2.5: A simple network and its dependency graph.

that VC0 would otherwise use. Transmission resumes for VC0 when credit arrives. VCs improve network throughput [27] and also makes it possible to build virtual networks on top of a physical topology. These virtual networks, or layers, can be used for various purposes such as efficient routing, deadlock avoidance, fault-tolerance and service differentiation [6, 12, 28]. We will discuss the use of VCs in the context of deadlock free routing and service differentiation in Chapter 3 and 4 respectively.

Deadlock

When designing a routing algorithm for interconnection networks one of the base criteria is that it must be deadlock free. A deadlock occurs when a packet gets blocked forever because of a resource conflict in the network. The resources can be buffers or links in the network where a conflict occurs when a packet holding one resource is allowed to request another. This happens when loss-less flow-control is used because a packet is not forwarded until the receiver is ready to accept it. Figure 2.5(a) shows a simple unidirectional network with nodes N_i and links C_i. If node N_0 sends to N_2, N_1 sends to N_0, and N_2 sends to N_1 all at the same time, a deadlock occurs. This is because the packet from N_0 to N_2 holds link C_0 while it waits for C_1 at N_1 which is busy. This propagates to the rest of the nodes since there is a circular dependency between the links in the network. To decide if a network has a deadlock we can construct the dependency graph as in Figure 2.5(b). To construct the graph we draw a vertex for each link (or resource) in the

network and a directed edge from link C_i to C_j if C_i depends on C_j. It has been shown that a network is free from deadlock if its dependency graph is acyclic [29]. This is an important result for deadlock avoidance, the details can be found in Chapter 3 of [1].

In practise, the above means that deadlocks can occur in networks where the topology contains loops *and* loss-less flow-control is used. In particular, when a set of packets is stalled because all paths toward the destinations are blocked by another packet in the set, forming cyclic dependencies between channel (buffer) resources due to flow-control. This behaviour stands in contrast to the Internet and conventional Ethernet, where packets are lost whenever a packet buffer overflows, which means that deadlock cannot occur.

To avoid deadlocks we could choose to avoid topologies with loops altogether or we could choose to just drop packets whenever a buffer overflow occurs. Ethernet actually does both through its routing algorithm, the Spanning tree protocol (STP) [30] and its (initial) lack of flow-control. The STP reduces any topology to a tree by disabling links until no cycles are left. This removes the deadlock potential, but it also removes a lot of bandwidth. Efficient topologies such as k-ary n-mesh and k-ary n-cubes contain lots of loops, and if we remove links the available bandwidth decrease. Thus, this is not a very efficient approach.

Three more advanced approaches to handle deadlock are *prevention*, *recovery*, and *avoidance* [1]. The deadlock prevention strategy prevents deadlocks by reserving all necessary resources before communications starts. If this reservation is not possible then the communication fails. This is the approach used in circuit switching. Deadlock prevention is also possible by using routing algorithms that reserve resources in an order such that conflicts will not happen. In deadlock recovery we do not reserve resources in advance, but grant any resource request. Since this allows for deadlocks to occur we need a way to detect it when it happens (usually by using timers where a timeout denounces a deadlock) and resolve the resource conflict. Since resolving the resource conflicts can be time consuming, deadlock recovery is only used when the probability of a deadlock is low. Finally, the most common way to eliminate deadlock is called deadlock avoidance. In this case resources are reserved and released as the packet travels through the network, but a reservation is only granted if it is safe to do so in a global perspective. I.e. we somehow need to know the global state of the network to be able to grant a reservation. In practise this is handled by the routing algorithm by making sure that no cycles exist in the routing table.

Classification

Routing algorithms can be classified according to two main criteria, flexibility and applicability. Flexibility refers to path selection and applicability refers to the class of topologies where the algorithm can be applied. With regards to flexibility we describe an algorithm as either deterministic, oblivious, or adaptive. A *deterministic* algorithm will always select the same path between a given source/destination pair. This removes the overhead of packet reordering at the destination, but might lead to lower network utilisation. *Oblivious* algorithms allow packets to use several paths between a given source/destination pair, but the path must be selected randomly without any knowledge of current network load. As this allows for out of order delivery the destination must handle packet reordering. The use of several paths improve network usage, but as current network conditions are not considered overloaded paths cannot be avoided. With *adaptive* routing we are allowed to select between several paths for a given source/destination pair and we are allowed to make this decision based on network feedback. Furthermore, for both oblivious and adaptive routing it is possible to select between different paths at both the source, as just described, and at every switch along the path between the source and destination. This is referred to as *source-adaptivity* and *switch-adaptivity*, and is support in layered routing as presented in Section 3.3. This degree of choice further improves network utilisation and makes it possible to avoid hot-spots. The main drawback of this increased flexibility is increased complexity in the design of network equipment and out-of-order packet delivery.

When classifying routing algorithms according to applicability the two main categories are topology *dependent* and topology *agnostic* algorithms. Topology agnostic algorithms make no assumptions about the topology and may be used with any type of topology including irregular networks, while topology dependent algorithms make certain assumptions about the topology and exploit this to improve performance. Historically, topology agnostic routing algorithms have lagged behind topology dependent algorithms with regards to performance, but recent proposals to a large extent match the performance of topology dependent routing strategies [31, 32]. Topology dependent algorithms has another important advantage, they are simple both in the algorithm and in implementation. This is especially true for routing algorithms designed for orthogonal networks as they are mathematically well defined, making it easy to perform deadlock free routing. A well know example is *dimension order routing* which can be used in meshes, tori, and hypercubes [33]. It works by crossing dimensions in strictly increasing or decreasing order. E.g. in a 2-D mesh we would first route along the x-dimension until we

reach the correct x-coordinate, then we would route along the y-dimension until we reach the destination (the algorithm is slightly more complex for tori due to wrap-around links). As the topology is well defined it is possible for the source to calculate the exact number of hops in each dimension. Then each switch only have to reduce the hop count by one for each hop, and change dimension whenever the hop count for the current dimension reaches zero. This allows for shortest-path routing as well as simple implementation. Moreover, it scales very well since the routing table can be expressed in a mathematical function. The drawback of such algorithms is that they require a strict regularity, making it necessary to maintain regularity when increasing network size and making it difficult to handle situations where network components fail since faults make the network irregular. Because of these drawbacks, topology agnostic routing algorithms are the preferred choice whenever the performance difference is negligible.

Performance

The performance of a routing algorithm depends on, in addition to the facts already discussed, the use of *shortest-paths* and a *balanced* use of network resources. Shortest-path routing means that we always select the path that gives the shortest number of hops between any source/destination pair, which reduce latency and improve throughput as we use the least amount of resources possible. By balanced, we mean that the routing algorithm evenly distributes traffic across then network when a uniformly distributed communications pattern is applied. Some algorithms, such as UpDown routing, are unbalanced since they rely on a spanning tree and the root of the spanning tree becomes a network hot-spot [34].

2.1.4 Selected Algorithms

This section contains an overview of the routing algorithms used in simulations and evaluations in this thesis.

Spanning Tree Protocol

The defalt routing algorithm for Ethernet switches is the Spanning tree protocol (STP), which is defined in IEEE Standard 802.1D [35]. It was designed to guarantee connectivity while preventing loops in the network. This is accomplished by turning any topology into a tree as described below.

In an Ethernet network every switch has an instance of the STP running and ready to perform the following operations: (i) Elect, among all switches,

the switch with the lowest identifier as the *root* of the spanning tree. (ii) Build the spanning tree by negotiating the deactivation of switch ports. In this step loops are avoided by allowing only one switch to forward frames *from* the direction of the root on to a given link. Thus, only one switch forwards packets to this branch and the switches lower down in the tree. (iii) Listen for changes in the topology (maintenance mode). When step (i) and (ii) are complete all switches listen for topology changes, and whenever a change is detected a new round of negotiations begins.

The main drawback of the STP is that we end up with a lot of unused links and therefore waste away resources. For local area networks this is not a severe problem, but in high performance computing clusters we need efficient topologies and we want to use every link available to achieve the best performance possible. In this scenario the STP is no longer a valid solution. As STP is the default routing algorithm in Ethernet switches we will use this as a reference when considering routing performance in Ethernet.

UpDown Routing

One of the most well known topology agnostic routing algorithms is UpDown (UD) [36]. UD can be used with any topology and it does not require VCs. This makes it suitable for a wide range of network technologies, including Ethernet. UD is a spanning three based routing algorithm that works in two steps. First it creates a breadth-first spanning tree of the topology to be used. Next it assigns either an up or down direction to each link in the topology. For host-to-switch links the up end is the switch end, and for switch-to-switch links the up end is the end closest to the spanning tree root. Now packets can be routed deadlock free as long as we follow the UD rule [36]: "a packet may never traverse a link in the up direction after having traversed one in the down direction". A deadlock free routing table can now be constructed and forwarded to every switch in the network. We will use UD as a benchmark when considering the performance in irregular topologies.

Dimension Order Routing

For topology dependent routing one of the simplest and most popular options is dimension-order routing [33]. This algorithm is applicable to meshes, tori, and hypercubes, and works by crossing dimensions in strictly increasing (or decreasing) order. E.g. in a 2-D mesh we would first route along the x-dimension until we reach the correct x-coordinate, then we would route along the y-dimension until we reach the destination. We will use dimension-order routing as a benchmark when considering performance in regular topologies.

Turn Based Tree Prohibition

Tree Based Turn Prohibition (TBTP) [37] is an algorithm based on turn prohibitions that guarantees that the number of prohibited turns is bounded to 50% regardless of the topology. This stands in contrast to UD where the maximum number of turn prohibitions depends on the topology, without any guaranteed upper bound.

TBTP works by first constructing a spanning tree from the topology. Then, in the second step, a potential set of prohibited and allowed turns are constructed for each switch S_i. The prohibited turns are all the turns around S_i except for turns between links in the spanning tree. Also, turns starting in S_i and following a link not in the spanning tree are permitted. As turns in the spanning tree are never prohibited, the connectivity of the network is preserved. In the third step, we select the switch S_m with the maximum difference between the number of allowed and the number of prohibited turns. It can be shown that there always exists at least one switch where the number of allowed turns is greater than or equal to the number of prohibited turns [37]. In the fourth step all the forbidden turns for the selected node S_i are added to the set of all forbidden turns. Finally, we delete all links connected to S_i except for the links that are part of the spanning tree. The process is then repeated for the remaining set of nodes. The algorithm terminates when all the links not part of the spanning tree are deleted, then the set of all prohibited turns contains the prohibited turns for the topology in question.

A deadlock free routing table can now be constructed by only using turns considered legal according to the above calculation. TBTP has recently been suggested as an alternative to STP in future Ethernet equipment [37]. We will therefore use TBTP as a benchmark for our Ethernet performance studies.

2.2 Quality of Service

Research in interconnection networks has traditionally paid little attention to QoS, while the Internet community has done a great deal of research on this topic. We try to exploit the results from the Internet community in order to support QoS in interconnection networks. It is, however, not possible to directly apply methods developed for the Internet and other lossy networks to interconnection networks. This is primarily due to the difference in operation between lossy networks and interconnetion networks when a link becomes saturated. In lossy networks, such as the Internet, a saturated link leads to packet loss for all flows using that link, but no other links or flows in

the network are affected. For interconnection networks the opposite happens. No packets are lost, but the saturation quickly spreads across the network reducing the overall performance for all links and flows including flows not using the link that is the source of congestion. This phenomenon is called a congestion tree and occurs because we use link level flow-control as described in Section 2.1.3. Therefore, it is necessary to carefully study existing results from the Internet community in the context of interconnection networks.

When discussing QoS in interconnection networks there are three properties of significant importance, *throughput, latency* and *packet loss*. The granularity of the object on which these metrics are applied are: single data streams, classes of traffic, and all network traffic. As most interconnection technologies use flow-control there is a strict guarantee of no packet loss that is valid for all data traffic. Ethernet can be viewed as an exception to this, which will be discussed in Section 3.1. With regard to latency and bandwidth, a combination of mechanisms are often defined, ranging from strict guarantees for single streams [38] via relative guarantees for classes of traffic [14] to no guarantees/over provisioning.

The capabilities that influence the technologies' ability to leverage QoS guarantees and differentiated treatment of traffic fall into four categories: flow-control, congestion control, traffic differentiation, and admission control. While the three former are usually embedded in a technology, the forth one is not.

Flow-control, as discussed in Section 2.1.3, aims to reduce or eliminate packet loss that is a result of contention and overflowing receive-buffers in switches, but its use can result in congestion in one part of the network spreading to other parts as link transfers slow-down or are temporarily halted. Congestion control aims to prevent or react to the onset of congestion, reducing or controlling its effect on overall throughput in the network. Traffic differentiation applies differential treatment to traffic in order to provide certain guarantees to particular streams. A complex application, e.g. a video server, deals with a multitude of traffic types, each with different requirements to timely delivery. One type of traffic may be sensitive to delay, but without strict bandwidth requirements. E.g. network control and management traffic. Other types of traffic may have strict bandwidth requirements, whereas the latency requirements are relaxed. Finally, admission control is the general concept of controlling the operation point of a network to avoid saturation. This is achieved by making all new flows a subject of admission control before any transmission can occur. The QoS capabilities proposed in this thesis belong to the service differentiation and admission control category.

2.2.1 Service Differentiation

Service differentiation is the unequal treatment of traffic based on a certain property. The granularity of differentiation ranges from no differentiation (i.e. best effort) at one extreme to flow level differentiation at the other extreme. In the middle we have class level differentiation, which is the strategy we will be discussing in Chapter 4. The Internet community, which is the sole proponent of the best effort strategy, has also standardised both a scheme for flow level and class level differentiation, respectively IntServ [39] and DiffServ [8]. The IntServ scheme differentiates on a per flow basis, therefore it is possible to get a very good match between requested and achieved QoS. The drawback is that the switch design becomes more complex and less scalable, because it requires per flow signalling and that state information for every single flow is kept in every hop from source to destination. This problem lead to the specification of DiffServ, which use the class of service paradigm. Following the DiffServ philosophy no core switch should hold status information about passing-through traffic neither should there be any explicit signalling on a per flow basis to these components. The switches are assumed to perform traffic discrimination only based on a QoS tag included in the packet header - all packets carrying the same QoS code will get equal treatment.

We have already discussed VC flow-control, which is also a key feature for supporting DiffServ in interconnects. The independent buffers dedicated to each VC in combination with mechanisms for differentiation between channels makes it possible to implement service classes. Certain VCs may be assigned a low priority, and then we assign all low priority traffic to that channel. Other traffic may require strict latency and jitter guarantees, which can be achieved by assigning another channel for this traffic. Basically, this becomes DiffServ implemented with VCs as suggested by Pelissier in [40]. The number of classes depends on the application and the number of VCs available. We will discuss this concept further in Chapter 4.

Integrated Services

The Integrated Services Architecture (IntServ) was one of the first proposed mechanisms to expand the service model of the Internet with QoS [39]. It was specified by the Internet Engineering Task Force[1] (IETF) and is an architecture that must be used in combination with a resource reservation mechanism (see below) to be fully functional. IntServ adds QoS by expanding the existing best effort class with other service classes labelled *guaranteed*

[1] http://www.ietf.org/

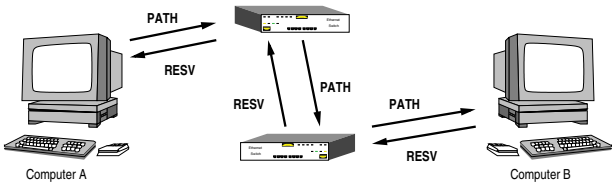


Figure 2.6: Resource reservation with RSVP.

services and *controlled-load services*, which corresponds to real-time traffic and elastic real-time traffic. To differentiate between these service classes IntServ uses flow specific information to map a flow to one of the service classes. This mapping is done by the resource reservation protocol when the initial connection is set up and all the information mapping a flow to a certain service class is contained in the routers along the path between source and destination. A major problem with this approach is that there will be a lot of such flow specification information to maintain when we are looking on core routers in the Internet. This limits the scalability of the IntServ mechanism.

Resource Reservation Protocol

To handle the resource reservation requirements introduced by IntServ the IETF created the Resource reservation protocol (RSVP) [41]. With RSVP we can request that the network reserves a certain amount of resources for our connection from source to destination when the connection is set up. RSVP was first used with IntServ, but has since then been used (and extended) as a signalling protocol for several different QoS mechanisms [39, 41, 42, 43].

The purpose of RSVP is to reserve resources by setting up and maintaining *soft state* information in all the routers along a certain path in the network. Via RSVP an application can request that a certain amount of resources reserved, as shown in Figure 2.6. Here host A wants to make a reserved connection to host B and starts by sending a *PATH* message that includes a unique ID and information about the resources it needs. As the *PATH* message travels towards host B the necessary soft state information is created in the intermediate nodes. When host B receives the *PATH* messages it responds with a *RESV* messages if it accepts the message. As the *RESV* message follows the reverse path back to host A resources are allocated all the way back to the sender. The reservation now consists of *soft-state* information set up and maintained by RSVP in all the nodes along the source/destination path.

Differentiated Services

The Differentiated Services Architecture (DiffServ) [8] is another mechanism to provide QoS in the Internet. It does not need a reservation stage, but it can be combined with RSVP or other mechanisms to improve functionality. Compared to IntServ the DiffServ solution is fundamentally different. One of the key differences is that DiffServ does not give absolute guarantees, but gives guarantees relative to other service classes specified. In other words a "better" service class will always give higher performance than a "worse" service class, but this might not always be good enough to satisfy the needs of the application.

DiffServ us a *stateless core* approach to providing QoS, i.e. the core switches does not contain any per-flow state. Therefore, the scalability problems of IntServ is not present in DiffServ. Instead DiffServ exploits a feature in version 4 of the Internet Protocol (IPv4) [8]. It uses a field in the IPv4 header called Type of Service[2] to label packets with the service class it belongs to. This is often referred to as the DiffServ code point (DSCP) in DiffServ terminology. As packets travel through the network the DSCP is used to differentiate the treatment of packets belonging to different service classes. At the network edge packets from different flows are aggregated in service classes based on a set of classifiers. In the core of the network packets are forwarded according to the per-hop behaviour specified for the DSCP that the packet belongs to.

2.2.2 Congestion Control

As flow-control is a mechanism to avoid packet loss due to buffer overflows, congestion control is a mechanism to aid switches and links in the network from becoming overloaded and depleting their credit supplies. When congestion develops, a depletion of credit supplies starts and the queues begin to fill up. This process spreads upstream through the network and results in the creation and growth of congestion trees which eventually terminate at the end-nodes. Obviously, this is a bad situation for a network as the growth of congestion trees can quickly preclude transmission of other flows, in the same virtual lane, that are not even destined towards the congested area.

The phenomenon causing spread of congestion to parts of the network that do not contribute to the congestion is called *head-of-line blocking*. Head-of-line blocking occurs, for instance, when the head of a FIFO queue is stalled due to heavy traffic. Then the head of the queue is rightly stalled because it is headed for a congested destination, but the packets behind that is not

[2]When used with IPv6 it uses the corresponding IPv6 field called *Traffic Class*.

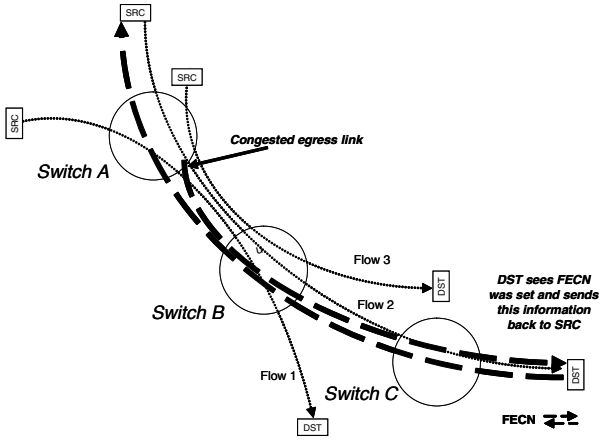


Figure 2.7: Forward explicit congestion notification.

headed for a congested destination is unjustly stalled. When we use loss-less flow-control this problem becomes even worse as the effect of back-pressure creates congestion trees upstream from the point of congestion.

Two common mechanisms to control congestion are Forward Explicit Congestion Notification (FECN) and Backward Explicit Congestion Notification (BECN). FECN is used to inform a packet's destination that it was subjected to congestion when traversing the network. This is achieved by setting a FECN flag in the packets header. This flag is observed by the destination, which can then signal the source about congestion, either by sending a congestion notification packet or, when acknowledgements are used, by setting the congestion flag in the next acknowledgement for the packet in question. See Figure 2.7 for an illustration of FECN. In this figure the link from Switch A to Switch B is oversubscribed. The FECN flag is set at this point and when the destination sees a packet that has this flag set, it sends the FECN status to the source. BECN is similar, but it shortens the feedback loop by going straight from the point of congestion to the source without the need for first going to the destination.

Several other schemes for congestion control exists and this field has received increased attention in recent years, but we will not delve further into the topic of congestion control in this thesis.

2.2.3 Admission Control

The purpose of an admission control (AC) algorithm is to control the operation point of a network in order to avoid saturation. This level of control is commonly attained by using either *rate control* or *call admission control* (CAC). CAC can be compared to the telephone system where there is always a risk of the line being busy if the load is high (call blocking), the lines are considered busy because adding any more calls to the network would reduce the quality all over. At the other end of the spectre we have rate control, which slowly reduce the rate for all clients as the number of clients grow. This allows for more people to participate at the cost of lower performance, and might be acceptable for services such as file transfer and web surfing, but disastrous for sensitive services such as voice over IP, video conferencing, and massive multi-player online games. Rate control is often used in combination with over-provisioning, the most prominent example being the Internet.

AC can be classified as two opposites, centralised versus distributed AC. There are drawbacks and advantages of both that we will discuss in Chapter 5. Where we study four different CAC schemes that differ in the way they collect information about the current network condition and how they make the admission decision.

2.3 Research Methods

When conducting research in communication networks it is common to either study an existing system or to create a new system. When we study existing systems our main tasks are to make measurements and to analyse the collected data. Data are collected in order to improve our knowledge about the system or to understand a certain phenomenon. When we create a new system we have two common options, we can either build a real life prototype or we can build a model. If we choose to build a model, we again have two choices. We can either make an analytical model or a simulation model (or both) [44]. How we conduct research depends on how we decide to build our model. Below we briefly discuss some advantages and disadvantages of these three options.

2.3.1 Prototyping

The main benefit of using prototypes is that it can be considered equal to performing measurements on a real system. Unfortunately, prototypes are costly and time consuming as they require the implementation of both hardware and software. Furthermore, if the first version of a prototype is incorrect, the cost

of building a second version is still expensive as new hardware have to be created. Secondly, it does not scale well. I.e. if we want to study large networks the cost of building a large number of prototypes is prohibitive. Thirdly, it takes a lot of manpower as experts in several fields are required. E.g. to build switching hardware we need people with experience from hardware design and embedded programming, as well as experts on switch architecture and algorithms.

Since the process of prototyping is so expensive and time consuming it is often the last step in a research process, performed after a combination of simulation and analytical work has been proven successful. Then, a prototype often serves as the final step in the verification of a new idea. In the context of this thesis prototyping was never an option due the time and cost constraints.

2.3.2 Analytics

Since prototyping is expensive and time consuming the use of analytical modelling is often preferred, especially for an initial study. The major benefit is that results can quickly be achieved if the case is simple and existing methods from queueing theory [45, 46] or network calculus [47] can be applied. But as the model gets more advanced, analytical modelling becomes complex. This makes it necessary to simplify and reduce the complexity of the model, which might reduce its value and correctness. For this reason, analytical modelling is often reserved for special cases and to prove certain limited aspects of the system under study. It can also be combined with simulations to verify certain observations acquired from simulations.

In the context of this thesis analytics has received little or no focus due to the complex nature of the systems studied and the lack of well developed analytical methods.

2.3.3 Simulation

Simulation represents the most flexible and available method for modelling new systems, and it is the method that we have relied on in this thesis. Several software packages for network simulations exist and many of them are available for free, such as J-Sim [48], OMNet++ [49], and NS2. Of commercial packages OpNet [50] and QualNet are examples targeting network simulation. Simulations are often used on its own, but can also be combined with analytical methods to further strengthen certain aspects. By using simulations we have a large degree of freedom in creating our models and we avoid any risk of disrupting running systems or processes. Furthermore, results can be quickly obtained if the necessary models are already available.

And when the source code is available new models can easily be created based on the code of existing models.

The main problems with simulation lies in deciding the necessary level of detail and verification of the simulation results. If the level of detail is too high our results might be affected by design issues unrelated to the issues we actually want to study. And it can make simulations so resource demanding that processing power becomes the bottleneck. On the other hand, if the level of detail is too low we might end up over simplifying the problem and missing out on essential problems. Finally, the problem of verification can be major drawback of simulations as the model is often too complex to verify by analytical means and too expensive to verify with a prototype. The result being that we build another, independent simulator to verify the same system.

Simulation setup

The simulations results presented in this thesis were performed on two different simulators, one Ethernet simulator and one InfiniBand simulator. The Ethernet simulator is based on the J-Sim framework[48], while the InfiniBand simulator was written in-house at Simula Research Laboratory. Both simulators operate at the link layer of the relevant technologies, which include individual packet buffering and arbitration in the switches.

When doing simulations the network goes through the tree phases: stabilisation, measurement, and drain. In the stabilisation phase no measurements are done since we are waiting for the network to enter a steady-state. When steady-state is reached the measurements phase starts and continues for a given number of cycles. During the measurement phase all packets entering the network are tagged, and when the tagged packets reach their destination statistics are collected. When the measurement phase is over the tagging of packets ends and we enter the drain phase. During the drain phase statistics for the packets remaining in the network are collected, and when all the tagged packets has left the network the simulation is over.

During simulations the throughput is gradually increased until we are beyond the saturation point of the network. This way we can identify the saturation point of the network and study the stability of the network when in saturation. In an unstable network the throughput will drop when we try to send more traffic than the network can handle, while in a stable network throughput remains at the saturation point.

Statistics are collected for throughput and latency. Throughput is calculated as the average number of packets received by a destination during a given number of cycles. This is called *accepted traffic*, while the actual number of packets that a host tries to send is called *offered traffic*. For regular

topologies the accepted traffic is calculated as the average of four measurements on one regular topology. For irregular topologies the accepted traffic is calculated as the average of 16 measurements, one for each of sixteen different irregular topologies. The set of irregular topologies remains the same for all simulations. Latency is measure as the time a packet spends in the network without any end-node queueing time. The calculation of average latency corresponds to the calculation of average throughput described above.

Chapter 3

Routing Algorithms

The routing algorithm is a core function in any interconnect. It describes how we compute the path a packet travels to get from source to destination. As described in Section 2.1.3 the routing algorithm has a major impact on both latency and throughput. Moreover, it has to be deadlock free in order to support a network with link level flow-control. Designing a routing algorithm that minimises latency, maximises throughput, and is deadlock free is difficult. Not only because they often conflict with each other, but also because the design is constrained by the features available in the target technology. In this chapter we will study the routing and flow-control used in conventional Ethernet [3], and we will propose two new algorithms that can be used to improve Ethernet performance in high performance computing.

In Section 3.1 we present and evaluate the routing and flow-control used in conventional Ethernet. Then in section 3.2 we propose and evaluate the Segment-based routing (SR) algorithm in comparison to the Spanning tree protocol (STP). In section 3.3 we propose and evaluate a second algorithm called Layered shortest-path routing (LASH). LASH is not directly compatible with Ethernet, but with some minor changes to Ethernet flow-control they can be combined.

3.1 Routing in Ethernet

The way conventional Ethernet operates differs in two aspects from dedicated interconnection networks. Firstly, interconnection networks are loss-less and only drop frames when bit errors occur. Conventional Ethernet drops frames whenever congestion occurs. Secondly, these networks use mechanisms that avoid deadlock situations while still using all available links. The problem of deadlock does not exist in Ethernet because of its routing algorithm and

because it originally only supported lossy operation. But flow-control was added to Ethernet in 1997 [51], and when we introduce flow-control and do not drop frames the possibility for deadlock appears in topologies with loops [1, 2]. Ethernet does, however, still avoid deadlock by using the STP to calculate routing tables. The STP turns *any* topology into a tree by disabling links until all loops are removed. It is a simple solution that avoids the problem of deadlock, but with one major drawback. When disabling links we waste resources, and for interconnection networks we want efficient topologies and we want to use every link to achieve the best performance possible. In this context the STP is not a good solution.

Our objective in this section is to show how to obtain a loss-less deadlock free network with the best possible performance, while adhering to the current Ethernet standard and using off-the-shelf Ethernet equipment. We achieve this (i) by introducing flow-control in all network nodes and (ii) by replacing the STP. Flow-control makes the network loss-less and replacing the routing algorithm avoids deadlock without disabling links. This makes it possible to leverage the performance that regular topologies such as meshes, tori, clos etc. offers. We also study the effect flow-control has on higher layer protocols, in this case TCP.

3.1.1 Xon/Xoff Flow-Control

To avoid packet loss we need a way to inform the upstream nodes about our buffer situation. In Xon/Xoff flow-control this is done by simple Xon/Xoff messages. When the downstream node has available buffer space it sends an X_{on} message to the upstream node telling it to start sending frames if any are available. As the transmission proceeds and the downstream node runs out of buffer space it sends an X_{off} message telling the upstream node to halt frame transmission. For this scheme to work we must make sure that these messages are sent in a timely manner. When the downstream node sends an X_{off} message it must do it at a point in time where it has enough space to buffer the frames received while it waits for the X_{off} message to take effect. There will be a delay between the transmission and the activation of the X_{off} message due to a propagation and processing delay. According to [2] the buffer requirements and trigger values can be calculated as follows[1]:

$$F \geq F_{\text{on}} + \frac{t_{\text{rt}} b}{L_{\text{f}}} \geq F_{\text{off}} + \frac{t_{\text{rt}} b}{L_{\text{f}}} \geq \frac{2 t_{\text{rt}} b}{L_{\text{f}}} \tag{3.1}$$

[1]The formula shown is for frames while the original is for flits. The end result is the same as mentioned in [2] Sect. 13.3 page 245.

Here F is the total buffer space, F_{on} the number of buffers triggering the on message, F_{off} the number of buffers triggering the off message, t_{rt} the propagation and processing delay for a frame, b the link bandwidth, and L_{f} the frame size. The minimum number of buffers needed to allow full speed operation is $2t_{\text{rt}}b$. We need $t_{\text{rt}}b$ bytes to make sure we have buffers available to receive data sent after the off message was sent, but before it was received and processed at the other end. We need another $t_{\text{rt}}b$ bytes to make sure we have data to send while we wait for an on message to be received and processed at the other end. If we want to further reduce the number of Xon/Xoff messages sent at the cost of more buffers, we can increase the number of buffers used for F, F_{on} and F_{off} according to the formula.

3.1.2 Ethernet Flow-Control

When flow-control was added to Ethernet the concept of *control frames* was introduced for the first time in Ethernet technology. Currently, there is only one flow-control scheme specified and this is an Xon/Xoff approach similar to the one described in the previous section. Here a *pause frame* is used to communicate the Xon/Xoff messages. A pause frame is a special instance of the control frame shown in Figure 3.1 [51]. According to the standard a pause frame must have the MAC control op-code set to 0x0001 and a MAC control parameter consisting of a two byte field called the *pause time*. The pause time P means the time the upstream node must wait before sending the next frame. This time is measured in 512 *bit-time* increments, where the bit-time B is the time it takes to send a single bit. For Gigabit Ethernet B equals 1 ns which gives P a range of 0–33.6 ms in 512 ns increments. A pause time P with a value of zero equals an X_{on} message and overrides any earlier pause times. A P between 1 and 255 equals an X_{off} message lasting $P \times 512$ bit-times. As time passes the pause time will eventually expire, this is a safety measure to avoid permanently pausing a link if the X_{on} message should be lost. If the situation persists, however, we must refresh the pause by sending another pause frame.

The exact algorithm for triggering the pause frame mechanism is unspecified, and it is up to the individual vendors to find their own solution. We use a threshold function to trigger the transmission of pause frames and a timer to check if the pause should be refreshed. This timer is a countdown to the expiration of the last pause frame transmitted. If the timer reaches zero and the current port is still congested, we have to resend the pause frame telling the upstream node to extend its pause time. The threshold function is tightly connected to the minimum buffer space we need to avoid dropping frames and to keep the link running at full speed as described in (3.1). With

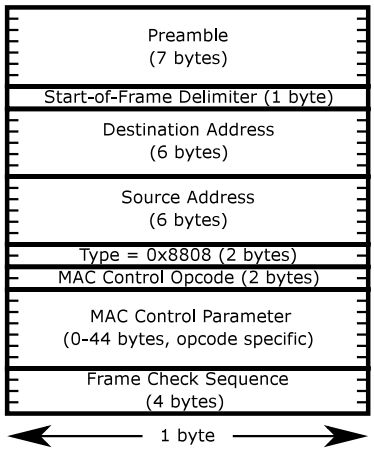


Figure 3.1: MAC control frame format.

the numbers from Table 3.1 we get the following buffer requirements when we replace $\frac{t_{\mathrm{rt}}b}{L_{\mathrm{f}}}$, F_{on} and F_{off} with 4566 bytes:

$$F \geq 2 \times 4566 \tag{3.2}$$

These 4566 bytes consist of the following fields from Table 3.1 [51]: *Frame on transit* is a frame that has just been flushed for transmission when flow-control has been activated on the sender side. This frame must be completed before we can send the pause frame. *Frame on receive* is a frame that has just been flushed for transmission when we have decoded the pause frame on the receiver side. We must finish transmission of this frame before we pause the link. *Pause frame* and *pause frame decode* is the time it takes to send and decode a pause frame respectively. The *propagation delay* is the delay over ten meters of unshielded twisted pair for Gigabit Ethernet. In total this becomes 3270 bytes, but we only buffer complete frames so we round this upwards to three maximum-length Ethernet frames which equals 4566 bytes. Thus, the minimum buffer space necessary for full speed operation becomes:

$$F \geq 9132 \text{ bytes} \tag{3.3}$$

As we try to minimise buffer space we set $F_{\mathrm{on}} = F_{\mathrm{off}}$. This gives us a total of 9132 bytes for each port and a pause frame trigger at 4566 bytes.

The granularity of Ethernet flow-control is per port, thus the upstream node can be told to stop frame transmission when the downstream node

Allotment	Bits
Frame on transit	12,336
Frame on receive	12,176
Pause frame	512
Pause frame decode	1024
Propagation delay (10m UTP)	114

Table 3.1: Minimum buffer requirements for Gigabit Ethernet flow-control.

runs out of buffer space without affecting traffic on any other ports. The effect of increasing and decreasing flow-control granularity is studied in [52]. The main conclusion being that increasing flow-control granularity to act on source/destination pairs further improves performance. Additionally, per port flow-control is incompatible with Ethernet's priority mechanism, this will be further discussed in Section 3.3.3. In the rest of this section we assume port based flow-control as this is the granularity that is supported by the Ethernet standard.

3.1.3 Switch Organisation

We have modelled our switch as a shared memory architecture. A shared memory architecture was chosen because it reduces the effect of head of line blocking, while in a crossbar approach this must be dealt with specifically. Shared memory is also the most deployed switch architecture in current Ethernet equipment [30].

Our switch model is shown in Figure 3.2. It has a shared memory used to exchange frames between ports, and each port have a dedicated lookup engine to allow for distributed address lookup. The output port lookup is completed before the frame is stored in shared memory, but it would also be possible to store the frame before or in parallel with the lookup. Then the frame could be updated with the output port information when the lookup is complete. On the output side each output port has a FIFO queue for outgoing frames. This queue is implemented as a pointer table with pointers to the corresponding frames in the shared memory. When an output queue is ready to transmit a frame it follows the pointer at the front of the queue, transmits the frame, and releases the memory previously occupied by the frame. In order delivery is ensured by using links among packets received through the same input port.

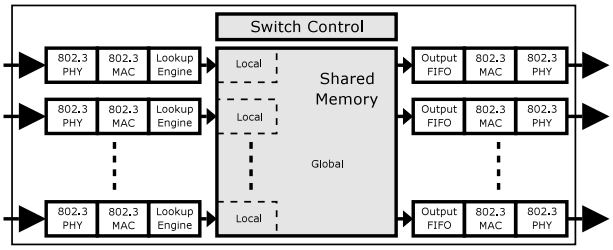


Figure 3.2: Switch Architecture.

3.1.4 Buffer Organisation

To combine our shared memory architecture with flow control we have divided our memory in global and local partitions. The global partition is common for all ports (*global memory*), while the local partition (*local memory*) is dedicated to a single port (Figure 3.2). The global memory is where frames are stored when there is no severe congestion in the switch. If short term congestion occurs the amount of global memory will be able to handle this without activating flow control, i.e. global memory is not subject to flow control. In case of long term congestion the global memory will be filled by frames destined for the congested port. As this happens additional frames destined for the congested port must use the local memory that belongs to the input port of the frame. Furthermore, as the local memory fills up flow control will be activated on this port. This scheme allows ports without frames destined for a congested port to continue operation as normal. To a certain extent it also removes head of line blocking from the ports with frames destined for a congested port, but as the local memory is filled no more progress can be made on this port until congestion resolves. This could be avoided if we, for each input port, had a local memory for each output port.

Flow control is triggered by the use of local memory as described in Section 3.1.2. When the local memory is filled a threshold function triggers the transmission of pause frames according to (3.3).

3.1.5 Performance Evaluation

Our evaluation consists of three routing schemes on regular and irregular topologies with the major performance properties being throughput, latency and frame loss. We also consider TCP throughput and latency as a means to evaluate the effect of flow-control on reliable transport protocols. We have

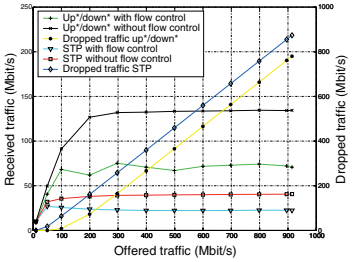
studied the performance of the STP, UpDown routing (UD), and Dimension order routing (DOR), which where all described in Section 2.1.4. STP is the default routing algorithm supported in Ethernet switches, while UD and DOR are well know algorithms from the field of interconnection networks.

We present packet level simulations for a set of regular and irregular topologies. All simulation results have been obtained with an Ethernet simulator developed with the J-Sim framework [48]. We simulate a shared memory Ethernet switch with support for 802.3x flow control and 1 Gbit/s Ethernet links. Each switch has five ports where one is connected to a computing node and up to four are connected to other switches. Our traffic model consists of uniform traffic, and a peak rate packet arrival process. The average bit rate is increased in steps from 10 to 1000 Mbit/s (1% - 100% load). The packet size is fixed at 1522 bytes which is the maximum Ethernet frame size.

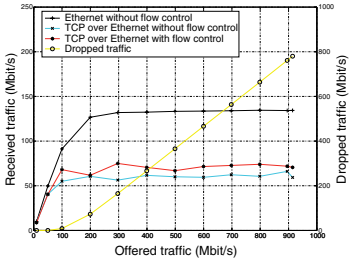
Throughput

We simulated the STP and UD routing on a set of irregular topologies, while for regular topologies we simulated UD and DOR on a 8x4 torus and a 8x4 mesh respectively. We have also studied TCP performance in combination with UD routing, DOR and flow-control. TCP in combination with STP was left out due to the poor performance of STP. The results for irregular topologies are presented in Figure 3.3(a) and 3.3(b). The x-axis show the amount of traffic that each node is trying to send in Mbit/s, the left y-axis shows the average per node receive rate in Mbit/s, and the right y-axis shows the link layer frame loss in Mbit/s. The link layer frame loss when TCP is used is left out as it is negligible due to TCP's built in congestion avoidance mechanism. It is in the order of tens of kilobytes while for a datagram service it is several megabytes (Figure 3.3(a)).

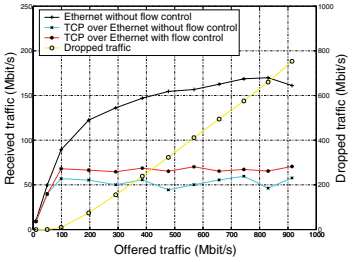
Figure 3.3(a) shows us that UD enabled Ethernet achieves more than tree times the throughput of conventional Ethernet when flow-control is disabled. UD gives a per node throughput of 131 Mbit/s compared to 39 Mbit/s for STP. These data rates, however, are only of theoretical interest since the frame loss is so high. Figure 3.3(a) shows that an injection rate of 300 Mbit/s results in a 60% frame loss for UD and 88% frame loss for conventional Ethernet. Few applications are usable under such conditions, something the TCP results in Figure 3.3(b) shows. Here the achieved throughput when running TCP over Ethernet without flow-control is only 55 Mbit/s, which is less than half the throughput we measured on Ethernet with UD routing. This decrease in throughput happens because TCP uses sliding windows and retransmissions to achieve a reliable service with in order delivery of packets. It is an example of the retransmission penalty that any application requiring



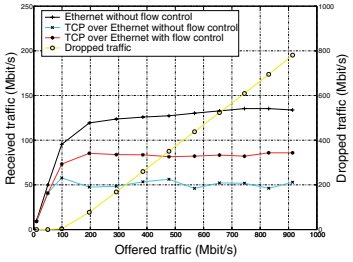
(a) Irregular network with 32 switches.



(b) Irregular network with 32 switches.



(c) 8x4 torus.



(d) 8x4 mesh.

Figure 3.3: Throughput.

a reliable transport layer can run into when using Ethernet without flow-control. Even if we are allowed to inject a lot of frames most of them are wasted since they will probably be dropped and retransmitted. The benefits of reducing retransmissions are larger than the benefits of blindly increasing the injection rate as we shall see below.

To improve performance we enable flow-control. The new achieved throughput, now with zero frame loss, is 65 Mbit/s for UD and 21 Mbit/s for STP (Figure 3.3(a)). When the network saturates, the end nodes are throttled since the switches in the network run out of buffer space. Throttling is achieved by pausing links in the network as described in Section 3.1.2. When more links in the network enters a paused state, the pause state propagates to the end nodes and reduced the number of frames injected into the network.

The achieved throughput for TCP is now 65 Mbit/s compared to 55 Mbit/s without flow-control (Figure 3.3(b)). Which shows that the introduction of flow-control has improved TCP throughput by 18% even if throughput for Ethernet with UD has been reduced from 131 Mbit/s to 65 Mbit/s (Figure 3.3(a)). TCP throughput is increased since we no longer drop frames, meaning that the TCP congestion mechanism is never activated. Every frame we inject arrives at its destination and there is never any retransmissions, which leads to increased throughput from the applications point of view even if the Ethernet injection rate has been reduced. With flow-control we inject less frames, but every frame is useful. Without flow-control we inject a lot of frames, but only a few of them reach the destination.

For regular topologies we obtain even better results when we introduce flow-control. With TCP the throughput is about 50 Mbit/s on both the torus and the mesh without flow-control. When we enable flow-control the TCP throughput increase by 30 % to 65 Mbit/s on the torus and by 60 % to 80 Mbit/s for the mesh (Figure 3.3(c) and 3.3(d) respectively). Again we see an increase in TCP performance even if the Ethernet injection rate is reduced. The 8x4 torus with UD routing and no flow-control achieves a throughput of 160 Mbit/s, when we enable flow-control this is reduced to 65 Mbit/s (Figure 3.3(c)). For the 8x4 mesh with dimension-order routing throughput is slightly lower with 130 Mbit/s and when flow control is enabled this is reduced to 80 Mbit/s (Figure 3.3(d)). It is easy to be seduced by these seemingly good numbers for Ethernet without flow-control, but the truth is that the frame loss and the resulting retransmission rate leads to very poor performance for applications, something the TCP numbers confirms.

The large difference between the torus and the mesh is due to the different routing algorithms. The UD algorithm is unable to utilise the extra connectivity that is present in the torus because it is vulnerable to congestion around the root of the UD tree when flow control is enabled. This weakness of the UD algorithm is studied in [34]. DOR on the other hand is tailored to exploit the regularity of the mesh topology and handles the situation well. The improvement in TCP follows from the reduction in packet loss as discussed earlier. In addition to the removal of frame loss TCP also benefits from the improvement in the routing algorithm. DOR is known to be better than UD routing, as can be seen in the differences between the torus and the mesh (Figure 3.3(c) and 3.3(d)).

Latency

We present latency results for both Ethernet and TCP simulations to see how they differ when flow-control is enabled and disabled. For irregular networks

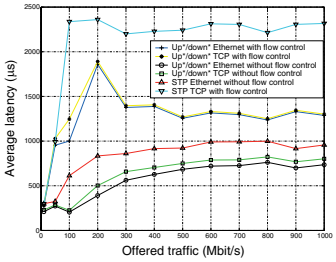
latency is increased with a factor of 2.5, from 1000 μs to 2500 μs, when flow-control is enabled and STP is used (Figure 3.4(a)). For UD routing latency is almost doubled from 700 μs to 1300 μs when flow-control is enabled. The use of UD routing gives lower latency and higher throughput since it can use all links in the network.

The increase in latency that we observe when flow-control is enabled is expected, and is caused by the back pressure created by the pause frame mechanism. When links are paused data frames must wait in buffers along the path from source to destination, where they, in case of no flow-control, would have been dropped. It is this waiting that causes the growth in latency. The worst case scenario is that a frame will wait at every hop towards its destination, resulting in a large increased in latency (See Section 4.3.4).

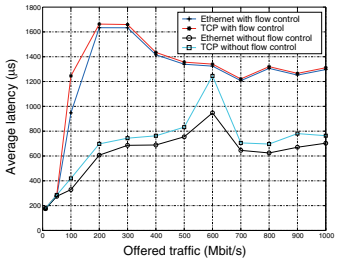
If we compare latency at the Ethernet and TCP level we see that there is almost no difference when flow-control is enabled, and a large difference when flow-control is disabled. When flow-control is enabled TCP latency is only slightly higher than Ethernet latency because the only difference is the protocol overhead in the end nodes[2]. When we disable flow-control we see an increase in TCP latency compared to Ethernet latency. This increase is caused packet loss. Packet loss triggers the TCP retransmission mechanisms and this affects latency in the same way we saw it affect throughput in the previous section. Thus, the introduction of flow-control increase throughput, but at the cost of increased latency.

The behaviour for regular topologies is very similar to that of irregular topologies. When we enable flow-control both the 8x4 torus and the 8x4 mesh see a doubling of latency from 750 μs to 1400 μs (Figure 3.4(b) and 3.4(c)). The results are very similar even though the torus has a higher connectivity than the mesh. This is again due to DOR and its ability to exploit the regularity of the mesh topology better than UD routing is able to exploit the torus. When we consider TCP latency we see the same differences between TCP and Ethernet latency as for irregular topologies.

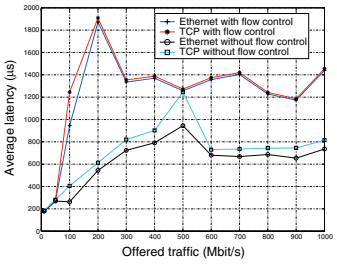
For applications where low latency is important actions must be taken in order to avoid saturation, since a saturated network increase latency. We will discuss this in detail in Chapter 4 and 5.



(a) Irregular topologies.



(b) 8x4 torus.



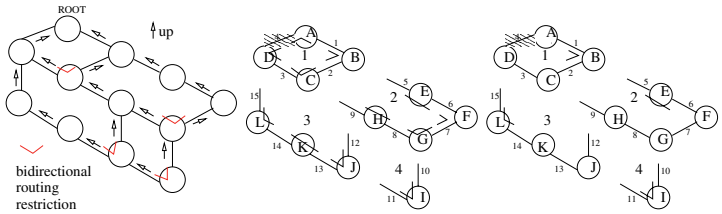
(c) 8x4 mesh.

Figure 3.4: Latency.

3.2 Segment-Based Routing

In the previous section we saw how the poor performance of the SPT can be improved by using the UD routing algorithm. While UD is able to improve performance it still has one major drawback. It is vulnerable to hot-spots around the root of the UpDown tree [34], which severely affects performance. We now suggest the Segment-based routing (SR) algorithm, where we have a larger degree of freedom when enforcing turn-restrictions compared to UD and related algorithms such as FX [57] and L-turn [58]. Furthermore,

[2]The latency introduced by the protocol stack in the end node can be large. As we are concentrating on the features of the network infrastructure the end node complexity has not been studied in detail. For more information on this topic please refer to [53, 54, 55, 56].



(a) Semi-regular topology.

(b) Segmented topology.

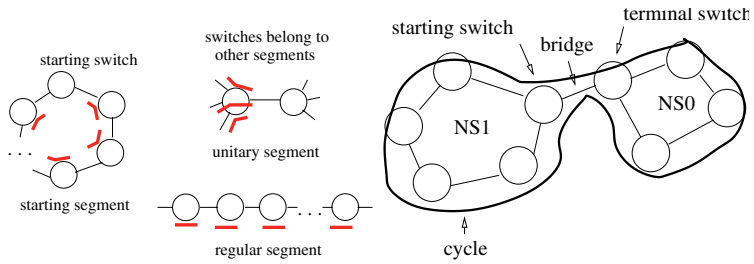
(c) A possible set of restrictions.

Figure 3.5: Segments and routing restrictions.

hot-spot vulnerability is reduced as the routing paths are better distributed throughout the network.

The key concept of the SR algorithm is the partitioning of a topology into sub-nets, and sub-nets into segments. This allows us to place bidirectional turn-restrictions *locally* within a segment. Segments are independent and we can place turn-restrictions within a segment independently from other segments. Figure 3.5(a) shows a semi-regular topology with UD routing restrictions, which we will use as an example. A segment is defined as a list of interconnected switches and links, as shown in Figure 3.5(b). Here we have four segments labelled $1-4$, where segment 1 consists of the switches $\{A, B, C, D\}$ and the links $\{1, 2, 3, 4\}$. Segment 2 consists of switches $\{E, F, G, H\}$ and the links $\{5, 6, 7, 8, 9\}$, and so on for the rest of the segments. All network links belong to one and only one routing segment (i.e. segments are disjoint), and every routing segment, except the initial segment, starts and ends on a switch already part of a computed segment. Furthermore, we group segments into sub-nets. A sub-net is a set of switches and links (i.e. one or more segments) that is connected to the rest of the network (other sub-nets) through only one link (see Figure 3.6(b)).

When the complete topology is partitioned into segments we can add routing restrictions. In Figure 3.5(c) we have enforced one routing restriction in each of the four segments from Figure 3.5(b), these routing restrictions guarantees deadlock-free routing and connectivity. In the first segment, the placement of a routing restriction in any one of the four switches will result in a deadlock-free routing algorithm for this segment. Moreover, as the second segment starts and ends on switches already belonging to the first segment, connectivity among the starting and ending switches is guaranteed through



(a) Segment types and locations for routing restrictions.

(b) Example of links connecting sub-nets.

Figure 3.6: Segment types.

the first segment. Thus, we can place a turn restriction in the second segment, breaking the cycle that can be found through the segment, without worrying about connectivity.

The main challenge is to find the segments, as they are critical to guarantee deadlock-freedom, preserve connectivity, and gain performance.

3.2.1 Segmentation Algorithm

The complete algorithm[3] is shown in Figure 3.7. It consists of the procedure `compute_segments`, which searches for all segments. And the procedure `find`, which tries to find a new segment starting in the switch received as an argument. Throughout the execution of the algorithm, switches and links can be in one of the following states:

- *Not visited.* Initially all switches and links are in the state not visited. This is denoted by the variable `.visited` being *false*.
- *Visited.* A switch or link becomes visited once it is made part of an already computed routing segment. This is denoted by the variable `.visited` being *true*.
- *Temporarily visited.* During the process of computing a routing segment, a switch or link may change state to temporarily visited. Only

[3]The algorithm assumes that a packet will never enter and leave a switch through the same link.

switches and links not marked as visited may be marked as temporarily visited. This is denoted by the variable `.tvisited` being *true*.

- *Starting*. A switch is marked as the starting switch if it is the first switch in the first segment within a sub-net. This is denoted by the variable `.starting` being *true*.

- *Terminal*. A switch is marked as terminal if, among all links, no new segment is found. This is denoted by the variable `.terminal` being *true*.

The `compute_segments` procedure (Figure 3.7) searches for all segments. First, it chooses a random switch as the starting point of the first segment in the first sub-net. Then the selected switch *sw* is marked as *starting* and *visited*, and added to the first sub-net. Second, the `find` procedure is used to find a segment starting in *sw*. Such a segment only exists if it is possible to arrive back at *sw* through a set of switches and links not already *visited*. On success, the `find` procedure updates all the switches and links belonging to the new segment, i.e. all of them are marked as *visited* and as belonging to the current sub-net. On fail, the `find` procedure leaves all links and switches in their initial state. If the procedure fails, it means that there are no new segments reachable from this switch, and the switch is marked as *terminal*. Third, the procedure `next_visited` is used to search for a switch marked as *visited*, belonging to the current sub-net, and with at least one link not marked as *visited*. If such a switch is found it is used to search for new segments as just described. Otherwise, the procedure `next_not_visited` is used to search for a switch that is not marked as *visited*, not marked as *terminal*, and attached to a terminal switch. If successful, a new sub-net is started and a new segment is searched for from this switch. On failure, we know that all switches have been searched and that all switches and links are part of a segment and sub-net.

The procedure `find` is responsible for finding, from a given starting point, a segment ending on a visited switch and made of switches and links not visited. During the search the current switch is marked as *tvisited* and is added to the current segment *segm*. Next, a set of links attached to the current switch is built (*suitable_links*). This set only includes links not marked as *visited*, nor as *tvisited*. If the set is empty, then there are no suitable links and the *find* procedure has failed in finding a new segment. Otherwise, the links in the set are considered in the order found. Order is important, since the segments found may be different if the order of search is changed. When the links are searched, they are first marked as *tvisited*, then added to the current segment *segm*. Then the switch at the other end of the link is inspected. If

```
procedure compute_segments()
var
 s : segment list
 sw : switch
 c : integer # current sub-net
 n : integer # current segment
 end : boolean
begin
 c = 0; n = 0
 sw = random
 sw.starting = true
 sw.sub-net = c
 sw.visited = true
 s[n] = empty
 end = false
 repeat
  if (find(sw,s[n],c))
   n++
  else
   sw.terminal = true
  sw = next_visited()
  if (sw == nil)
  begin
   sw = next_not_visited()
   c++
   sw.starting = true
   sw.sub-net = c
   sw.visited = true
  end
  if (sw == nil) end = true
 until (end)
end procedure
```

```
procedure find(sw, segm, snet) : bool
var
   nsw : switch
begin
 sw.tvisited = true
 segm = segm + sw
 links = suitable_links(sw)
 if (links==nil) begin
  sw.tvisited = false
  segm = segm - sw
  return false
 end
 for each link ln in links begin
  ln.tvisited = true
  segm = segm + ln
  nsw = aTop[sw,ln]
  if ((nsw.visited and nsw.sub-net =
snet) or
                find(nsw, segm, snet)) begin
   ln.visited = true
   sw.visited = true
   ln.tvisited = false
   sw.tvisited = false
   return true
  end
  else begin
   ln.tvisited = false
   segm = segm - ln
  end
 segm = segm - sw
 sw.tvisited = false
 return false
end procedure
```

Figure 3.7: Main procedure for searching segments.

this switch is marked as *visited*, or if a recursive call of the `find` procedure from the neighbour switch returns true (i.e. such a switch is found at a later stage), then a new segment has been found. If we are unable to find a new segment we return with failure.

Figure 3.8 shows an example run of the algorithm. The topology is made of twelve switches connected with fourteen links. We start by randomly selecting a switch, which yields switch I. From switch I we find segment ns_1

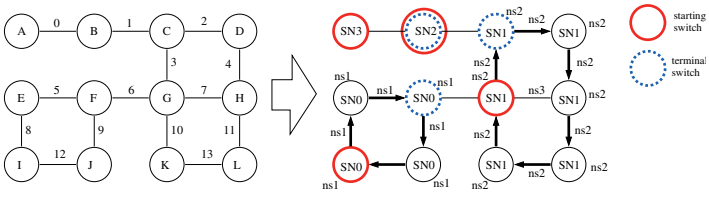


Figure 3.8: Example of computing routing segments.

consisting of the following links and switches: $\{I, 8, E, 5, F, 9, J, 12\}$. This is the only segment that can be found from I since a segment should always end in a visited switch. Next, we search for a switch marked as *visited*, which includes $\{I, E, F, J\}$, and that has a link not visited, which reduces the candidates to $\{F\}$. But, from F no new segment can be found, therefore, the switch is marked as *terminal*. All switches in the segments computed so far belong to the first sub-net (SN_0). Next, we search for a switch not visited and attached to a terminal switch. The unique solution is switch G. At this step, a new sub-net is started (SN_1), and G is marked as *starting* and *visited*. From G a new segment is found: $\{G, 3, C, 2, D, 4, H, 11, L, 13, K, 10\}$. Next, we search for a switch marked as *visited* and with one or more attached links not visited, which results in $\{G, C, H\}$. G is searched and a new segment containing only link 7 is found. Next, from C no new segment can be found and the switch is marked as *terminal*.

We continue by inspecting B, as it is attached to a terminal switch and not marked as *visited*, we decide that no new segment is found. Thus, it is marked as *terminal*. Then we perform the same action on A. When finished, three segments and four sub-nets are found, including four starting switches and three terminal switches.

Through the previous steps we found three types of routing segments:

- *Starting segment*. This type of routing segment will start and end on the same switch, thus forming a cycle. This routing segment will probably be found every time a new sub-net is initiated.

- *Regular segment*. This type of routing segment will start on a link, will contain at least one switch, and will end on a link.

- *Unitary segment*. This type of routing segment consists of only one link.

In order to ensure deadlock-freedom and preserve connectivity, the routing algorithm must enforce routing restrictions in each routing segment as shown in Figure 3.6(a). In particular, for a starting segment, the cycle can be broken by enforcing a bidirectional routing restriction on any switch except the starting one as it could introduce a cycle between two sub-nets. For regular segments cycles are broken by enforcing one bidirectional routing restriction on any of the switches belonging to the segment. Finally, for unitary segments, no traffic can be allowed to cross the link in order to avoid deadlock. Thus, on one side of the link a bidirectional routing restriction must be enforced between this link and every other link attached to this switch.

3.2.2 Segment-Based Routing in Ethernet

The STP is embedded in all conventional Ethernet switches and it makes the configuration of Ethernet equipment an effortless task. Every switch has an instance of the algorithm running and ready to perform the following operations: (a) Elect a *root switch*. The switch with the lowest identifier is selected and becomes the root of the spanning tree. (b) Negotiate the deactivation of ports in order to build a spanning tree. This process avoids loops by making sure that only one switch is responsible for forwarding frames from the direction of the root on to a given link. (c) Listen for changes in the topology (maintenance mode). While in maintenance mode all switches are listening for topology changes, and when a change is detected a new round of negotiations begins. During the negotiation phase routing might be inconsistent.

In order to replace STP with SR we will consider two approaches. The first approach requires that the standard organisations embrace SR and that the firmware in future Ethernet switches are shipped with SR embedded. The second approach provides a less elegant solution, but with some effort it can be used in current of-the-shelf equipment.

Clearly, the best way to embed SR into Ethernet is to replace STP by SR, and deliver auto-configuration support at the same levels as STP currently does. This requires the presence of the SR algorithm in all switches and the support for the following four operations [30]: (i) Elect a master switch. (ii) Collect topology information. (iii) Calculate SR tables. (iv) Distribute routing tables. In step (i) the master switch is selected through negotiation as with the root switch in step (a) above. After the master switch is selected, it collects topology information (ii) from all other switches and creates a complete image of the network. Then, it calculates (iii) and distributes (iv)

routing tables to all switches. Whenever the topology changes, step (ii), (iii), and (iv) have to be repeated. However, due to the segment-based approach of SR the number of switches involved in reconfiguration will be reduced because many changes can be handled locally within a segment. The challenge of this approach is that we must convince the standard organisations to adopt SR.

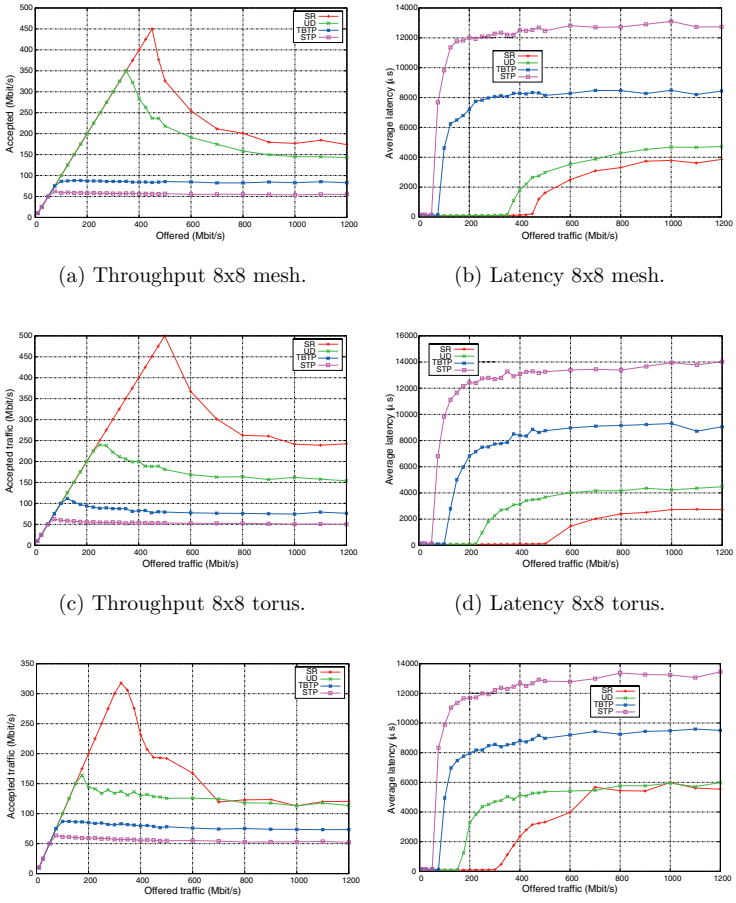
Our second approach does not require any changes to conventional Ethernet switches and relies on features available in all *managed* Ethernet switches. A routing table for SR requires that the route look-up function is able to consider both the destination address and the input port when selecting the output port for the frame to be forwarded. A requirement that is met by most Ethernet switches classified as *managed* switches. This makes it possible to precalculate and then distribute routing tables to the switches involved by the use of the simple network management protocol. The necessary tools must be written and the switches must operate in manual mode (i.e. STP and auto-learning have to be disabled). And, if the topology changes new routing tables will have to be uploaded. Clearly, this approach is somewhat cumbersome as we have no auto-configuration, but the performance gains are significant and can make it worthwhile in a high performance computing context.

3.2.3 Performance Evaluation

We present packet level simulations for a set of regular and irregular topologies. All simulation results have been obtained with an Ethernet simulator developed with the J-Sim framework [48]. We simulate a shared memory Ethernet switch with support for 802.3x flow control and 1 Gbit/s Ethernet links. Each switch has five ports where one is connected to a computing node and up to four are connected to other switches. Our traffic model consists of uniform and pairwise traffic patterns, and a peak rate packet arrival process. The average bit rate is increased in steps from 10 to 1000 Mbit/s (1% - 100% load). The packet size is fixed at 1522 bytes which is the maximum Ethernet frame size.

Regular Topologies

We have studied three sets of regular topologies, an 8x8 torus, an 8x8 mesh, and an 8x8 mesh with 5% link faults. The latter is interesting as it shows the strength of SR when applied to semi-regular networks (regular topologies with link failures). For each topology we have performed simulations for uniform and pairwise traffic patterns.



(a) Throughput 8x8 mesh.

(b) Latency 8x8 mesh.

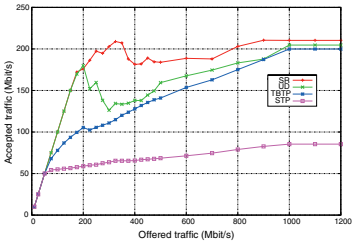
(c) Throughput 8x8 torus.

(d) Latency 8x8 torus.

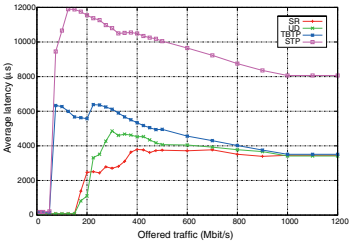
(e) Throughput 8x8 faulty mesh.

(f) Latency 8x8 faulty mesh.

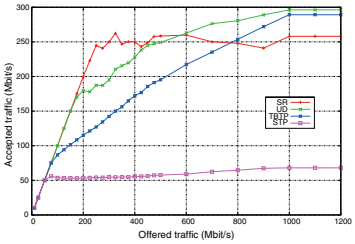
Figure 3.9: Throughput and latency for regular topologies with uniform traffic.



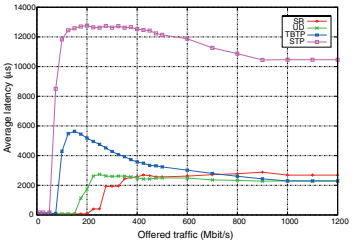
(a) Throughput 8x8 mesh.



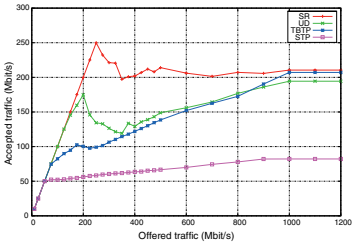
(b) Latency 8x8 mesh.



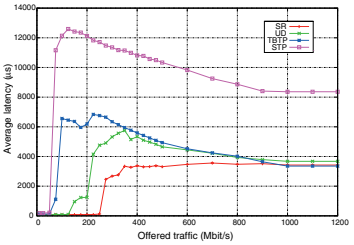
(c) Throughput 8x8 torus.



(d) Latency 8x8 torus.



(e) Throughput 8x8 faulty mesh.



(f) Latency 8x8 faulty mesh.

Figure 3.10: Throughput and latency for regular topologies with pairwise traffic.

The results from an 8x8 torus in Figure 3.9(c) show that SR outperforms all alternatives when uniform traffic is used. UD, being second best, is outperformed by a factor of 2.2. For the 8x8 mesh (Figure 3.9(a)) the trend is maintained, but now SR only outperforms UD by a factor of 1.2. The difference between SR and UD is reduced because there are less links for SR to exploit. Never the less, UD is still affected by early saturation due to the associated hot-spots close the root node. Finally, for a mesh with 5% link faults (Figure 3.9(e)) SR outperforms UD by a factor of 2. Here, SR is able to retain much of the regularity of a full 8x8 mesh even in the presence of faults, while the alternatives suffer from a significant reduction in performance. In general, STP and TBTP are unable to exploit the network since they do not distribute routes evenly in the topology. Instead, they use the links of the spanning tree, which quickly saturates.

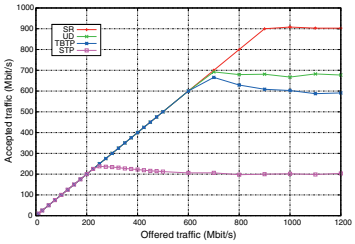
Stability problems are visible for all algorithms, but in particularly for SR with uniform traffic. Figure 3.9(c) shows that throughput is more than halved when SR hits the saturation point. This is caused by unfair flow-control, unfair in the sense that flows with a short path receives more bandwidth than flows with long paths. A flow risks that its share of bandwidth is reduced for each hop towards the destination, since the number of flows it must share the bandwidth with might increase. UD and TBTP also suffer from this phenomenon, but not as severely. This phenomenon can be avoided by the use of virtual channels where packets in different channels do not interfere with each other (see the results in Section 3.3.5) or by the use an age aware flow-control where the packet that has spent the longest time in the network is always selected for transmission [2]. This phenomenon, however, is seldom seen in real life situations because it only occurs in a heavily saturated network. A situation that is avoided whenever possible, usually by admission control or over-provisioning.

Latency closely follows the pattern seen with throughput, as the network reaches saturation latency rapidly increase. Improving latency will be studied further in Chapter 4.

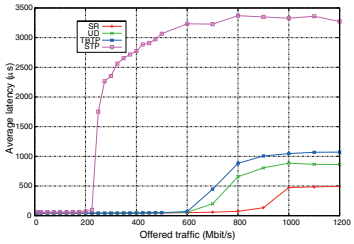
For pairwise traffic in Figure 3.10 we see similar results, but with an overall reduction in throughput as the pairs cause a non-uniform use of network resources causing an early saturation of the network. Latency is also reduced as there is less congestion in the network.

Irregular Topologies

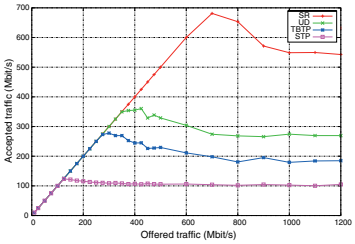
For irregular networks we have studied random topologies with sixteen, thirty-two, and sixty-four switches. The evaluation of different sized networks gives



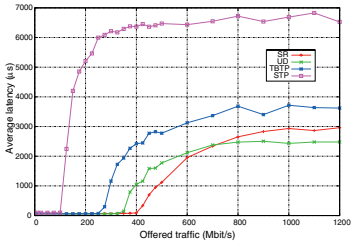
(a) Throughput 16 switches.



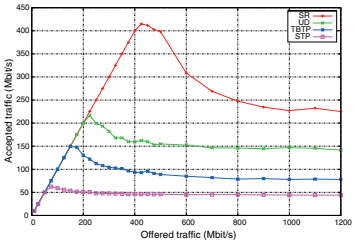
(b) Latency 16 switches.



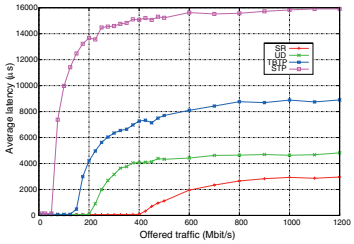
(c) Throughput 32 switches.



(d) Latency 32 switches.



(e) Throughput 64 switches.



(f) Latency 64 switches.

Figure 3.11: Throughput and latency for irregular topologies with uniform traffic.

us an indication of the scalability of each algorithm, as well as a performance evaluation on irregular networks. The switch with the lowest ID has been selected as the starting point for SR, and the segments have been computed using the shortest distance to already visited switches. When enforcing turn-restrictions within a segment we have randomly selected the switch where we enforce the turn-restriction, while source/destination paths have been calculated following the path balancing algorithm described in [59]. This method minimises the deviation of link weight.

In Figure 3.11(a) we see that SR increases throughput by a factor of 1.5 compared to UD, while UD and TBTP are similar in performance. For thirty-two switches (Figure 3.11(c)) the performance difference is 1.84 in favour of SR compared to UD, and 2.4 in favour of SR compared to TBTP. So as the network size grows the performance of UD is reduced compared to SR, for TBTP it is even more so. For sixty-four switches (Figure 3.11(e)) the trend continues, and SR now outperforms UD and TBTP with a factor of 2.0 and 3.2 respectively. SR both perform and scale better than the other alternatives. This is due to the advantage of having an even distribution of traffic across the network generated by its flexibility and locality independence when enforcing turn-restrictions. This combination of a local (segment) and global view of the network makes it possible to ensure better decisions when enforcing turn-restrictions compared to the global only perspective of UD and TBTP. An aspect which becomes more important as the network size grows.

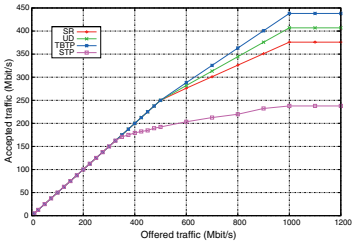
The stability has improved since there are fewer loops in an irregular networks than in meshes and tori. Latency follows the pattern seen with throughput as was the case with regular networks.

Again, the results are similar for pairwise traffic (Figure 3.12), but with smaller differences due to the non-uniform use of the network.

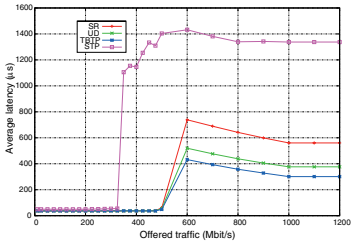
The use of turn-prohibition algorithms (SR, UD, TBTP) improves performance when compared with link-prohibition algorithms (STP), because we do not disable links.

3.3 Layered Routing

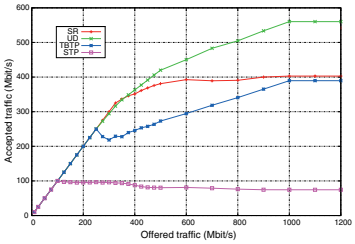
As an alternative to routing algorithms based on turn-restriction such as SR, we now suggest an algorithm, Layered shortest-path routing (LASH), that depends on virtual channels. This algorithm requires more complex network hardware, but the benefits are guaranteed shortest-path routing and higher performance than turn-restriction based algorithms. It can be applied to existing technologies such as the InfiniBand Architecture (IBA) [4] and



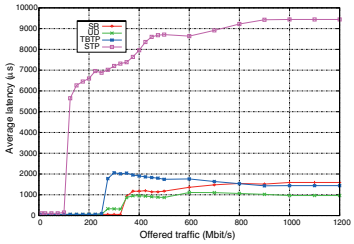
(a) Throughput 16 switches.



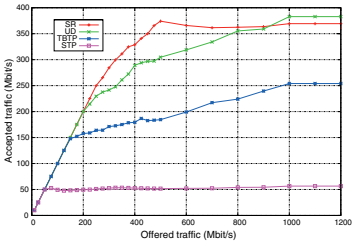
(b) Latency 16 switches.



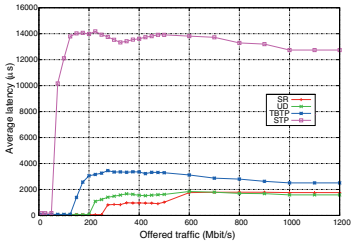
(c) Throughput 32 switches.



(d) Latency 32 switches.



(e) Throughput 64 switches.



(f) Latency 64 switches.

Figure 3.12: Throughput and latency for irregular topologies with pairwise traffic.

Advanced Switching Interconnect (ASI) [5], while Ethernet requires modifications in order to work. This will be clarified in Section 3.3.3.

Layered routing is a concept that uses virtual channels (see Section 2.1.3) to avoid deadlock and guarantee shortest-path routing. The virtual channels are divided into *layers* and network deadlocks are avoided by preventing portions of traffic from using specific layers. This contrasts with SR, which avoid deadlock by preventing data packets from using specific paths and thereby restrict routing freedom (see Section 3.4). Our method increases the performance of deterministic routing using only a limited number of virtual channels. This makes it directly applicable to present-day technologies such as IBA and ASI.

A general problem that arises in irregular networks is the combination of deadlock avoidance and shortest-path routing. As a result, most existing methods for shortest-path routing in irregular networks provide shortest-paths only relative to some constraint. UD, SR, and TBTP are all examples of this, in that they support shortest-paths only relative to the active constraints enforced on turns.

3.3.1 Layered Shortest-Path Routing

In this section, we describe how to use LASH for guaranteed shortest-path routing in any topology. This is done by first identifying a routing function R that finds one shortest physical path between every source and destination. Then we generate a traffic assignment function T that assigns the source/destination pairs to different layers in such a way that freedom from deadlock in each individual layer is guaranteed.

Below, we give an algorithm for mapping source/destination pairs onto virtual layers. We assume that a network I, and a layering L of that network is given. Furthermore, we assume that L has n layers, and that we have an address assignment function A that assigns exactly one address to each traffic node in I.

Step 1: Let $T(\langle s, a\rangle) = \textit{undefined}$ for all source/address pairs, let R_i be empty for all i such that $1 \leq i \leq n$ and let $A(a)$ be undefined for all addresses a.

Step 2: Take one source/destination pair $\langle s, d\rangle$ that has not yet been processed. For an arbitrary shortest-path between this pair do the following:

Step 2.1: Find a new unused address a, and let $A(a) = d$. (For some technologies this amounts to generating a source routing header that contains all routing information; for others, such as IBA, it amounts to assigning a new address to the destination.)

Step 2.2: Find an existing layer L_i such that letting R_i be enriched to support the path, and letting $T(\langle s, a\rangle) = \{L_i\}$ will not close a cycle of dependencies in the layered dependency graph of I. If one exists, let $T(\langle s, a\rangle) = \{L_i\}$, otherwise leave $T(\langle s, a\rangle)$ unchanged.

Step 3: If there are source/destination pairs that have not yet been processed, go to Step 2.

Step 4: (Optional balancing step) Find two existing layers L_j and L_k, and a source/address pair $\langle s, a\rangle$ such that the following is true:

- there are more source/address pairs assigned to L_j than to L_k,
- $T(\langle s, a\rangle) = L_j$,
- letting $T(\langle s, a\rangle) = L_k$ instead and updating R_j and R_k accordingly will not create a cycle of dependencies.

Let $T(\langle s, a\rangle) = L_k$ instead and update R_j and R_k accordingly. Repeat this step until no combination of L_j, L_k, and a source/address pair $\langle s, a\rangle$ with the given properties exists.

If the network contains fewer layers than are needed to provide shortest-path routing, the algorithm fails by leaving the traffic assignment function undefined for some source/address pairs. The method can, however, easily be adapted in such a way that these pairs are assigned to a layer, but are not routed according to their shortest-paths. This can be done either by identifying non-shortest-paths that fit into some layer, or by assigning all of these paths to a separate layer that is routed according to UpDown routing.

The complexity of the algorithm is given by the number of source/destination pairs (N^2) times the number of layers (n), times the complexity of checking for cycles (N). Our tests show that this does not become a problem for modern machines until the networks are quite large (256 switches or more). For larger networks we can circumvent the problem by considering more than one source/address pair at a time. In [60] it is demonstrated that such an approach has only a minor effect on the number of virtual channels needed for shortest-path routing.

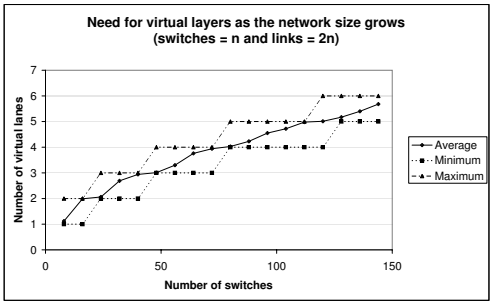


Figure 3.13: Required number of layers.

Required number of layers

An important issue in the evaluation of LASH is the number of layers that are necessary to grant shortest-path routing to every $\langle s, d \rangle$ pair. The required number of layers depends on network size and connectivity, not on the number of hosts. The number of hosts is only limited by the number of ports on each switch.

If we have minimal connectivity so that the network has the shape of a tree, one virtual layer suffices, because no cycle of dependencies can be closed as long as all routing follows a shortest-path. Furthermore, networks with maximal connectivity also need only one layer, because no packet will traverse more than one link and then no channel dependencies can exist. But most network topologies that are used in practise will fall somewhere between these two extremes. For networks with 16, 32, 64, and 128 switches and any connectivity, covering all $\langle s, d \rangle$ pairs (i.e. all switches) requires a maximum of 3, 3, 5, and 6 layers, respectively (Figure 3.13). The variance in the required number of layers is small. For a set of one hundred random topologies the difference between the most demanding and the least demanding topology was never more than one layer.

3.3.2 Ethernet Flow-Control Granularity

Recall our discussion of Ethernet flow-control in Section 3.1.2, where we briefly mention the incompatibility between Ethernet's flow-control mechanism and its priority mechanism. This is problematic for the use of LASH in Ethernet switches because LASH requires virtual channels. Ethernet does

not explicitly support virtual channels, but the IEEE 802.1Q standard introduced priority tagging of Ethernet frames, where each frame can have a priority from 0-7, and each priority has dedicated buffering resources. We will exploit this priority mechanism as virtual channels. Figure 3.14(a) shows the format of a IEEE 802.1Q compliant Ethernet frame. The three bits labelled priority are used to indicate the priority of a given frame. To enable the concept of virtual channels we change the semantics of these three bits from priority to *virtual channel identifier*, which allows for eight virtual channels. Thus, we have support for eight virtual channels and one of our two requirements for LASH routing is satisfied.

Our second requirement is virtual channel flow-control, but Ethernet only supports per port flow-control. This makes it impossible for flow control on a per-priority basis. Not only does this limit the performance of the priority scheme itself [52], but it makes it impossible to support virtual channels in combination with flow-control. Since, when flow control is enabled, the virtual channels no longer have independent buffering resources.

3.3.3 Layered Routing in Ethernet

In order to use LASH in Ethernet we need to change the granularity of Ethernet flow-control. We suggest a simple change to the Ethernet flow control mechanism to allow for per-priority flow-control.

The pause frame that relay the on/off message must include a field that tells the receiver what priority it should pause. This can be done by either changing the current MAC control frame or introduce a new one. Changing the current frame format requires that the priority value is embedded in the zero-padding following the pause time (Figure 3.14(b)). This makes it possible to extract the priority constraint for a pause frame by reading the first three bits after the pause time. Thus, per-priority aware switches will be able to find per-priority information, while it will be ignored by other switches. The introduction of a new control frame is quite similar, but now we create a new *OpCode* to distinguish between pause frames. The new pause frame will contain the same information in the same place, but with a different OpCode (Figure 3.14(c)). In this case per-priority aware switches will use the OpCode field to decide the type of flow-control received and it will then scan for the necessary information accordingly. Per-port aware switches will use pause frames with $OpCode = 1$ as before, while pause frames with $OpCode = 2$ will be ignored. This change is currently considered in the IEEE 802.3ar Congestion Management Task Force, but in the context of congestion control.

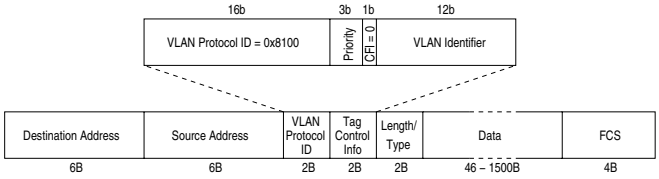
Independent of the strategy chosen, the main change is the inclusion of priority information in the pause frame that enables the concept of virtual channel flow-control. The STP can now be replaced or extended (for backwards compatibility see the next section) by LASH. Such a replacement can be either centralised or distributed. In a centralised approach all switches will go through an election process where a *master* is selected based on identity or another property (e.g. it could be pre-configured). The master switch builds and maintains a topology map based on the information collected from all the other switches. The topology map is then used by the master switch to calculate and distribute routing tables. In the distributed scheme a process similar to *link-state routing* can be applied if we exchange the random selection of layers in LASH with a deterministic method. In a distributed approach every switch in the network builds and maintains a topology map and independently calculates the routing table necessary to reach all other destinations in the network. This approach avoids single point of failure and it reduces the amount of control traffic in the network since only topology information is passed between switches. Either way, self-management can be similar to what we have in the STP today.

3.3.4 Layered Routing and QoS

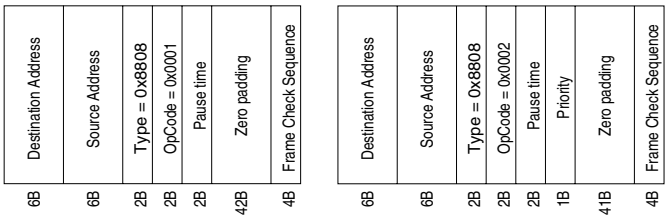
As described in Section 3.3.1 LASH needs a modest number of layers, but when combined with service differentiation the number of layers used is increased. A network with thirty-two switches needs three layers for routing, which leaves us with five layers for other purposes such as QoS. E.g. if we have a network with thirty-two switches we can have a maximum of two classes of service, using a total of six layers. Three of these layers are required for LASH routing, but as we need separate queues for each of the classes of service we end up with a total of six layers. For large networks it is possible to avoid the limit of eight layers by only using seven layers for LASH and reserving one layer either for UD or TBTP (or even STP). Then all paths that can not be routed deadlock free within one of the seven layers can be routed according to UD or TBTP in the eight layer. This layer could then be dedicated to best effort traffic. This removes the size constraint, but it also reduces the performance compared to a network only using LASH.

Backwards Compatibility

To provide a convenient upgrade path LASH enabled switches should be backwards compatible with older switches. This enables a gradual upgrade



(a) Data frame.



(b) Pause frame.

(c) New pause frame.

Figure 3.14: Ethernet frame formats.

of network equipment, where an *island* of LASH enabled switches can work together with switches not supporting LASH. Backwards compatibility can be achieved by requiring new switches to support both STP and LASH, and dedicating virtual channel 0 to STP. How frames are routed will then be determined by the type of switch the source and destination refers to. LASH switches will route according to LASH for all LASH destinations within its island, and according to STP for all other destinations. STP switches will route according to STP for all destinations.

LASH capable switches (LASH switches) will recognise each other as LASH switches, and STP capable switches (STP switches) as STP switches. While STP switches will recognise both STP and LASH switches as STP switches. A LASH routing domain consists of a set of connected LASH switches forming a LASH island, but at the same time these switches are part of a spanning tree of the complete topology together with the STP switches.

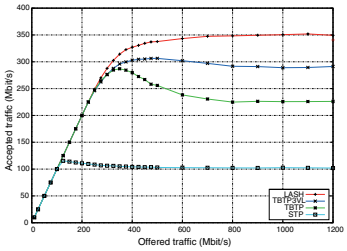
Since all LASH switches use per-priority flow-control, while the STP

switches use per-port flow-control we must make sure that the exchange of pause frames between STP and LASH switches is done correctly. The STP switches do not understand per-priority flow-control, and the LASH switches only sends frames on virtual channel zero when connected to an STP switch. Therefore, the priority field remains unused between LASH and STP switches. But when a LASH switch propagates pause frames initiated on a port connected to an STP switch to other LASH switches the virtual channel identifier (priority field) should be set to zero to avoid interference with the virtual channels used for LASH routing. This approach will, of course, reduce the overall performance compared to a strictly LASH enabled network, but allows for a gradual replacement of equipment.

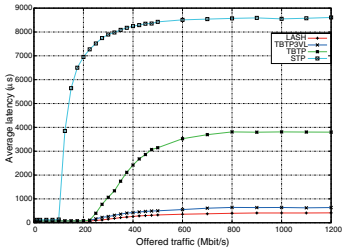
If we want to combine LASH, STP and QoS, everything becomes more complex. We must now map between the priorities used in the STP domain and the LASH domain. In the STP domain we have from zero to eight priorities, while in the LASH domain the number of priorities available depends on the network size. In a network with thirty-two switches we need to dedicate three priorities to LASH *for each* priority we want to use for QoS. Thus, we end up with six priorities dedicated to LASH routing and two priorities dedicated to the STP. In the worst case we need to map eight priorities to these two priorities, which can be done by mapping the first four to priority zero, and the last four to priority one. The mapping from eight to two priorities might seems coarse, but experience shows that two to four service classes are often enough [30, 61].

3.3.5 Performance Evaluation

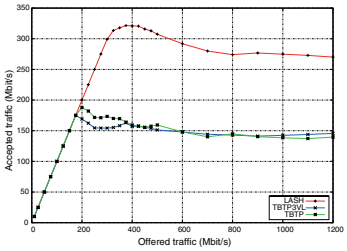
Again we present packet level simulations for a set of regular and irregular topologies. And, as before, all simulation results have been obtained with an Ethernet simulator developed with the J-Sim framework [48] as described in Section 3.2.3. We simulate a shared memory Ethernet switch with support for 802.3x flow-control and 1 Gbit/s Ethernet links. Each switch has five ports, where one is connected to a computing node and up to four are connected to other switches. Our traffic model consists of uniform traffic patterns and a peak rate packet arrival process. The average bit rate is increased in steps from 10 to 1000 Mbit/s (1% - 100% load). The packet size is fixed at 1522 bytes, which is the maximum Ethernet frame size. The link length is ten meters with a propagation delay of 1.14e-6 seconds, which corresponds to Gigabit Ethernet over UTP. Each run simulates two seconds of real time on each topology and the average throughput and latency is then calculated from the observed results.



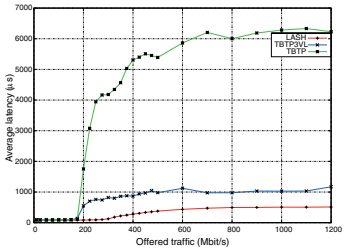
(a) Throughput for irregular topologies with 32 switches.



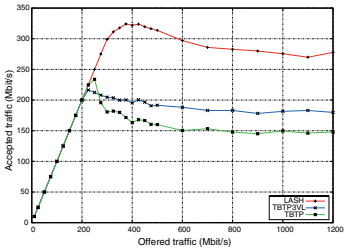
(b) Latency for irregular topologies with 32 switches.



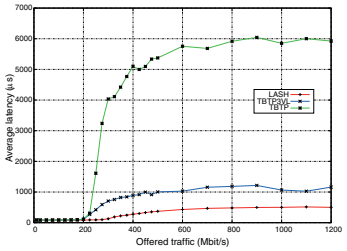
(c) Throughput for 8x4 mesh.



(d) Latency for 8x4 mesh.



(e) Throughput for 8x4 torus.



(f) Latency for 8x4 torus.

Figure 3.15: Throughput and latency.

We compare the results for LASH with TBTP and STP, the latter is the default routing algorithm for Ethernet and the former is a recent proposal for Gigabit Ethernet. TBTP was presented in Section 2.1.4. In order to have a fair comparison between LASH and TBTP we have included results for TBTP where the algorithm is allowed to use the same number of channels as LASH. This means that the network traffic is spread over several layers instead of one. These results are labelled TBTPnVL in Figure 3.15, where n is the number of layers used. Results for STP are included as a reference. All results are for Ethernet where per virtual channel flow-control is available as discussed in Section 3.3.3.

Throughput

Throughput results where generated for a 4x8 mesh, a 4x8 torus, and irregular networks with sixteen (not shown) and thirty-two switches. Figure 3.15(a), 3.15(c) and 3.15(e) summarise the average throughput for the different routing algorithms.

It is evident that LASH achieves the highest throughput, it outperforms both TBTP and STP because all links are used, no turns are forbidden and all frames are routed via shortest-paths. Thus, we see an average increase in throughput of 16% compared to TBTP3VL for irregular networks (Figure 3.15(a)). For the 4x8 torus the increase is 25% (Figure 3.15(e)) and for the 4x8 mesh an immense 83% (Figure 3.15(c)). For TBTP3VL the improvement over TBTP varies as the improvement yielded by adding more layers depends on the topology. An improvement of 25% and 20% is achieved for irregular networks and the torus respectively. On the 4x8 mesh the addition of more layers does not benefit TBTP so the performance of TBTP and TBTP3VL is similar. This shows that the addition of further layers gives TBTP some room for improvement, but as the number of turns allowed and the available paths are still the same the performance increase is limited.

LASH is stable and without any drop in performance when saturation is reached. This is due to the use of virtual channels. This also benefits TBTP when used with several channels, but as TBTP depends on a spanning tree a drop in performance is seen.

Latency

The latency measurements show only end-to-end network delay without the queueing time in the hosts. Figure 3.15(b), 3.15(d) and 3.15(f) shows the average latency. As can been seems from the figures, LASH improves latency

when compared to the other alternatives. In Figure 3.15(b) LASH has an average latency of 450μs which is a 25% reduction compared to the 600μs of TBTP3VL, and a 89% reduction of that of TBTP. For the torus latency is reduced by 55% between LASH and TBTP3VL, and finally for the mesh the reduction is 52%. No turn-prohibition in combination with shortest-path routing are the reason for these improvements for LASH. For the TBTP3VL the addition of more layers leads to a reduction in latency compared to only one layer. This is due to less head of line blocking when several layers are available for the same path.

3.3.6 Layered Routing in the InfiniBand Architecture

The concept of layers as described above maps directly to the virtual channel mechanisms supported in IBA (see Section 4.2) and ASI. Moreover, since LASH only needs a limited number of virtual channels, even for large network sizes, it is directly applicable to these technologies without the changes required for Ethernet. The performance results for these technologies are similair to the above results, but with regards to IBA it is possible to enhance LASH to support oblivious routing [6].

In the LASH algorithm just described, we assumed a priori that only one shortest-path was chosen. IBA does, however, allow for several paths to exist between any pair of nodes by defining multiple addresses for each destination. The routing table can then be set up so that each address follows a different path to the same destination. This added freedom can be exploited within the framework of layered routing by using source adaptivity, where all shortest-paths are allowed and the choice between them is made by the source node. In [6] we show that a source adaptive version of LASH, called MP-LASH, improves performance compared to deterministic LASH when non-uniform traffic patterns are used.

3.4 Contributions and Related Work

In this chapter we first studied the performance of Ethernet with regards to routing and flow-control, then we proposed Segment-based routing, and finally we proposed Layered shortest-path routing.

In Section 3.1 we reviewed how conventional use of Ethernet has severe performance limitations, which we illustrated through the use of TCP as a higher layer protocol. Furthermore, we showed how to improve performance

by activating flow-control and replacing the routing algorithm. The activation of flow-control turned Ethernet into a loss-less network where we avoided triggering the TCP congestion control mechanism, while the replacement of the STP allowed us to use any topology without having to worry about deadlocks. Combined these changes gave a throughput increase of 60%. This results shows the impact that link layer optimisations has on higher layer protocols, and in the end for applications.

Others have studied the effect of Ethernet flow-control on TCP congestion control and how TCP benefits from link layer flow-control, such as [52, 62, 63, 64]. Furthermore, W. Noureddine et al. have proposed several improvements to the current mechanism by increasing the flow-control granularity from port based to source/destination based [52, 65]. Common for all of the above studies are that they only consider scenarios with one or two switches and that they do not consider the deadlock problem.

In the Section 3.2 we proposed the Segment-based routing algorithm, where the novelty resides in the introduction of a *locality independence* property. This property adds a new dimension to the enforcment of routing restrictions, and allows us to reduce the restrictions compared to current turn-based algorithms. The concept of segments used in SR makes it possible to exploit the semi-regularity found in irregular topologies, such as meshes and tori with faults. Our studies showed that SR increases performance compared to other routing algortihms suggested for Ethernet. Furthermore, SR is similar in performance to state-of-the-art turn-based algorithms such as FX [57] when used in regular topologies, while it supersedes FX for irregular topologies [7].

Several routing algorithms have been suggested for improving Ethernet, such as SmartBridge [66], STAR [67], OSR [68], Viking [69], and VLAN-Based Minimal Paths [70]. But all of these strategies have been designed for lossy networks, without addressing the deadlock problem. Others, such as L-turn [58], smart-routing [71], and TBTP [37] include deadlock avoidance, but their performance is lower than that of SR and LASH. Finally, we have a contribution from M. Karol et al., where they suggest a deadlock prevention scheme for Ethernet using advanced buffer management [72]. This is a novel approach, but with the drawback of changing the semantics of the pause frame and adding extra housekeeping to the switches, which makes it incompatible with current off-the-shelf Ethernet equipment. Furthermore, it is unclear how this scheme would work with multiple priorities and it has not been evaluated with regards to Ethernet.

In Section 3.3 we presented the concept of layered routing and the LASH algorithm. In layered routing network resources are divided into layers and deadlocks are avoided by preventing portions of traffic from using specific

layers. This contrasts with the turn prohibition approaches such as SR that avoid deadlock by preventing data packets from using specific paths and thereby restricting routing freedom. Layered routing increases the performance of deterministic and oblivious routing, and it needs only a limited number of virtual channels even for large networks. This makes it directly applicable to present-day technologies such as IBA and ASI. And, as shown in Section 3.3.3, suitable for Ethernet with only a minor change to the Ethernet flow-control protocol. Moreover, this proposal can serve as an alternative to the solution proposed by the IEEE 802.1aq Shortest-Path Bridging task force, which currently suggests a solution using multiple spanning trees, but without any deadlock avoidance when lossless operation is used.

Several routing algorithms related to layered routing exist, including the approach taken in the Avici Terabit Switch/Router [73], where separate virtual networks (layers) are used for each destination port in a torus topology. This approach will, however, require a high number of virtual channels for large networks. It is therefore not suited for present-day technologies such as IBA, ASI, and Ethernet. Other related work aims at increasing the switch adaptivity, such as [74], where Linder and Harden achieved deadlock-free, minimal, and adaptive layered routing using virtual channels for regular networks, in particular for k-ary n-cubes. This method does not, however, generalise easily to arbitrary topologies, and the need for virtual channels grows exponentially with n. Yet another idea is to order the layers. Packets escape from possible deadlocks in a higher layer by making a transition down to a lower layer. If the lowest layer is deadlock-free, so will the entire system be [75, 76, 77, 78, 79]. There are problems with all these approaches. Some of them require extra functionality in the switches or the hosts that not all technologies provide. Others aim at maximising adaptivity, which results in out-of-order delivery. The extra protocol overhead involved in sorting the packets at the destination is, in some cases unacceptable, and this is the main reason why many technologies only use deterministic routing.

3.5 Critique

Th SR algorithm improves performance through a more flexible way of enforcing turn-restrictions, but this flexibility makes it harder to find the optimal solution since the solution space is so large. In Section 3.2.3 we also observed that SR results in unstable operation when the network is saturated. This is caused by unfair flow-control, but is nevertheless a weakness that can make SR unsuitable for certain applications. Furthermore, the way we apply SR

to Ethernet i Section 3.2.2 disables auto-configuration. This makes network management harder.

LASH routing uses virtual channels for deadlock avoidance and shortest-path routing, and as the network size increases so does the number of channels required. This can be a problem with large networks, even if we have shown that LASH only requires a moderate number of layers. Another problem with the use of virtual channels is that they are, in many cases, intended for other use such as QoS or traffic isolation, which reduce the number of layers available for other purposes.

Finally, LASH is easily used in technologies such as IBA and ASI, but Ethernet requires some changes as described in Section 3.3.3. These changes, even if simple, must be adopted by the correct standardisation organisations, which is hard to realise.

3.6 Further Work

The incompatibility between flow-control and priorities in Ethernet is an issue that should be fixed. A possible fix has been proposed in Section 3.3.2 and in [52], but the standardisation organs and equipment vendors remain to be convinced.

The SR algorithm might be improved in two ways. First, the way routing restrictions are placed can be optimised according to some criteria, such as throughput or load balancing. Second, the way segments are used can possibly be exploited for fault-tolerance. For LASH it is a challenge to combine it with QoS in networks where virtual channels are scarce. It is also interesting to see how well it scales, with regards to the number of virtual channels required, for networks with thousands of switches.

For routing in general the field is well understood for both regular and irregular topologies. The challenge for the future is to combine efficient routing with QoS and fault-tolerance, which is difficult since the virtual channels required for these features are a scarce resource. Either other ways to solve these problems must be found or the number of virtual channels available in future technologies must be increased.

Chapter 4

Service Differentiation

When discussing Quality of service (QoS) in interconnection networks there are three properties of significant importance, *bandwidth, latency* and *packet loss*. Packet loss is avoided by using flow-control, which was discussed with regards to Ethernet [3] in Chapter 3.1.2. In Chapter 3 we also discussed improvements in throughput and latency from the point of routing. The routing algorithm does not, however, differentiate traffic, nor does it guarantee bandwidth or latency. Thus, with regard to latency and bandwidth guarantees, a combination of mechanisms are necessary, these are *service differentiation* and *admission control*. In this chapter we will combine the LASH routing algorithm from the previous chapter with a scheme for service differentiation in order to reduce latency and improve throughput. In the next chapter we will expand this with admission control.

In Section 4.1 we describe how Differentiated Services (DiffServ) [8] can be applied to interconnection networks, in this case to the InfiniBand Architecture (IBA) [4]. Then in Section 4.2 we give an overview of the QoS mechanisms supported by IBA. This is followed by simulation results and an evaluation of the proposed scheme in Section 4.3.

4.1 Differentiated Services

The Internet Engineering Task Force has provided the Internet community with several QoS concepts and mechanisms. The best known are Integrated Services [39], the Resource reservation protocol [41], and DiffServ [8] as described in Section 2.2.1. We will apply the DiffServ philosophy to intercon-

nections networks through the use of the available QoS mechanisms found in IBA.

DiffServ assumes no explicit reservation mechanism in the interior network elements. QoS is here realised by giving data packets differentiated treatment relative to QoS header information (see Section 2.2.1). Even if this mechanism has been suggested with the Internet in mind, the concepts are also valid for interconnection networks. Interconnection network technologies such as IBA and Advanced Switching Interconnect (ASI) [5] support several QoS features, while Ethernet [3] is more limited. The DiffServ concept can be adapted to any of these technologies, but we focus on IBA as it has the most complete feature set.

4.2 QoS in InfiniBand

IBA is a switch-based network architecture first standardised in 2000 [4]. It is designed for high performance, high connectivity applications, and is suitable for both internal and external interconnects. IBA has built in features to support QoS, which makes it a strong candidate for system area networks and for advanced applications in high performance computing. IBA consists of the following four elements: Channel adaptors, Switches, Routers, and Subnet managers. Channel adaptors are IBAs name for network interfaces and are interconnected through IBA switches.

IBA networks are often referred to as subnets. A subnet is comparable to a local area network. As two or more local area networks must be interconnected with a router so must two or more IBA subnets. An IBA subnet requires at least one subnet manager, which is responsible for configuring switches, routers and channel adaptors in the subnet. Whenever the network changes (e.g. a links go down, a device is added, or a link is removed) the subnet manager must reconfigure the network accordingly.

4.2.1 QoS Mechanisms

IBA supports four mechanisms for QoS: *Service levels*, *virtual lanes* (channels), *virtual lane weighting*, and *virtual lane priorities*.

A service level (SL) is a field in the packet header that denotes what type of service a packet shall receive as it travels toward its destination. This corresponds to the packet marking approach described in DiffServ and can be used to implement per hop forwarding rules. IBA supports 16 SLs, but

how to use these SLs is not specified.

To complement its sixteen SLs IBA also supports a maximum of sixteen *virtual lanes* (VLs). VLs are logical channels on the same link, but with separate buffers and flow control as described in Section 2.1.3. A minimum of two VLs must be supported, VL0 must be supported as the default data lane and VL15 must be supported for dedicated subnet management traffic. The sixteen SLs are mapped to the corresponding VL by the SL number, i.e. SL12 is mapped to VL12. If a direct SL to VL mapping is not possible the SL will be degraded according to a SL to VL mapping table. In the worst case only one data VL is supported and all SLs will be mapped to VL0.

Virtual Lane Weights and Priorities

Each VL can be configured with a weight and a priority, where the weight is the proportion of link bandwidth this VL is allowed to use and the priority is either high or low. If we include control lanes we get the following arbitration hierarchy:

Level 1: Control lanes have the highest priority and will preempt anything else.

Level 2: High priority lanes preempts low priority lanes, but to ensure forward progress of low priority lanes a parameter called *limit of high-priority* is used. The limit of high-priority is the maximum number of packets that can be scheduled on a high priority lane before a packet *must* be scheduled on a low priority lane — if there is one waiting.

Level 3: Arbitration between lanes with the same priority is done by a weighted fair arbitration scheme. The weight assigned to a lane decides the number of 64 byte blocks it is allowed to send when its turn occurs. Lanes are scheduled in a round robin fashion.

When a lane is scheduled it is marked active and allowed to send up to the number of blocks specified for the VL. The weight counter for the VL is decreased for each packet sent and the lane is rescheduled as long as its weight is larger than zero and no lane with higher priority has data to send. When the weight reaches zero it is reset and the next lane that has data to send is marked active.

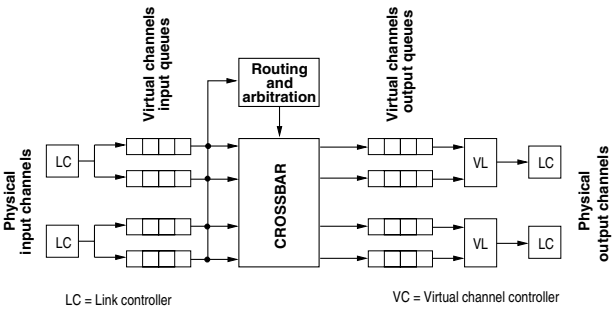


Figure 4.1: Switch architecture.

4.2.2 Switch architecture

In our simulations we are modelling a switch with the above features. The overall design is based on the canonical router architecture described in [1]. The architecture is flit based and uses virtual cut-through switching [24]. The switching core consists of a crossbar where each link has dedicated access and supports sixteen VLs (Figure 4.1). The VLs on the same link have multiplexed access to the crossbar. Thus, there is flit level scheduling of lanes over the crossbar, but there is no flit interleaving between lanes. For each link, only one packet is allowed to traverse the crossbar at a time.

Each VL has its own buffer resources that consist of an input buffer large enough to hold a packet and an output buffer large enough to hold two flits in order to increase performance. VL arbitration is done at the input side to select which VL is allowed to send next. Link arbitration at the output link is done to select which input is the next to send to this output. Output link arbitration is done in a round robin fashion.

There is always a one-to-one mapping of SL-to-VL and each VL is assigned a SL at start-up without any possibility for change during operation. This excludes run-time reconfiguration, but makes the scheme simpler and is in line with the DiffServ philosophy. An alternative would be to dynamically reconfigure the SL-to-VL mapping through a subnet manager as done in [80], but this is incompatible with the DiffServ philosophy.

4.2.3 Routing

As a routing algorithm we could use any algorithm compatible with InfiniBand, but we have chosen to use LASH as presented in the previous chapter. As IBA supports VLs we can directly apply LASH to this architecture, whereas for Ethernet we had to make changes to the flow-control scheme. Furthermore, LASH guarantees shortest-path routing, which is important for QoS because it reduces latency.

The combination of LASH and service differentiation requires additional layers compared to LASH with only a single service class. In a network with thirty-two switches we need to dedicate three lanes to LASH for each class of service. If we have five service classes, as we will use in our evaluation, we need need a total of fifteen lanes (3×5).

4.3 Performance Evaluation

We now endeavour to provide QoS in InfiniBand or any similar cut-through network by adhering to the DiffServ philosophy. We approach the problem by studying the provision of QoS without any explicit admission control mechanism. We start by carefully examining the sensitivity of different QoS properties under various load and traffic mixture conditions, including the effect of back-pressure as created by flow-control. These experiments give valuable information regarding the QoS behaviour of cut-through networks when used as a pure relative service model. Specifically we study (i) the effect of using VLs with a weighted arbitration scheme to do throughput differentiation, (ii) the robustness of a weighted arbitration scheme when VL load and weight is unbalanced and (iii) the latency and jitter characteristics of VLs with a weighted arbitration scheme.

We present results for topologies with thirty-two switches and 160 nodes, with five nodes connected to each switch and a maximum of ten links per switch. Traffic is modelled by a normal approximation of the Poisson distribution and the address distribution is random pairs, where each source sends to only one destination and no destination receives packets from more that one source. The topology consists of sixteen randomly generated irregular topologies. Simulations have also been performed on networks with eight and sixteen switches with similar results.

The five different end nodes send traffic on five different service levels (Table 4.1) where SL 1 and 2 are of the expedited forwarding (EF) class in DiffServ terminology. SL 3 and 4 are of the assured forwarding (AF) class and SL5 is best effort (BE) traffic.

SL	DA Eq.	Load	BW	UW	Pri
1	EF	10	4	6	high
2	EF	15	6	1	high
3	AF	20	8	6	low
4	AF	25	10	1	low
5	BE	30	1	1	low

Table 4.1: The service levels used in all simulations.

4.3.1 Throughput

Our first results are from a balanced configuration under increasing load conditions. In the balanced configuration the weighting of each SL is according to the load on that SL, i.e. the SL with the highest load also has the highest weight — except for SL5 which always has the lowest weight. This makes the configuration vulnerable to changing load conditions where a mismatch between applied load and VL weight will cause trouble.

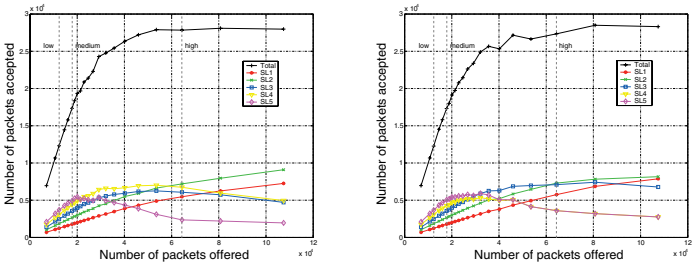
Figure 4.2(a) shows the throughput of each SL as well as the total throughput. As can be seen from the figure throughput is differentiated. Throughput for all SLs is consistent with the weighting when the network is below saturation since there is enough bandwidth for everyone. When the network reaches saturation low priority SL throughput is reduced as high priority SLs preempts low priority bandwidth, which is consistent with our configuration.

4.3.2 Robustness

We have considered a well-behaving configuration where we require VL weights and SL load to be matched. This is not always the case so we have looked at how an unbalanced configuration performs. In this configuration the weights for SLs with the highest load have been swapped with the weights for SLs with the lowest load. Figure 4.2(b) shows the throughput of all SLs as well as the total throughput. From the figure we see that the performance is almost identical to the previous configuration as long as the network is below saturation. Only when the network reaches saturation does the mismatch between

[1]Balanced weight. With this configuration the proportion of applied load matches the assigned SL weights.

[0]Unbalanced weight. With this configuration there is a mismatch between the applied load and the assigned SL weights, i.e. the network is mis-configured.



(a) Balanced weights.

(b) Unbalanced weights.

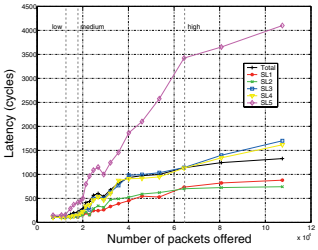
Figure 4.2: Throughput for a network with thirty-two switches.

weighting and load become visible. As long as we are below the saturation level there is enough bandwidth for everyone and there is no problem fulfilling demands, but when the network gets saturated the weighting takes effect as we saw in the previous configuration. Only now the wrong SLs preempts the bandwidth because of the mismatched weights. In Figure 4.2(b) this is visible for SL4 which should be just above SL3, but is actually at the level of SL5.

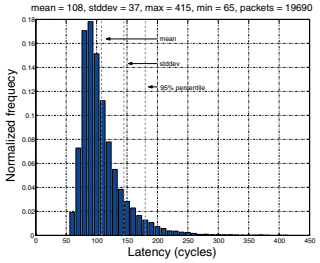
From this we can conclude that the weighting configuration is *not* crucial in a network below saturation. It is crucial when the network is in saturation, but in a QoS setting we want to avoid a saturated network. Thus the weighting configuration can be considered robust in a non-saturated network and the need for on the fly reconfiguration of the SL weights is not necessary. This encourages over provisioning when possible, because when there is enough bandwidth we do not need QoS. Unfortunately, over provisioning only works when bandwidth is cheap, which is not the case in most HPC scenarios. For such use we need to avoid saturation while still making efficient use of the bandwidth available. Avoiding saturation can be done with admission control as we will see in Chapter 5.

4.3.3 Latency

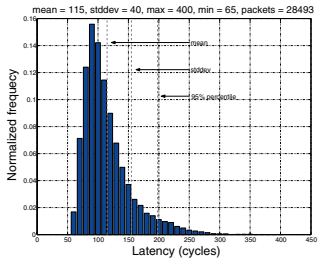
We have seen that we are able to differentiate traffic with regards to bandwidth, we will now study the latency characteristics. Recall that we measure



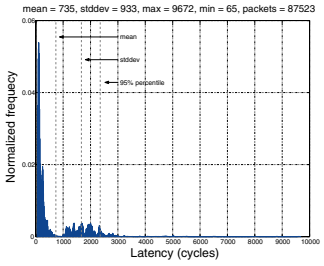
(a) Average latency.



(b) Latency distribution for SL1 packets at low load.



(c) Latency distribution for SL1 packets at medium load.



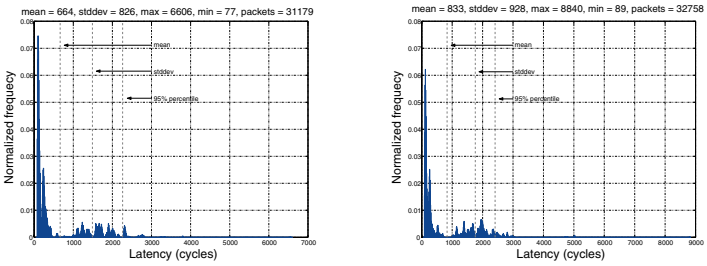
(d) Latency distribution for SL1 packets at high load.

Figure 4.3: Latency for a network with thirty-two switches.

latency from the time a packet enters the network until it leaves the network. Queueing time in the transmitter is not included in the measured latency. Figure 4.3(a) shows the average network latency for individual SLs and for all traffic. The latency is low as long as the load is well below saturation, but compared with Figure 4.2(a) we see that low priority SLs suffer from an increase in latency at the same point as the throughput graph starts to level out. The same effect can be seen for high priority SLs, but at a point closer to saturation. The low priority SLs suffer when we are unable to linearly increase throughput. At this point the high priority SLs takes away bandwidth from the lightly weighted low priority SLs. They have used most of the free bandwidth up to this point, but now the number of low priority packets preempted by high priority packets increase and we see a rise in latency for these SLs. When the network approaches saturation this also affects high priority SLs. Because the network is highly loaded the number of packets affected by the back-pressure mechanism increase, which cause delays and increased overall latency. Again we see a need to keep the network below saturation, preferably in the linear area of the throughput graph. This can, as mentioned earlier, be achieved with a suitable admission control mechanism. For latency sensitive high priority traffic, admission control can be used to keep the load at a level necessary to ensure low latency. Low priority traffic can be admission controlled based on bandwidth requirements, while best effort traffic can be left out of an admission control scheme since this type of traffic only use leftover resources. We will study admission control in Chapter 5.

4.3.4 Jitter

Let us now turn our attention to the jitter characteristics. Figure 4.3(b) shows the latency distribution for SL1 traffic at the load level marked as *low* in Figure 4.2(a) and 4.3(a). The mean, standard deviation and the 95 % percentile are marked with a dashed line in the graph. The distance between the mean mark and the standard deviation mark reflects the standard deviation. The histogram has a sharp peak and a short tail and the standard deviation is low. The 95% percentile is 180 cycles so 95% of the packets has a latency of 180 cycles or lower. For *medium* load, shown in Figure 4.3(c), the situation is slightly worse. Here we have a low average latency and a 95% percentile at 198 cycles. Moving on to *high* load level in Figure 4.3(d), things are worse since we now have moved out of the linear area of the throughput graph. The jitter potential is substantial with a inter-quartile range of 1262 cycles and the 95% percentile at 2356 cycles.



(a) 3 hops.

(b) 4 hops.

Figure 4.4: Latency histograms per hop for SL1 traffic at high load.

To investigate the jitter issue further Figure 4.4 shows per hop latency histograms for SL1 traffic at high load with three and four hops which where the most frequent path lengths. The interesting part here is how much the latency properties increase when the path length is increased with one hop. The 95% percentile increases from 2262 cycles to 2410 cycles and the inter-quartile range increases from 1125 cycles to 1478 cycles as well as substantial increases for mean, standard deviation and maximum observed latency. The same behaviour can be seen for other path lengths as long as there are a substantial number of packets. This effect can be explained by the back-pressure caused by the use of flow-control. When a packet somewhere in the network is delayed by back-pressure the packet latency for this packet is increased, but the latency for the following packets is also affected. Thus the effect of back-pressure in a node propagates throughout the whole network and introduces jitter. Moreover, this will result in an exponential increase in the theoretical maximum latency. A simple equation to calculate the maximum network latency in a network of switches with only one VL per port and where the input buffer is one packet wide is as follows:

$$P = \sum_{sw=1}^{n} (Q-2)^{sw} \tag{4.1}$$

Here P is the maximum number of packets we could end up waiting for, Q is the number of links per switch and n is the number of switches in our path. We ignore two of the links because one is the link we are entering on

and one is the link we are leaving on and they will not delay our packet. If our packet pass through three switches A, B and C in that order, and each switch has four ports. It will have to wait for two packets, one for each link, in switch C. In switch B it will also have to wait for two packets, but both of these packets will have to wait for two packets in switch C. In switch A it will again have to wait for two packets and these two packets will have to wait for two packets each in switch B, which will each have two wait for two packets in switch C. Thus, we ends up with $P = 14$.

The equation for maximum network latency now becomes:

$$L = T \times P \tag{4.2}$$

Here L is the maximum network latency and T is the transmission speed in cycles. As (4.1) grows exponentially this will have significant impact on the jitter characteristics of the network. If we consider our network and a packet with path length three (as in Figure 4.3.4) we get the following:

$$L = T \times P = 32 \times 584 = 18688 \tag{4.3}$$

In our simulations the maximum observed latency for a path length of three is 6606 cycles so we are far from the theoretical maximum. For a path length of four the maximum observed latency is 8840 cycles while L is 149760 cycles. We assume that the large difference between theoretical and observed latency can be explained by a very low probability for the theoretical worst case to occur. But still, in order to avoid the actual increase in latency the network load must be kept low.

4.4 Contributions and Related Work

In our study of a DiffServ inspired QoS concept for interconnection networks we have found that: (i) Throughput differentiation can be achieved by weighting of VLs and by classifying VLs as either low or high priority. (ii) The balance between VL weighting and VL load is not crucial when the network is operating below the saturation point. (iii) Latency increase with load and yields substantial jitter. Good jitter characteristics seems unobtainable since the theoretical maximum latency grow exponentially, even if the measured maximum latency does not show exponential behaviour its growth rate is substantial. From the application viewpoint these results shows that both message passing applications and bulk transfer applications can coexist in the same network without dedicated link resources. This level of service differentiation could reduce cost and complexity in compute clusters as only one

network technology is required, as opposed to the approach where one network technology is used for low latency message passing (e.g. Myrinet [81]) and another for bulk transfer (e.g. 10 Gigabit Ethernet [3]).

Recently we have seen several research contributions to this field. Jaspernite et. al. [82, 83] and Skeie et. al [53] discuss latency improvements for switched Ethernet relative to IEEE Standard 802.1p, but they do not consider flow-control. Many other contributions regarding Ethernet QoS also exists, but common for them all are that they do not consider QoS in combination with flow-control and deadlock avoidance schemes.

In [40] Pelissier gives an introduction to the set of QoS mechanisms offered by InfiniBand and the support for DiffServ over IBA. In this approach the presence of admission control is assumed,but no evaluation is given and he suggests to implement admission control with RSVP, which is incompatible with the DiffServ scheme. Alfaro et. al builds on Pelissier's work and present a strategy for computing the arbitration tables of IBA networks and a methodology for weighting of virtual lanes using the arbitrator defined in IBA [84]. The concept is evaluated through simulations assuming that only priority traffic requests QoS. In [80] Alfaro et. al also include time sensitive traffic, besides calculating the worst case latency through various types of switching architectures. But this strategy has to recompute the IBA arbitrator every time a new connection is accepted [80, 84], which is not compatible with DiffServ and the concept of a stateless core network. Other QoS efforts include The Multimedia Router [85, 86] and some earlier work in [87] and [88], but none of these fits well with the DiffServ philosophy in the context of interconnection networks.

4.5 Critique

The service differentiation scheme has been shown to work well for prioritising high priority low bandwidth flows over more bandwidth demanding low priority flows. There remains, however, one major drawback. Deterministic guarantees are not possible because there is no protection from saturating the network. The absence of admission control means that we must rely on over-provisioning in the high priority class to be able to deliver QoS. If the high priority class becomes saturated its performance will degrade along with the performance of all the lower priority classes.

4.6 Further Work

Improving latency and the effect back-pressure has on latency is the main challenge remaining. Avoiding saturation by the use of admission control is one possibility that we will explore in the next chapter. Another possibility is mechanisms for reducing the effect of head-of-line blocking on uncongested flows. Some contributions targeting this has recently appeared [89, 90] with very promising results.

As always, other topologies, traffic patterns, and other combinations of service classes could be evaluated. E.g. hot-spots is one interesting scenario that is missing, another is the use of adaptive routing. More advanced ways of populating the IBA arbitration tables, such as multiple entries and large versus small allocations could have given further insight.

A grand challenge remaining is the resource use when combining QoS, routing and fault tolerance, which are all important features for future interconnects. Since these features require many of the same resources, e.g. virtual channels, they conflict with each other when all three are required at the same time. To have enough resources to support QoS, layered routing, and fault tolerance is currently prohibitively expensive. Alternatives for reducing the resource needs and for combining these mechanisms should be sought.

Chapter 5

Admission Control Algorithms

In Chapter 4 we achieved a relative differentiation of network traffic based on traffic classes. We showed how to differentiate between classes, but we where unable to give quantitative guarantees since the amount of traffic allowed into the network was unrestricted. In order to give soft guarantees with regards to both bandwidth and latency we will now study several types of *admission control* (AC) in combination with the service differentiation scheme from the previous chapter.

Admission control is the general concept of controlling the operation point of a network, at one extreme we have *connection admission control* (CAC) and at the other extreme we have *rate control*. CAC resembles the telephone system where there is always a risk of the busy signal (call blocking), because the load is high. Rate control, on the other hand, slowly reduce the rate for all callers as the number of callers grow. This allows for more people to participate at the cost of lower performance. It is acceptable for services such as file transfer and web surfing, but disastrous for services such as Voice over IP and massive multiplayer online games. Rate control is often used in combination with over-provisioning, when the risk of overloading the network is low. The AC schemes that we propose in Section 5.2 and 5.3 are all CAC schemes, but they differ in how they collect information about current network conditions and the information available for making the AC decision. We evaluate the proposed methods in Section 5.4.

5.1 Avoiding Saturation

One of the main findings in Chapter 4 where that the balance between VL weights and VL load is not crucial when the network is operating below the saturation point. In general this sets the target load for AC, since as long as we can ensure that the load of the various service classes are below saturation we can also guarantee that each class get the bandwidth they request. The target for AC is therefore the point where the amount of offered traffic is equal to the amount of accepted traffic. The effective bandwidth at this point will be used as a guideline for the Calibrated load and Link-by-link AC schemes presented below.

Another finding in Chapter 4 was that latency below saturation was acceptable, but with significant jitter. This problem we challenge in Section 5.3.1 by using delay to make the admission decision instead of bandwidth.

5.2 Centralised Admission Control

In centralised AC there exist an entity that has knowledge of the current network state. This entity can be used as the decision maker for AC. Below we propose two schemes which follow this approach by using a bandwidth broker (BB). The BB has knowledge about the amount of traffic entering the network, but the details available to the BB are different for the two schemes.

5.2.1 Calibrated Load Admission Control

Calibrated load (CL) is a scheme relying on a BB that knows the total rate of traffic entering the network. Our *admission control parameter* is the rate of traffic that can be injected into the network while still operating the network below saturation. When the rate of traffic entering the network reaches the admission control parameter no more traffic will be admitted. The admission control parameter must be decided by measurements on the network in order to find the saturation point, ours is deduced from the simulations performed in Chapter 4.

To separate high and low priority traffic we use two different rate parameters, one for the total high priority traffic and one for the total low priority traffic. For high priority traffic this can be expressed as follows:

$$\sum_{i=0}^{n} L_i + P < CL \tag{5.1}$$

Here CL is the calibrated load for outgoing traffic, L_i is the load in node i and P is the peak rate for the requesting flow. The flow is admitted if the total load $\sum_{i=0}^{n} L_i$ plus the requested increase P is below the calibrated load CL. Low priority traffic can be expressed similarly just substituting the high priority values with low priority values. The strength of this scheme lies in its simplicity, its weakness lies in its inaccuracy. It is inaccurate because it does not take into account the distribution of flows in the network. E.g. it is not suitable for handling hot spots.

5.2.2 Link by Link Admission Control

Our second scheme is the *Link-by-link* (LL) approach. Here the BB knows the load on every link in the network and will consult the available bandwidth on every link between the source and the destination before accepting or rejecting a flow. Compared to the CL approach, this solution requires both topology and route information about the network.

For the admission decision we adopt the *simple sum* approach as presented in [91]. This algorithm states that a flow may be admitted if its peak rate p plus the peak rate of the already admitted flows s is less than the link bandwidth bw. Thus the requested flow will be admitted if the following inequality is true [91]:

$$P + S < BW \tag{5.2}$$

As for the CL method we deduce the effective bandwidth from the measurements obtained in Chapter 4. Since we are dealing with service levels where each SL has different bandwidth requirements it is natural to introduce differentiation in (5.2). We achieve this by dividing the link bandwidth into portions relative to the traffic load of the SLs, and include only the bandwidth available to a specific service level BW_{sl} as follows

$$P + S_{sl} < BW_{sl} \tag{5.3}$$

where

$$BW_{sl} = BW_{link} * \frac{L_{sl}}{L_{total}} \tag{5.4}$$

and S_{sl} is the sum of the admitted peak rates for this service level, BW_{link} is the effective link bandwidth, L_{sl} is the fraction of bandwidth allowed for this SL, and L_{total} is the total load.

5.3 Decentralised Admission Control

In decentralised AC there is no single entity responsible for making the admission decision. Rather, a local decision is made whenever a new admission request arrives. Below we will discuss two different variations of distributed AC which we have adopted from the Internet world. The first one use measurements while the second one use probes.

5.3.1 Measurement Based Admission Control

The measurement based approach, called Egress Based (EB) admission control is different from the other three schemes in that it uses latency as the target for AC instead of throughput. The idea behind this approach is that by targeting delay it should be possible to give more precises latency guarantees as well as higher utilisation of the available bandwidth.

EB AC is a fully distributed AC scheme where the egress nodes are responsible for conducting the provisions. Basically, we adopt the Internet AC concept presented by Cetikaya and Knightly in [92]. This method does not assume any pre-knowledge of the network behaviour as is the case with our previous solutions. Also different from the previous approaches is the use of a delay bound as the primary AC parameter. For clarity we give a brief outline of the algorithm below, a more detailed description can be found in [93].

The method is entirely measurement based and relies on the sending node to time-stamping all transmitted packets. This enables egress nodes to calculate two types of statistical data. First, by dividing time into time-slots of length τ and counting the number of arriving packets, the egress nodes can deduce the arrival rate of packets in a specific time-slot. By computing the maximum arrival rate for increasingly longer time intervals we get a peak rate arrival envelope $R(t)$, where $t = 0, ..., T$ time-slots. Second, by comparing the originating time-stamp relative to the arrival time, the egress node can calculate the transfer time of a packet. With this information the egress node can estimate the time needed by the infrastructure to service the k following packets; i.e. a consecutive stream of packets, where the next packet in the service class enters the infrastructure before the previous packet has departed the egress node. By doing this for larger and larger k sequences of packets within a measuring interval of length $T\tau$ and subsequently inverting this function we achieve the service envelope $S(t)$, giving the amount of packets processed by the network in a given time interval t. Now repeating this M times, the mean $\overline{R}(t)$ and the variance $\sigma^2(t)$ of $R(t)$, and the mean $\overline{S}(t)$ and variance $\Psi^2(t)$ of $S(t)$ may be calculated. If a flow request has a peak

rate P and a delay bound D it may be accepted if the peak rate P plus the measured arrival rate $R(t)$ is less than the service rate required for the delay $D, S(t+D)$. This is expressed by the following equation:

$$t\overline{R}(t) + Pt - \overline{S}(t+D) + \alpha\sqrt{t^2\sigma^2(t) + \Psi^2(t+D)} < 0 \tag{5.5}$$

for all interval lengths $0 \leq t \leq T$. Additionally there is a stability criterion that must be satisfied

$$\lim_{t\to\infty} \overline{R}(t) + P < \lim_{t\to\infty} \frac{\overline{S}(t)}{t} \tag{5.6}$$

According to [93] the right-hand side of (5.6) may, in a network using weighted fair queueing, be replaced by the link capacity C. Then (5.6) may be rewritten as

$$\lim_{t\to\infty} \overline{R}(t) + P < C \tag{5.7}$$

which simply states that the mean arrival rate in the network's lifetime plus the peak rate of the flow never should exceed the link capacity of the network.

The EB scheme derives its knowledge of the network from measurements of the traffic passing through the egress nodes. It is therefore difficult for the egress nodes to have a complete picture of the load in the network, moreover the packet latency is used to infer the network load utilising the fact that increased network load will increase latency as well. From that viewpoint it is difficult to give a service class bandwidth guarantees since it has no concrete knowledge of the network load. The algorithm will admit as much traffic as it can without breaking the delay bound, and the efficiency of the algorithm is linked to its ability to limit service levels to operate within the delay bounds.

5.3.2 Probe Based Admission Control

As an alternative to passively monitoring the network activity in the egress nodes of the network, it is possible for the end nodes to take a more active role in the admission decision. This can be done by actively sending probe packets through the network from source to destination and monitor the arrival of the probes at the egress of the network [94, 95]. If the size and rate of the probes are designed correctly they should give the end-node the opportunity to calculate how the new flow will be treated by the network. Several probing schemes have been proposed in the literature, some of which are described in [94, 95]. The approach in [95] require either that packets be

dropped to indicate congestion or that congested packets be marked in the switches in the network. Packet dropping is implausible in interconnection networks such as the IBA and ASI since packets are not dropped, but flow-controlled. Explicit marking of packets requires intelligence in the switches and partly removes the point of end-point admission control.

In [94] Bianchi et.al. propose a probing scheme where the load is inferred by measuring the jitter of the probes. This requires that the probes are forwarded through the network with low priority to ensure that the probe packets are unable to steal bandwidth from the already existing traffic in the network while giving worst-case measurements of the network jitter and thus guaranteeing that the traffic, when admitted, will get at least the service of the probe packets. It does, however, introduce a chance for under utilisation. We adopt and apply this scheme to IBA. And in our scenario it is natural to let the probes be forwarded on one of the low priority SLs with a weight equal to 1. The admission decision is based on the following equation:

$$T_{max} - T_{min} < J \tag{5.8}$$

and

$$P = 0 \tag{5.9}$$

Where T is the transmission time and P the number of packets rejected. For each probe received the destination registers the packet's transmission time, i.e. the time the packet spends in the network. When an adequate amount (6 probes for each flow was used in the evaluation) of probes have been sent and received, the receiver calculates the jitter by subtracting the minimum packet transmission time from the maximum packet transmission time. This value is compared to the jitter requirements embedded in the probes and an admission decision is sent back to the sender. If the perceived jitter is below the upper bound given in the probe the flow is accepted, otherwise the flow is rejected. Additionally if any of the probes are rejected by the sender due to back-pressure the flow is also rejected.

5.4 Performance Evaluation

The performance evaluation consists of two parts, one where we consider throughput and latency characteristics at the class level and another where consider these characteristics at the flow level. Evaluation on the class level corresponds to the level of detail where we did service differentiation in Chapter 4. Therefore, we expect a close match between requested and achieved

performance. At the flow level, however, there might be a mismatch between how individual flows are treated compared to the aggregated results for the class. This makes it necessary to study the flow level characteristics in order to understand how individual flows within a class perform.

5.4.1 Class Level QoS

The class level QoS from Chapter 4 is now extended with AC. Admission control is performed centrally or at the network edge so we are still in compliance with the DiffServ philosophy, where complexity in the core of the network is avoided.

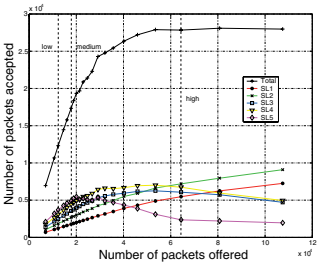
In the previous section we described four AC schemes. We will evaluate three of them through extensive simulation in this section, while the the fourth will be evaluated in the next section.

Throughput

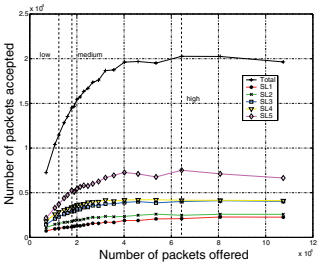
The purpose of AC is to keep network load below saturation, because below this point we can guarantee that all SLs get the bandwidth they require. It is, however, important to get as close to the saturation point as possible to avoid wasting resources. For the evaluation we will use the same SLs as in Chapter 4, where Table 4.1 reflects the percentage of link bandwidth that each SL requires.

When a network enters saturation we are no longer able to give all service classes the bandwidth they request. This is shown in Figure 5.1(a) for a network without AC. Without AC we are no longer able to give all service classes the bandwidth they require and high priority SLs take bandwidth away from low priority SLs, i.e. the bandwidth differentiation is no longer according to the percentages in Table 4.1. Let us now consider the performance of the CL scheme. We see from Figure 5.1(b) that we are able to keep the accepted traffic below the saturation point independently of the offered load, which goes beyond the saturation point. But there is still a problem with the differentiation, since as the load increases toward the *high mark* the differentiation between SLs of the same priority is diminished. The CL scheme is able to keep the load below saturation, but the information used to make the admission decision is too coarse to maintain bandwidth differentiation between SLs of the same priority, since it makes the decision based on the total load for the network and not individual SL's.

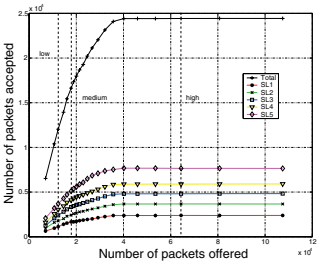
The LL scheme corrects the weakness found in the CL scheme by providing exact differentiation between SLs, as shown in Figure 5.1(c). The



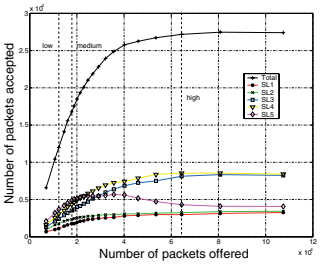
(a) No admission control.



(b) Calibrated load.



(c) Link by link.



(d) Egress measurements.

Figure 5.1: Throughput.

achieved differentiation is now exactly as specified in Table 4.1, making it possible to guarantee each SL the required bandwidth. Furthermore, the LL scheme estimates the saturation point more precisely than CL, which improve network utilisation since we operate it closer to the saturation point. This improvement is a direct effect of the increased amount of information that LL scheme has about the network state. Because it knows the load of every link in the network it is able to make a decision based on the exact load along the requested source/destination path.

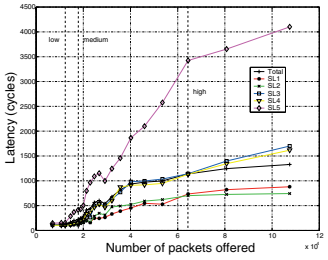
Finally, we have the EB method, where we use required versus measured latency to make the admission decision, instead of throughput. It is apparent from Figure 5.1(d) that this method is unable to give bandwidth guarantees and unable to keep the load below the saturation point. While the EB scheme fails with regards to bandwidth it might be more successful for latency, which we consider next.

Latency

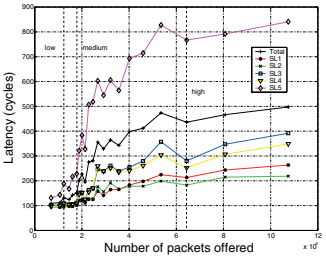
Latency increase with the load, and latency figures close to the network lower bound is only possible to achieve with a lightly loaded network, i.e. we can improve latency by sacrificing bandwidth. But a major increase in latency does not occur until we reach saturation, therefore it is possible to make a compromise where we achieve close to saturation throughput and low latency. This is what we are targeting in this section. Further improvement to latency could be achieved by lowering the operation point of the network.

Figure 5.2(a) shows the average latency (in cycles) for both individual SLs and total traffic without AC. Latency grows slowly at first, but when we reach the *medium mark*, which signify saturation, latency rapidly increases. Compared with the CL results in Figure 5.2(b) we see that the CL scheme is able to improve latency, even if all it does is to keep the network load below saturation. The average latency for all packets at the high mark is 1120 cycles without AC and 436 cycles with CL, a 61% reduction. Furthermore, we have only minor differentiation difficulties compared to what we had for throughput. We do, however, have problems with jitter, but we postpone that discussion to the next section.

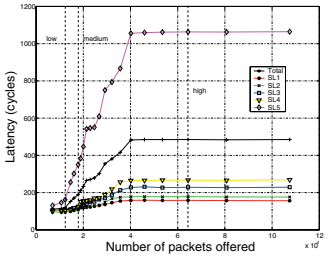
With the LL scheme we have no visible jitter at the class level (Figure 5.2(c)), which can be attributed to the precision of this scheme when it comes to differentiation between SLs and its estimation of the saturation point. The latency value for high load is 485 cycles, which is 10% above the CL latency at this point. This shows us that the LL scheme is better than the CL scheme, because it gives lower latency to SL 1-4 and higher latency to SL 5 (best



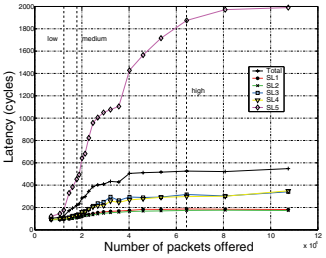
(a) No admission control.



(b) Calibrated load.



(c) Link by link.

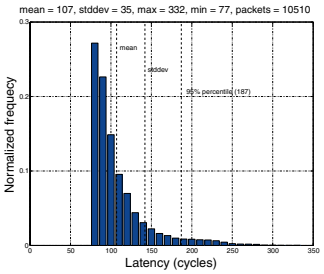


(d) Egress measurements.

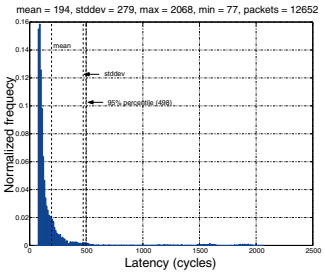
Figure 5.2: Average latency.

effort) — even if it has a higher average latency for all packets compared to CL. The increase in the average latency for all traffic is caused by the increase in throughput for SL5.

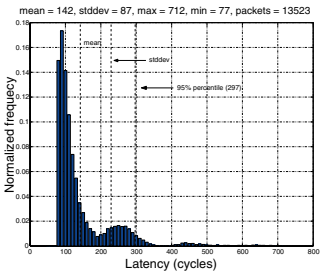
The EB scheme performed poorly for bandwidth, but gives decent results for latency as can be seen in Figure 5.2(d). It is capable of giving the same latency to SLs fairly independently of weight , e.g. SL 1 and 2. But it is unable to satisfy the delay bound of 100 cycles given for SL 1 and 2, and 250 cycles given for SL 3 and 4. The measured latency at the high mark is 187 cycles for SL 1 and 316 cycles for SL 3. So even using a measurement based method we are unable to give hard delay guarantees. The poor performance



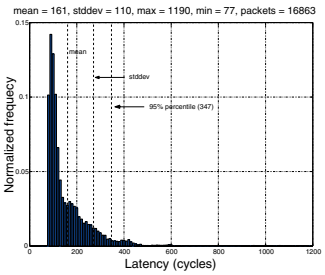
(a) No admission control (medium load).



(b) Calibrated load (high load).



(c) Link by link (high load).



(d) Egress measurements (high load).

Figure 5.3: Latency distribution for SL1 packets at with three hops.

of the EB scheme can be ascribed to the use of delay as the primary AC parameter, this results in the bandwidth requirements being ignored.

Jitter

The latency distribution for SL1 packets with a path length of three hops is shown in Figure 5.3, this was the most frequent path length observed in our simulations. In each figure the mean, the standard deviation and the 95 %

percentile are marked with a dashed line. The distance between the mean mark and the standard deviation mark reflects the standard deviation.

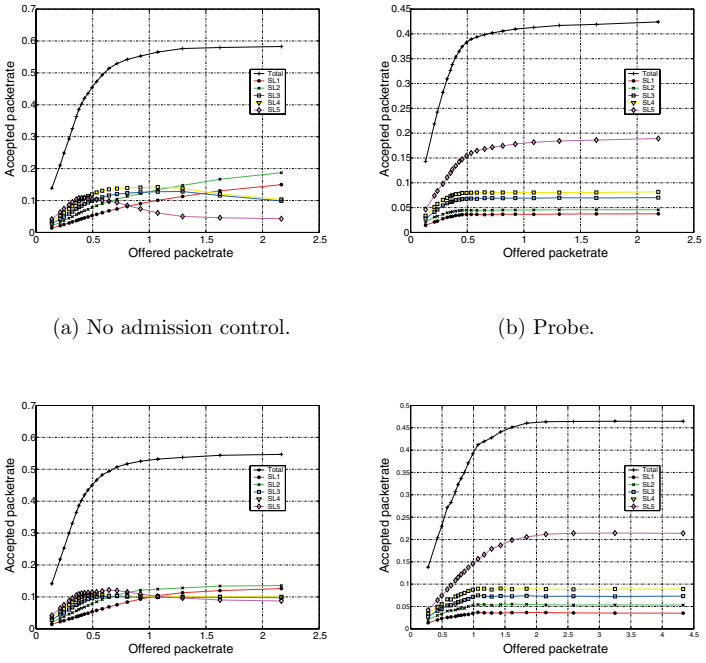
Figure 5.3(a) shows the latency distribution achieved without AC and at *medium mark* in Figure 5.1(a). This is the behaviour we would like to guarantee by limiting the operation with AC. Figure 5.3(b) shows the distribution for the CL scheme at the *high mark* in Figure 5.1(b). The load here is 20% higher than in the former. We see that the CL scheme has poor jitter characteristics, something we already suspected from the latency plots in Figure 5.2(b). The mean is 194 cycles, the standard deviation 279 cycles and the 95% percentile is 498 cyles. Thus 95% of the packets have a latency of 498 cycles or below. The problem here is the high standard deviation, which indicates unpredictable packet arrival times. The reason are as previously describe for throughput.

The LL scheme has better jitter characteristics as can be seen from Figure 5.3(c). The histogram has a shorter tail compared to Figure 5.3(b), and a mean of 142 cycles, a standard deviation of 87 cycles, and a 95% percentile of 297 cycles. Thus, jitter is reduced by 40% for 95% of the packets. Furthermore, the load for LL at this point is about 7% above the CL load so lower jitter is achieved at a higher load.

The EB scheme, shown in Figure 5.3(d), falls between the CL and LL scheme with a 95% percentile of 357 cycles, which is a 30% improvement over CL. This is, however, achieved at a load 25% above the CL load. This improvement is due the fact that the EB scheme does not consider throughput when making the admission decision, while the CL scheme use a very coarse estimate of available bandwidth when making the admission decision. Still, even a delay bound scheme shows a limited ability to achieve good jitter characteristics.

5.4.2 Flow Level QoS

We will now study the flow level characteristics of three admission control schemes in combination with service differentiation. Two schemes, LL and EB, are the ones that we studied in the previous section. The third, is a probe based (PB) approach which replace the CL scheme because of its poor performance. It is described in Section 5.3.2. Our goal is a QoS concept with *flow aware* AC and *flow negligent* traffic classes. We will study throughput, latency and jitter characteristics as we did in the previous section, but at the flow level.



(a) No admission control.

(b) Probe.

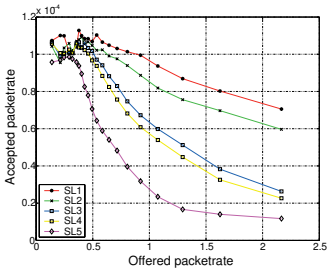
(c) Egress measurements.

(d) Link by link.

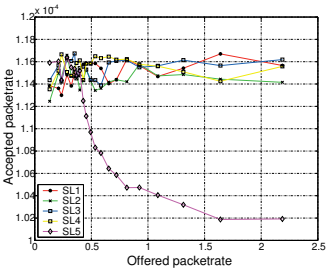
Figure 5.4: Average class throughput.

Throughput

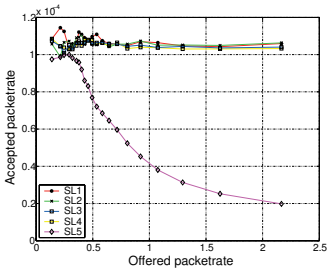
Throughput results at the class level is presented in Figure 5.4, and throughput results at the flow level is presented in Figure 5.5. As before, Figure 5.4(a) shows that without AC we are unable to give all service classes the requested bandwidth due to saturation. Furthermore, the high priority classes preempts the low priority classes, i.e. the bandwidth differentiation is no longer according to the percentages in Table 4.1. This behaviour is reflected at the flow level in Figure 5.5(a) where we see that the throughput per flow decrease as the number of flows increase. The PB scheme in Figure 5.4(b) performs similar to the LL scheme, the admission decision is precise and the



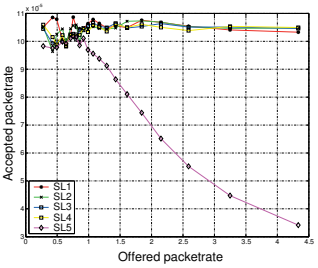
(a) No admission control.



(b) Probe.



(c) Egress measurements.



(d) Link by link.

Figure 5.5: Average flow packet rate.

bandwidth differentiation is according to the requirements. For flow level throughput in Figure 5.5(b) we see that flows in SL 1 - 4 get the requested bandwidth, while SL5 gets less. Thus, we are able to give bandwidth guarantees with this scheme, at both the class and flow level, and we are able to utilise the network resource well since we get close to the saturation point. We have already seen that the EB scheme is unable to give bandwidth guarantees at the class level. At the flow level all accepted flows receive the required bandwidth (Figure 5.5(c)), but the number of flows accepted in each class is not differentiated accordingly to the actual specifications. This can, again, be ascribed to the use of delay as the AC parameter, which results in the bandwidth requirements being ignored and that the number of flows accepted in each class is not differentiated according to the actual requests.

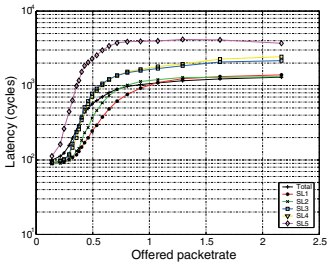
The LL scheme worked well for classes, and works equally well for individual flows. At the flow level the PB approach is as good as the LL scheme. It is able to give all flows the requested bandwidth at the cost of less bandwidth available to best effort traffic in SL5.

Comparing Figure 5.5(b) and 5.5(d) in detail we observe that the PB scheme gives slightly higher throughput to most classes. This is caused by the slightly higher load applied in the PB scheme to balance the lack of normal SL5 traffic.

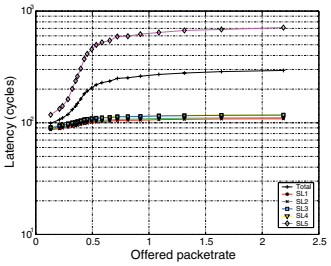
Latency

Typically we want to have low latency for flows in high priority SLs, while we accept higher latency for low priority SLs. For best effort traffic in SL5 we do not try to meet any latency requirements.

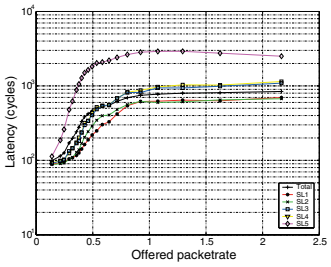
Figure 5.6(a) shows the average per flow latency (in cycles) for increasing load values without AC. Compared with the results from PB in Figure 5.6(b) we see that latency is improvement by one order of magnitude for high priority flows. The EB scheme in Figure 5.6(c) only slightly improves latency, but the differentiation between flows in different SLs is better than for PB. For high priority SLs latency levels out at 700 cycles, while latency for low priority SLs levels out at 1000 cycles. For the PB scheme both high and low priority SLs levels out at 100 cycles. The EB scheme is unable to achieve latencies as low as the other schemes even if its using latency as the primary AC parameter. The reason is the unpredictable latency characteristics in cut-through networks, which we discussed in Section 4.3.4. Finally, the LL scheme also improves latency by close to an order of magnitude in Figure 5.6(d). The strength of the LL scheme, that we have seen all along, is its ability to differentiate between flows in different SLs. This can be attributed



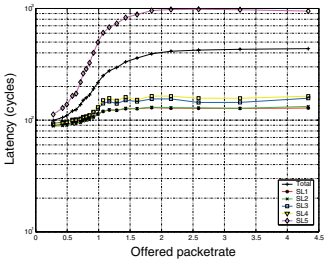
(a) No admission control.



(b) Probe.



(c) Egress measurements.



(d) Link by link.

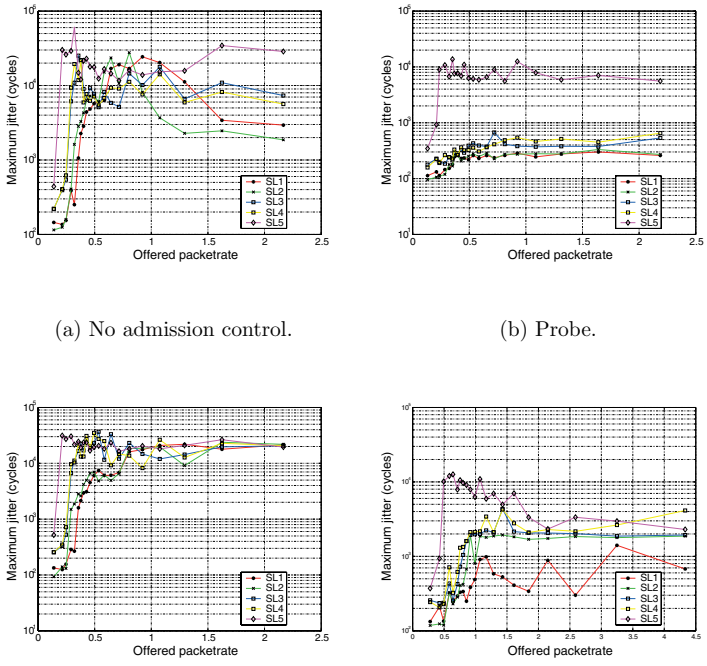
Figure 5.6: Average flow latency.

to its detailed knowledge of the network conditions.

Jitter

We have evaluated per flow jitter by plotting the maximum observed jitter for all our schemes in Figure 5.7. The plots contain the maximum per flow jitter observed throughout a simulation run.

Without AC the results show that there is substantial increase in jitter for all flows even at very low load, this verifies what we have seen at the class level. With the PB scheme jitter is reduced by one order of magnitude



(a) No admission control.

(b) Probe.

(c) Egress.

(d) Link by link.

Figure 5.7: Maximum flow jitter.

for high priority flows (Figure 5.7(b)), and slightly less for low priority flows. This scheme use jitter for the admission decision, but we are still unable to reduce jitter significantly as is is still in the order of 300 to 600 cycles. The EB scheme yields a jitter of 12000 cycles regardless of priority (Figure 5.7(c)), which is similar to the observed jitter without AC. This can be attributed to focus on latency rather than jitter. Finally, for the LL mechanism in Figure 5.7(d), we see an improvement in overall jitter, but not as good as for the PB scheme. This is caused by the fact that this scheme ignores latency and jitter properties and only concentrates on throughput when making the admission decision. Moreover, we can conclude that jitter is the most challenging at-

tribute to control in cut-through networks due to backpressure.

5.5 Contributions and Related Work

We have suggest three AC schemes for interconnection networks. Our main findings are as follows. First, bandwidth guarantees for a set of service classes are achievable with the use of the Link-by-link and the Probe based schemes. The Calibrated Load and the Egress Based methods are unable to achieve any guarantees. Second, latency and jitter guarantees are hard to achieve regardless of the method used. This is due to the nature of interconnection networks and the way back-pressure affects latency. Third, flow level guarantees are possible by combining *flow aware* admission control with *flow negligent* service differentiation. Fourth, each mechanism works best when measured on the parameter used to make the admission decision.

These results are important in several ways. First, they are important as a means to achieve QoS in interconnection networks and making it possible to have low latency message passing applications and high throughput bulk transfer applications share the same network without risk of interference. Second, they are important to bridge QoS between traffic coming from the Internet and entering a local cluster, e.g. when a compute cluster is a part of a compute grid that is accessed through the Internet using IETF standards such as DiffServ or IntServ. As these protocols are used for some applications on the Internet we need ways to represent these QoS attributes on our cluster to be able to serve the application according to its specified QoS.

Many ways of doing AC do exists in literature, but they almost exclusively consider either ATM networks [96] or the Internet with TCP/IP type networks [97]. And they do not consider loss-less communication and back-pressure which are important issues in most interconnection networks. Furthermore, these contributions can not be directly adopted by interconnection networks since they exploit features and flaws of either ATM or TCP/IP. E.g. the approach in [95] require either that packets be dropped to indicate congestion or that congested packets be marked in the switches in the network. Packet dropping is implausible in interconnection networks such as the InfiniBand Architecture since packets are not dropped, but flow-controlled. Furthermore, explicit marking of packets requires intelligence in the switches, which defeats the whole purpose of end-point AC. Another scheme designed for interconnection networks is presented in [98] by Yum et. al, that use hop by hop bandwidth reservations and requires recomputations of a weighted round robin scheduler at every hop towards the destination. This is incom-

patible with DiffServ and the stateless core philosophy, and requires additional support in the switches, which increase complexity.

5.6 Critique

For the centralised approach there are three problems. First, the BB is a single point of failure, should it fail the AC becomes unavailable. This can be levigated by the use of a backup unit, but at increased cost and complexity. Second, for large networks the load on the BB can be very high resulting in a lot of strain on the BB and slow response times for admission requests. This can also make the BB expensive and increase the probability for failure. Third, as the network size increase the information that the BB has about the current network conditions are delayed compared to the actual conditions, which might lead to inaccurate AC.

The distributed approach overcomes the fault-tolerance and scalability problems of centralised AC, but introduce two other problems. First, it is hard to estimate the current global network state since we only have local information about the network. This makes it difficult to make accurate decisions. Second, the network communication overhead might increase as the AC entities must use network resources to make an admission decision.

5.7 Further Work

Common for all of the above methods are that they yield the best results when measured on the same parameter that they use for the admission decision. E.g. LL is gives the best results for bandwidth and use this as a parameter for AC. PB use jitter to make its decision and gives the best results for jitter. So in order to further improve AC a scheme that considers throughput, latency, and jitter might be sought. The study could also be extended with other traffic patterns, e.g. hot-spots, in order to understand the impact the traffic pattern has on the AC algorithm used.

Further work includes the investigation of congestion control as a replacement for AC, where we could use different congestion thresholds for different service classes in order to tell the end-node that no more connections are acceptable. Some results along these lines can be found in [99]. Another challenge is to improve latency and jitter characteristics. This could possibly be solved by methods that reduce head of line blocking, recent proposals include [90].

Chapter 6

Conclusion

The contributions of this thesis are the evaluation of four and the proposal of three admission control mechanisms, two routing algorithms, and one method for service differentiation in interconnection networks.

6.1 Routing

We started by pointing at some of the weaknesses in the current Ethernet [3] standard, as well as the performance benefits of loss-less operation for higher level protocols. Motivated by these findings we proposed Segment-based routing (SR) and Layered shortest-path routing (LASH), two topology agnostic routing algorithms for interconnection networks.

Segment-based routing is a versatile algorithm that guarantees deadlock free routing and can be used with almost any type of interconnection technology. The novelty of SR resides in its locality independence property. This property increases the number of ways a set of routing restrictions can be distribute, which makes it possible to improve performance compared to other turn-based algorithms. Furthermore, the concept of segments makes it possible to exploit the semi-regularity found in irregular topologies, such as meshes and tori with faults, to improve performance.

Layered shortest-path routing routing is an algorithm that guarantees topology agnostic, deadlock free, shorts-path routing. Its novelty lies in the idea of layered routing. Layered routing divides resources into layers, and deadlocks are avoided by preventing portions of traffic from using specific layers. This contrasts with the turn-prohibition approach of SR where

deadlocks are avoided by preventing data packets from using specific paths, which restricts routing freedom. For deterministic routing the performance of LASH supersedes all known topology agnostic algorithms, while for oblivious routing it is close to state-of-the-art. Layered shortest-path routing requires virtual channels so it can not be used by all technologies, but the number of layers required is modest. This makes it suitable for existing technologies such as the InfiniBand Architecture [4] (IBA) and the Advanced Switching Interconnect [5] (ASI). It can also be used with Ethernet by doing some minor changes to the Ethernet flow-control mechanism.

6.2 Service Differentiation

In order to deliver service differentiation we adopted the Differentiated Services [41] philosophy and applied it to IBA. It can also be used with ASI and in Ethernet, but for Ethernet it cannot be combined with flow-control unless the granularity of the flow-control mechanism is changed (as is now being addressed by the IEEE 802.3ar Congestion Management Task Force).

We showed that throughput differentiation is possible by a combination of weighted virtual lanes and classifying lanes as either high or low priority. These mechanisms where found necessary to give relative bandwidth guarantees when the network is highly loaded, but on a network operating below saturation they where of less importance. Moreover, we showed that the theoretical worst-case latency grow exponentially because of flow-control. Through simulations we found that, even if the observed latency did not show exponential behaviour its growth rate where substantial as soon as the network became saturated. This motivated the need for admission control to reduce latency.

6.3 Admission Control

With service differentiation we where able to differentiate between classes, but unable to give quantitative guarantees since we did not restrict the amount of traffic allowed into the network. By introducing admission control (AC) we where able to limit the network load. We have suggested three different AC algorithms and we have evaluated these three as well as a fourth proposal originally intended for the Internet.

Each algorithm can be combined with a DiffServ based QoS scheme, without increasing complexity in network switches. The Calibrated load

and Link-by-link schemes assume pre-knowledge of the network's saturation point and require a bandwidth broker. The Egress based method is based on endpoint (egress) latency measurements to make a statistical assessment of load. While the Probe based scheme uses a probe packet to check if there is room for more flows in the network based on experienced jitter.

Our findings where that bandwidth guarantees for aggregated flows are achievable with the Link-by-link and Probe schemes. While the Calibrated load and Egress based methods are unable to guarantee bandwidth. Latency and jitter is hard to handle for all schemes due the nature flow-control and the way it introduces back-pressure. Strict AC can be used to improve latency, but at the cost of lower throughput. Overall, the Probe and Link-by-link schemes make it possible to give soft guarantees for throughput and latency, while they are unable to give any guarantees for jitter.

6.4 Future work

Several of the topics discussed in this thesis deserve further study. The algorithm used for segmentation in SR can probably be improved, especially for irregular topologies. Moreover, a study of how to exploit segments for fault-tolerance and local reconfiguration is a topic for future research.

When combining layered routing with QoS we quickly need more layers than any existing technology supports. Therefore, how to reduce the number of layers required, while retaining the routing and QoS functionality, is an important question to answer. This also extends to the combination of QoS or LASH or both, with fault-tolerance. I.e. how to design a scalable interconnect that support high performance routing, QoS, and fault-tolerance remains a challenge for the future.

Common for all the proposed AC schemes are that they yield the best results when measured on the same parameter that they use to make the admission decision. So in order to further improve AC a scheme that considers throughput, latency, and jitter might be sought. Furthermore, an investigation of congestion control as a replacement for AC remains. In this approach we could use different congestion thresholds for different service classes in order to tell the end-node that no more connections are acceptable.

Bibliography

[1] José Duato, Sudhakar Yalamanchili, and Lionel Ni. *Interconnection Networks An Engineering Approach*. Morgan Kaufmann, revised edition edition, 2003.

[2] William J. Dally and Brian Towles. *Principles and practices of interconnection networks*. Morgan Kaufmann, 2004.

[3] IEEE Standards Association. *IEEE Standard 802.3-2005 LAN/MAN CSMA/CD Access Method*, 2005.

[4] InfiniBand Trade Association. *Infiniband architecture specification*, 1.2 edition, October 2004.

[5] Advanced Switching Interconnect Special Interest Group. *Advanced Switching Core Architecture Specification*, revision 1.1 edition, August 2004.

[6] O. Lysne, T. Skeie, S.-A. Reinemo, and I. Theiss. Layered routing in irregular networks. *IEEE Transactions on Parallel and Distributed Systems*, 17(1):51–65, 2006.

[7] Andres Mejía, José Flich, José Duato, Sven-Arne Reinemo, and Tor Skeie. Segment-based routing: An efficient fault-tolerant routing algorithm for meshes and tori. In *In proccedings of the 20th IEEE International Parallel and Distributed Processing Symposium (IPDPS 2006)*. IEEE Computer Society, April 2006.

[8] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. *RFC 2475 Differentiated Services*. IETF, December 1998.

[9] Sven-Arne Reinemo and Tor Skeie. Ethernet as a lossless deadlock free system area network. In Yi Pan, Daoxu Chen, Minyi Guo, Jiannong Cao, and Jack J. Dongarra, editors, *Parallel and Distributed Processing*

and Applications: Third International Symposium, ISPA 2005, Nanjing, China, November 2-5, 2005. Proceedings, volume 3758 of *Lecture Notes in Computer Science*, pages 901–914. Springer Berlin/Heidelberg, October 2005.

[10] Sven-Arne Reinemo, Andres Mejía, Tor Skeie, José Flich, and José Duato. Boosting ethernet performance by segment-based routing. In *To appear in Proceedings of the 15th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing.* Euromicro, IEEE Computer Society, February 2007.

[11] Tor Skeie Sven-Arne Reinemo and Olav Lysne. Effective shortest path routing for gigabit ethernet. In *Submitted*, 2006.

[12] Sven-Arne Reinemo, Tor Skeie, and Olav Lysne. Applying the diffserv model in cut-through networks. In *Proceedings of the 2003 International Conference on Parallel and Distributed Processing Techniques and Applications*, pages 1089–1095, June 2003.

[13] Sven-Arne Reinemo, Tor Skeie, Thomas Sødring, Olav Lysne, and Ola Tørudbakken. An overview of qos capabilities in infiniband, advanced switching interconnect, and ethernet. *IEEE Communication Magazine*, 44(7), July 2006.

[14] Sven-Arne Reinemo, Frank Olaf Sem-Jacobsen, Tor Skeie, and Olav Lysne. Admission control for diffserv based quality of service in cut-through networks. In Timothy Mark Pinkston and Viktor K. Prasanna, editors, *High Performance Computing - HiPC 2003: 10th International Conference, Hyderabad, India, December 2003. Proceedings*, volume 2913 of *Lecture Notes in Computer Science*, pages 118–129. Springer Berlin/Heidelberg, December 2003.

[15] Frank Olaf Sem-Jacobsen, S. A. Reinemo, T. Skeie, and O. Lysne. Achieving flow level qos in cut-through networks through admission control and diffserv. In Hamid R. Arabnia, editor, *Proceedings of the 2004 International Conference on Parallel and Distributed Processing Techniques and Applications*, volume 3, pages 1084–1090, June 2004.

[16] Marco Fillo, Stephen W. Keckler, William J. Dally, Nicholas P. Carter, Andrew Chang, Yevgeny Gurevich, and Whay S. Lee. The m-machine multi-computer. In *Proceedings of the 28th Annual International Symposium on Microarchitecture*, pages 146–156, 1995.

[17] R.E. Kessler and J.L. Schwarzmeier. Cray t3d: a new dimension for cray research. In *Compcon Spring '93, Digest of Papers*, pages 176–182, February 1993.

[18] N.R. Adiga, M.A. Blumrich, D. Chen, P. Coteus, A. Gara, M.E. Giampapa, P. Heidelberger, S. Singh, B.D. Steinmacher-Burow, T. Takken, M. Tsao, and P. Vranas. Blue gene/l torus interconnection network. *IBM Journal of Research and Development*, 49, March 2005.

[19] Charles Clos. A study of non-blocking switching networks. *The Bell System Technical Journal*, March 1953.

[20] J. Beetem, M. Denneau, and D. Weingarten. The gf11 supercomputer. In *Proc. 12th Annu. Int. Sym. Computer Architecture*, pages 108 – 115, June 1985.

[21] K. Hajikano, K. Murakami, E. Iwanbuchi, O. Isono, and T. Kobayashi. Asynchronous transfer mode switching architecture for broadband isdn. In *Proc. IEEE ICC 88*, pages 911–915, June 1988.

[22] Y. Sakurai, N. Ido, S. Gohara, and N. Endo. Large scale atm multistage switching network with shared buffer memory switches. In *Proc. ISS 90*, volume 4, pages 121–126, May 1990.

[23] H. Suzuki, H. Nagano, T. Suzuki, T. Takeuchi, and S. Iwasaki. Output buffer switch architecture for asynchronous transfer mode. In *Proc. IEEE ICC 89*, pages 99–103, June 1989.

[24] P. Kermani and L. Kleinrock. Virtual cut-through: A new computer communication switching technique. *Computer Networks*, 3(4):267–286, September 1979.

[25] W. J. Dally and C. L. Seitz. The torus routing chip. *J. Distributed Computing*, 1(3):187–196, 1986.

[26] SGS-Thomson Microelectronics. *STC104 Asynchronous Packet Switch - Data sheet*, June 1996.

[27] William J. Dally. Virtual-channel flow control. *IEEE Transactions on Parallel and Distributed Systems*, 3(2):194–205, March 1992.

[28] Ingebjørg Theiss and Olav Lysne. Froots - fault handling in up*/down* routed networks with multiple roots. In Timothy Mark Pinkston and Viktor K. Prasanna, editors, *High Performance Computing - HiPC 2003:*

10th International Conference, Hyderabad, India, December 2003. Proceedings, volume 2913 of *Lecture Notes in Computer Science*, pages 106–117. Springer Berlin/Heidelberg, December 2003.

[29] W. J. Dally and C. L. Seitz. Deadlock-free message routing in multiprocessor interconnection networks. *IEEE Transaction on Computers*, C-36(5):547–543, May 1987.

[30] R. Seifert. *The Switch Book: The Complete Guide to LAN Switching Technology*. John Wiley & Sons, Inc., 2000.

[31] Jose Carlos Sancho, Antonio Robles, and Jose Duato. A flexible routing scheme for networks of workstations. In *ISHPC '00: Proceedings of the Third International Symposium on High Performance Computing*, pages 260–267, London, UK, 2000. Springer-Verlag.

[32] T. Skeie, O. Lysne, J. Flich, P. Lopez, A. Robles, and J. Duato. Lashtor: A generic transition-oriented routing algorithm. In *Proceedings of the IEEE International Conference on Parallel and Distributed Systems (ICPADS)*, pages 595–604. IEEE Computer Society Press, 2004.

[33] Herbert Sullivan and T R Bashkow. A large scale, homogeneous, fully distributed parallel machine, i. *SIGARCH Comput. Archit. News*, 5(7):105–117, 1977.

[34] Olav Lysne and Tor Skeie. Load balancing of irregular system area networks through multiple roots. In *Proceedings of the International Conference on Communication in Computing*, pages 165–171, June 2001.

[35] IEEE Standards Association. *IEEE Standard 802.1D Media access control (MAC) bridges*, 1998.

[36] Michael D. Schroeder, Andrew D. Birrell, Michael Burrows, Hal Murray, Roger M. Needham, and Thomas L. Rodeheffer. Autonet: a high-speed, self-configuring local area network using point-to-point links. *IEEE Journal on Selected Areas in Communications*, 9(8), October 1991.

[37] Francesco De Pellegrini, David Starobinski, Mark G. Karpovsky, and Lev B. Levitin. Scalable cycle-breaking algorithms for gigabit ethernet backbones. In *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, volume 4, pages 2175–2184, March 2004.

[38] Francisco J. Alfaro, José L. Sanchez, and José Duato. Qos in infiniband subnetworks. *IEEE Transactions on Parallel and Distributed Systems*, 15(9), September 2004.

[39] R. Braden, D. Clark, and S.Shenker. *RFC 1633 Integrated Services*. IETF, June 1994.

[40] *Providing quality of service over InfiniBand architecture fabrics*, 2000.

[41] R. Braden et al. *Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification*. IETF, rfc2205 edition, September 1997.

[42] Lixia Zhang, Stephen Deering, and Deborah Estrin. RSVP: A new resource ReSerVation protocol. *IEEE network*, 7(5):8–?, September 1993.

[43] X. Xiao and L. M. Ni. Internet qos: A big picture. *IEEE Network Magazine*, pages 8–19, March 1999.

[44] Jerry Banks, John S. Carson II, Barry L. Nelson, and David M. Nicol. *Discrete-Event System Simulation*. Prentice Hall, 3 edition, 2000.

[45] W. L. Winston. *Operations Research: Applications and Algorithms*. Wadsworth Publishing Co., Belmont, CA, 3 edition, 1997.

[46] D. Gross and C. Harris. *Fundamentals of Queueing Theory*. John Wiley, New York, 3 edition, 1997.

[47] Jean-Yves Le Boudec and Patrick Thiran. Network calculus: A theory of deterministic queuing systems for the internet. In G. Goos, J. Hartmanis, and J. van Leeuwen, editors, *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet*, volume 2050 of *Lecture Notes in Computer Science*. Springer Berlin/Heidelberg, January 2001.

[48] Hung ying Tyan. *Design, Realization, and Evaluation of Component-Based Compsitional Software Architecture for Network Simulation*. PhD thesis, Ohio State University, 2002.

[49] A. Varga. Using the omnet++ discrete event simulation system in education. *IEEE Transactions on Education*, 42(4), November 1999.

[50] Xinjie Chang. Network simulations with opnet. In *Simulation Conference Proceedings*, pages 307–314, 1999.

[51] Rich Seifert. *Gigabit Ethernet*. Addison Wesley Pub Co., 1998.

[52] W. Noureddine and F. Tobagi. Selective back-pressure in switched ethernet lans. In *Proceedings of GLOBECOMM*, December 1999.

[53] T. Skeie, J. Johannessen, and Ø. Holmeide. The road to an end-to-end deterministic ethernet. In *Proceedings of 4th IEEE International Workshop on Factory Communication Systems*, August 2002.

[54] Helen Chen and Pete Wyckoff. Simulation studies of gigabit ethernet versus myrinet using real application cores. In *Communication, Architecture, and Applications for Network-Based Parallel Computing*, pages 130–144, 2000.

[55] Giuseppe Ciaccio and Giovanni Chiola. Gamma and mpi/gamma on gigabit ethernet. In *Proceedings of the 7th European PVM/MPI Users' Group Meeting on Recent Advances in Parallel Virtual Machine and Message Passing Interface*, pages 129–136. Springer-Verlag, 2000.

[56] Giuseppe Ciaccio, Marco Ehlert, and Bettina Schnor. Exploiting gigabit ethernet capacity for cluster applications. In *Proceedings of the 27th Annual IEEE Conference on Local Computer Networks (LCN'02)*, pages 669–679, November 2002.

[57] José Carlos Sancho, Antonio Robles, and José Duato. A flexible routing scheme for networks of workstations. In M. Valero, K. Joe, M. Kitsuregawa, and H. Tanaka, editors, *High Performance Computing: Third International Symposium, ISHPC 2000, Tokyo, Japan, October 16-18, 2000. Proceedings*, volume 1940 of *Lecture Notes in Computer Science*, pages 260–267. Springer Berlin / Heidelberg, October 2000.

[58] M.Koibuchi, A.Funahashi, A.Jourakua, , and H.Amano. L-turn routing: an adaptive routing in irregular networks. In *Proceedings of the 2001 International Conference on Parallel Processing*, pages 383–392. IEEE Computer Society, September 2001.

[59] Josè Flich, Pedro López, Manuel P. Malumbres, Josè Duato, and Tom Rokicki. Combining in-transit buffers with optimized routing schemes to boost the performance of networks with source routing. In M. Valero, K. Joe, M. Kitsuregawa, and H. Tanaka, editors, *High Performance Computing: 3rd International Symposium, ISHPC 2000, Tokyo, Japan, October 16-18, 2000. Proceedings*, volume 1940 of *Lecture Notes in Computer Science*, pages 300–309. Springer Berlin / Heidelberg, October 2000.

[60] T. Skeie, O. Lysne, and I. Theiss. Layered shortest path (lash) routing in irregular system area networks. In *Proceedings of Communication Architecture for Clusters*, 2002.

[61] IEEE Standards Association. *IEEE Standard 802.17-2004*, September 2004.

[62] Oliver Feuser and Andre Wenzel. On the effects of the ieee 802.3x flow control in full-duplex ethernet lans. In *Proceedings of the 24th Conference on Local Computer Networks*, October 1999.

[63] Ren Jing-Fei and R. Landry. Flow control and congestion avoidance in switched ethernet lans. In *IEEE International Conference on Communications*, volume 1, pages 508–512, June 1997.

[64] J. Wechta, Armin Eberlein, and F. Halsall. The interaction of the TCP flow control procedure in end nodes on the proposed flow control mechanism for use in IEEE 802.3 switches. In *HPN*, pages 515–534, 1998.

[65] Wael Khalil Noureddine. *Improving the performance of TCP applications using network-assisted methods*. PhD thesis, Standford university, June 2002.

[66] Thomas L. Rodeheffer, Chandramohan A. Thekkath, and Darrell C. Anderson. Smartbridge: a scalable bridge architecture. *SIGCOMM Comput. Commun. Rev.*, 30(4):205–216, 2000.

[67] King-Shan Lui, Whay Chiou Lee, and Klara Nahrstedt. Star: a transparent spanning tree bridge protocol with alternate routing. *SIGCOMM Comput. Commun. Rev.*, 32(3):33–46, 2002.

[68] Román García, José Duato, and José Serrano. A new transparent bridge protocol for lan internetworking using topologies with active loops. In *ICPP '98: Proceedings of the 1998 International Conference on Parallel Processing*, pages 295–303, Washington, DC, USA, 1998. IEEE Computer Society.

[69] S. Sharma, K. Gopalan, S. Nanda, and T. Chiueh. Viking: a multi-spanning-tree ethernet architecture for metropolitan area and cluster networks. In *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, volume 4, pages 2283–2294, March 2004.

[70] Tomohiro Otsuka, Michihiro Koibuchi, Akiya Jouraku, and Hideharu Amano. Vlan-based minimal paths in pc cluster with ethernet on mesh and torus. In Wu chun Feng and José Duato, editors, *Proceedings of the 2005 International Conference on Parallel Processing*, pages 567–576. IEEE Computer Society, June 2005.

[71] L. Cherkasova, V. Kotov, and T. Rokicki. Fibre channel fabrics: Evaluation and design. In *Proceedings of the Twenty-Ninth Hawaii International Conference on System Sciences*, volume 1, pages 53–62, January 1996.

[72] Mark Karol, S. Jamaloddin Golestani, and David Lee. Prevention of deadlocks and livelocks in lossless backpressured packet networks. *IEEE/ACM Transactions on Networking*, 11(6):923–934, December 2003.

[73] William James Dally. *Scalable switching fabrics for internet routers.* Avici Systems, Inc., 1999. White paper.

[74] Daniel H. Linder and Jim C. Harden. An adaptive and fault tolerant wormhole routing strategy for k-ary n-cubes. *IEEE Trans. Comput.*, 40(1):2–12, 1991.

[75] K. V. Anjan and Timothy Mark Pinkston. An efficient, fully adaptive deadlock recovery scheme: Disha. *SIGARCH Comput. Archit. News*, 23(2):201–210, 1995.

[76] Josè Duato. A necessary and sufficient condition for deadlock-free adaptive routing in wormhole networks. *IEEE Trans. Parallel Distrib. Syst.*, 6(10):1055–1067, 1995.

[77] Josè Duato. A necessary and sufficient condition for deadlock-free routing in cut-through and store-and-forward networks. *IEEE Trans. Parallel Distrib. Syst.*, 7(8):841–854, 1996.

[78] Josè Duato and Timothy Mark Pinkston. A general theory for deadlock-free adaptive routing using a mixed set of resources. *IEEE Trans. Parallel Distrib. Syst.*, 12(12):1219–1235, 2001.

[79] Ziqiang Liu and A. A. Chien. Hierarchical adaptive routing: a framework for fully adaptive anddeadlock-free wormhole routing. In *Proceedings of the Sixth IEEE Symposium on Parallel and Distributed Processing 1994*, pages 688–695, October 1994.

[80] F. J. Alfaro, J. L. Sanchez, J. Duato, and C. R. Das. A strategy to compute the infiniband arbitration tables. In *Proceedings of International Parallel and Distributed Processing Symposium*, April 2002.

[81] N. J. Boden, D. Cohen, R. E. Felderman, A. E. Kulawik, C. L. Seitz, J. N. Seizovic, and W. K. Su. Myrinet – a gigabit-per-second lan. *IEEE MICRO*, 1995.

[82] J. Jaspernite and P. Neumann. Switched ethernet for factory communication. In *Proceedings of 8th IEEE International Conference on Emerging Technologies and Factory Automation*, pages 205–212, October 2001.

[83] J. Jaspernite, P. Neumann, M. Theiss, and K. Watson. Deterministic real-time communication with switched ethernet. In *Proceedings of 4th IEEE International Workshop on Factory Communication Systems*, August 2002.

[84] F. J. Alfaro, J. L. Sanchez, and J. Duato. A strategy to manage time sensitive traffic in infiniband. In *Proceedings of Workshop on Communication Architecture for Clusters*, April 2002.

[85] Jose Duato, Sudhakar Yalamanchili, Blanca Caminero, Damon S. Love, and Francisco J. Quiles. MMR: A high-performance multimedia router - architecture and design trade-offs. In *HPCA*, pages 300–309, 1999.

[86] B. Caminero, C. Carrion, F. J. Quiles, J. Duato, and S. Yalamanchili. A solution for handling hybrid traffic in clustered environments: The multimedia router mmr. In *Proceedings of International Parallel and Distributed Processing Symposium*, April 2003.

[87] A. A. Chien and J. H. Kim. Approaches to quality of service in high-performance networks. In *Lecture Notes in Computer Science*, volume 1417, 1998.

[88] M. Gerla, B. Kannan, B. Kwan, E. Leonardi, F. Neri, P. Palnati, and S. Walton. Quality of service support in high-speed, wormhole routing networks. In *International Conference on Network Protocols*, 1996.

[89] Venkata Krishnan and David Mayhew. A localized congestion control mechanism for pci express advanced switching fabrics. In *Proceedings of the 12th Annual IEEE Symposium on High Performance Interconnects*, 2004.

[90] J. Duato, I. Johnson, J. Flich, F. Naven, P. Garcia, and T. Nachiondo. A new scalable and cost-effective congestion management strategy for lossless multistage interconnection networks. In *HPCA '05: Proceedings of the 11th International Symposium on High-Performance Computer Architecture*, pages 108–119, Washington, DC, USA, 2005. IEEE Computer Society.

[91] S. Jamin, S. Shenker, and Peter B. Danzig. Comparison of measurement-based call admission control algorithms for controlled-load service. In *INFOCOM (3)*, pages 973–980, 1997.

[92] Coskun Cetinkaya and Edward W. Knightly. Egress admission control. In *INFOCOM (3)*, pages 1471–1480, 2000.

[93] J. Schlembach, A. Skoe, P. Yuan, and E. Knightly. Design and implementation of scalable admission control. *QoS-IP*, 2001:1–15, 2001.

[94] G. Bianchi, F. Borgonovo, A. Capone, L. Fratta, and C. Petrioli. Endpoint ad-mission control with delay variation measurements for qos in ip networks. *ACM/SIGCOMM Computer Communications Review*, 32(2):61–69, 2002.

[95] Lee Breslau, Edward W. Knightly, Scott Shenker, Ion Stoica, and Hui Zhang. Endpoint admission control: Architectural issues and performance. In *Proceedings of the ACM SIGCOMM*, pages 57–69, 2000.

[96] Kohei Shiomoto, Naoaki Yamanaka, and Tatsuro Takahashi. Overview of measurement-based connection admission control methods in atm networks. *IEEE Communications Surveys and Tutorials*, 2(1):2–13, 1999.

[97] Edward W. Knightly and N. B. Shroff. Admission control for statistical qos: theory and practice. *IEEE Network*, 13(2):20–29, 1999.

[98] K. H. Yum, E. J. Kim, C. R. Das, M. Yousif, and J. Duato. Integrated admission and congestion control for qos support in clusters. In *Proceedings of IEEE International Conference on Cluster Computing*, pages 325–332, September 2002.

[99] Gary McAlpine, Manoj Wadekar, Tanmay Gupta, Alan Crouch, and Don Newell. An architecture for congestion management in ethernet clusters. In *Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05)*, 2005.