Cost functions in Optical Burst-Switched networks

Bartlomiej Klusek

under the supervision of

Dr John Murphy and Dr Liam Barry

A thesis submitted in June 2006 for the degree of
Doctor of Philosophy in
the School of Electronic Engineering,
Dublin City University

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of Doctor of Philosophy is entirely my own work and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work

Signed [illegible]

ID No 50162616

11th of June, 2006

To Alicja

Abstract

Optical Burst Switching (OBS) is a new paradigm for an all-optical Internet It combines the best features of Optical Circuit Switching (OCS) and Optical Packet Switching (OPS) while avoiding the main problems associated with those networks Namely, it offers good granularity, but its hardware requirements are lower than those of OPS

In a backbone network, low loss ratio is of particular importance Also, to meet varying user requirements, it should support multiple classes of service In Optical Burst-Switched networks both these goals are closely related to the way bursts are arranged in channels Unlike the case of circuit switching, scheduling decisions affect the loss probability of future bursts

This thesis proposes the idea of a *cost function* The cost function is used to judge the quality of a burst arrangement and estimate the probability that this burst will interfere with future bursts Two applications of the cost function are proposed A scheduling algorithm uses the value of the cost function to optimize the alignment of the new burst with other bursts in a channel, thus minimising the loss ratio A cost-based burst dropping algorithm, that can be used as a part of a Quality of Service scheme, drops only those bursts, for which the cost function value indicates that are most likely to cause a contention Simulation results, performed using a custom-made OBS extension to the *ns-2* simulator, show that the cost-based algorithms improve network performance

List of publications

- **Bartlomiej Klusek**, John Murphy and Liam Barry, "Cost-based burst dropping strategy in Optical Burst Switching networks", *In Proceedings of IEEE 7th International Conference on Transparent Optical Networks,* July 2005

- **Bartlomiej Klusek**, John Murphy and Liam Barry, "New Fiber Delay Line Usage Strategy in Optical Burst Switching Node," *in Proceedings of IASTED Networks and Communication Systems,* Apr 2005

- **Bartlomiej Klusek**, John Murphy and Liam Barry, "Traffic sources for Optical Burst Switching simulations," *in Proceedings of SPIE OPTO Ireland,* Apr 2005

- **Bartlomiej Klusek**, John Murphy and Philip Perry, "Cost-based wavelength allocation algorithms in optical burst switching networks," *in Proceedings of Asia-Pacific Optical Communications,* Nov 2004

- Dawid Nowak and **Bartlomiej Klusek**, "Analysis of Waveband Connecting Probability in Hybrid Optical Networks," *in Proceedings of 2nd IFIP-TC6 International Conference on Optical Communications and Networks,* Oct 2003

- Dawid Nowak, **Bartlomiej Klusek**, and Tommy Curran, "New Method for Calculating Probability of Waveband Connection in All Optical Networks," *in Proceedings of IEI/IEE Telecommunications System Research Symposium,* May 2003

- **Bartlomiej Klusek**, Dawid Nowak, and Tommy Curran, "Wavelength Assignment Algorithms in an OBS Node," *in Proceedings of IEI/IEE Telecommunications System Research Symposium,* May 2003

Contents

List of Figures

List of acronyms

AAP	Adaptive Assembly Period
ABR	Aggressive Burst Rescheduling Algorithm
ATM	Asynchronous Transfer Mode
AWGM	Arrayed Waveguide Grating Multiplexer
BAS	Broadcast-And-Select
BORA	Burst Overlap Reduction Algorithm
DARPA	Defense Advanced Research Projects Agency
DIA	Defense Intelligence Agency
D-PWS	Dynamic Preferred Wavelength Sets
DS-OBS	DiffServ over OBS
DWDM	Dense Wavelength Division Multiplexing
FAP	Fixed Assembly Period
FDL	Fiber Delay Line
GMPLS	Generalized Multiprotocol Label Switching
IETF	Internet Engineering Task Force
IP	Internet Protocol
JET	Just-Enough-Time
JIT	Just-In-Time
JITPAC	Just-in-Time Protocol Acceleration Circuit
LAUC-VF	Last Available Unused Channel with Void Filling
LTS	Laboratory for Telecommunications Science (NSA)
Min-EV	Minimum Ending Void
Min-SV	Minimum Starting Void

MSSIS	Maximum Stable-Set Interval Scheduling Algorithm
NRL	Naval Research Laboratory
OBS	Optical Burst Switching
OCS	Optical Circuit Switching
ODBR	On-Demand Burst Rescheduling Algorithm
OEO	Optical-Electrical-Optical
OPS	Optical Packet Switching
OTSI	Optical Time Slot Interchanger
OWns	Optical WDM Network Simulator
PLC	Planar Lightwave Circuit
PP	Probabilistic Preemptive scheme
PWA	Priority-based Wavelength Assignment
PWRP	Preemptive Wavelength Reservation Protocol
QoS	Quality of Service
SDH	Synchronous Digital Hierarchy
SOA	Semiconductor Optical Amplifier
SONET	Synchronous Optical Network
TAG	Tell-And-Go
TAS	Tune-And-Select
TCP	Transmission Control Protocol
TSOBS	Time Sliced OBS
VoIP	Voice over IP
WDM	Wavelength Division Multiplexing
WGR	Wavelength grating router
WSQ	Waiting-Queuing-Scheduling

Chapter 1
Introduction

The history of optical communication is probably as long as history of humanity and possibly longer. When two humans (or apes) needed to exchange some information while being out of earshot of each other, they gestured. By doing so, they formed a primitive light-based communication circuit. Their source of light was almost certainly the sun. It was modulated by the transmitting person's hands, propagated in the free space and was finally received by the other person's eye.

Millenia later, the development of laser and optical fiber became the enabling technology for high-speed telecommunications. Without it, the information society of today would not be possible.

The first widely deployed optical networks were based on optical point-to-point links and electronic routers, with all multiplexing being done in the electrical domain. This SONET/SDH based architecture was inherited from telephony networks, and is well suited for its original purposes, i.e. carrying voice traffic.

Rapid expansion of the Internet brought several important changes, however. The most obvious one is an ever-increasing demand for bandwidth. According to [1], for 18 years the data traffic doubled every twelve months. Since 1997, the rate of increase has increased, and today's networks have to cope with traffic doubling every six months.

The bandwidth offered by an optical fiber is enormous, several orders of magnitude bigger than this of twisted pair or coaxial cable. Although not even close to utilizing the theoretical capabilities, the introduction of *wavelength division multiplexing* (WDM) made it apparent that the limiting factor would be not the transmission, but switching. The electronic routers are the bottlenecks of first gen-

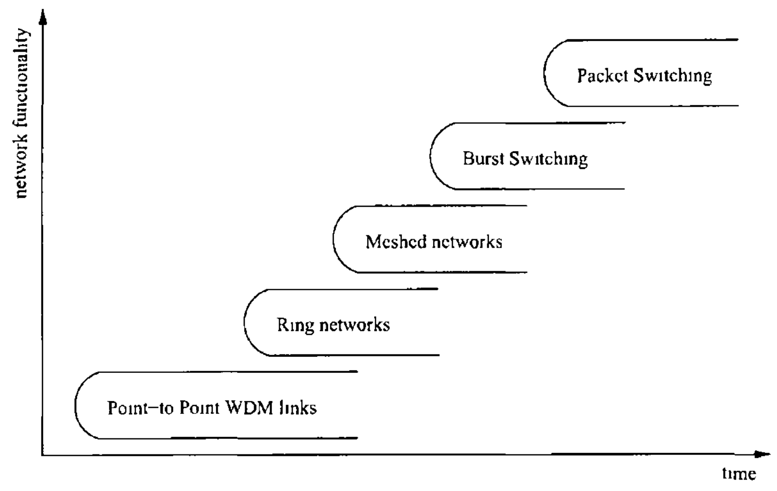


Figure 1 1 Evolution of optical networks

eration optical networks Additionally, Moore's Law states that the processing power of computers doubles every eighteen months This means that the demand for bandwidth will always grow faster than the switching capabilities of electronic routers

The bandwidth problem can be alleviated by the introduction of wavelength-switched, fully optical networks Here, instead of performing optical to electrical to optical (OEO) conversion in each core router, a lightpath is set up for each connection The data stream traverses the network without being analyzed and is only converted back into the electrical domain at its final destination As the theoretical limit of the capacity offered by a fiber is estimated to be 100 terabits per second [2], eliminating electrical switches dramatically increased network bandwidth The architecture of those networks today is described by the IETF GMPLS standards [3–5]

However, this approach has its disadvantages as well Highly aggregated IP traffic, carried by backbone networks, is known for its burstiness and long-range dependence [6,7] Low granularity, offered by wavelength-switched networks, means that either link utilization is low, or that a high proportion of the traffic will be lost Additionally, because each connection requires its own wavelength, the number of concurrent connections originating at a given node is limited This means that in a large network, it may be impossible to create a full mesh of connections Another factor is connection setup time In a wavelength-switched network this may be of the order of days if the network has to be reconfigured manually Such a network will be incapable of reacting to changes in traffic loads

It is widely accepted that the ultimate solution for the future is Optical Packet Switching (OPS) [8] In an OPS network, each IP packet is sent separately to its destination, solving most of the above mentioned problems The packer header is sent along with the packet itself as an in-band information, that has to be extracted and analyzed in each traversed core node However, if such a network is to be feasible, the switching times have to be much shorter than the packet length Currently available hardware does not offer such capabilities

Optical Burst Switching (OBS) [9,10] has been proposed as an alternative solution It combines the best features of circuit and packet switching and - hopefully - avoids the worst of both In particular, its hardware requirements will be lower than in the OPS case, but the granularity and flexibility will be significantly higher than those of wavelength-switched networks

Burst switching in itself if not a new idea It was first proposed in the eighties for use in copper-based networks as *fast circuit switching* [11,12] This concept did not gain popularity back then, as the limited bandwidth offered by a twisted pair cable did not justify the additional complexity Two decades later, the situation is entirely different Conventional switches became the bottlenecks and cannot be expected to support the bandwidths an optical fiber is capable of Burst switching enables us to use full capabilities of a fiber, without the limitations imposed by OEO conversions [9]

The evolution of optical networks can be presented in a graphical format, as shown in Fig 1 1 The figure is similar to that presented in [13]

1.1 Thesis contribution

In Optical Burst Switching networks the arrangement of bursts within a channel is an important issue Manipulating it can affect the overall burst loss probability

Unfortunately, the 'quality' of burst arrangement is diffcult to describe in a numerical way This thesis attempts to achieve this goal and describe its applications

- *Cost function* The main contribution of this thesis is the idea of a *cost function* Cost function value is expected to be an indicator of how likely is a given burst to cause contention in future Therefore, cost function can be used to judge how well a burst is aligned with other bursts in a channel and to asses suitability of a particular channel for a particular burst

- *Soft decision making* Most scheduling algorithms and contention resolution mechanisms use *hard decisions* This thesis demonstrates that the decision whether to use a certain resource or not can be made in a *soft* way, based on the value of a cost function

- *Fiber Delay Line (FDL) usage algorithm* FDLs can be used to reduce the burst loss ratio This thesis shows how the idea of a cost function can be used to design a schedulmg algorithm in a FDL-equipped node Use of the additional information, provided by a cost function, allows us to prevent contention, instead of resolving it Also, the cost-based algorithm allows us to match the needs of the node with the size of the available FDL-bank in an optimal way In effect, the overall burst loss ratio is minimized

- *An early dropping algorithm* A cost-based burst dropping algorithm is presented This algorithm can be used as a part of Quality of Service scheme, and allows us to ensure good QoS parameters for high-priority classes while dropping a minimal amount of low-priority traffic Alternatively, for a given amount of low-priority traffic to be dropped, the burst loss ratio in the high-priority class can be minimized

- *Simulation tool* The last contribution is an OBS extension to the *ns-2* simulator The efficient and robust underlying *ns-2* engine and its two-language architecture, coupled with an extensive set of OBS classes created a powerful simulation tool

1.2 Thesis overview

The rest of this thesis is organized as follows

- **Chapter 2** This chapter introduces the basic idea of Optical Burst Switching In particular, it discusses the burst switching and burst assembly algorithms

- **Chapter 3** In the first part of this chapter schedulmg algorithms used in Optical Burst Switching are presented The significance of optimal burst scheduling is emphasized and several existing scheduling algorithms are described The rest of this chapter deal withs ensuring Quality of Service (QoS) in Optical Burst Switching networks

- **Chapter 4** This chapter describes the idea of a *cost function* along with theoretical analysis, its proposed formula and possible applications

- **Chapter 5** All the simulations described in this thesis were performed using a custom-built OBS extension to the ns-2 simulator This chapter describes this extension, its internal architecture and usage

- **Chapter 6** The performance of cost-based schedulmg algorithm in Optical Burst-Switched networks was tested m several simulations Results of these simulations are presented and discussed here

- **Chapter 7** In this chapter the simulation results concerning the cost-based burst dropping algorithm are presented and discussed This and two previous chapters form the core of this thesis

- **Chapter 8** In this chapter final conclusions are presented

Chapter 2

Optical Burst Switching

For a fully transparent optical network to work efficiently without ultra-fast hardware, the basic data unit has to be significantly larger that IP packets. Thus, incoming packets are assembled into bigger entities, called bursts, at the edge node.

While it is usually assumed that bursts will consist of IP packets, an OBS network can carry any type of data. For example, ATM cells can be assembled into bursts. Even analog data can be carried by an OBS network. For simplicity, in the rest of this thesis, only IP packets will be considered.

An important feature of OBS networks is that control information is separated from the payload. The control packet is sent ahead of the actual data, using a different channel. When received by a core node it it converted to the electrical domain, analyzed and sent to the next node. At each stage a routing decision is made (if necessary) and a connection is set up. The data burst is then sent by the edge node without waiting for any acknowledgment. This one-way reservation protocol is similar to tell-and-go (TAG) [14, 15]. Several dedicated OBS protocols have been designed; their details will be discussed later in this chapter.

The time difference between sending the control packet and the data burst is called offset time. It is necessary to allow the intermediate nodes to receive the control packet, prepare a lightpath for the incoming burst and transmit the control packet to the next node. It is possible to eliminate offset time if at each node data bursts are delayed in a Fiber Delay Line (FDL) while the control packet is being analyzed and the connection is being set up.

The data burst traverses the entire network in optical domain. Only at the destination node is it converted to the electrical domain. It is then disassembled, and all the IP packets are sent to their respective destinations On the other hand, the control packet has to be converted to electrical domain at each core node However, as it is significantly smaller than the data burst, it can be transmitted at a much lower bitrate, allowing use of relatively cheap hardware

2.1 Burst switching protocols

In circuit-switched networks, the path is only considered complete after the initiating station has received positive acknowledgment, and no data can be sent before this happens The resulting delay is of little consequence, if the average length of connection is much greater than the required set-up time

Optical bursts, while larger than packets, are still relatively small, and waiting for a positive acknowledgment would be inefficient Therefore, other switching protocols had to be developed

2 1.1 Just-In-Time (JIT)

The idea of Just-In-Time switching was proposed by Mills in his HIGHBALL project [16] He used the analogy of trains leaving a station, without setting all the switches across the country at this moment Instead, the switches are set as the train approaches This idea can be adapted to optical burst switching [17]

When a burst is ready to be sent, the originating node sends the control packet with a SETUP message first As it traverses the network, at each core node bandwidth is reserved for the incoming burst Depending on the version of the protocol, the reservation starts either immediately after the control packet is received (explicit setup), or at the expected arrival time of burst (estimated setup) The latter option is more efficient, but also more complicated After the burst has been transmitted, the originating node sends a RELEASE message, letting the downstream nodes know that the transmission has ended and the lightpath can be torn down The reservation process is shown in Fig 2 1

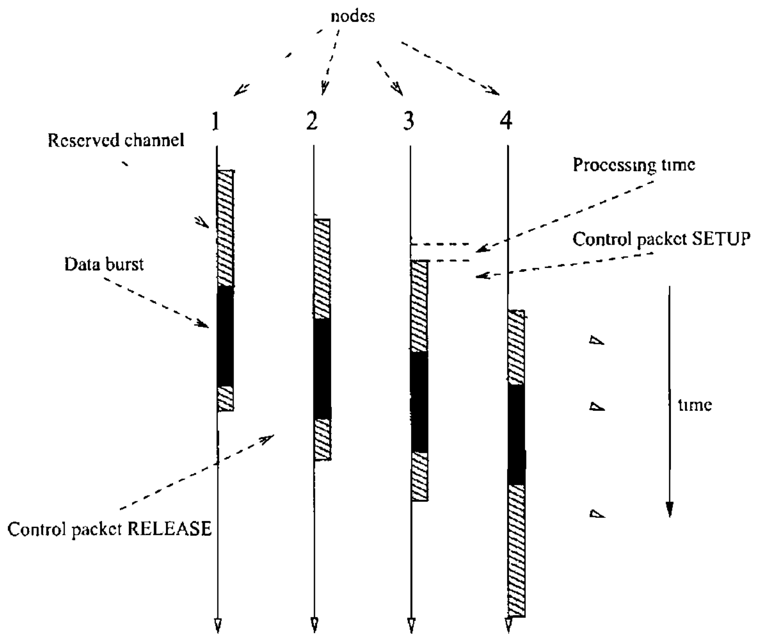


Figure 2 1 Just-In-Time (JIT) protocol

Note that when the reservation is made, the length of the burst is not known This simplifies the signalling and makes transmitting very long bursts easy, but at the price of suboptimal efficiency As the bandwidth is reserved until the RELEASE message arrives, it is possible that another burst will be dropped simply because it is not known that the bandwidth will be available at the time of its arrival In Fig 2 1 such a situation occurs in node 4 - the RELEASE message has not arrived yet, and the entire channel is unavailable

A variant of JIT with partial acknowledgement was used in the JumpStart project [18, 19] In this case, the burst source uses a SETUP message to initiate a connection The appropriate offset time is determined by the ingress switch, and transmitted back to the source in a SETUP ACK message This approach makes it possible to vary the offset time depending on traffic conditions

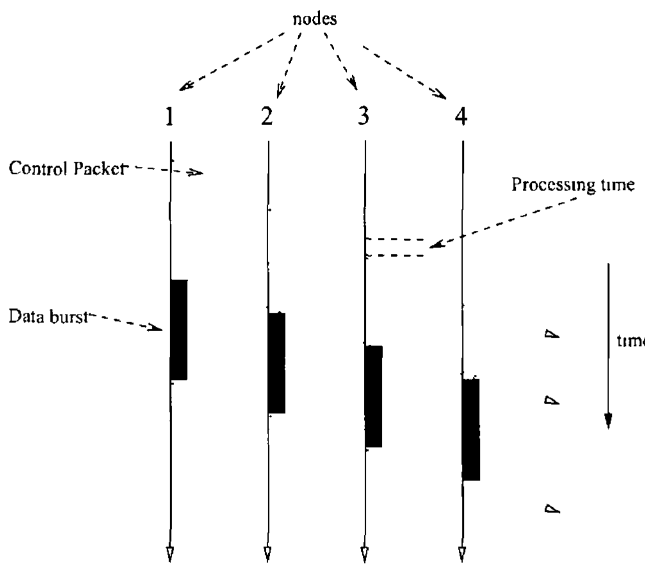


Figure 2 2 Just-Enough-Time (JET) protocol

2.1 2 Horizon

Horizon, proposed by Turner [20,21] is sometimes considered to be a reservation protocol [22] It does, however, specify a way to choose a particular channel, and therefore is better classified as a *scheduling algorithm* (and was originally presented as such) and will be described in Chapter 3

2.1 3 Just-Enough-Time (JET)

Just-Enough-Time (JET) is a burst switching protocol proposed by Yoo *et al* [23] The main difference, when comparing it to JIT, is that the SETUP message contains information about the burst length It makes it possible to only reserve the bandwidth for the exact amount of time that is necessary to transmit the burst This offers several advantages

1 No RELEASE message is necessary This simplifies signalling

2 Bandwidth is available immediately after the transmission is complete, without waiting for a RELEASE message

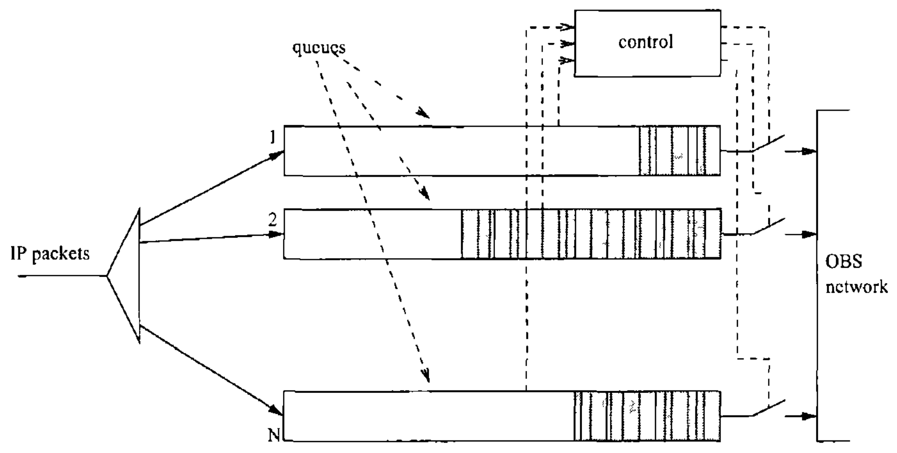


Figure 2 3 Structure of a burst assembler

3 Reservations can be made *after* another existing reservation, or in a *gap* between reservations This improves efficiency and reduces burst loss probability

Fig 2 2 illustrates the reservation process Note that when JET is used, the state of a channel that has to be kept by each core node, is more complex that in case of JIT At any time there can be more that one reservation, and for every reservation its starting and ending time has to be remembered Simulation results presented in [22, 24] show that JET outperforms JIT significantly However, some authors postulate that under realistic conditions the void-filling possibilities are limited and thus simpler protocols like JIT can offer better performance [25]

2.2 Burst assembly

As mentioned earlier, in an OBS network, IP packets have to be aggregated into bursts, as a single packet is too small to be efficiently processed by an optical network This process is called burst assembly or burstification and is performed by a burst assembler at an edge node A burst assembler is essentially an electrical buffer where the incoming data is stored until a burst is ready to be sent

Data packets grouped within a burst need to have at least one common characteristic - destination This means that in each node, there has to be at least one assembler for each other edge node There may be more assemblers than that if, for example, packets belonging to different classes have to be separated from each other or if it is desirable to group packets according to any other characteristic

It has been postulated [26] that the assembly process can reduce the self-similarity of aggregated IP traffic However, subsequent research [27,28] suggests that it is not the case

A structure of a generic burst assembler is shown in Fig 2 3 Incoming IP packets are first directed to an appropriate queue according to their characteristics At a certain time all packets from a given queue are transmitted together as a burst, the queue is emptied and the assembly process is repeated

Burst assembly has to balance the need for large bursts (to lower overhead) and low delay There may also be other considerations, for example it can be used as a part of a Quality of Service scheme The rest of this section will show how different assembly algorithms achieve those goals

2.2.1 Time and size - based algorithms

The Fixed-Assembly-Period (FAP) [29] algorithm, as its name implies, stores the incoming packets for a certain, pre-determined time At the timeout, the burst is sent, regardless of its size This means that bursts are generated at regular intervals, and may result in sending very large bursts if at a given time the load is high Round-robin burst assembly, proposed by Tachibana *et al* is a variant of the FAP algorithm [30] Similar algorithm has also been proposed in [31]

Min-BurstLength [29] is an alternative algorithm, based on size threshold IP packets are buffered until the size of the burst exceeds some pre-determined value In this case, the burst size is almost constant, and interarrival time variable One possible disadvantage of this approach is that, for low loads, the assembly delay may be excessively high A similar algorithm has been proposed in [32]

Algorithm 1 Fixed-Assembly-Period

```
EVENT IP packet arrives
d ⇐ destination address of the packet
if timer d is not running then
    start timer
end if
add packet to the queue d
EVENT timeout
send control packet
schedule burst
stop timer
```

Algorithm 2 Min-Burst-Length

```
EVENT IP packet arrives
d ⇐ destination address of the packet
add packet to the queue d
update burst length information
if minimum burst length exceeded then
    send control packet
    schedule burst
    reset burst length
end if
```

Those two algorithms can be combined, creating the Min-BurstLength-Max-Assembly-Period algorithm [29]. In this case, the burst is sent when either the pre-defined burst size is exceeded, or when the assembly timer times out. In effect, the burst is never unmanageably long, nor is the assembly time excessive.

2.2.2 Adaptive algorithms

Algorithms with fixed time- or size-based thresholds cannot adapt to varying traffic loads. Therefore, it can be expected that their performance will be sub-optimal for any non-typical situation. This can be prevented by using an adaptive algorithm that changes its behavior according to real-time traffic measurements.

Cao *et al.* proposed the Adaptive-Assembly-Period (AAP) algorithm [29]. Here, the burst length threshold is adjusted depending on lengths of previously sent bursts. Their simulation results show that AAP outperforms FAP. They also noted that AAP tends to synchronize with TCP congestion control mechanism, enhancing performance.

Algorithm 3 Adaptive-Assembly-Period

```
EVENT: IP packet arrives
d $\Leftarrow$ destination address of the packet
if timer d is not running then
  initialize timer to current (or initial) assembly period
  start timer
end if
add packet to the queue $d$
EVENT: timeout
send control packet
schedule burst
stop timer
update current assembly period
```

Another, hysteresis-based adaptive assembly algorithm has been proposed by S Oh *et al* in [33] Here, the maximum burst size is adjusted in steps, depending on sizes of previous bursts Also, to avoid excessive assembly times in case of low load, a timer is used

2.2.3 Composite burst assembly

Burst assembly algorithm may constitute a part of Quality of Service scheme VM Vokkarane *et al* proposed the idea of *composite burst assembly* [34], where packets belonging to different QoS classes are placed into the same burst They are arranged in such a way, that the low priority packets are grouped in the tail of the burst Alternatively, if there are more than two classes, the packets are arranged in order of decreasing priority Later, the policies used within the network make the tail more likely to be dropped Details of this and other QoS schemes will be discussed in Chapter 3

2 2 4 Predictive burst assembly

There are three main sources of delay in Optical Burst-Switched networks queuing delay, offset time and propagation delay Of these, the propagation delay cannot be changed, but it is possible to manipulate the other two

Typically the sequence of events is as follows IP packets start arriving at the edge node where they are stored until burst assembly is completed Then, a control packet is sent with a reservation request Processing the packet and setting up the route takes a certain time, and while this is being done the burst waits in a buffer at the edge node Finally, after some pre-determined offset time, the burst is sent

It is possible to improve this mechanism if we take into account the fact that IP traffic can be fairly accurately predicted, at least in a short timescale Then the size of burst and the time of its assembly being finished can be guessed with

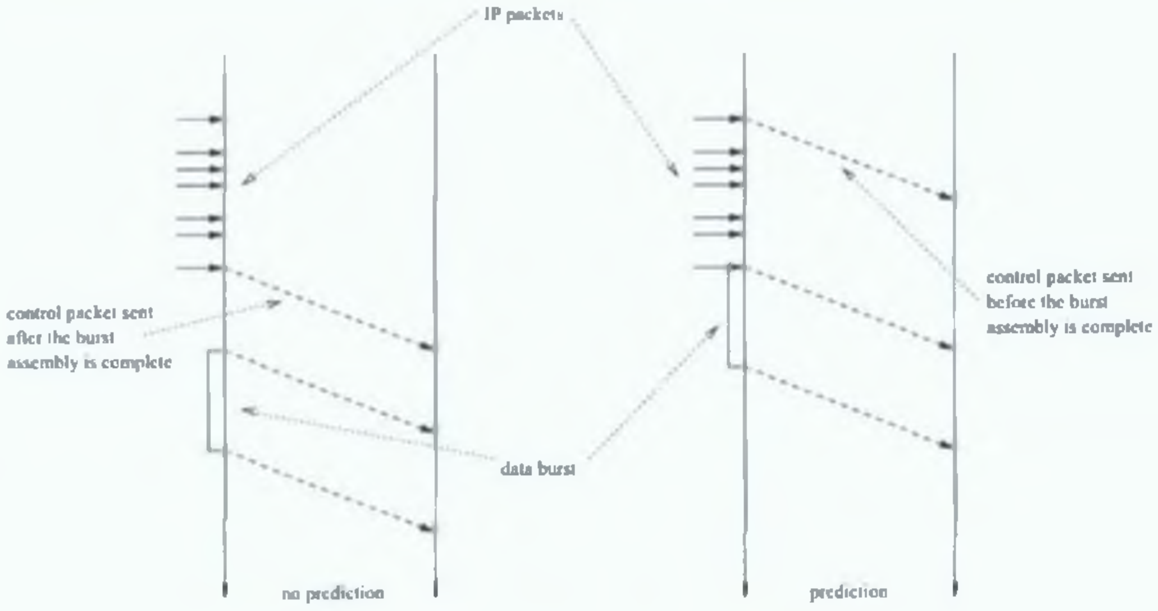

IP packets
control packet sent after the burst assembly is complete
control packet sent before the burst assembly is complete
data burst
no prediction
prediction

Figure 2.4: Predictive burst assembly

an acceptable degree of certainty even when the first bursts start arriving. With this information available, a control packet can be sent earlier, thus minimizing overall delay.

The process is illustrated in Fig. 2.4. In the first case, no prediction is used. The control packet is only sent as the last IP packet arrives. In the second case, both burst size and assembly time are predicted before the process is completed, possibly at the time of arrival of the first IP packet. This allows the edge node to send the control packet much earlier, minimizing delay.

The prediction algorithm can either overestimate or underestimate the burst size. This will result in either poor bandwidth utilization as more bandwidth is reserved than necessary, or in additional delay, as a proportion of packets will have to wait until the next burst is ready.

One such algorithm has been proposed by Hashiguchi *et al.* [35]. In their approach the assembly time is constant, so only burst size has to be predicted. It is assumed that IP packets arrive at approximately the same rate during the assembly process, so the amount of data buffered within a certain fraction of assembly time can be used to estimate the total burst length

It has been proposed that a *linear predictor* can be used to predict network traffic [36–38] Morato *et al* used this technique in burst assembly [39] and found that it reduces end-to-end delay while maintaining low bandwidth waste

2.3 Node architecture

A WDM node should be capable of switching any wavelength from any input port to any wavelength in any output port A typical example of such a node is presented in Fig 2 5 The incoming signal is first split into individual wavelengths and then fed into the switching matrix Then, after being routed to the appropriate output port, each wavelength is - if necessary - converted to another channel and sent to an output fiber

OBS network characteristics place additional requirements on a node that is to be used in such a network First of all, information incoming in a control channel needs to be extracted and analyzed, and then sent again to an appropriate downstream node A node capable of performing this function is presented in Fig 2 6

Optical switches can be realized using the Microelectromechanical Systems (MEMS) technology [40] In this case, the optical signal is directed to an appropriate port by means of microscopic, movable mirrors An example of a 2D MEMS switch is presented in Fig 2 7 There is an array of mirrors in a crossbar configuration Each mirror can be either raised or lowered By raising an appropriate mirror, it is possible to switch signal from any input port to any output port Larger switches can be implemented using 3D MEMS approach Unfortunately, the swithcing times of MEMS switches (tens of milliseconds) are too long for Optical Burst Switching

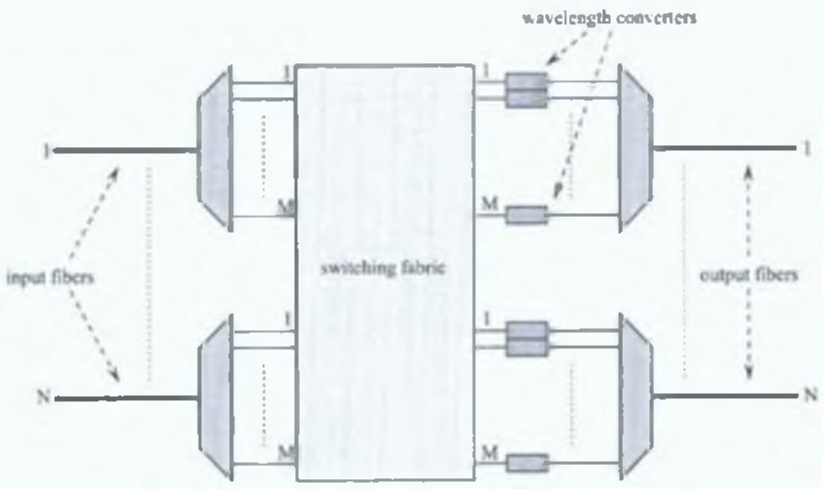


Figure 2.5: Architecture of a WDM node

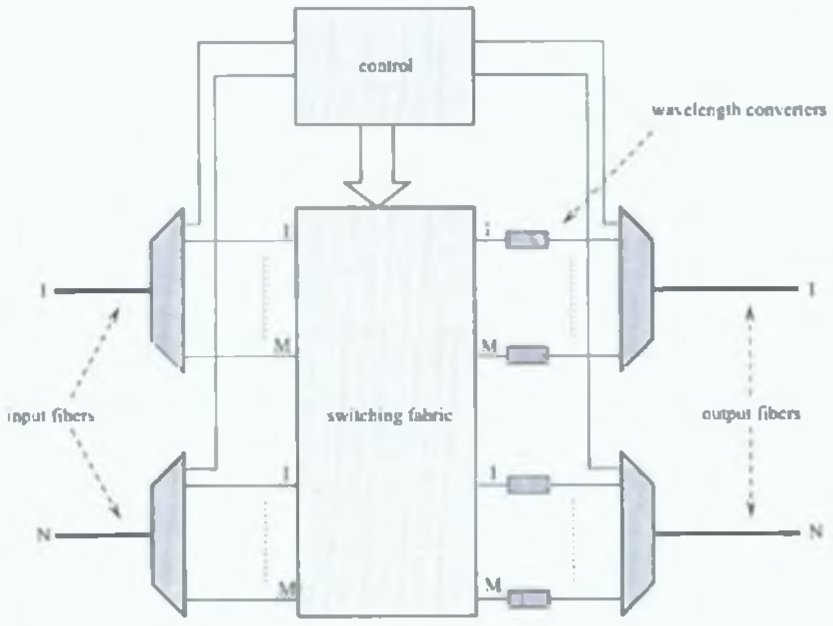


Figure 2.6: Architecture of a generic OBS node

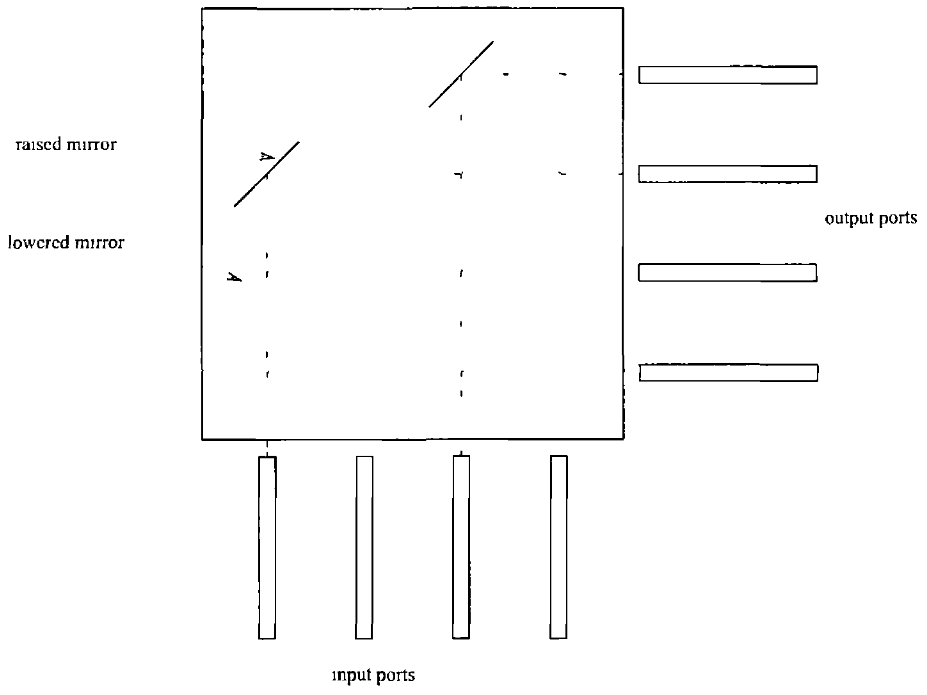


Figure 2 7 2D MEMS switch

2 3.1 Broadcast-and-Select (BAS)

Broadcast-and-Select (BAS) architecture was proposed for use in OPS networks [41, 42], and can be adapted for OBS [43] It is based on Semiconductor Optical Amplifiers (SOAs) [44, 45], acting as optical on/off switches here SOAs are an attractive choice because of their fast switching time, high on/off ratios, loss compensating capabilities and broad amplification bandwidth [43]

An example of such a node is presented in Fig 2 8 [43] Signal from each of N fibers (all wavelengths) is split into M*N parts, where M is the number of channels and N is the number of fibers The SOAs are divided into N*M banks of N, each bank switching signals that are meant to be sent on the same wavelength Therefore, those signals are then directed to filter/converter pairs with the same output wavelength Within the pair, the filter is tuned to extract the appropriate input wavelength

So, if wavelength m_1 in fiber n_1 is to be switched to wavelength m_2 in outgoing

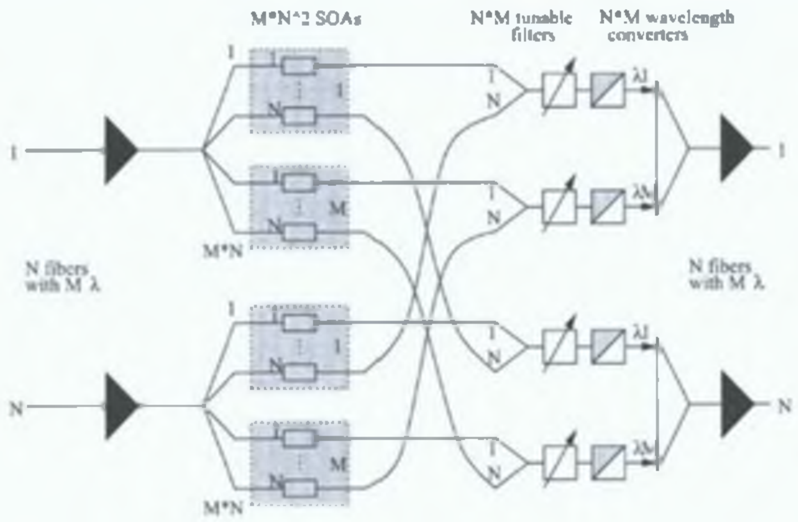


Figure 2.8: Broadcast-And-Select (BAS) node architecture.

fiber n_1, then the following will happen:

- An appropriate SOA will be switched on,
- The appropriate filter will be tuned to wavelength m_1
- The signal will be converted to m_2 in the wavelength converter

Unfortunately, only small nodes (for example one with 4 fibers and wavelengths per fiber [43]) can be built using this architecture, due to power budget issues [46].

2.3.2 Tune-and-Select (TAS)

In a Tune-And-Select architecture the incoming signal is split into individual wavelengths, as shown in Fig. 2.9. Then, each wavelength is fed into a tunable converter, and converted to desired output wavelength. This signal is broadcast to all output fibers via a bank of SOAs.

This type of node requires tunable wavelength converters. Such converters can be realized using OEO conversion with tunable lasers [47–49]. An all-optical solution is possible as well [50].

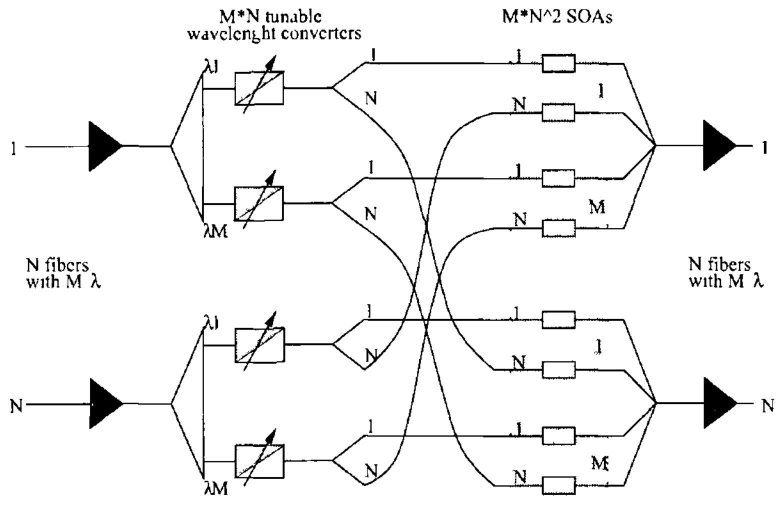


Figure 2 9 Tune-And-Select (TAS) node architecture

2.3.3 Wavelength grating router (WGR) based node

The development of fast tunable lasers and tunable wavelength converters made it possible to use wavelength grating routers [51] (also known as arrayed waveguide grating multiplexers or AWGMs) as switching elements Ramamurtham *et al* described a WGR-based OBS switch [52] Its architecture is presented in Fig 2 10 The incoming signal is first split into individual wavelengths, which are then fed into tunable wavelength converters Then, WGRs direct the signal to one of the output ports, according to its wavelength Output ports of WGRs and the output multiplexers are connected by a fixed *permutation pattern* The blocking properties of the switch depend on the choice of this pattern The authors claim that by using their algorithm, the throughput of such a node can achieve 89% of throughput of a strictly non-blocking node

More advanced switch architectures based on WGRs are also possible Ngo *et al* proposed both strictly non-blocking and rearrangeably non-blockmg architectures, at a cost of a slight increase in complexity [53]

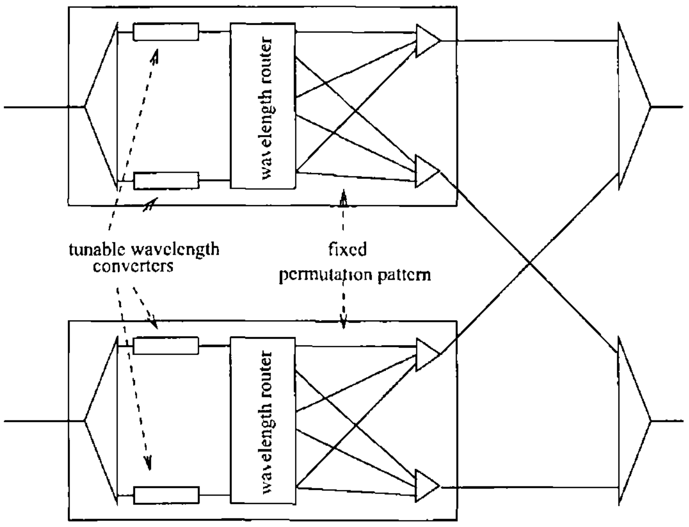


Figure 2 10 Wavelength grating router based node

2.4 Scheduling algorithms

Often, when a connection is to be made, there is more than one channel that can be used In a circuit-switched network the choice is of secondary importance - for example algorithms like round-robin can be used to ensure uniform wear on all components In an optical burst-switched network, however, the choice of a particular channel can and will affect network performance

In an OBS network, reservations are made in advance This may cause gaps or voids to be formed Depending on the choices made, those gaps may be larger or smaller, subsequently more or less likely to be used by other bursts Additionally, bursts may be delayed, either by electrical means in an edge node, or by buffering in a Fiber Delay Line (FDL) in a core node

For those reasons, the choice of a scheduling algorithm is a very important consideration in an OBS network In Chapter 3 a detailed description of algorithms proposed so far will be presented

2.5 Time Sliced Optical Burst Switching (TSOBS)

In an OBS node, a burst arriving on wavelength n might need to be converted to wavelength m in case of a contention. This operation is done by a *wavelength converter*. Unfortunately, those devices are considered to be the largest single cost component of an optical node. Therefore it was proposed by Ramamirtham *et al.* to replace wavelength conversion with switching in time domain [54]. In their approach, each wavelength carries a structure of time slots of equal length. A sequence of slots in consecutive frames is called a channel. A control packet carries information about the incoming burst:

- its arrival time,
- its channel (i.e. slot number),
- its length (i.e. number of slots).

A key building block of TSOBS router is an *Optical Time Slot Interchanger* (OTSI). It is capable of delaying the contents of time slots by different amounts of time, thus changing their relative positions. The structure of an OTSI is shown in Fig 2.11.

The authors claim that their solution offers good statistical multiplexing performance without the need for costly wavelength converters.

2.6 OBS testbeds

Presently, there are no functioning large-scale OBS networks. However, several laboratory testbeds have been implemented:

- Baldine *et al.* described an OBS demonstration network overlaying the Advanced Technology Demonstration Network (ATD*net*) all-optical testbed [55]. ATD*net* is high performance networking testbed in the Washington

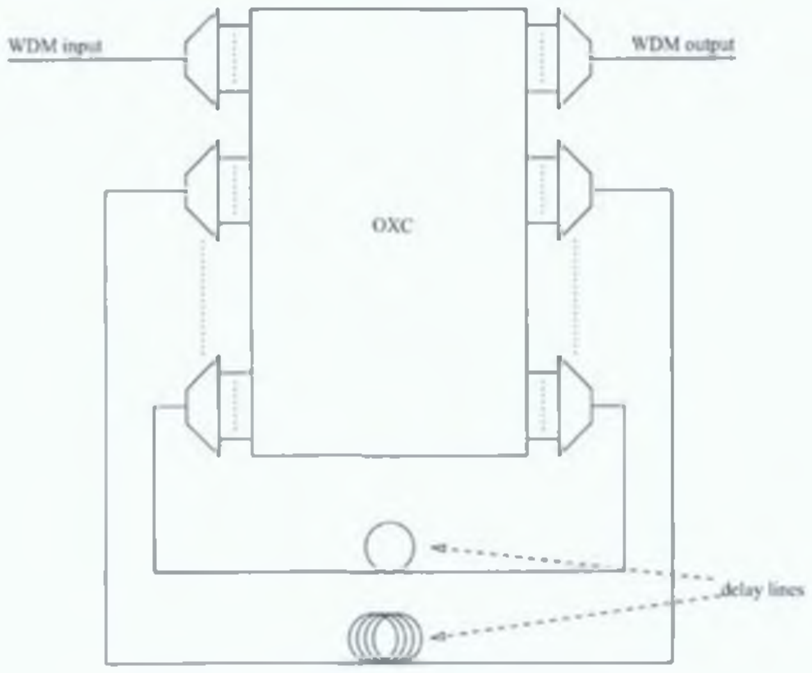


Figure 2.11: ATD*net*Optical Time Slot Interchanger

D.C. area, established by the Defense Advanced Research Projects Agency (DARPA) [56].

Experimental network controllers were placed at three ATD*net* sites: the Defense Intelligence Agency (DIA), the Naval Research Laboratory (NRL), and the National Security Agency's Laboratory for Telecommunications Science (LTS). Each site was equipped with a Firstwave SIOS 1000 MEMS optical crossconnect. The network configuration is shown in Fig. 2.12 [55].

The described demonstration network uses the Just-In-Time (JET) reservation protocol [16], implemented by JITPACs (Just-In-Time Protocol Acceleration Circuit), placed at each node.

- Kitayama *et al.* described work performed by NTT Network Innovations Laboratories, University of Tokyo, Osaka University, and Fujitsu Laboratories along with the coordinator, NTT Communications [57]. In the PHASE 1, a six node network was built, with 3D MEMS optical switches in two

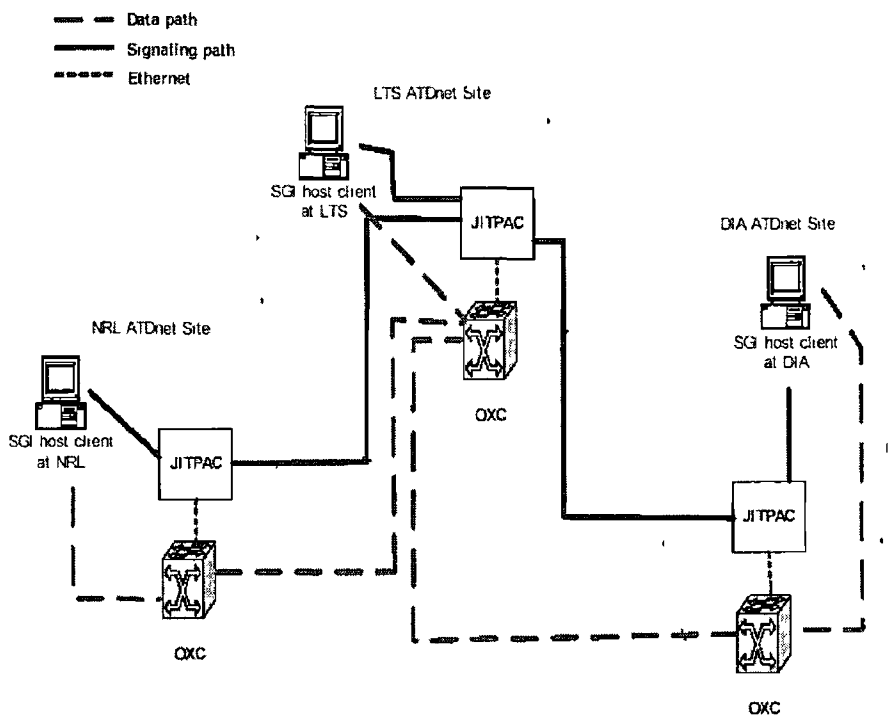


Figure 2 12 ATD*net* OBS testbed configuration nodes, and planar lightwave circuit (PLC) switches in four nodes The overall switching time was 30 ms and the network throughput of 0 87 was obtained

- Sun *et al* built an OBS testbed, consisting of one core node and three edge nodes [58] The architectures of both types of nodes were discussed, along with the format of data burst and control packet The switching time of 3ms was achieved using thermo-optic switches

2.7 Summary

In this chapter the fundamentals of Optical Burst-Switched networks were introduced In particular the evolution of optical networks and motivation for optical burst switching were discussed The basic OBS protocol (JIT and JET) were described along wit their respective advantages and disadvantages The following sections dealt with burst assembly algorithms and OBS node architecture Finally, an OBS variant called Time Sliced Optical Burst Switching was introduced and several existing OBS testbeds were described

Chapter 3

Scheduling algorithms and Quality of Service

The first part of this chapter explains the importance of scheduling in Optical Burst-Switched networks and describes in detail several scheduling algorithms Distinction is made between *legacy* scheduling algorithms and dedicated OBS algorithms The latter are further divided into *per-burst* algorithms and *batch* or *group* scheduling algorithms Additionally, a *rescheduling* algorithm is described

The second part of this chapter deals with Quality of Service provisioning in Optical Burst-Switched networks Starting with the popular *offset-based* QoS scheme, several other approaches are described, including *composite burst assembly* and *intentional burst dropping*

3.1 Scheduling algorithms

The scheduling decision is of a particular importance in Optical Burst Switching networks This is because reservations are made in advance The control packet, carrying information about the data burst arrives ahead of it Therefore, when a reservation request arrives with a relatively small offset time, it is possible that there are other existing reservations in the future In other words, there may be available *voids* or *gaps* Creation of large gaps is considered undesirable, as their bandwidth is likely to be wasted

Fig 3 1 shows two examples of burst scheduling In both cases there are five existing reservations (represented by unfilled rectangles) and one incoming reser-

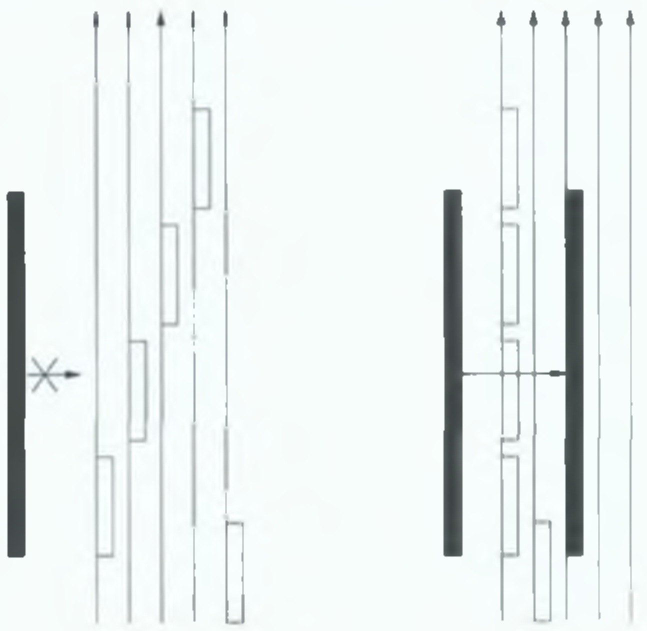


Figure 3.1: Different scheduling algorithms

vation request (represented by a black rectangle). The difference is in the way the existing reservations are arranged. In the first case they are spread over all available channels. In the other one, the reservations are concentrated in the first channel. Even though at any given time there is at least one channel available, the new reservation cannot be made in the first case.

Scheduling algorithms can be classified as either *void-filling* or *non void-filling*. The former are capable of utilizing voids between previously allocated bursts, therefore improving efficiency. The latter can only schedule a burst in a completely free channel, or in a channel that is free from a certain time onwards, i.e. after all existing reservations. This is a simpler approach, but less bandwidth-efficient. It should be noted that most algorithms can have two versions - void-filling and non void-filling. Also, void-filling protocols require use of JET reservation protocol.

Another possible classification depends on whether the algorithm allocates each burst individually (*per-burst* algorithms), or if it operates on groups or *batches* of bursts. Batch scheduling algorithms offer better bandwidth utilization, but are usually more complex and ıntroduce addıtıonal delay

3 1.1 Legacy scheduling algorithms

Certaın well-known algorıthms, for example those used ın tradıtıonal telephony can be used ın burst-swıtched networks While not necessarıly optımal for optıcal burst swıtchıng, theır behavıor ıs well-studıed and they are generally uncomplıcated

- Fırst Fıt - the channels are checked ın some pre-determıned order, most lıkely from the lowest numbered one to the hıghest one The new burst ıs allocated ın the fırst avaılable channel
- Most Used - the average load ın each channel ıs measured ın a certaın tıme wındow The connectıon attempts are made ın order of decreasıng channel load
- Random - connectıon attempts are made ın a random order
- Round Robın - the channels are checked ın a pre-determıned order, but the startıng poınt changes wıth each attempt

Of the algorıthms lısted above, Fırst Fıt and Most Used tend to concentrate traffıc ın a certaın set of channels Thıs results ın a better performance than ın the case of Random and Round Robın, whıch use all wavelength channels evenly [59]

3 1 2 Per-burst schedulıng algorıthms

Per-burst schedulıng algorıthms schedule one burst at a tıme, takıng ınto account only exıstıng reservatıons The decısıon ıs made as soon as possıble and a control packet ıs sent to the downstream node ımmedıately afterwards Thıs approach ıntroduces mınımal delay, but ıs less bandwıdth-effıcıent than batch schedulıng, dıscussed later ın thıs chapter

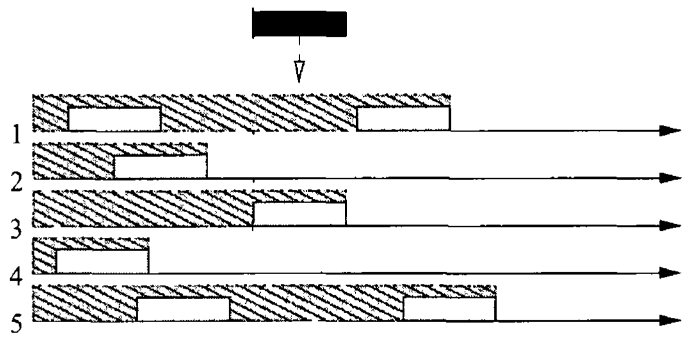


Figure 3 2 Horizon schedulmg algorithm

Horizon

Horizon, proposed by Turner [20] is a fast, non void-filling schedulmg algorithm It uses a concept of *scheduling horizon*, which is defined as the ending time of the latest reservation in a channel The value of the scheduling horizon is kept for each channel When a reservation request arrives, the channel with the largest scheduling horizon smaller than the burst's arrival time is chosen, and its scheduling horizon is updated

An example is presented in Fig 3 2 There are five channels and an incoming reservation request Of the two channels with scheduling horizons smaller that the burst's arrival time (2 and 4), channel 2 will be chosen, because its scheduling horizon is greater

This algorithm only keeps one variable per channel - the ending time of the last reservation If, on the other hand, a full list of reservations was kept (starting and ending times of all reservations), then the channels 1 and 5 could be used as well

Last Available Unused Channel with Void Filling (LAUC-VF)/Minimum Starting Void (Min-SV)

When Last Available Unused Channel with Void Filling (LAUC-VF) [60] is used, the preference is given to a channel, where the gap between the new burst and

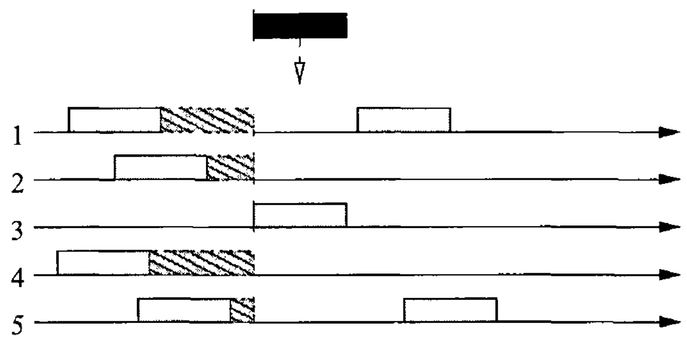


Figure 3 3 Last Available Unused Channel with Void Filling (LAUC-VF)

the previous one will be the smallest In other words, the size of the starting void is used as a metric, as shown in Fig 3 3 In the presented example, there are five channels and one incoming reservation request Channel 3 is unavailable Of the remaining four, channel 5 will be chosen, because the newly created void will be the smallest

Xu *et al* proposed an algorithm called Minimum Starting Void (Min-SV) [61] which is, in fact, an effective implementation of LAUC-VF Here, a geometric techniques and balanced binary search trees are used to optimize the search for a channel with the smallest starting void Also, a non void-filling version has been proposed [60], and is essentially identical to the Horizon algorithm

LAUC-VF is a well-known algorithm and it offers relatively good performance Therefore, it is commonly considered to be a benchmark, and many new scheduling algorithms are compared against it

Minimum Ending Void (Min-EV)

Minimum Ending Void (Min-EV) is another algorithm proposed by Xu *et al* [61] It uses the size of an ending void as a metric, and chooses the channel where its size will be the smallest An effective implementation, similar to the one m case of Min-SV was proposed Note that the endmg void only exist when allocating a burst in an existing void, so Min-SV cannot possibly have a non void-filling

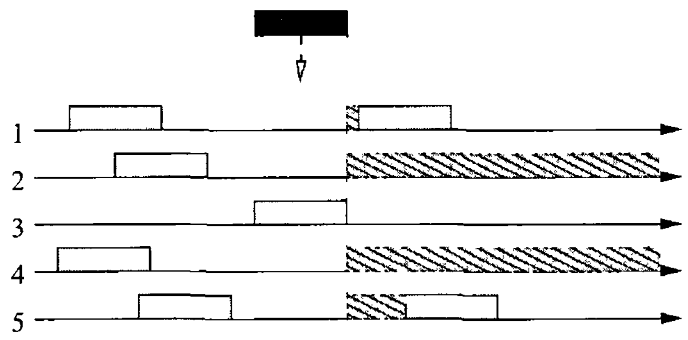


Fıgure 3 4 Mınımum Endıng Voıd (Mın-EV)

versıon

BestFıt

The BestFıt algorıthm, proposed by Xu *et al* [61] trıes to mınımıze the weıghted sum of startıng and endıng voıd sızes Thıs approach ıs obvıously more complıcated than ın the case of MınSV or MınEV, but the authors proposed a sımılarly effıcıent ımplementatıon

Burst Overlap Reductıon Algorıthm (BORA)

Lı *et al* noted that contentıon probabılıty depends on the amount of overlap between bursts arrıvıng at the same node [62] As presented ın Fıg 3 5, the four ıncomıng bursts cannot be allocated ın the outgoıng lınk Z ıf the node ıs not equıpped wıth FDLs, or ıf all ıts FDLs are occupıed However, ıf an upstream node delayed bursts b1 and b4 so that they dıd not overlap wıth b2 and b3 respectıvely, then all four bursts could be easıly allocated as shown ın Fıg 3 6

The authors proposed a *Burst Overlap Reducıng Algoıthm* (BORA) that - as ıts name ımplıes - attempts to reduce the burst overlap at the edge node Bursts can be delayed for arbıtrary perıods of tıme there, as they are stıll ın the electrıcal domaın The algorıthm can also be used ın the network core, but then ıt ıs necessary to use FDLs, and ıts functıonıng ıs lımıted to fıxed delays

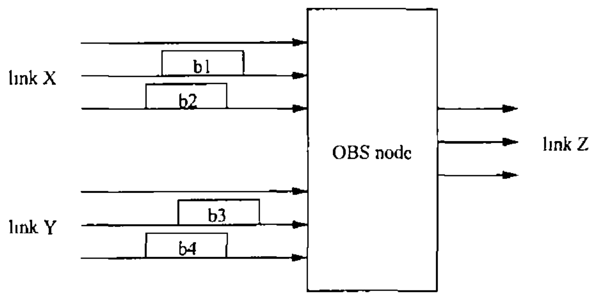


Fıgure 3 5 Overlappıng bursts arrıved

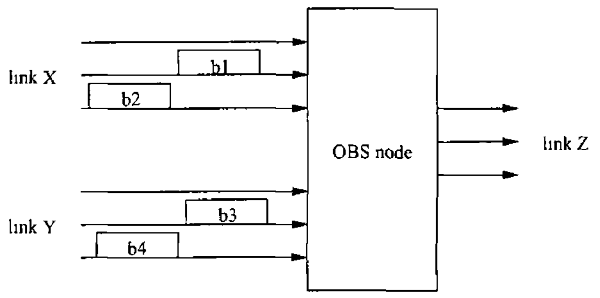


Fıgure 3 6 Reduced overlap

The reductıon ın overlap ıs done at the cost of delayıng bursts To reduce the maxımum delay experıenced by a burst, a maxımum delay ıs defıned If a burst would have to be delayed more, then the channel ıs consıdered unavaılable

Two varıatıons of the BORA algorıthm have been descrıbed

- *Fıxed Order Search* In thıs case, the node always attempts to allocate the burst by searchıng the channels ın a predefıned order, untıl eıther the burst can be allocated or ıs dropped Thıs versıon of the algorıthm ıs called BORA-FS

- *Destınatıon-based Order Search* (BORA-DS) Here, for each destınatıon a set of preferred wavelengths (called *home channels*) ıs defıned When a burst ıs to be sent, the edge node fırst searches the channels belongıng to the preferred set for ıts destınatıon, and only later the remaınıng channels Thıs approach ensures that the overlap of bursts that have the same destınatıon ıs mınımızed

Additionally, both versions can be either void filling or non-void filling The authors provided simulation results proving that both BORA-FS-VF and BORA-DS-VF overperform LAUC-VF m terms of burst loss probability The difference is particularly big (two orders of magnitude) for small loads

Priority-based Wavelength Assignment (PWA)

Wang *et al* considered an OBS network with no wavelength conversion capability In such a network, proper wavelength assignment is of particular importance, as decisions made an one node will affect burst loss probability in downstream nodes Also, only the edge nodes can choose a wavelength

To minimize burst loss m the network core, the authors proposed a *Priority-based Wavelength Assignment* algorithm [63] Each sender using PWA keeps a priority database for every destination node Each time a burst is sent, the database is updated, depending on whether the transmission was successful or the burst was lost Therefore, the wavelengths that offer the best chance of reaching a given destination are assigned the highest priority

When a burst is ready to be sent, the edge node first attempts to allocate it in the highest priority channel, then the second best and so on, until the burst is allocated or lost

3 1 3 Batch scheduling algorithms

Most of the proposed scheduling algorithms consider only one burst at a time Therefore, at the decision time, the following information is available

- Information about the incoming burst - mainly its starting and ending time, but also its source, destination, class and other relevant information
- The existing set of reservations For each reservation, at minimum its starting and endmg time is known

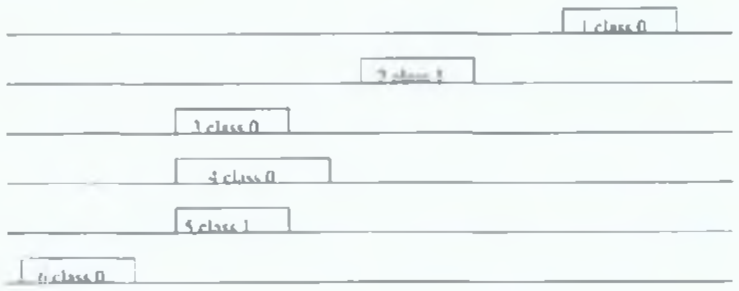


Figure 3.7: Waiting-Queuing-Scheduling algorithm.

Unfortunately, as the decision is made without any delay, no information is available about future reservation requests. This results in a suboptimal allocation of bursts. An alternative approach is called *batch scheduling*. Here, after a control packet is received, it is delayed before the decision is made. During this delay, other control packets are received and likewise delayed. In effect, when the burst is about to be scheduled, the following information can be utilized:

- Information about the burst itself.
- The existing set of reservations.
- Information about bursts that will be scheduled during a period of time equal to the scheduling delay

Tan *et al.* described a Waiting-Queuing-Scheduling (WQS) algorithm based on this principle [64]. In their approach, scheduling is combined with Quality of Service provisioning. Basically, incoming reservation requests are delayed and queued. The position of a new control packet in a queue depends on its class, size and whether it overlaps with packets already in the queue. Basically, bursts with higher priority or longer bursts are placed ahead of other overlapping bursts. When the waiting time expires, the burst is scheduled using one of the existing algorithms, for example LAUC-VF.

An example of the queuing process is shown in Fig. 3.7 [64]. There are six control packets in the queue, belonging to classes 0 and 1, class 1 being the high priority class Control packets 3, 4 and 5 overlap Packet 5 belongs to class 1, so it will be scheduled first, and packet 4 will be the next one, because of its size

It has been demonstrated that this approach can offer low loss ratio in high priority class and also decrease the total burst loss ratio It supports unlimited number of traffic classes Its drawbacks include the necessity of using larger offset times due to the additional delay experienced by control packets

Charcranoon *et al* proposed a *Maximum Stable-Set Interval Scheduling Algorithm* (MSSIS) [65] Here, incoming reservation requests are briefly delayed, and then graph theory is used to schedule them Presented simulation results show that this approach offers about 5% improvement over traditional immediate reservation Another batch scheduling algorithm has been described in [66]

3.1 4 Burst rescheduling algorithms

Once a burst has been scheduled, the possibilities of modifying the reservation are very limited In particular, changing its arrival time would require using FDLs On the other hand, moving an existing reservation to another wavelength (rescheduling) is easier, and only notifying downstream nodes of the change will be necessary This may make it possible to schedule a burst that would otherwise be dropped Consider the situation presented in Fig 3 8 a) There are two channels, four existing reservations and one incoming reservation request Unfortunately, it is not possible to make the reservation But if reservation 3 is rescheduled in the other channel, as presented in Fig 3 8 b), the first channel becomes available for the new burst

Tan *et al* proposed two algorithms using the idea of rescheduling existing reservations in order to accommodate new bursts [67] *On-demand burst rescheduling algorithm* (ODBR) first attempts to schedule the new burst without modifying existing reservations Only when it fails, it tries rescheduling in order to accommodate the new burst All the wavelengths are tested one by one, and the one where the newly created void would be the smallest is chosen

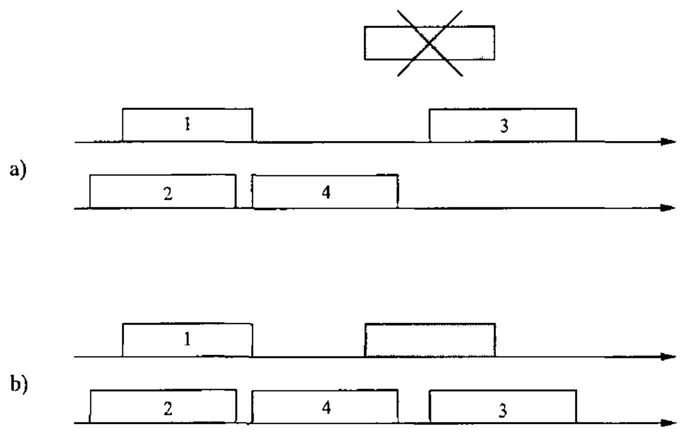


Fıgure 3 8 Burst reschedulıng

Aggressıve burst reschedulıng algorıthm (ABR) attempts reschedulıng *after* schedulıng a new burst When a burst ıs allocated ın wavelength C_i, the algorıthms searches for a burst allocated ın some other wavelength C_j that can be moved to C_i Thıs reduces the probabılıty of contentıon for bursts arrıvıng later

The authors present sımulatıon results that show that the performance of theır algorıthms ıs sımılar to that of LAUC-VF and better than ın the case of LAUC The sıgnallıng overhead caused by reschedulıng bursts ıs estımated to be ın the range of 2% ın case of ODBR and 20% ın case of ABR

3.2 Quality of service

Qualıty of Servıce provısıonıng ıs stıll an ımportant ıssue ın the Internet as many servıces, lıke Voıce over IP (VoIP) requıre certaın bandwıdth or delay guarantees On the other hand, bulk data transfers are ınsensıtıve to delay and faırly ınsensıtıve to loss ratıo Much effort has been devoted to provıdıng QoS ın the Internet Unfortunately, most of the proposed schemes are buffer-based In the optıcal domaın, sıgnal can only be delayed for a short perıods ın Fıber Delay Lınes (FDLs) Thıs makes those QoS schemes ımpractıcal ın optıcal networks

Provıdıng QoS ın OBS networks requıres fındıng new schemes, that eıther do not requıre bufferıng, or can be used wıth the lımıted bufferıng capabılıty of FDLs In thıs chapter several such schemes are descrıbed, theır areas of applıcatıon and potentıal weaknesses are ıdentıfıed

3 2 1 Offset-based QoS

In OBS networks, bandwıdth ıs reserved ın advance The control packed ıs followed by the data packet after a certaın perıod of tıme, called *offset tıme* It has been noted [9] that loss probabılıty experıenced by bursts wıth larger offset tıme ıs lower than ın case of bursts wıth smaller offset tıme The logıcal explanatıon of thıs effect ıs that ıf the offset tıme ıs relatıvely large, then bandwıdth wıll be reserved *before* other bursts have a chance to make a reservatıon

Consequently, one of the most wıdely proposed solutıons was to use *addıtıonal offset tıme* to provıde QoS guarantees to certaın classes of bursts [68–70] In thıs scheme, bursts belongıng to hıgh prıorıty classes use larger offset tıme than strıctly necessary, achıevıng a certaın degree of *ısolatıon* from lower-prıorıty bursts Yoo *et al* [70] demonstrated that when burst lengths are exponentıally dıstrıbuted, then usıng addıtıonal offset tıme equal to fıve tımes the average burst length ensures over 99% of class ısolatıon Alternatıvely, offset tıme can be adjusted to achıeve desıred loss probabılıtıes by usıng a heurıstıc formula proposed by So *et al.* [71]. A detailed analytical model of an offset-based QoS scheme has been prosented in [72].

A big advantage of this approach is that it exploits effects that are natural in Optical Burst Switching networks. Therefore, its added functionality resides at the edge and no modifications in the core of the network are required. The scheme can support multiple classes of service with good class isolation.

However, there are some disadvantages as well:

- A *burst selecting effect* has been observed [68]. Offset based QoS scheme creates significant amount of voids. Due to their presence, loss probability will be higher in case of longer bursts. This in turn means that the loss probability alone becomes an unreliable metric of network performance.

- To ensure sufficient class isolation, the additional offset time has to be on the order of several mean burst lengths [68]. The increase is even bigger if multiple classes are to be supported. The resulting end-to-end delay in high priority class may be unacceptable.

- In a complex network, any link may carry aggregated traffic, with bursts destined to several different edge nodes. As the hop distances to their respective destinations are unlikely to be identical, their offset times will vary as well. This makes it difficult to ensure that traffic classes are isolated in the entire network.

3.2.2 Composite burst assembly

Vokkarane *et al.* proposed an interesting QoS scheme [34, 73], where IP packets belonging to different classes are assembled into one burst. However, the high priority packets are placed in the head of the burst, and the low priority packets in its tail. During a contention, only the tail of contending burst is dropped, as shown in Fig 3.9. Authors proposed several composite burst assembly algo-

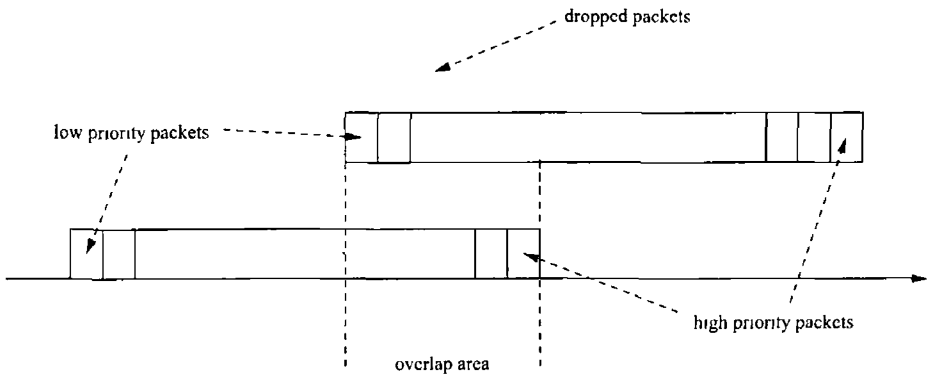


Figure 3 9 Tail dropping technique

rithms Their simulations show that this approach outperforms those with only a single class of packets per burst

An essentially identical technique has been proposed by Arakawa *et al* [74] The only difference is that packets are assembled in opposite order, and the network core drops burst head in case of contention

3 2 3 Intentional burst dropping

Chen et al proposed a proportional QoS scheme for OBS networks [75] In their model, network provider sets a set of factors, and for all service classes, a given QoS metric is proportional to its factor

$$\frac{q_i}{q_j} = \frac{s_i}{s_j}$$

where q_i and s_i are the QoS metric and a QoS factor, respectively, for class i The proportional QoS is also defined for short time scales

$$\frac{c_i(t,t+T)}{c_j(t,t+T)} = \frac{s_i}{s_j}$$

where $c_i(t,t+T)$ is average value of a QoS metric over the time period $t,t+T$

This scheme keeps two counters, of all bursts and dropped bursts, for each traffic class If the loss ratio in any class is lower than it should be, according to the equation above, then a certain proportion of bursts belonging to this class will be intentionally dropped Those counters are periodically reset to make sure that the most recent history is taken into account

Another QoS scheme utilizing intentional burst dropping was proposed by Zhang et al [76–78] In this case, the technique is called an *early drop scheme* and is used to provide absolute QoS guarantees An *early dropping probability* is calculated for each class, based on measured loss ratio in the immediate higher class This mechanism is similar to random early detection (RED), except that congestion is detected by measuring loss ratio instead of queue size

Zhou *et al* suggested that intentional burst dropping can also be used to ensure fairness in Optical Burst-Switched networks [79]

3 2.4 Assured Horizon

Dolzer proposed a QoS framework called *Assured Horizon* [31, 80] Its functionality is split between network edge and network core

- *Edge node* Incoming packets are classified according to their class and assembled separately A fixed-time assembly algorithm is used On timeout, the assembler checks if the queue size does not exceed its predefined maximum value If not, then the burst is marked as compliant (C) and sent into the network Otherwise, the remaining packets will either wait until the next burst is generated, or, if their total length exceeds a certain threshold, they will be assembled into separated burst and marked as non-compliant (NC)

- *Core node* Each node in the network core is preceded by a *burst dropper* When in a *regular state*, the dropper does not drop any bursts However, when the number of allocated wavelengths in the node exceeds a certain threshold, the dropper switches to a *congestion state* and drops all bursts marked as NC

Sımulatıons prove that thıs QoS scheme offers servıce dıfferentıatıon that can be adȷusted ın a wıde range, comparable to electrıcal solutıons

3 2 5 Preferred wavelength sets

Wan *et al* [81] proposed an Optıcal Burst-Swıtched network based on *dynamıc wavelength routıng* Theır proposal dıffers from a "standard" OBS network ın that ıt contaıns a *control node* - a central entıty that serves as an ıntermedıary ın all reservatıon requests In other words, the control ın the network ıs centralızed Another dıfference ıs that the edge node waıts for an acknowledgment before sendıng data

The authors proposed a Qualıty of Servıce scheme based on wavelength quotas - dynamıc preferred wavelength sets (D-PWS) Basıcally, for each QoS class, a floor level and ceılıng level of wavelength quota are defıned No traffıc class can be assıgned more wavelengths at once than defıned by ıts ceılıng level, but at the same tıme, the mınımum set of wavelengths ıs guaranteed by the floor level

The authors compare theır scheme wıth a scheme usıng statıc preferred wavelength sets (S-PWS) and fınd that D-PWS can guarantee QoS effectıvely, and offers better network utılızatıon than S-PWS

3.2 6 Preemptıve wavelength reservatıon protocol (PWRP)

Lıao *et al* proposed an algorıthm called PWRP, capable of provıdıng proportıonal QoS for an arbıtrary number of traffıc classes [82] Each node keeps an *usage profile* for each traffıc class A class ıs consıdered to be *ın pıofile* ıf ıt does not exceed a predefıned usage lımıt If a gıven class ıs out of profıle, then ıts reservatıons can be preempted A sımplıfıed dıagram of the algorıthm ıs presented ın Fıg 3 10

If the burst beıng preempted ıs already beıng transmıtted, then the node wıll send a termınatıon sıgnal, so that downstream nodes learn about the preemptıon However, ıf the transmıssıon has not started yet, then no sıgnal wıll be sent,

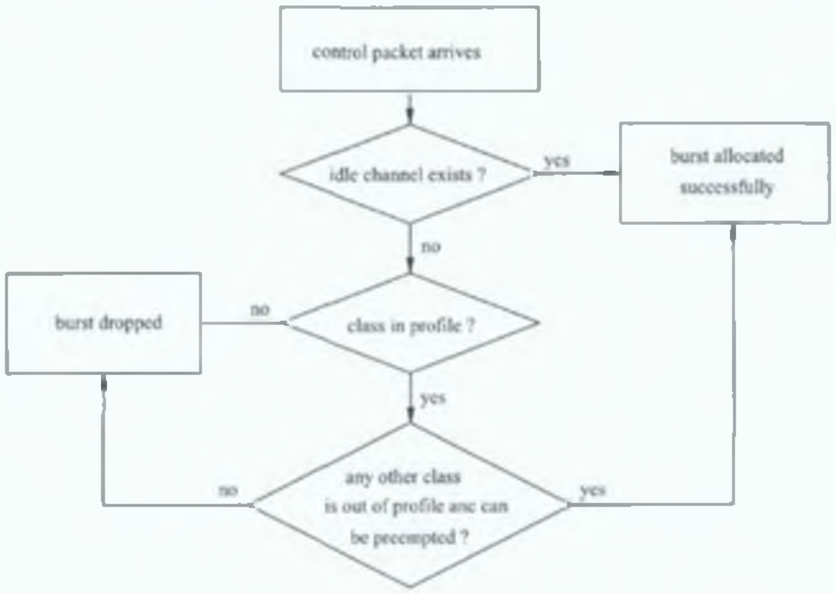


Figure 3.10: PWRP algorithm

and all other nodes will rely on signal detection to decide if the burst has been preempted or not.

The authors present simulation results that prove that PWRP algorithm outperforms both random dropping algorithm and offset-based QoS in terms of blocking probability and resource utilization.

3.2.7 Probabilistic Preemptive scheme (PP)

Yang *et al.* proposed a flexible preemptive scheme, capable of supporting multiple classes of service with arbitrary blocking ratios between classes [83]. When a high priority reservation request arrives, an attempt is made to allocate it in a free wavelength. If this is impossible, then all existing reservations are re-examined. If a reservation for a lower priority burst that could be preempted to accommodate the new burst is found, then it is preempted with a predefined probability p.

The process is illustrated in Fig. 3.11. There is one channel with one existing low priority reservation and an incoming high-priority reservation request. There are two possibilities. The existing reservation will be preempted with probability

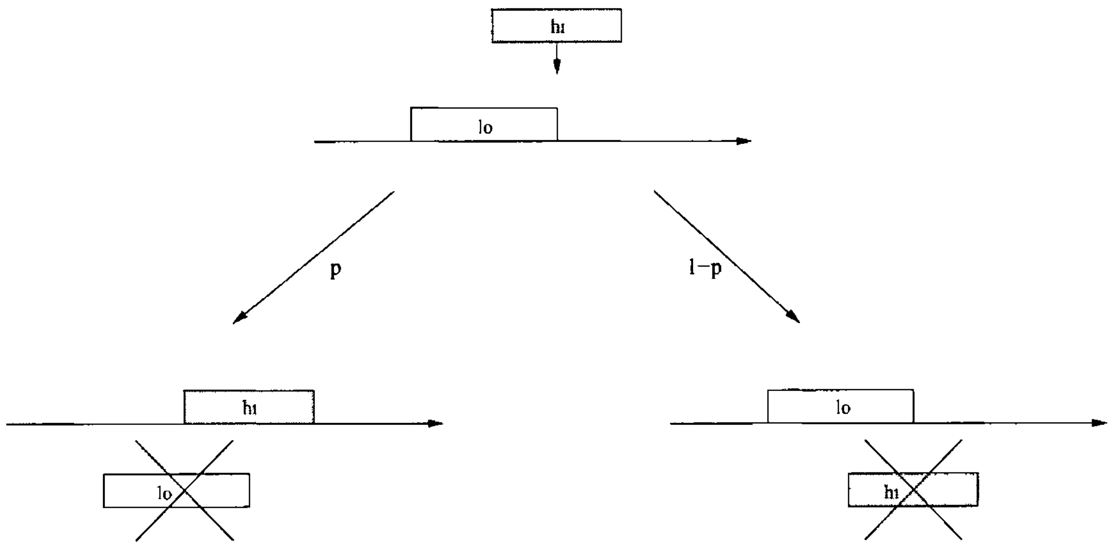


Figure 3 11 Probabilistic Preemptive scheme

p Alternatively, the low priority reservation will be kept and the incoming burst will be dropped with probability *1-p*

By adjusting the preempting probability it is possible to set an arbitrary ratio of blocking probabilities between classes Also, according to the presented results, the scheme does not increase the overall blocking ratio and does not introduce additional delay

Tachibana *et al* proposed another preemptive scheme [84], where both the downstream and upstream nodes are notified if a preemption occurs This reduces the waste of bandwidth and improves network utilization

3 2.8 Partially Preemptive Burst Scheduling with Proportional Differentiation

Cankaya *et al* proposed a scheme that combines partial burst preemption with providing proportional QoS guarantees [85] In this scheme, when a contention occurs, two courses of action are possible Either the overlapping part of existing reservation will be dropped, and the entire new burst will be scheduled, or the exıstıng reservatıon wıll be left untouched and only the non-overlappıng part of the new burst wıll be scheduled The exact choıce depends on whether the contendıng bursts conform to theır usage profiles The usage profiles are defined ıdentically as ın Sectıon 3 2 3

Thıs QoS scheme not only provıdes servıce dıfferentıatıon, but also, accordıng to presented sımulatıon results, ımproves overall burst loss probabılıty and utılızatıon

3 2.9 DiffServ over Optical Burst Swıtchıng (DS-OBS)

Blake *et al* presented a framework for supportıng IETF *DıffServ* [86] ın Optıcal Burst-Swıtched networks [87] In theır approach an OBS edge node ıs combıned wıth a DıffServ edge router and ıs responsıble for

- Traffic classıfyıng,
- DıffServ traffic condıtıonıng,
- Burst assembly,
- Offset Management,
- Burst and control packet generatıon

The authors proposed a burst assembly algorıthm that supports the DıffServ servıces and adapts ıts parameters accordıng to traffic condıtıons

In a core node, arrıvıng control packets are first buffered ın multıple queues, accordıng to theır class and destınatıon Instead of schedulıng them on a first come - first served basıs, the node performs prıorıty schedulıng, thus provıdıng dıfferentıated servıce Consequentıally, dıfferentıated servıce ıs achıeved for the data bursts as well The authors claım, based on theır sımulatıon results, that thıs scheme not only provıdes QoS dıfferentıatıon, but also enhances the whole network performance

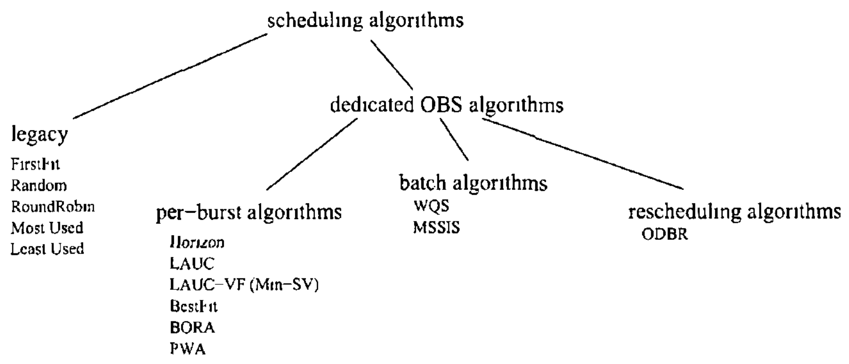


Figure 3 12 Classification of schedulmg algorithms

QoS mechanism	used in
extra offset time	[9,69,71,72]
partial dropping	[34,73,74,85]
intentional dropping	[31,75–78,80]
admission control	[31,80]
centralized control	[81]
wavelength sets	[81]
preemption	[82–84]
priority scheduling	[87]

Table 3 1 QoS mechanisms

3.3 Summary

This chapter presents two problems in Optical Burst-Switched networks - scheduling algorithms and Quality of Service provisioning The first section discusses the importance of burst schedulmg in OBS networks, describes existing algorithms and classifies them The summary of this classification is presented in Fig 3 12

The second part of this chapter deals with Quality of Service mechanisms in OBS networks Firstly, it underscores the need for QoS provisioning in backbone networks The, several QoS mechanisms are described along with the proposed schemes that utilize them Table 3 1 summarizes this information

Chapter 4

Cost functions

In this chapter the idea of a *cost function* is described Firstly, the motivation is presented, along with a formal definition of a cost function, theoretical analysis and a proposed formula Then several possible applications of cost functions are presented, in particular

- scheduling algorithms,
- FDL management,
- Quality of Service - intentional burst dropping

4.1 Motivation

In Optical Burst Switched networks reservations are made in advance This, due to differing offset times, leads to the creation of *gaps* or *voids* In other words, the available bandwidth is being fragmented, resulting in lower bandwidth utilisation and higher loss ratio Because of that, the alignment of bursts in a channel is of a great importance

Many existing scheduling algorithms use some sort of metric to judge the suitability of a particular channel for a particular burst However, the decision is always *hard*, i e the metric is only used to choose the best channel and is not used elsewhere In particular, while a certain algorithm may decide that channel m is better that channel n, it does not try to estimate the magnitude of the difference

It can be reasonably expected that the network performance can be improved in two ways

- A better type of metric (a *cost function*) can be designed,
- Its value can be used not only for burst scheduling, but also for other purposes

As will be shown in this and the following chapters, the cost function achieves both of those goals

4.2 Definition

When any burst is allocated in any channel, it makes its bandwidth unavailable for other bursts By doing so, it increases the probability of contention for future bursts $P_B(B_m|B_n) > P_B(B_m|{}^{\neg}B_n)$, for $m > n$, where $P_B(B_m|B_n)$ - blocking probability for burst m, if burst n was allocated and $P_B(B_m|{}^{\neg}B_n)$ - blocking probability for burst m, if burst n was **not** allocated

The cost function $C()$ is defined as a **value that is an indicator of the probability of future contention, caused by allocating a particular burst in a particular channel** $C(B_n) \sim P_B(B_x|B_n)$

4.3 Cost function formulas

The state of a channel can be described by a set of reservations Each reservation consists of its starting time and ending time If we denote the starting time of the i-th reservation by S_i and its ending time by E_i then the state of the entire channel can be represented by vectors $\bar{S} = [S_1, S_2, \ldots, S_n]$and $\bar{E} = [E_1, E_2, \ldots, E_n]$ where n is the number of reservations The newly arriving burst is described by its starting and ending times, S and E, respectively

The cost function is defined in relation to the channel state, the new burst and possibly other variables

$$C = f\left(\bar{S}, \bar{E}, S, E, \ldots\right) \tag{4.1}$$

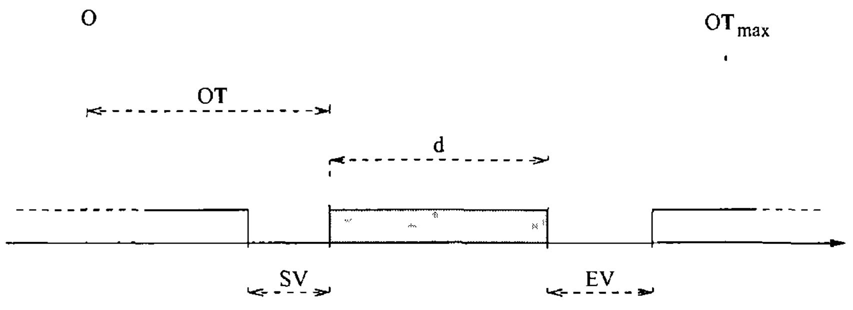


Figure 4 1 A single-channel OBS system

The value of the cost function is called the channel cost Some of the existing channel allocation algorithms can be described using the idea of cost functions For example, in the case of Last Available Unused Channel with Void Filling (LAUC-VF) or Min-EV [60, 61] the value of the cost function is simply the size of the starting or ending void created by allocating the burst, respectively For example in the case of LAUC-VF the cost function is

$$C_{LAUC-VF} = \min(S - E_i) \tag{4 2}$$

for all non-negative values of $(S - E_i)$ For Minimum Ending Void (Min-EV)

$$C_{Min-EV} = \min(S_i - E) \tag{4 3}$$

for all non-negative values of $(S_i - E)$ Both algorithms allocate the burst in the channel where the value of the cost function is the lowest

It is difficult to estimate the probability of interfering with future bursts based on the channel state alone Therefore, other information will be included to achieve better results The general form of the cost function is

$$C = f\left(\bar{S}, \bar{E}, S, E, OT_{min}, OT_{max}\right) \tag{4 4}$$

where $OT_{min}$and OT_{max} are the minimum and maximum offset times expected at the node, respectively

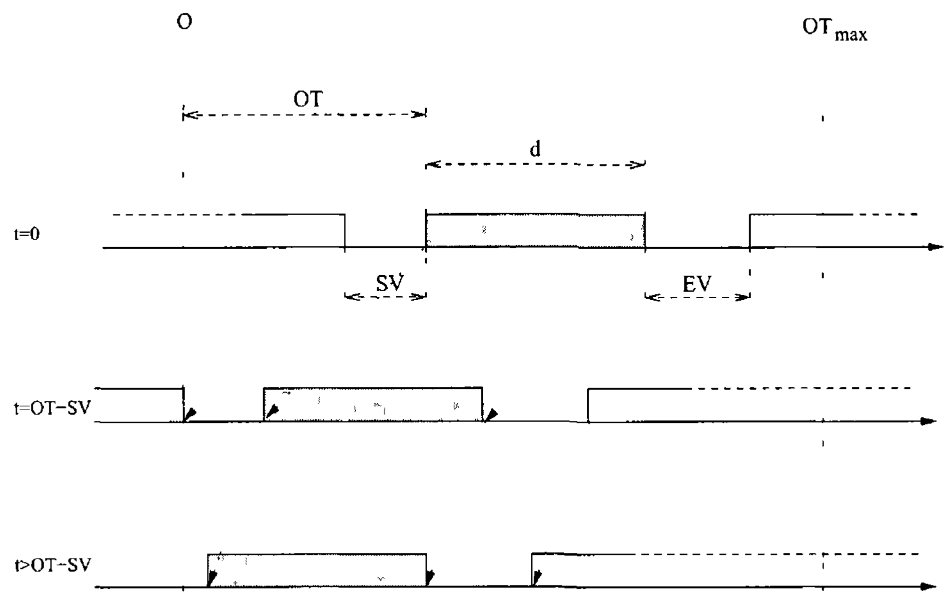


Figure 4 2 Burst transmission process

To design a cost function formula, let us consider a simple, single channel OBS system presented in Fig 4 1 In the channel, there are two existing reservations (possibly more), and a new burst, represented by a grey rectangle, is allocated This creates two voids, starting void and ending void, whose size is denoted by SV and EV, respectively

Let us further assume the following

- The length of a burst is constant and denoted d,
- The interarrival time of reservation requests is exponentially distributed and its mean value is denoted T_{arr},
- Offset times are uniformly distributed between 0 and OT_{max},
- The offset time of the new burst is denoted by OT,
- SV and EV are smaller than d

Let us recall the cost function definition from Section 4 2 We will assume that a burst caused a contention if, before its transmission is finished, another burst is dropped, *that could be accepted, had the original burst not been allocated* In other words, if removing the original burst would allow the new reservation to be made, it is considered to have caused a contention

The transmission of the burst in Fig 4 1 will be finished in $OT + d$ We need to calculate the probability that the burst will cause a contention within this timeframe This time can be divided into two distinctive periods, as shown in Fig 4 2

1 First period, $0 < t < OT - SV$, when the burst is not being transmitted yet, and the entire starting void is available,

2 Second period, $OT - SV < t < OT + d$, when the available part of the starting void is decreasing or the burst is being transmitted

The situation at the end of the first period is shown in Fig 4 3 If, at any time during this period another reservation request arrives, its starting time will fall anywhere within the rectangle marked "offset range" The contention, as previously defined, will occur, if it falls within the rectangle marked "contention range" As the offset time distribution is uniform, the probability of any single burst contending with the original burst can be calculated as the proportion of contention range and offset range The beginning and ending of the contention range can be calculated as

$$CR_{begin} = OT - SV - t$$

$$CR_{end} = OT + EV - t$$

The length of the contention range, calculated by substracting those values, is equal to the sum of starting and ending voids, so

$$P_{1burst} = \frac{SV + EV}{OT_{max}}$$

where P_{1burst} is the probability of contention if one burst arrived within this period If, two reservation requests arrive, then

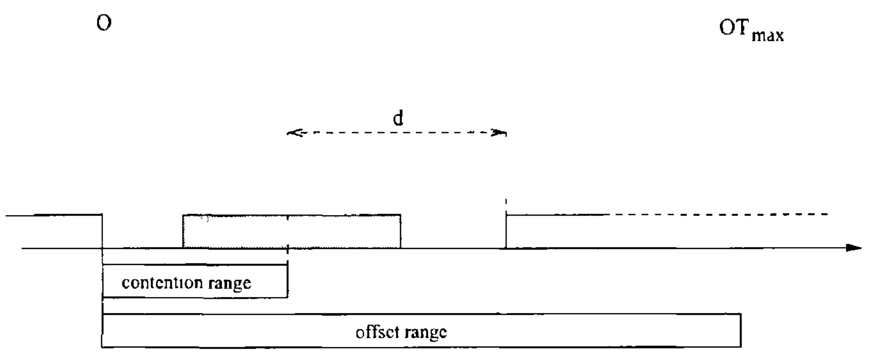


Figure 4 3 Contention range in the first period

$$P_{2bursts} = 1 - (1 - \frac{SV + EV}{OT_{max}})^2$$

And for k reservation requests

$$P_{kbursts} = 1 - (1 - \frac{SV + EV}{OT_{max}})^k$$

where $P_{2bursts}$ and $P_{kbursts}$ are the probabilities of contention if two and k bursts arrive, respectively

The probability of k reservation requests arriving within a certain period is given by the Poisson distribution

$$f(k\ \lambda_a) = \frac{e^{-\lambda_a}\lambda_a^k}{k!}$$

where λ is the expected number of reservation requests arriving within this period

$$\lambda_a = \frac{OT}{T_{arr}}$$

Therefore, the total probability of a contention occurring within the first period is given by the formula below

$$P_a = \sum_{k=1}^{\infty} \frac{e^{-\lambda}\lambda^k}{k!}(1 - (1 - \frac{SV + EV}{OT_{max}})^k)$$

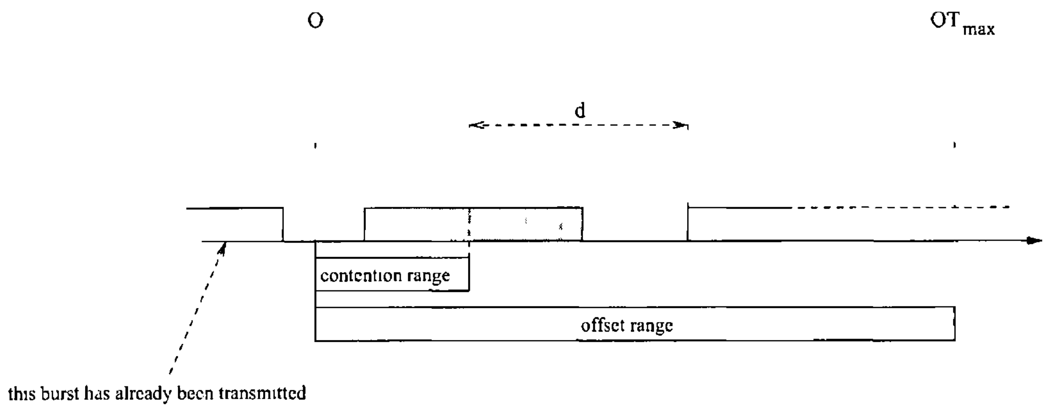


Figure 4 4 Contention range in the second period

The situation is more complicated in the second period As shown in Fig 4 4, the contention range is continuously decreasing

$$P_{1burst} = \frac{OT + EV - t_2}{OT_{max}}$$

$$t_2 = t - (OT - SV)$$

To simplify the calculations let us divide this period into n ranges, each Δt long

$$\Delta t_2 = \frac{SV + EV}{n}$$

If n is large enough, the size of the contention range and thus the contention probability can be considered constant within each Δt

$$t(i) = OT - SV + i\Delta t_2$$

$$P_{1burst}(i) = \frac{OT + EV - (OT - SV + i\Delta t_2)}{OT_{max}}$$

$$P_{1burst}(i) = \frac{EV + SV - i\Delta t_2}{OT_{max}}$$

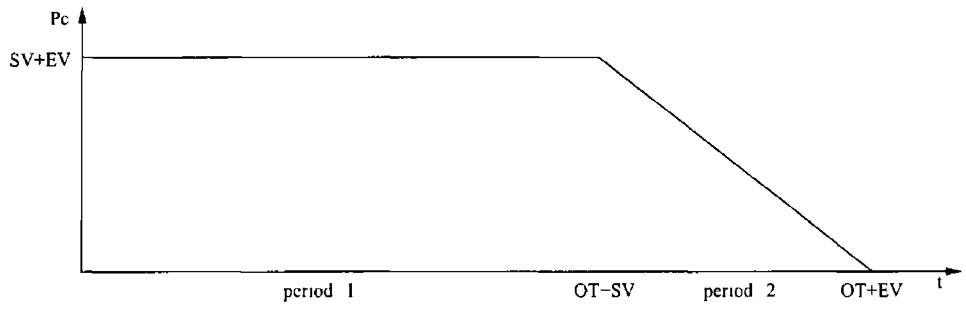


Figure 4 5 Graphical representation of contention probability

$$P_{kbursts} = 1 - (1 - \frac{EV + SV - i\Delta t_2}{OT_{max}})^k$$

The probability of k bursts arriving within any Δt period is given by the Poisson distribution

$$f(k\ \lambda_b) = \frac{e^{-\lambda_b}\lambda_b^k}{k!}$$

To obtain the probability of contention in the second period, we have to sum all the fractional probabilities

$$P_b = \sum_{i=0}^{n}\sum_{k=1}^{\infty} \frac{e^{-\lambda_b}\lambda_b^k}{k!}(1 - (1 - \frac{EV + SV - i\Delta t_2}{OT_{max}})^k)$$

The total probability of contention until the burst gets transmitted is the sum of probabilities in the first and second periods

$$P = P_a + P_b$$

Let us now examine the obtained formula more closely, to determine the relative weights of parameters that influence the contention probability The three parameters are the starting void (SV), the ending void (EV) and the offset time (OT)

In the first period, the probability of contention is constant, does not change with time, and is proportional to $SV + EV$ In the second period, it gradually decreases to 0, as shown in Fig 4 5 The total probability of contention is represented by the area under the curve This area can be expressed as a function of SV, EV and OT Then, in order to determine the relative weights of starting and ending voids, the obtained formula is differentiated with respest to S and EV

$$\frac{dP}{dSV} = OT - SV$$
$$\frac{dP}{dEV} = OT + EV$$

It is apparent that, for very small offset times, the influence of the starting void decreases to 0 (the starting void cannot be smaller than the offset time, so $OT - SV \to 0$) This leads us to the conclusion that for small offset time the ending void will be more important than the startmg void, and for large offset times the difference will be negligible

For example, the size of the starting void does not matter if the burst arrived with the smallest possible offset time No other burst will be allocated in this gap anyway Therefore, the proposed cost function first calculates the channel price according to LAUC-VF and Min-EV, then applies appropriate weights to those results and chooses the smaller value

$$C = \min\left(\frac{C_{LAUC-VF}}{OT - OT_{min}}, \frac{C_{Min-EV}}{OT_{max} - OT}\right) \tag{4 5}$$

This formula reflects the relative weights of SV and EV, depending on the offset time However it does nor approximate the previously derived probabilities in a strict sense

The average value of the cost function as presented above depends on the average void size which, in turn, depends on the link load In some cost function applications (in particular the intentional burst droppmg, described later in this chapter) it is desirable to reduce the dependence of the average cost function value on the load This can be achieved by multiplying the void sizes by the link load

$$C = \min\left(\frac{C_{LAUC-VF}}{OT - OT_{min}}, \frac{C_{Min-EV}}{OT_{max} - OT}\right) L_{link} \qquad (4.6)$$

where L_{link} is the link load at a given time, as a fraction of maximum link load

4.4 Applications of a cost function

A cost function gives us the knowledge about the quality of the alignment of a given burst with other bursts It can be seen as an imaginary cost of using a particular channel for this burst This section explains how this information can be used to improve the performance of OBS networks A cost-based scheduling algorithm will be presented in Section 4 4 1 and in Section 4 4 2 it will be explained how a cost function can be used to match the size of an FDL bank with the needs of a node Finally, in Section 4 4 3 a cost-based burst dropping algorithm will be presented

4.4 1 Scheduling algorithms

Let us imagine an OBS node Certain sets of resources will be contained within this node Examples of such resources are

- outgoing links,
- wavelength converters,
- Fiber Delay Lines

When a reservation request for an incoming burst arrives at the node, it will use some or all of those resources For example, the burst may be forwarded on the same wavelength, it may also be converted to another wavelength and/or delayed in an FDL Frequently, there is more than one way of handling a burst A contention may be resolved by wavelength change, buffering or both

It can be expected that bursts will have to compete for access to these resources A burst may have to be dropped because there is no available wavelength in the outgoing link, but also because it cannot be converted to the available wavelength due to lack of a wavelength converter (if the node does not have full wavelength conversion capability) or because a FDL channel is occupied by other bursts

A cost function can be used to assign an imaginary price to a link A logical extension of this idea is to put a price tag on all other resources in a node Then, we could define total cost of allocating a burst as the sum of all fractional costs

$$C_{total} = C_{channel} + C_{converter} + C_{FDL} +$$

The above formula is the core of a cost-based scheduling algorithm When a reservation request arrives, the combination of resources that offers the lowest total cost is chosen

Using the cost-based approach means that a burst might be converted to another wavelength or buffered in an FDL even if it is not strictly necessary, if this will result in better alignment of this burst with other bursts In other words, the additional cost of using a converter or FDL will be offset by the resulting reduction in channel cost An example of such a situation is presented in Fig 4 6 (a) There is one channel with two existing reservations and an incoming reservation request It is possible to allocate the new burst without buffering, as shown in Fig 4 6 (b) However, this results in creation of two voids, just slightly smaller than a typical burst length This makes their bandwidth extremely unlikely to be utilized by later bursts Alternatively, the new burst may be buffered in a FDL, as shown in Fig 4 6 c) Then, it will be aligned with an existing reservation and only one, larger gap will be created For this course of action to be chosen, the difference between the values of cost function for the original burst and delayed burst has to be greater than the cost of using a FDL

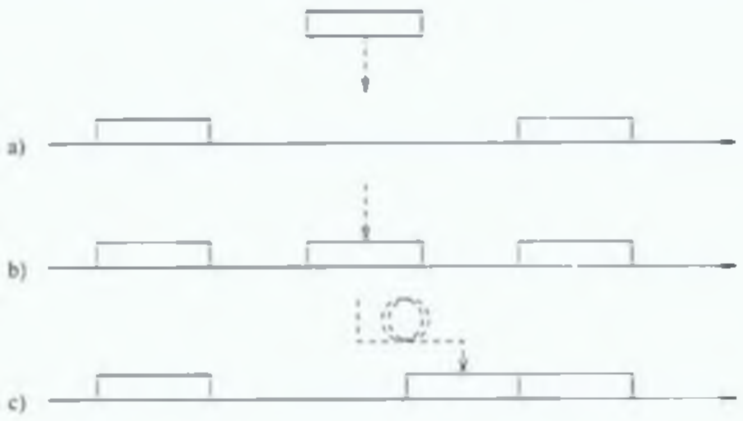


Figure 4.6: Cost-based scheduling

The cost of a channel is calculated using a cost function, presented earlier in this chapter. When costs of other resources are concerned, there are several possible strategies:

- the price may be constant,
- the price may be linearly dependent on the number of currently used resources,
- the price may be a square function of the number of currently used resources.

It is expected that varying the price of a resource, depending on the number of resources available, will make them widely available when they are plentiful, but at the same time will tend to conserve the resources when there are only a few left. Simulation results comparing those strategies will be presented in Chapter 6.

4.4.2 FDL management

It is typically assumed that a reservation attempt is first made without delaying a burst. Only if it fails, a FDL is used. This leads to a situation, where for most of the time only a few FDL channels will be used, and the probability of a large number

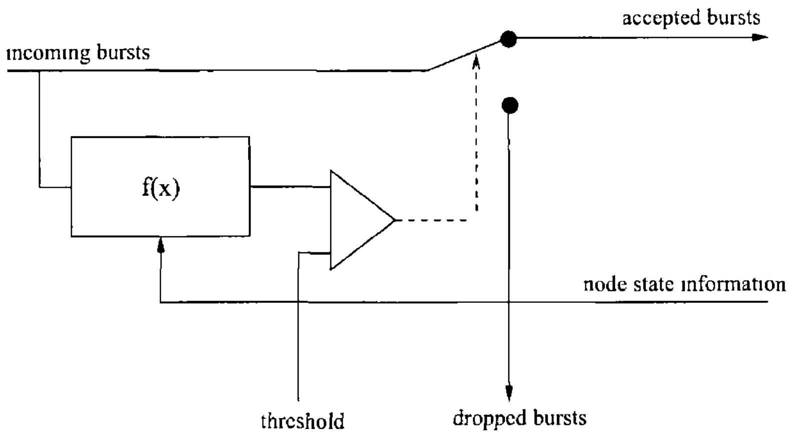


Figure 4 7 Structure of a generic burst dropper

of channels being used at the same time is very small It follows, that increasing the size of the FDL bank will offer gradually decreasing benefits in such a case

However, when a cost-based scheduling algorithm is used, the delay lines are used not only to resolve contention, but also to prevent it from happening in the first place By adjusting the price of an FDL, it is possible to ensure that most of the available channels are utilized The simulation results presented in Chapter 6 show how this approach can improve the burst loss ratio in nodes with various FDL bank sizes

4.4 3 Quality of Service - selective burst dropping

Several Quality of Service schemes proposed in literature use *intentional burst dropping,* also called *early burst drop* Basically, a certain proportion of low priority bursts is dropped before even entering a node in order to improve the loss ratio in higher classes

Fig 4 7 shows a structure of a generic burst dropper When a control packet arrives with a reservation request, the dropper calculates the value of a certain function *f()* and compares it with a pre-set threshold *th* If the value of *f()* exceeds *th* the burst will be dropped Otherwise an attempt to forward the burst to the next node will be made

Burst dropper can be

- Random, when the value of function *f()* is obtained from a random number generator [76–79],
- Deterministic, if for example *f(SEQ) = SEQ mod 10*, where *SEQ* - sequence number of incoming burst, or if all bursts belonging to a certain class are dropped [31,80],
- Cost-based, if *f()* is a cost function

The value of a cost function reflects the amount of resources within a node that are needed to forward the burst It is also an indicator of the probability that accepting this burst will cause contention in the future Therefore, dropping only those bursts with cost function values above a certain threshold can be expected to offer better performance than dropping the same percentage of bursts randomly

4.5 Summary

In this chapter the idea of a *cost function* was presented Cost function is an indicator of the probability of the future contention caused by allocating a certain burst Therefore, its value can be considered to be an imaginary cost of using the channel

Several applications of a cost function have been proposed

- Scheduling algorithms - the value of a cost function along with the cost of other resources (like FDLs) is considered to be the total cost of allocating a burst, and the lowest-priced combination is chosen,
- FDL management - when a cost-based scheduling algorithm is used, it is possible to effectively utilize FDL banks of arbitrary, especially large sizes,

- Quality of Service - if a QoS scheme uses intentional burst dropping, then a cost-based algorithm, where only those bursts with the cost function value above a certain threshold are dropped, offers better overall performance than a random one

Chapter 5

Simulator

Several analytical models of Optical Burst-Switched networks have been presented (e g [72, 88, 89]) However in many cases such analysis is difficult or impossible Therefore, it was necessary to develop a suitable simulator to test the ideas presented in this thesis This chapter describes the structure of this simulator, discusses the choices that had to be made and provides its short usage manual and a few example scripts

5.1 Existing OBS simulation programs

Optical Burst Switching is a rather new paradigm, and there are relatively few simulators available The non-commercial tools include

- Optical WDM Network Simulator (OWns) has been developed by B Wen *et al* at the University of Maryland DAWN lab [90] It is a WDM extension to the *ns*-2 simulator Its later versions allow to simulate OBS networks as well

- OBS-ns simulator, developed at WINE laboratory (Information and Communications University, Taejeon, Korea) is based on the OWns simulator

- Also, many stand-alone simulators are mentioned in the literature [91, 92], but are not made publicly available

5.2 Stand-alone simulator vs. extension to an existing simulator

When a simulation program with desired capabilities does not exist, it is necessary to develop a new simulator. The first choice that has to be made is whether to develop one from scratch or extend/modify an existing simulator. The answer is non-obvious and depends on particular circumstances and desired characteristics of the final program.

If a completely new simulator is written, it is likely to be well-suited for the intended purpose and fast. Unfortunately, it will also be rather inflexible, and any serious changes to it will require much effort. Also, this approach might be time-consuming, as the entire simulator core has to be developed from scratch.

Extending an existing simulator has many advantages. First of all, the simulator core and user interface are already in place. Their functionality is likely to have already been validated. Also, in case of OBS simulator, it is useful to have a TCP/IP-capable simulator in place. However, a significant amount of time may be needed to get to know the simulator's internals in order to extend it.

Keeping all of the above in mind, I decided to extend the well-known *ns-2* simulator. This extension will be described in Section 5.4.

5.3 Ns-2 simulator

Ns-2 is a discrete event network simulator [93]. It is mainly intended as a tool to investigate TCP/IP based wired and wireless networks. A wide range of TCP variants, routing protocols and traffic sources are supported.

Ns-2 uses a *two language* concept [94]. OTcl scripting language is used as a frontend to C++ based simulator core. This solution offers the best features of an interpreted language (ease of development and modification, interactivity) with those of a compiled language (speed, efficiency).

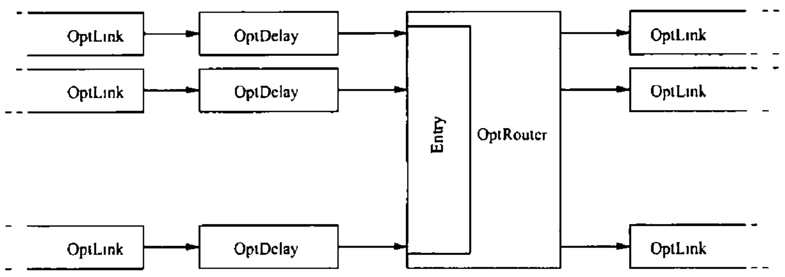


Figure 5 1 Structure of an *OBSCoreNode*

The simulator maintains two separate, similar but not identical object hierarchies - the interpreted one and the compiled one When the user creates an OTcl object, a compiled object can be created automatically Also, mechanisms exist to use common member variables in compiled and interpreted objects

As a principle, any functionality involving processing individual packets will be implemented in the compiled hierarchy If, on the other hand, speed is not critical, but the ease of modification is (configuration, one-time calculations), then using OTcl will be more convenient

5.4 OBS extension to ns-2 simulator

This section describes the objects that together form the OBS extension to *ns*-2 simulator In each case, the object's internal structure is presented, along with its configuration parameters and usage instructions

5 4 1 OBS object

OBS object represents the Optical Burst-Switched network "cloud" in ns-2 reality It contains edge nodes, core nodes and optical links

Usage *set obs [new OBS]*

5 4.2 OBSCoreNode

OptRouter is the main building block of an *OBSCoreNode* Similar to *ns-2* object *Classifier*, it keeps a list of available destinations When a control packet is received, it is examined and sent to one of the destinations from the list, based on its address

Before being passed to *OptRouter*, the control packet will first be delayed in an *OptDelay* object This delay represents the time needed to convert the control packet to the electrical domain, analyze it, and make a routing/schedulmg decision Fig 5 1 shows the internal structure of a OBSCoreNode object

Core nodes are not visible from ns-2 network An OBSCoreNode object is created within a given OBS network and automatically belongs to this network

Usage *set cn [$obs core-node]*

5.4.3 OBSEdgeNode

The *OBSEdgeNode* object contains three main building blocks, as shown in Fig 5 2

- *ns-2* object *Node*,
- *OBSCoreNode* object,
- A set of *BurstAssemblers* and/or *BurstGenerators*

OBS edge nodes are represented by Node/OBSEdgeNode object and are created by a *Simulator* object

Usage *set ns [new Simulator]*

set en [$ns obs-node]

Later, the edge node has to be assigned to an OBS network

$obs add-edge-node $en

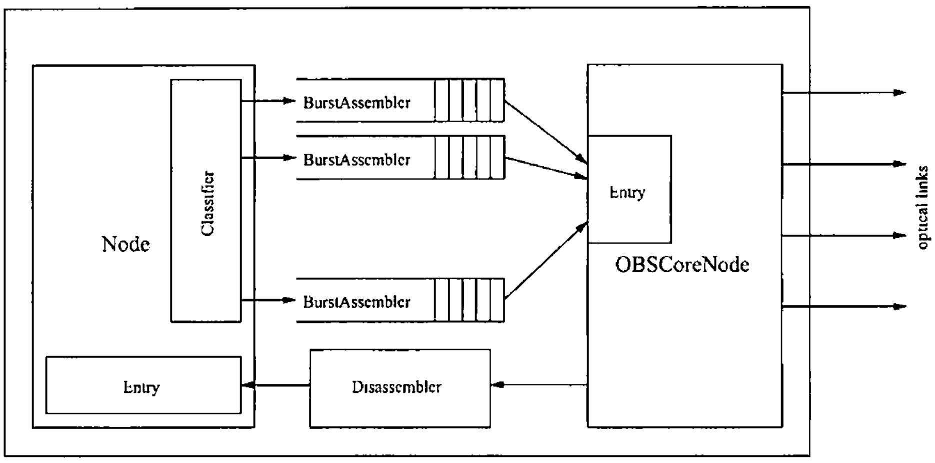


Figure 5 2 Structure of an *OBSEdgeNode*

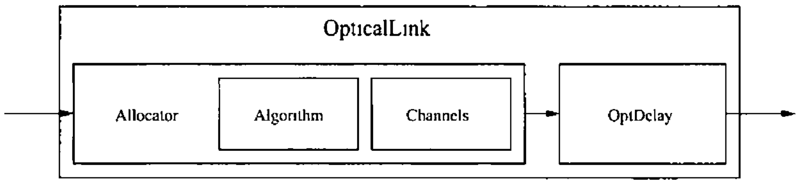


Figure 5 3 Structure of an *OpticalLink*

5.4.4 OpticalLink

The *OpticalLink* object, presented in Fig 5 3 is composed of an *Allocator* that contains a set of *OptChannels* and scheduling *Algorithm* and an *OptDelay* object, representing the propagation delay

Usage *set link [obs optical-link from_node to_node link_size delay algorithm]*

where *from_node* and *to_node* are edge or core nodes, *link_size* is the number of wavelengths in the link and *algorithm* is the scheduling algorithm

5 4 5 BurstAssembler

The burst assembly process at an edge node is performed by a *BurstAssembler* object The assembly algorithm uses a time trigger, size trigger (packet count and/or byte length) or both The offset time can be constant, or can be uniformly distributed over a predefined range

A *BurstAssembler* object uses the followıng varıables that can be modıfied by user

- *max_count* ıs the maxımum number of IP packets ın the queue
- *max_sıze* ıs the maxımum sıze of queue ın bytes If ıt (or the maxımum number of packets) ıs exceeded, a burst wıll be generated
- *max_tıme* ıs the maxımum assembly tıme
- *offset* ıs the offset tıme ıf a constant offset tıme ıs used or mınımum offset tıme ıf random offset tıme ıs used
- *offset_max* defınes the upper bound of the offset range when random offset tıme ıs used
- *offset_jıtter* ıs set to "0" ıf constant offset tıme ıs desıred or to "1" ıf the offset tıme ıs to be randomly dıstrıbuted over a predefıned range
- *ıd* ıs the ID number of all bursts generated by thıs assembler
- *BW* ıs the bandwıdth of the outgoıng lınk

After a burst assembler has been created, ıt needs to be assıgned to a node-node paır

$ba set-src-dst fromnode tonode, where *fromnode* ıs the source node and *tonode* ıs the destınatıon node

5 4 6 BurstLogger

The OBS extensıon does *not* use the *ns-2* loggıng functıonalıty Therefore, separate loggıng facılıtıes had to be provıded

All loggıng objects derıve from *BurstLogger* The *BurstLogger* ıtself ıs capable of countıng the receıved bursts and lost bursts On request, ıt wıll dısplay those number along with the loss ratio, or write them to a specified file It can also filter the incoming bursts according to their source, destination, and ID

Each *BurstLogger* implements the following commands

- *$bl set-src source_address* - only bursts with this source address will be taken into account,
- *$bl set-dst destination_address* - only bursts with this destination address will be taken into account,
- *$bl set-id burst_ID* - only bursts with this ID will be taken into account,

-1 is considered to be a wildcard and will match any source address, destination address and burst ID For example, if $src = 10$, $dst = -1$ and $ID = -1$, then all bursts with a source address of 10 will be logged, regardless of their ID and destination address

Other logging objects are

- *BurstLogger/Distribution* calculates the distribution of burst length as the function of interarrival time and vice versa
- *BurstLogger/LossDistribution* in addition to measuring the burst loss ratio, measures the *loss distribution*, e g the loss probability as a function of burst length
- *BurstLogger/Time* averages the traffic in a desired interval and writes the results to a file
- *BurstLogger/Traffic* calculates the total load within the simulation time
- *BurstLogger/Trace* logs every received burst to a file

A *BurstLogger* or any of its subclasses can be added to the following network components, each of which implements an *add-logger* command

- *OpticalLink* A *BurstLogger* will track traffic characteristics and loss ratio in this particular link
- *BurstAssembler* and its derivative objects A *BurstLogger* added to a *BurstAssembler* can be used to analyze traffic characteristics However, no loss ratio will be reported as no bursts can be lost within a burst assembler
- *OBSEdgeNode* A *BurstLogger* can be used to analyze the traffic received by an edge node

An arbitrary number of *BurstLoggers* can be added to a network component This is useful when different classes of bursts have to be tracked separately

5.5 Traffic sources

Using *ns-2* based IP traffic sources and burst assembly can be very realistic However, as each burst may contain hundreds of packets, this approach is also time consuming Therefore, in many cases it is beneficial to use a *burst generator* instead of a burst assembler As long as statistical properties of generated traffic match those of assembled traffic, the simulations will yield proper results in much shorter time

The OBS extension to *ns-2* provides several objects that can be used to generate burst traffic Each of those objects extends the *BurstAssembler* class shares much of its functionality In particular the *offset, offset_max* and *offset_jitter* variables need to be set Also, the *set-src-dst* procedure needs to be called

All burst generator objects also define procedures *start* and *stop*, used to start and stop the generation process, respectively

5 5.1 BurstAssembler/Exponential

BurstAssembler/Exponential generates traffic where both burst length and interarrival time are exponentially distributed This type of traffic is characteristic for telephony networks and therefore well-studied Although it does not perfectly approximate the properties of assembled IP traffic, many authors use it in their simulations [9,23,88] Also, it is useful for simulator validation as it can be relatively easily studied analytically

In addition to those inherited from *BurstAssembler* this generator defines the following variables

- *avlength* is the average length of a burst in seconds
- *load* is the desired load in Erlangs
- *const_length* is set to "0" if the burst length is to be exponentially distributed or to "1" if it is to be constant

5.5.2 BurstAssembler/Generator

By analyzing the functioning of a burst assembler it can be noted than the burst length and its assembly time are, in fact, dependent For example, if the assembly time is shorter than the maximum value, then it it obvious that it was the length trigger that caused the burst to be generated, and the burst will be of the maximum length Conversely, maximum burst size implies that the particular burst generation was triggered by exceeding the maximum assembly time

Those dependencies can be described by correlated or conditional distributions Conditional distribution is a function of two arguments, the first one being the value of the previously generated assembly time or burst length It can be viewed as a set of distributions - the previously generated value determines which *distribution is to be used next* [95]

The conditional distributions to use with this generator can be obtained by using IP traffic assembled using a desired algorithm and *BurstLogger/Distribution*

The following methods are available

- *load-n-1-distributions file_name*

- *load-l-a-distributions file_name*

Also, the member variable *correlated* can be set to "0" if burst lengths and assembly times are to be generated independent of each other, or to "1" if conditional distributions are to be used.

5.5.3 BurstAssembler/EnvelopeGenerator

In some simulations, it is desirable to use traffic that exhibits self-similarity and long-range dependence. Using an appropriate IP traffic source and assembling its traffic is not always suitable, mainly for performance reasons. The burst generation itself would consume significant part of simulator's time.

Envelope generator, is a new type of burst generator that can efficiently generate long-range dependent traffic. It exploits the fact that the process of assembling IP packets into bursts is similar to calculating bandwidth within the assembly window. Therefore, if the bandwidth at any time can be known, the process of generating and assembling IP packets can be skipped altogether.

In this traffic source, a set of Pareto-distributed on/off sources is used. No IP packets are generated, however. Instead, the state of each source is used to calculate bandwidth envelope. Immediately after a burst is generated, the envelope is sampled, and its value is used to decide when the next burst will be generated, and what will its length be. In effect, much less simulator events are necessary to generate a burst.

The generation process is presented in Fig. 5.4. There are three on/off sources. The rectangles on the corresponding axes represent IP packets that would normally be generated. In an envelope generator, however, the sources only report their state when pooled. This information is then used to calculate bandwidth envelope, shown below. Its value determines the size of the next burst and the simulated assembly time. The same principle can be used to generate a different type of traffic - for example, exponentially distributed on/off sources can be used.

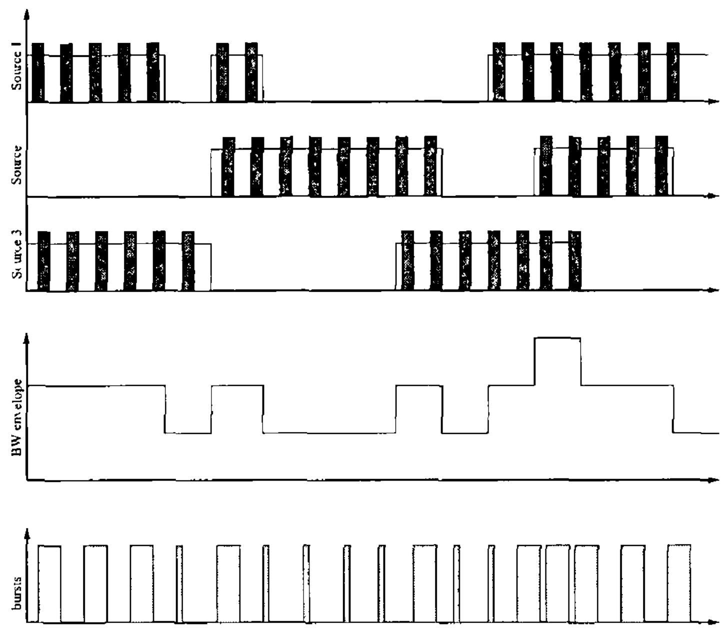


Figure 5 4 An envelope generator

The *BurstAssembler/Envelope* generator uses the above described method The following options are available

- *sources* is the number of on/off sources that is to be used,
- *rate* is the average bitrate,
- *avg_on* and *avg_off* are the average on and off times of each source,
- *shape* is the shape parameter of Pareto distribution

The distribution of the on and off times of each source can be either Pareto or exponential

$bg set-type pareto

$bg set-type exponential

5.5 4 *Generators* class

In some simulation scenarios the number of burst generators can be very large In particular this may be the case when automatically generated network topologies are used This makes starting and stopping each burst generator individually difficult and impractical *Generators* is a helper object used to simplify management of many burst generators

The class *Generators* defines the following methods

- *add-generator* adds a generator to the object,
- *start* starts all generators,
- *stop* stops all generators

5.6 Scheduling algorithms

When creating an *OpticalLink*, the user has to specify the scheduling algorithm that is to be used The OBS simulator offers the following scheduling algorithms

- Algorithm/FirstFit,
- Algorithm/RoundRobin,
- Algorithm/MostUsed,
- Algorithm/LeastUsed,
- Algorithm/Random,
- Algorithm/LAUC-VF,
- Algorithm/Min-EV,
- Algorithm/GapFilling

All of the above algorithms except *GapFilling* are described in detail in Chapter 3 They also do not require any additional configuration

GapFilling is a cost-based algorithm, using as a metric either one of the cost functions proposed in Chapter 4 It defines the following variables which need to be set before creating the link

- *min_offset* is the minimum offset that is expected at the node,
- *max_offset* is the maximum offset expected at the node,
- *fdl_price* is the base price of an FDL, if FDLs are used

5.7 Burst dropping

A burst dropping technique, described in Section 3 2 3 and Section 4 4 2 can be implemented by using a *BurstDropper* object There are two types of a burst dropper available

- *BurstDropper/Random* discards bursts in a random fashion,
- *BurstDropper/Cost* drops a burst when its cost function value exceeds a certain threshold

A *BurstDropper* can be configured to only drop certain bursts This is done using the same syntax as in case of a *BurstLogger*

- *set-src*
- *set-dst*
- *set-id*

The default value of source, destination and ID are -1, which means that all bursts can be dropped

The proportion of dropped bursts is determined by setting the variable *threshold* In case of a random dropper it can be set to any value between 0 (no burst is dropped) to 1 (all bursts are dropped) When a cost-based dropper is used, the burst will be dropped if the value of a cost function exceeds the threshold value

load	measured	calculated
12	0 000000	7 9817e-07
14	0 000012	1 499e-05
16	0 000153	1 4555e-04
18	0 000916	8 5441e-04
20	0 003424	3 3803e-03
22	0 009770	9 7788e-03
24	0 022191	2 2095e-02
26	0 041190	4 1219e-02
28	0 066684	6 6498e-02
30	0 096446	9 6266e-02

Table 5 1 Measured and calculated burst loss ratios

5.8 Simulator validation

Optical Burst-Switched network do not yield easily to theoretical analysis Therefore, validating the simulator output is only possible in very simple cases

Let us consider a simple OBS system, composed of two edge nodes and an unidirectional 32-channel link between them There is an exponential burst generator attached to one of the nodes The offered load varies between 12 and 30 Erlangs The offset time for all bursts is constant

The loss ratio in such a system can be calculated analytically by using the Erlang-B formula

$$P_B = \frac{\frac{E^N}{N!}}{\sum_{i=0}^{N} \frac{E^i}{i!}}$$

Table 5 1 lists the measured and calculated loss ratios The same results are presented in a graphical form in Fig 5 5 Apparently, the simulation results are very close to the calculated values, and any deviations are likely to be caused by relatively short simulation times

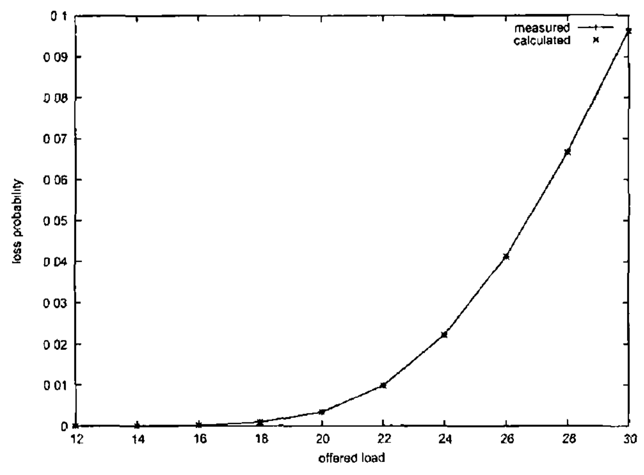


Figure 5 5 Measured and calculated burst loss ratios

5.9 Routing

The problem of routing in the OBS extension can be divided into two parts Firstly, whatever routing scheme is supported by the IP part of the network has to be supported Then, routing functionality has to be provided withing the OBS network

5.9 1 IP routing

In *ns-2* simulator, routing tables are calculated based on connection tables This capability needs to be preserved, so that IP nodes can communicate using the OBS network In order to achieve that, a *PseudoLink* is created between each pair of edge nodes within a given OBS network A *PseudoLink* is seen by *ns-2* as an ordinary link, and thus will be used when populating the routing tables, but does not provide any connection capabilities The *entry* point of a *PseudoLink* relays all IP packets to an appropriate *BurstAssembler*

Fig 5 6 shows how *PseudoLinks* "connect" OBS edge nodes Note that this is independent on the actual topology of the OBS network

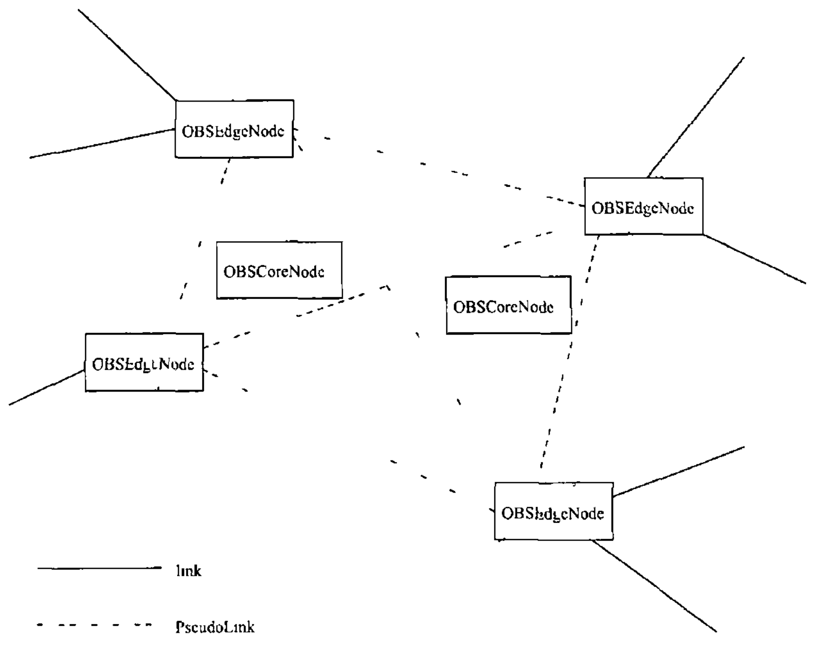


Figure 5 6 *PseudoLinks* withing an OBS network

5.9.2 OBS routing

The routing functionality in an OBS network resides in *OptRouter* objects The *OBS* object is aware of all connections between nodes and uses an implementation of the Dijkstra algorithm, to populate their routing tables The *compute-routes* and *populate-routers* procedures needs to be invoked manually after the network topology is defined

5.10 Example simulations

The following subsections discuss two relatively simple simulation scenarios, designed to show two possible approaches to OBS simulations More simulation scripts will be presented in Appendix A

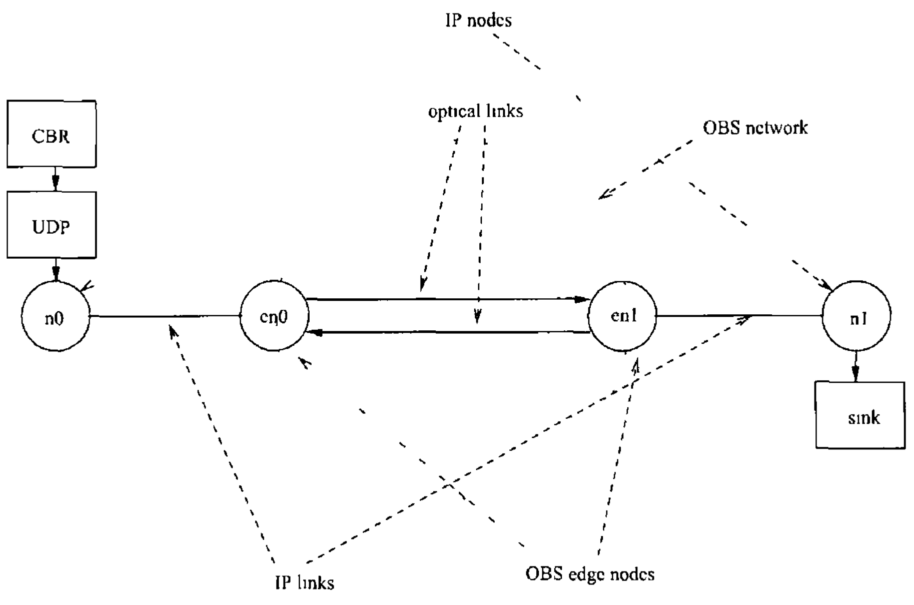


Figure 5 7 Example network topology

5 10.1 Network with burst assemblers

Fig 5 7 shows a topology of a simple network Its IP part consists of two nodes, *n0* and *n1* They are connected to two OBS edge nodes, that in turn are connected to each other with two unidirectional optical links A constant bitrate source using an UDP protocol is attached to the node *n0* and a sink to the node *n1* A Tcl script describing this network is presented below

First of all, create the Simulator object, just like in every ns-2 simulation

```
set ns [new Simulator]
```

Then, create an OBS network

```
set ob [new OBS]
```

Create IP nodes

```
set n0 [$ns node]
set n1 [$ns node]
```

then create OBS edge nodes For now, they do not belong to any OBS network

```
set en0 [$ns obs-node]
set en1 [$ns obs-node]
```

The edge nodes need to be added to an OBS network

```
$ob add-edge-node $en0
$ob add-edge-node $en1
```

The IP nodes and OBS edge nodes are connected with ordinary links

```
$ns duplex-link $n0 $en0 2Gb 1ms DropTail
$ns duplex-link $n1 $en1 2Gb 1ms DropTail
```

Now two optical links with two wavelengths, one millisecond delay and "Random" scheduling algorithm are created It is not strictly necessary to use variables (*ol0* and *ol1*) to point to them, but it is useful if the links are to be manipulated later

```
set ol0 [$ob optical-link $en0 $en1 2 1ms "Algorithm/Random
   "]
set ol1 [$ob optical-link $en1 $en0 2 1ms "Algorithm/Random
   "]
```

Only one burst assembler is needed at each node, as there is only one possible destination

```
set ba0 [new BurstAssembler]
$ba0 set max_size 1MB
$ba0 set max_time 1ms
$ba0 set offset 5ms
$ba0 set BW 0 1Gb
$ba0 set-src-dst $en0 $en1
```

```
set ba1 [new BurstAssembler]
$ba1 set max_size 1MB
$ba1 set max_time 1ms
$ba1 set offset 5ms
$ba1 set BW 0 1b
$ba1 set-src-dst $en1 $en0
```

The *ns*-2 logging functionality is not used within the OBS network Therefore, *BurstLogger* objects have to be used to extract information from the network

```
set bl0 [new BurstLogger]
set bl1 [new BurstLogger]
```

The loggers are added to the optical links, so that the burst loss probabilities can be collected

```
$ol0 add-logger $bl0
$ol1 add-logger $bl1
```

The following commands compute the next hop information for each node within the OBS network and populate *OptRouters* using this information

```
$ob compute-routes
$ob populate-routers
```

An IP traffic source is created and configured

```
set udp [new Agent/UDP]
$ns attach-agent $n0 $udp
set null [new Agent/Null]
$ns attach-agent $n1 $null
$ns connect $udp $null
```

```
set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
$cbr set type_ CBR
$cbr set packet_size_ 1000
$cbr set rate_ 0 2Gb
$cbr set random_ false
```

Finally, the events are scheduled and the simulation is run

```
$ns at 0 "$cbr start"
$ns at 50 "$cbr stop"
$ns at 200 "$bl0 print-stats"
$ns at 200 "$bl1 print-stats"
$ns at 200 "exit 0"
$ns run
```

The simulation produces the following output

```
 BurstLogger total bursts  62501, lost  15151
 BurstLogger  bursts lost  0 242412
 BurstLogger total bursts  0, lost  0
```

Each logger is attached to an optical link and reports the total number of received bursts and the number of bursts that could not be transmitted due to a contention The next line contains the calculated burst loss probability Note that no data is transmitted from edge node *en1* to *en0*, and no loss probability is reported for this link

The obtained loss probability of over 24% might seem to be rather high, as the source bandwidth is exactly equal to the bandwidth of two available optical channels This is because the used source is deterministic and bursts may be lost due to synchronization effects If the source bitrate is reduced by a very small amount

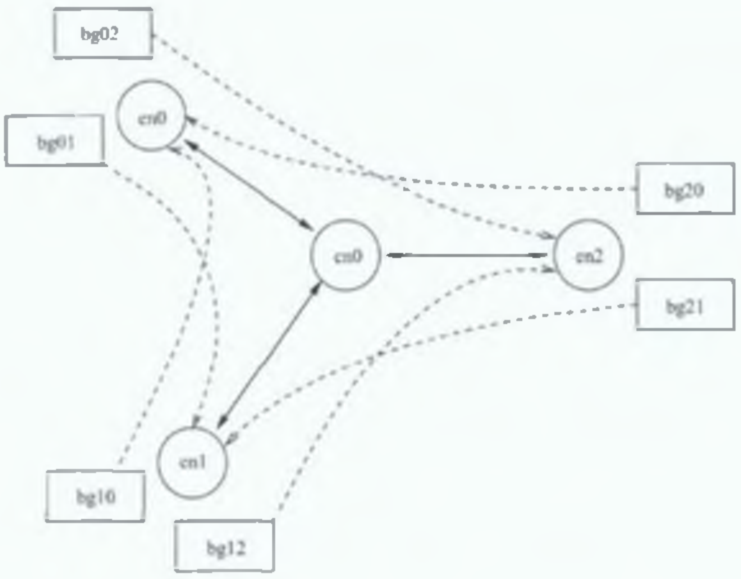


Figure 5.8: Network with burst assemblers

```
$cbr set rate_ 0.199Gb
```

then all bursts will be transmitted with no losses:

```
BurstLogger:total bursts: 62188, lost: 0
BurstLogger:total bursts: 0, lost: 0
```

5.10.2 Network with burst generators

Simulation times can be greatly reduced when burst generators are used instead of burst assemblers. Fig. 5.8 shows a network topology example. There are three edge nodes and one core node. Note the absence of any IP nodes, the entire network is OBS-based. In each edge node there are two burst generators, one for each of the two remaining edge nodes.

The following script can be used to simulate such a network.

Create *Simulator* and *OBS* objects, and define the load offered by each burst generator.

```
set ns [new Simulator]
set ob [new OBS]
set load 50
```

Create edge nodes and add them to the OBS network

```
set en0 [$ns obs-node]
set en1 [$ns obs-node]
set en2 [$ns obs-node]
$ob add-edge-node $en0
$ob add-edge-node $en1
$ob add-edge-node $en2
```

Create the core node Unlike edge nodes it is created by the OBS network itself and belongs to it automatically

```
set cn0 [$ob core-node]
```

Create optical links

```
set olen0cn0 [$ob optical-link $en0 $cn0 100 1ms "Algorithm
    /Random"]
set olcn0en0 [$ob optical-link $cn0 $en0 100 1ms "Algorithm
    /Random"]
set olen1cn0 [$ob optical-link $en1 $cn0 100 1ms "Algorithm
    /Random"]
set olcn0en1 [$ob optical-link $cn0 $en1 100 1ms "Algorithm
    /Random"]
set olen2cn0 [$ob optical-link $en2 $cn0 100 1ms "Algorithm
    /Random"]
set olcn0en2 [$ob optical-link $cn0 $en2 100 1ms "Algorithm
    /Random"]
```

Create burst generators *BurstAssembler/Exponential* generates traffic where both burst length and assembly time are exponentially distributed

```
#from node en0 to node en1
set bg01 [new BurstAssembler/Exponential]
$bg01 set offset 5ms
$bg01 set avlength 1ms
$bg01 set load $load
$bg01 set-src-dst $en0 $en1
#from node en0 to node en2
set bg02 [new BurstAssembler/Exponential]
$bg02 set offset 5ms
$bg02 set avlength 1ms
$bg02 set load $load
$bg02 set-src-dst $en0 $en2
#from node en1 to node en0
set bg10 [new BurstAssembler/Exponential]
$bg10 set offset 5ms
$bg10 set avlength 1ms
$bg10 set load $load
$bg10 set-src-dst $en1 $en0
#from node en1 to node en2
set bg12 [new BurstAssembler/Exponential]
$bg12 set offset 5ms
$bg12 set avlength 1ms
$bg12 set load $load
$bg12 set-src-dst $en1 $en2
#from node en2 to node en0
set bg20 [new BurstAssembler/Exponential]
$bg20 set offset 5ms
```

```
$bg20 set avlength 1ms
$bg20 set load $load
$bg20 set-src-dst $en2 $en0
#from node en2 to node en1
set bg21 [new BurstAssembler/Exponential]
$bg21 set offset 5ms
$bg21 set avlength 1ms
$bg21 set load $load
$bg21 set-src-dst $en2 $en1
```

There are six burst generators, and starting/stopping each of them separately would be rather inconvenient Therefore, a *Generators* helper object is used

```
set generators [new Generators]
$generators add-generator $bg01
$generators add-generator $bg01
$generators add-generator $bg10
$generators add-generator $bg12
$generators add-generator $bg20
$generators add-generator $bg21
```

In this simulation, only the total traffic loss will be investigated In order to do so, only two burst loggers are necessary one attached to all burst generators and one attached to all nodes A *BurstLogger/Traffic* object will, among others, report the total traffic transmitted by a generator or received by a node

```
set bl_out [new BurstLogger/Traffic]
set bl_in [new BurstLogger/Traffic]
#add logger to generators
$bg01 add-logger $bl_out
$bg02 add-logger $bl_out
$bg10 add-logger $bl_out
```

```
$bg12 add-logger $bl_out
$bg20 add-logger $bl_out
$bg21 add-logger $bl_out
#add loggers to nodes
$en0 add-logger $bl_in
$en1 add-logger $bl_in
$en2 add-logger $bl_in
```

Finally, the routes are computed, router tables populated and events are scheduled

```
#compute routes within the OBS network
$ob compute-routes
$ob populate-routers
#schedule events
$ns at 0 "$generators start"
$ns at 5 "$generators stop"
$ns at 2000 "$bl_out display-traffic"
$ns at 2000 "$bl_in display-traffic"
$ns at 2000 "exit 0"
#run the simulation
$ns run
```

This script produces the following output

```
total bursts    1249643
 time   5 000802
 traffic   1250 875260
 load   250 134930
av  burst length    0 001001
total bursts    1136544
 time   5 006243
 traffıc   1137 429390
 load   227 202172
av  burst length   0 001001
```

Thıs allows us to calculate the total loss ın terms of both burst loss probabılıty (9 05%) and traffıc loss (9 3%) The dıfference between those probabılıtıes ıs a result of a *burst selectıng effect*, descrıbed ın Chapter 3 2

5.11 Summary

In thıs chapter, the OBS extensıon to the *ns*-2 sımulator has been ıntroduced The buıldıng blocks of the OBS sımulator have been descrıbed ın detaıl, along wıth theır usage and confıguratıon optıons Then, several helper objects were dıscussed Fınally, a sımple sımulatıon scenarıo was used to valıdate the sımulator output, and two example sımulatıon scrıpts were presented and descrıbed ın detaıl

Chapter 6

Results and discussion of cost-based scheduling

The performance of the cost-based scheduling algorithm described in Chapter 4 was tested in a series of simulations Several different network topologies and traffic types were considered In the following section the simulation procedure is first introduced, then the performance of the cost-based algorithm and its dependence on several parameters is tested in a simple, single-node scenario Then, the results for three more complex network topologies are presented

6.1 Introduction

The cost-based scheduling algorithm used in all simulations in this chapter considers two types of resources outgoing channels and FDL channels In other words, when a reservation is to be made, a lowest-priced outgoing channel is found first Then, a lowest priced channel is found again, for a burst delayed in a FDL, and the cost of using a FDL is added The cost of using a FDL channel is either constant (*constant* strategy), or it is a function of the number of occupied FDL channels (*linear* strategy) Finally, the lower of those two values is chosen

Obviously, the initial price of using a FDL channel is an important parameter and its choice will affect the network performance Fig 6 1 shows how the burst loss probability is expected to change when the FDL price is adjusted At first (point "A"), when the price is low, the algorithm will tend to use most of the FDL channels to improve the alignment of bursts within a channel However, when the contention actually occurs, there is a high probability that no FDL channel will be available to resolve it At the other end of the scale (point "B"), when the price is high, bursts are delayed only in case of a contention This means that only a few FDL channels are used, as contention is a relatively rare occurrence Between those extremes (point "C"), there is an optimal FDL price, for which the loss ratio is the lowest It balances the need to use FDLs to improve the burst alignment and to resolve a contention when it occurs

A *FDL usage histogram* is a useful tool to understand the functioning of a cost-based algorithm It can be obtained by periodically sampling the state of a FDL bank, and recording the number of channels used When the simulation is finished, the results are presented in a graphical form Fig 6 2 presents three examples of such histograms Curve "1" means that for most of the time only a few FDL channels are used, and the most probable occurrence is that no channels are used at all Curve "2" represents a case, where for most of the time most of the channels are used, but at the same time the probability of *all* channels being used is relatively high Finally, the curve "3" means that while a significant proportion of channels is usually used, there almost always are available channels to resolve a contention The relationship between the curves presented in Fig 6 1 and 6 2 is as follows "A"-"2", "B"-"1" and "C"-"3"

The simulation procedure in all cases presented in this chapter was similar First, the burst loss ratio was determined for a range of possible FDL prices and loads The results were then presented in a graphical form, similar to Fig 6 1 Also, the FDL usage histograms were usually obtained to verify that the algorithm was functioning as expected

In all cases the cost function formula described in Chapter 4, Equation 4 6 was used The value of a cost function was considered to be infinity if a burst could not be allocated m a channel

The optimal FDL price, obtained in those simulations, was then used to compare the performance of the cost-based algorithm with that of LAUC-VF for a

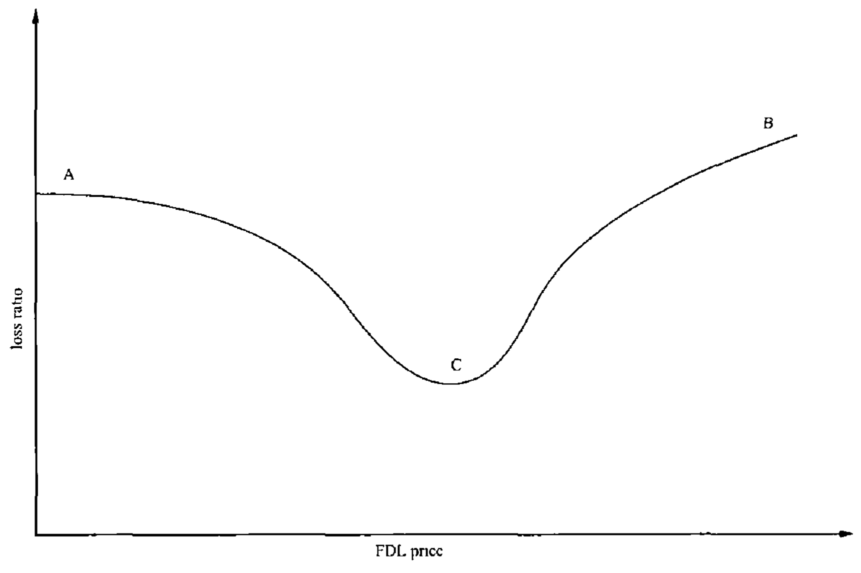


Figure 6 1 Example loss vs FDL price graph

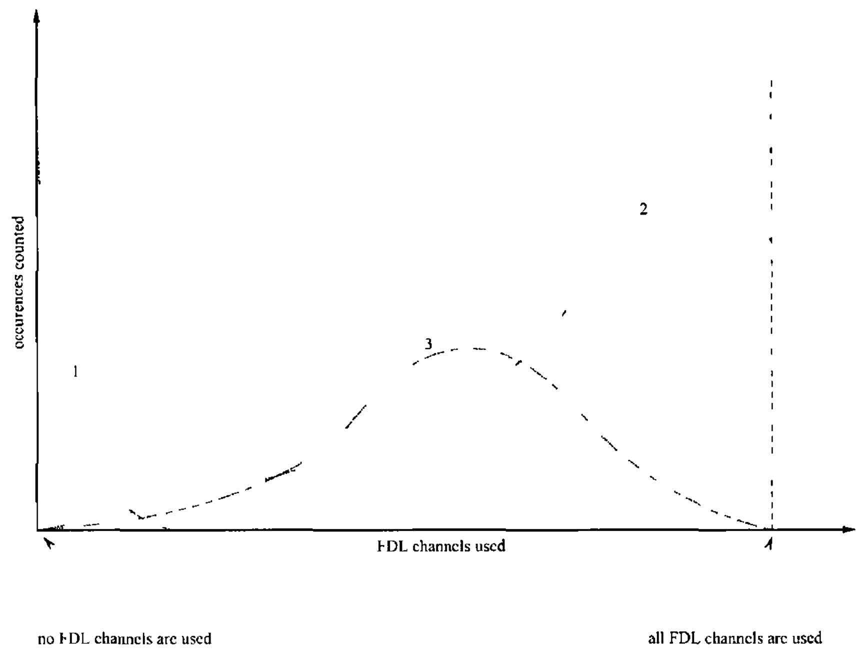


Figure 6 2 Example FDL usage histograms range of loads The reasons for choosing LAUC-VF are described in Section 3 1 2 In all cases the simulation time was increased, until the differences in results obtained in subsequent simulation runs became insignificant

6.2 Single node scenario

Let us consider the simple network presented in Fig 6 3 There are eight edge nodes four called *source nodes* and four *destination nodes* In each source node, there are four burst generators, sending equal amounts of traffic to each of the destination nodes The source nodes are connected to the core node with unidirectional *incoming links* and the core node is connected to the destination nodes with unidirectional *outgoing links* There is a shared FDL bank in the core node

A *Random* scheduling algorithm is used in all incoming links, and the cost-based one in the outgoing links Therefore, only the core node utilizes the cost-based algorithm

Except where stated otherwise, the simulation parameters are as follows

- The burst length is constant and the assembly time is exponentially distributed
- There are 32 channels in all links and in the FDL
- The FDL length is equal to one-third of burst length
- The offset time is uniformly distributed over a range of 10 burst lengths
- The load is measured at the source nodes, and the loss ratio is averaged over all four outgoing links

6.2 1 Basic simulation

A set of simulations was performed, using the network topology and procedure described above The relationship between the burst loss and FDL price is pre-

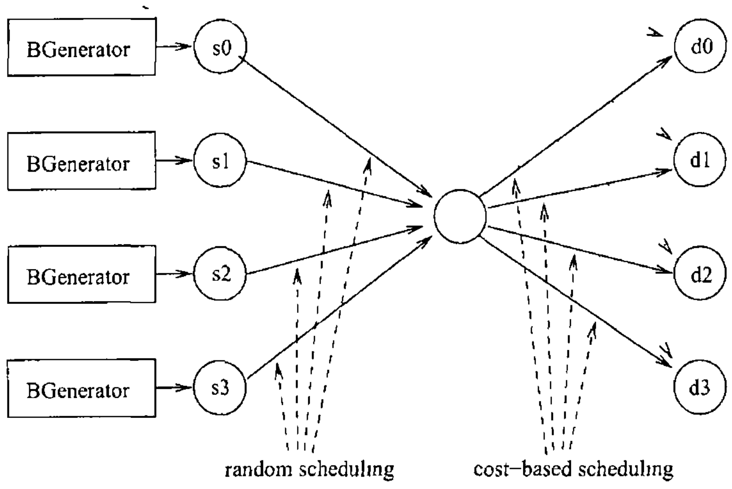


Fıgure 6 3 Sıngle-node network topology

sented ın Fıg 6 4-6 6

It ıs apparent that the loss ratıo ıs relatıvely hıgh when the cost of usıng a FDL ıs zero, then ıt drops markedly and fınally rıses agaın Thıs effect can be explaıned as follows when the FDL prıce ıs very low, a burst wıll be delayed every tıme when ıt wıll result ın a better alıgnment wıth other bursts However, thıs wıll also result m most of the FDL channels beıng occupıed for most of the tıme Thıs ıs not an optımal solutıon, because when a contentıon occurs there wıll be no way to resolve ıt

At the other end of the range, when the FDL prıce ıs very hıgh, a burst wıll only be delayed when ıt ıs ımpossıble to allocate ıt otherwıse Unfortunately, as no bursts wıll be delayed ın order to ımprove theır alıgnment wıth other bursts, the probabılıty of contentıon wıll be quıte hıgh Also, thıs approach results ın a majorıty of FDL channels beıng unused for most of the tıme

Consequently, between those extremes, there ıs a certaın FDL cost for whıch the traffıc loss ıs the lowest An examınatıon of the results presented m Fıg 6 4-6 6 leads to the followıng conclusıons

- Both strategıes offer sımılar mınımum loss ratıo,

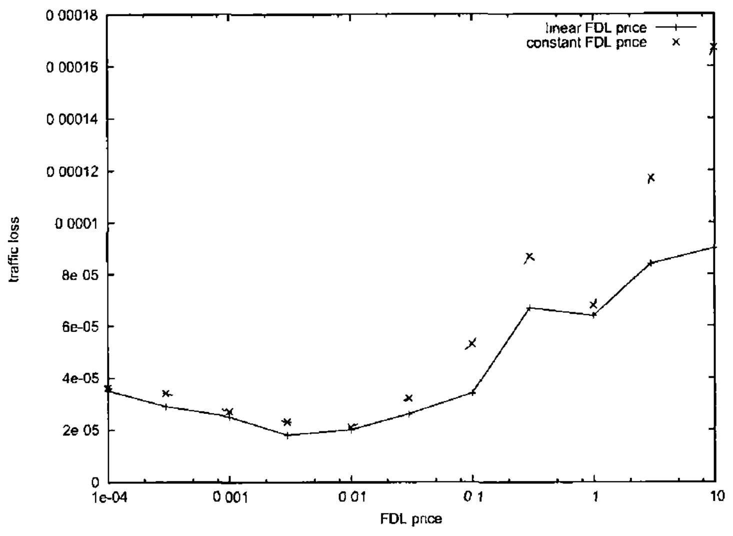


Figure 6 4 Traffic loss vs FDL price, 0 5 link load

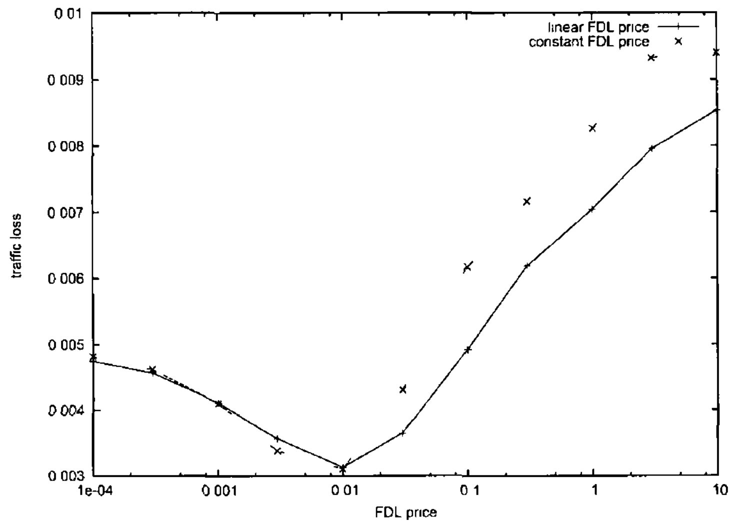


Figure 6 5 Traffic loss vs FDL price, 0 75 link load

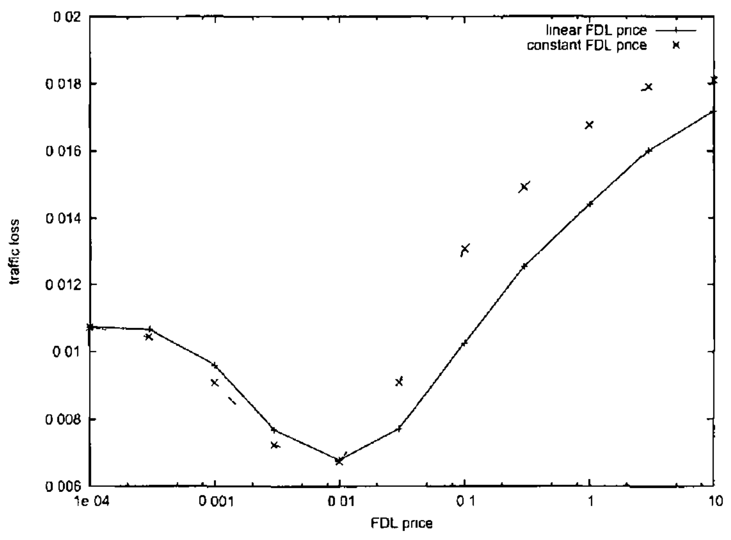


Figure 6 6 Traffic loss vs FDL price, 0 93 link load

- The optimal FDL price is similar for both strategies and equal to 0 01
- The "linear" strategy seems to be more tolerant of initial price deviations

Therefore, while the impact of the FDL price on the loss ratio will be studied for both strategies in the rest of this chapter, the *linear* strategy will be used to compare the performance of the cost-based algorithm with LAUC-VF

It can be noted that the relative improvement that can be achieved by using the optimal FDL price is the biggest for low loads For example, when the link load is 0 5, it approaches an order of magnitude For higher loads the difference is smaller, but still significant

The FDL usage histograms presented in Fig 6 7-6 9 reflect the probability of a certain number of FDL channels being used at any time Those results confirm the previous assertion regarding the behaviour of a cost-based algorithm for different FDL prices When the price is too low, then a large number of FDL channels is used for most of the time and consequently the probability of a FDL being unavailable to resolve a contention is relatively large On the other hand, when the price is excessively high, then a FDL is only used when a contention occurs, and most of the FDL channels are effectively wasted

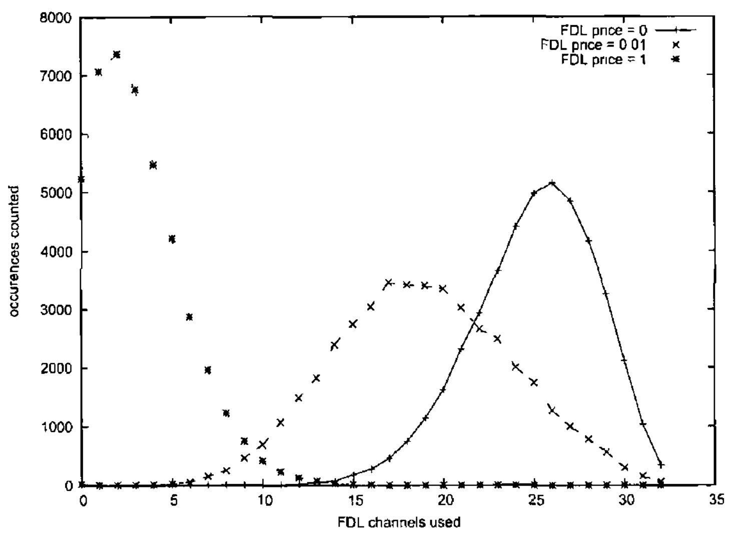


Figure 6 7 FDL usage histograms, linear FDL pricing, 0 5 link load

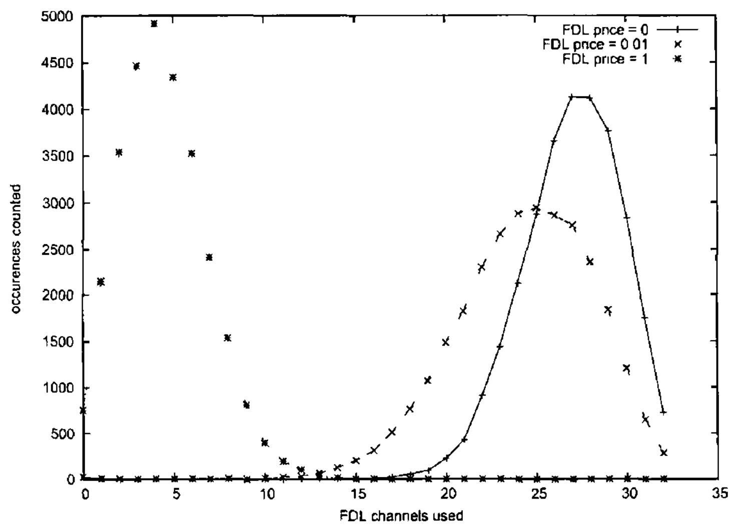


Figure 6 8 FDL usage histograms, linear FDL pricing, 0 75 link load

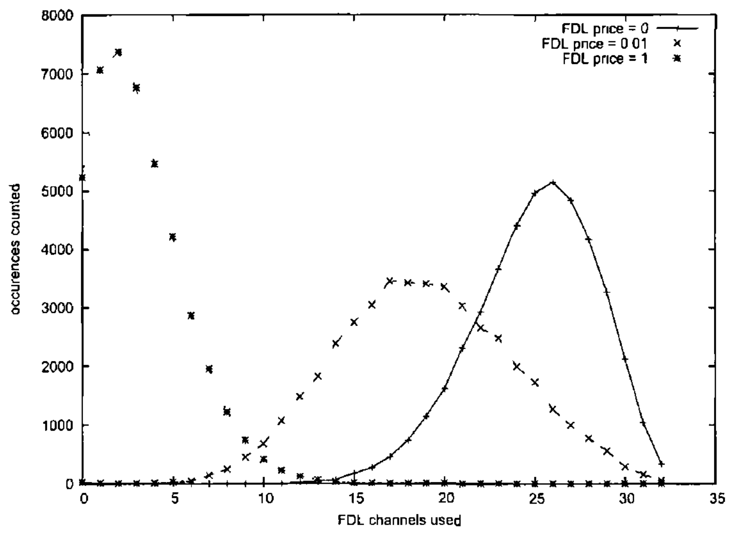


Figure 6 9 FDL usage histograms, linear FDL pricing, 0 93 link load

To fully assess the performance of the cost-based algorithm, it is necessary to compare it to that of other algorithms Fig 6 10 shows the burst loss ratio as a function of link load for the cost-based algorithm and LAUC-VF It is apparent that the cost-based algorithm outperforms LAUC-VF significantly for the entire range of loads For example, when the link load is 0 75, then the loss ratio of the cost-based algorithm is over 3 5 times lower than that of LAUC-VF For lower loads, the difference increases

The histograms presented in Fig 6 11 offer an explanation for those results LAUC-VF only uses an FDL when a contention occurs This means that it tends to only use a few FDL channels and, in the simulated network, a significant proportion of FDL channels remains unused The cost-based algorithm uses a FDL for two purposes to resolve a contention when it occurs, but also to prevent a contention *before* it occurs Therefore, it can effectively utilize FDL channels that would otherwise remain unused

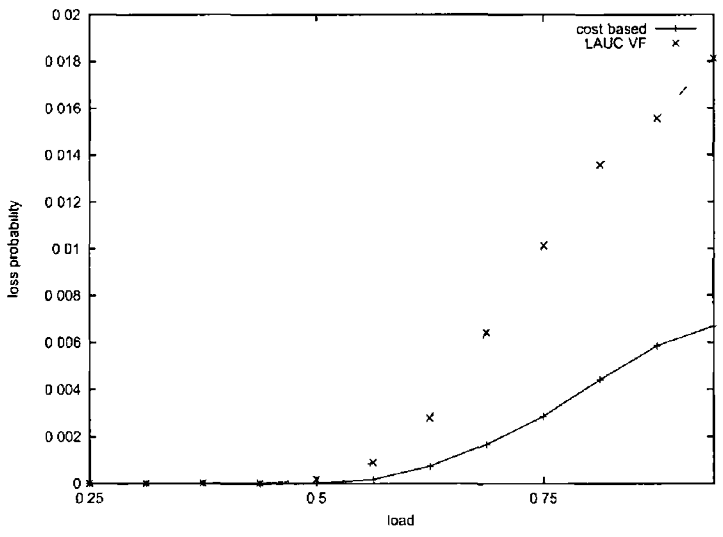


Figure 6 10 Burst loss probability for the cost-based algorithm and LAUC-VF as a function of link load

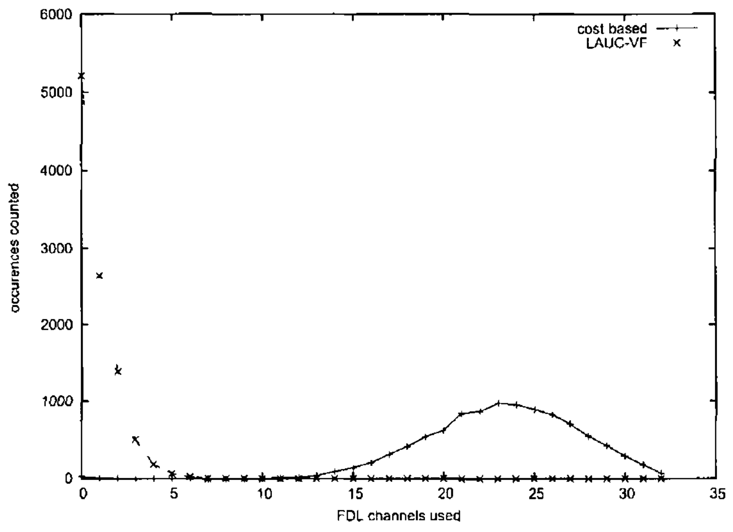


Figure 6 11 FDL usage histograms for the cost-based algorithm and LAUC-VF

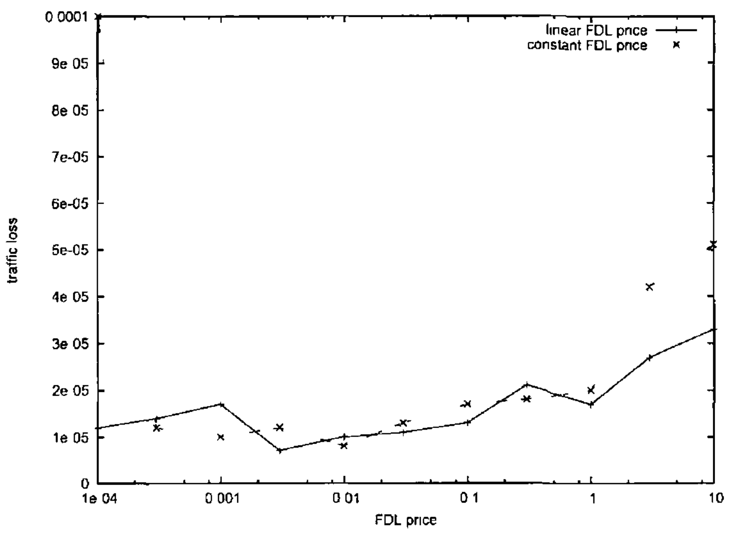


Figure 6 12 Traffic loss vs FDL price, 0 5 link load, variable burst length

6.2 2 Effects of traffic type

A traffic with constant burst length and exponentially distributed assembly time is well-understood and easy to generate Therefore, it is often used in simulations However, its statistical properties differ markedly from those of an assembled IP traffic

In this section, the performance of the cost-based scheduling algorithm is tested for two other types of traffic In the first case, both the burst length and assembly time are exponentially distributed Then, traffic sources utilizing realistic assembled traffic distributions are used

Variable burst length

The simulation scenario described in Section 6 2 was used All simulation parameters remained unchanged, with the exception of the burst length, which was variable and exponentially distributed

The simulation procedure was the same first, the relationship between burst loss and FDL price was investigated, as shown in Fig 6 12-6 14, and the optimal value of the price was determined

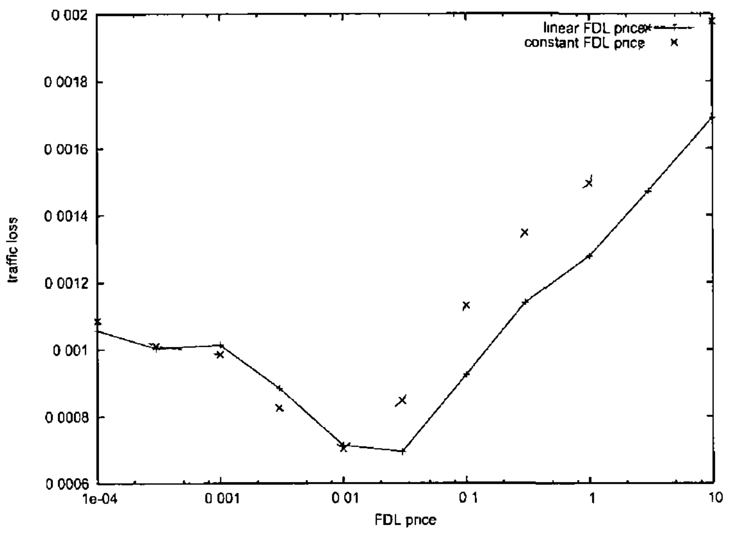


Figure 6 13 Traffic loss vs FDL price, 0 75 link load, variable burst length

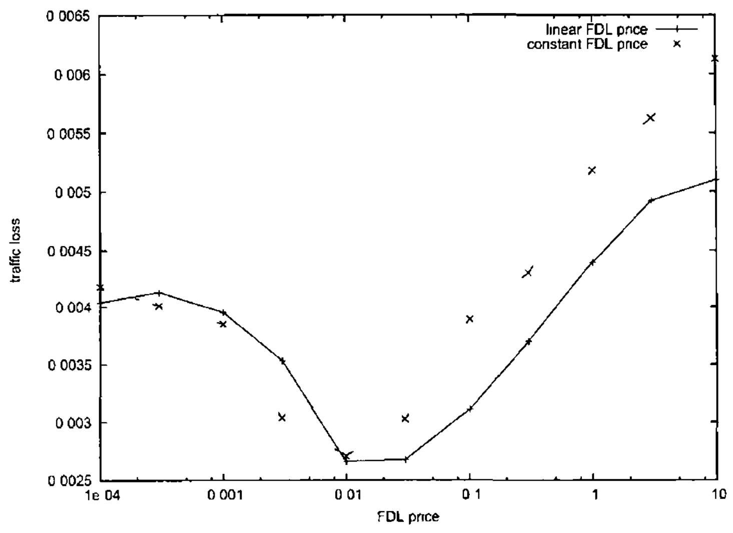


Figure 6 14 Traffic loss vs FDL price, 0 93 link load, variable burst length

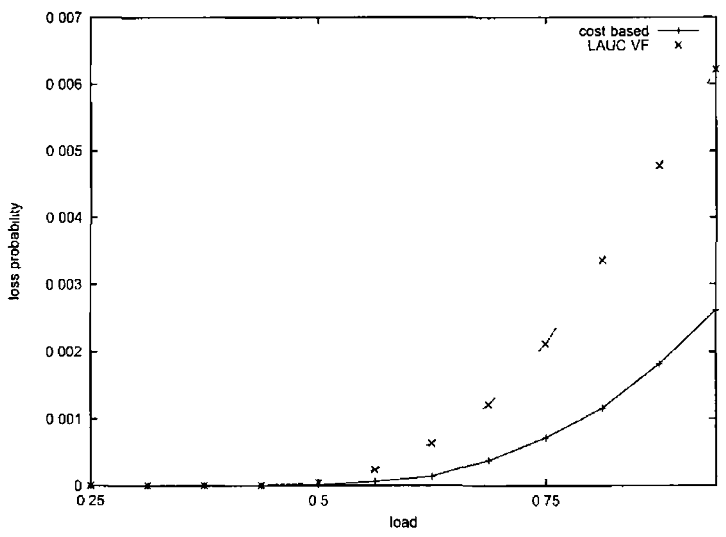


Figure 6 15 Burst loss probability as a function of link load, variable burst length

Then, the obtained value was used to compare the performance of the cost-based algorithm and LAUC-VF Fig 6 15 presents the burst loss ratio offered by both algorithms as a function of offered load While the results cannot be directly compared to those presented in Section 6 2 1 due to the *burst selecting effect*,[1] it is apparent that the cost-based algorithm outperforms LAUC-VF The difference is approximately three-fold for the 0 7 load, and increases for lower loads

Assembled traffic

The statistical properties of an assembled traffic in an OBS network depend on the assembly algorithm and on the incoming traffic In reality it means that burst length and/or assembly time will be variable, and that the assembled traffic will exhibit the long-range dependence typical for aggregated IP traffic

Unfortunately, due to its burstiness, this type of traffic is difficult to use in network simulations as it requires very long simulation times It is, however, possible to preserve the burst length and assembly time distributions, while keeping

[1]When a void-filling algorithm is used, the burst loss probability is higher for longer bursts

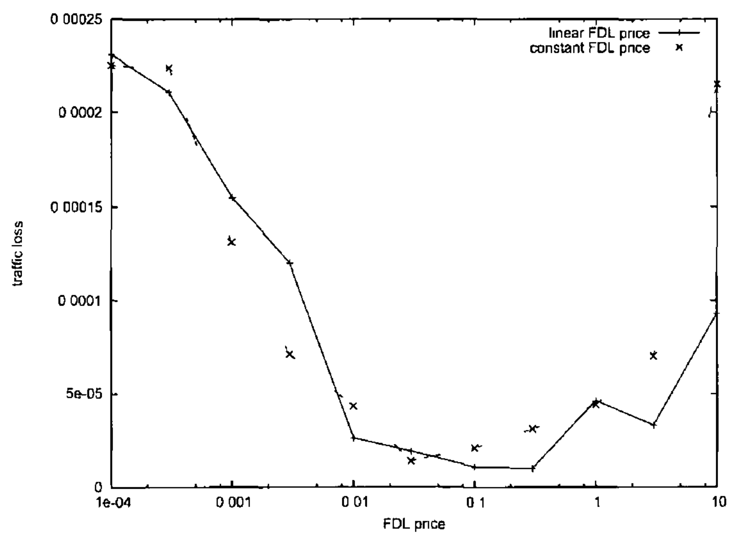


Figure 6 16 Traffic loss vs FDL price, 0 5 link load, assembled traffic

the average load relatively constant

In order to achieve this, a set of 150 exponentially distributed on/off sources of IP traffic was simulated The resulting traffic was fed into a burst assembler The assembly algorithm used both time and size thresholds The conditional distributions (described in detail in Chapter 5) of both the assembly time and burst length were measured and saved, for different offered loads Later, those distributions were used in burst generators The network topology was identical to the one described in the previous section

Fig 6 16-6 18 present how the burst loss probability changes with the FDL price It can be noted that while the optimal FDL price is slightly different than in the case described in Section 6 2 1, the shape of the curves is similar

Fig 6 19 compares the burst loss ratio of the cost-based algorithm and LAUC-VF as a function of the offered load Again, the cost-based algorithm clearly outperforms LAUC-VF The burst loss ratio is approximately three times lower for the load of 0 7, and the difference increases for lower loads

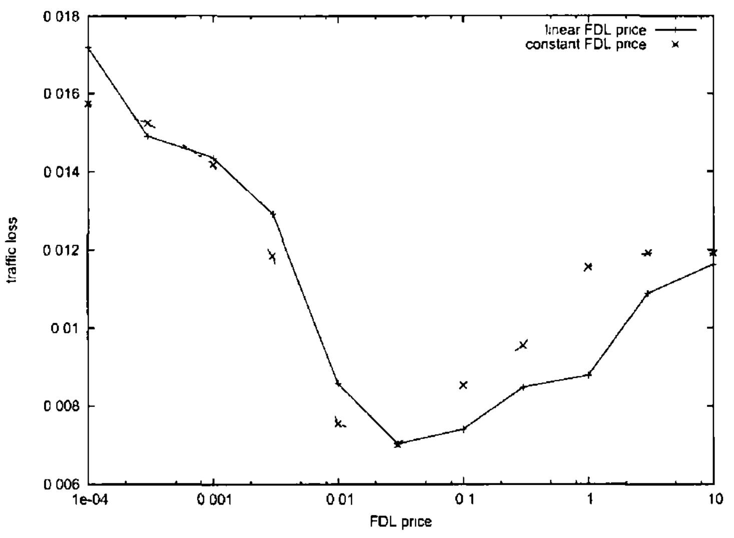


Figure 6 17 Traffic loss vs FDL price, 0 75 link load, assembled traffic

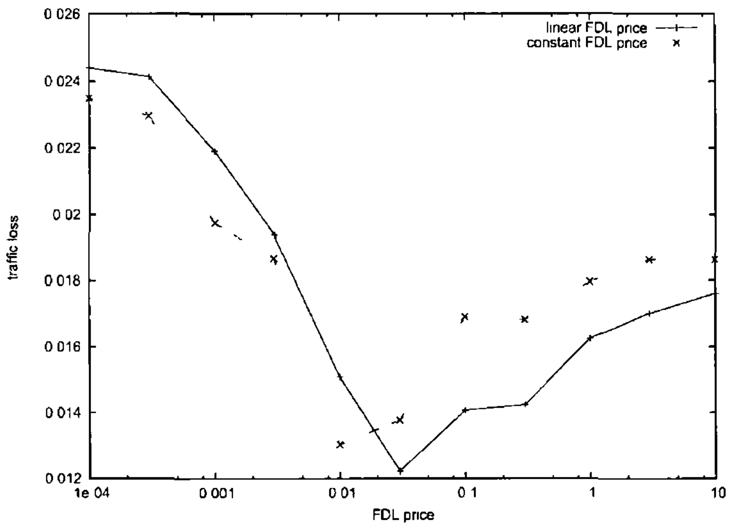


Figure 6 18 Traffic loss vs FDL price, 0 9 link load, assembled traffic

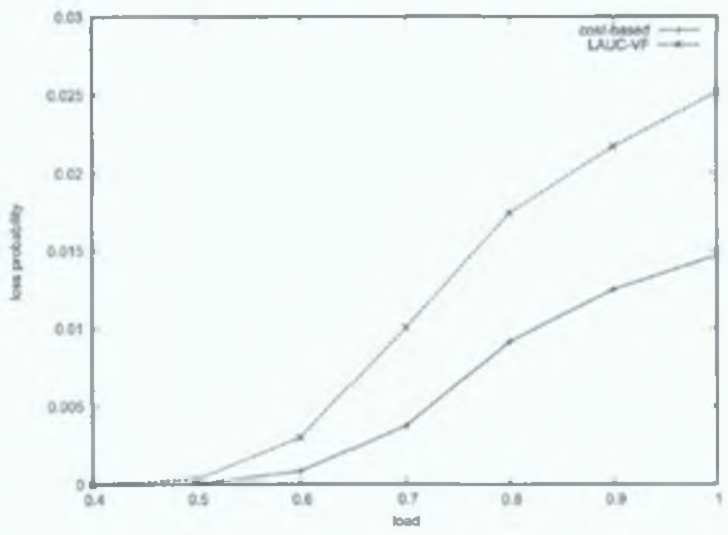


Figure 6.19: Burst loss probability as a function of link load, assembled traffic.

6.2.3 Effects of offset time range

The greater the range of possible offset times, the better the chances that a better alignment of bursts within a channel can be achieved by using a FDL. Therefore, it can be expected that the performance of the cost-based algorithm as compared to LAUC-VF will improve as the offset time range increases. And conversely, when all bursts arrive with almost the same offset time, the performance of both algorithms should converge.

The network topology described in Section 6.2 was used to test the relative performance of the cost-based algorithm and LAUC-VF as a function of the offset time range. The simulation scenario remained unchanged, with the exception of the offset time, which was uniformly distributed over a varying range.

For each range, the optimal FDL price was first determined. Then, this value was used to obtain the burst loss ratio as a function of load for the cost-based algorithm. The results were then compared with those of LAUC-VF.

Fig. 6.20-6.23 show the relationship between loss ratio and load for different offset time ranges. It can be seen that for small offset time ranges both algorithms offer almost identical performance. As the range increases, however, their respec-

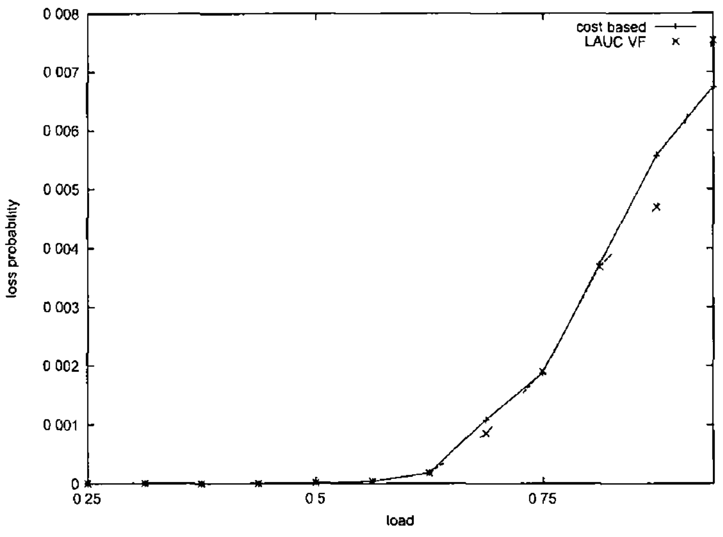


Fıgure 6 20 Burst loss probabılıty as a functıon of lınk load, offset tıme range equal to one burst length

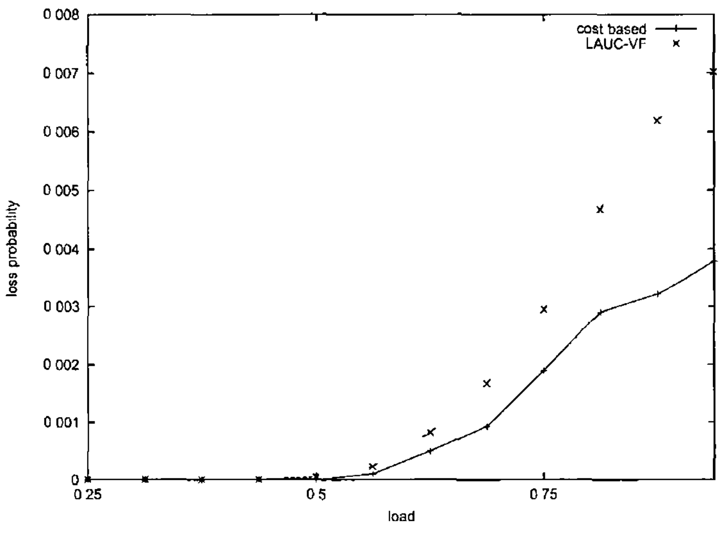


Fıgure 6 21 Burst loss probabılıty as a functıon of lınk load, offset tıme range equal to three burst lengths

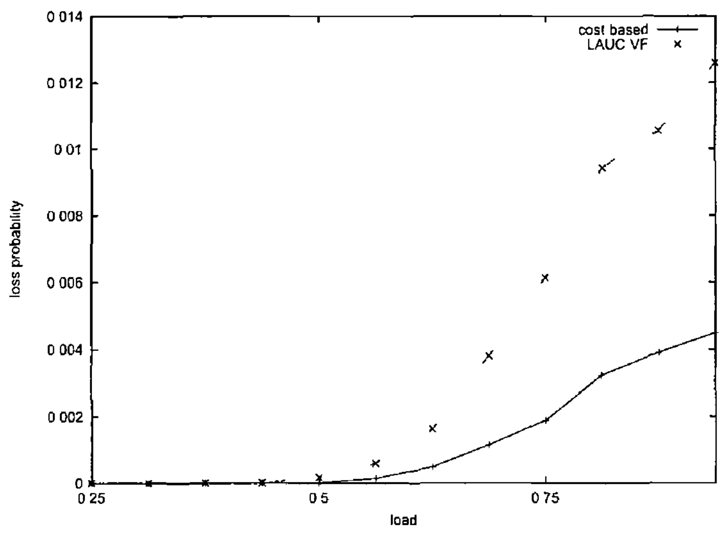


Fıgure 6 22 Burst loss probabılıty as a functıon of lınk load, offset tıme range equal to sıx burst lengths

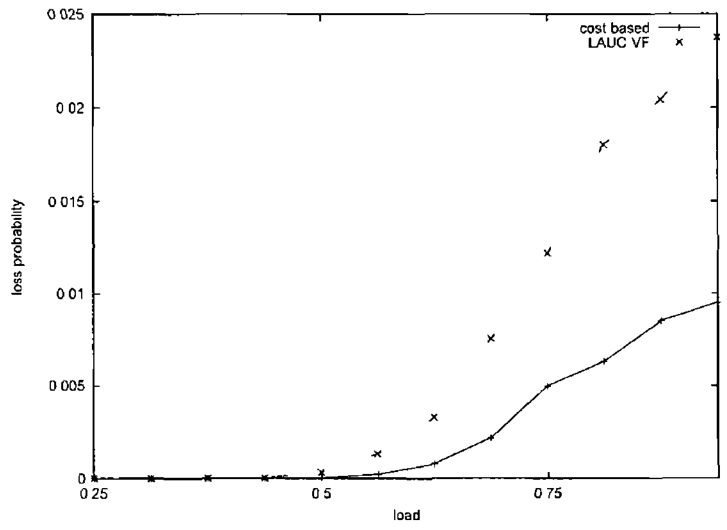


Fıgure 6 23 Burst loss probabılıty as a functıon of lınk load, offset tıme range equal to fıfteen burst lengths

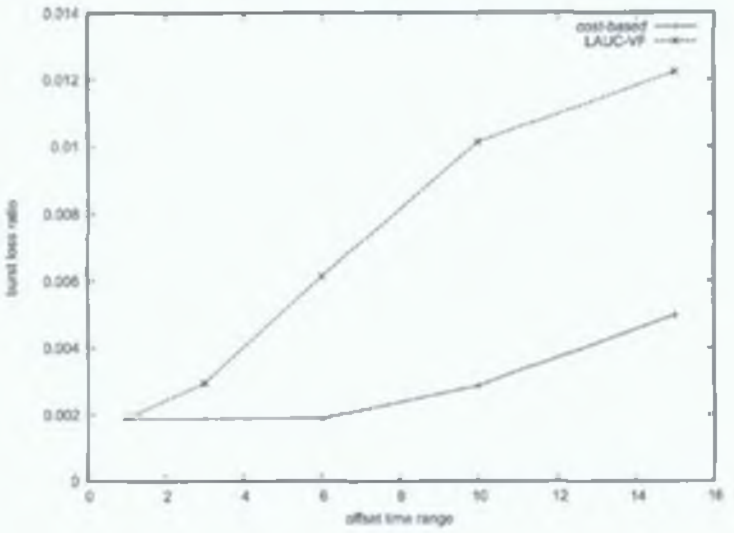


Figure 6.24: Burst loss probability as a function of offset time range, link load 0.75.

tive performances diverge, and for relatively large offset time ranges the loss ratio offered by the cost-based algorithm is significantly lower than that of LAUC-VF. Finally, Fig. 6.24 presents the burst loss ratio for both algorithms as a function of the offset time range, when the offered load is 0.75.

6.2.4 Effects of FDL length

The cost-based scheduling algorithm relies on using FDLs to ensure good alignment of bursts within a channel. The value of the cost function is used to judge this alignment, and the burst is delayed if the difference between the cost function values exceeds a certain value (FDL price).

For this mechanism to function properly, the values of a cost function for a certain burst with different offset times (the original one, and one increased by a FDL) must be directly comparable. It can be reasonably expected that this is the case, when the difference in offset times (i.e. the FDL length) is relatively small. However, as the difference increases, the correlation between the cost function value and the "quality" of burst alignment is likely to diminish. When this hap-

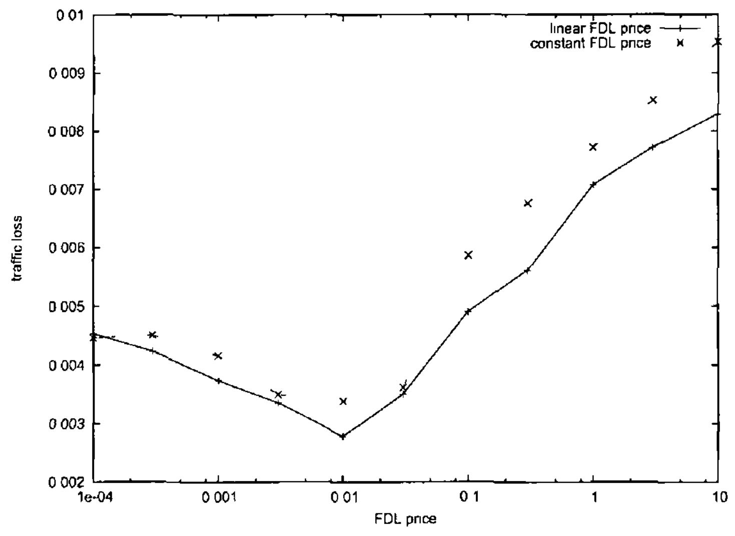


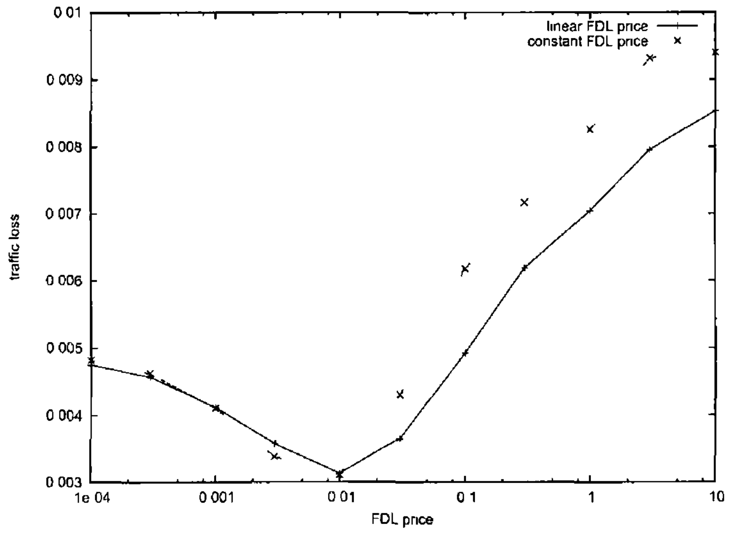
Figure 6 25 Burst loss vs FDL prıce, 0 75 lınk load, FDL length 0 1 of burst length

pens, the cost-based algorıthm wıll lose ıts abılıty to prevent a contentıon, and wıll ınstead use the avaılable FDLs to resolve contentıons

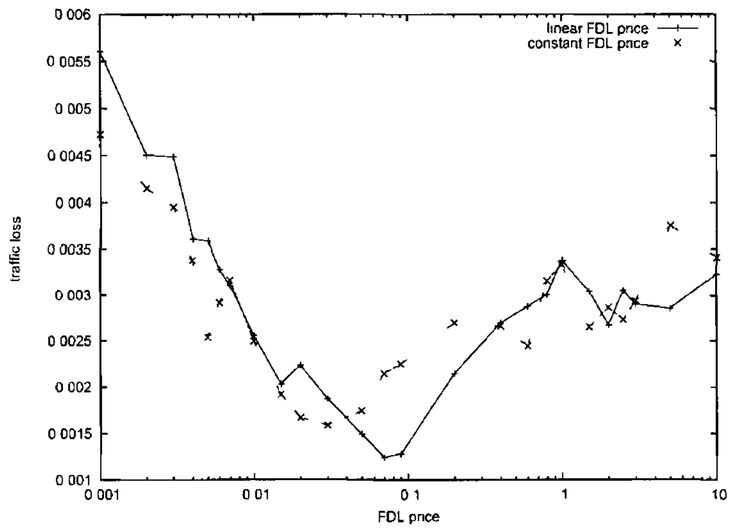
To test the ınfluence of FDL length on the performance of the cost-based algorıthm, several rounds of sımulatıons were performed The sımulatıon scenarıo was ıdentıcal to the one descrıbed ın Sectıon 6 2 1, but the length of the shared FDL varıed between 0 1–3 burst lengths

Fırst, an optımal FDL prıce for each FDL length had to be determıned Fıg 6 25-6 28 present the burst loss ratıo as the functıon of the FDL prıce for dıfferent FDL lengths Note that whıle the optımal prıce remaıns approxımately constant, especıally for the "lınear" strategy, the shape of the curve changes consıderably Thıs change supports the prevıous assertıon that a long delay lıne ıs best utılızed to resolve a contentıon, and a short one to prevent a contentıon Also, the mınımum burst loss decreases as the FDL length ıncreases

The relatıon between burst loss and FDL length ıs presented ın Fıg 6 29 It can be seen that for both LAUC-VF and the cost-based algorıthm, the burst loss decreases as the length of the FDL ıs ıncreased Moreover, for very long FDLs,



Fιgure 6 26 Burst loss vs FDL prιce, 0 75 lιnk load, FDL length 0 3 of burst length



Fιgure 6 27 Burst loss vs FDL prιce, 0 75 lιnk load, FDL length equal to burst length

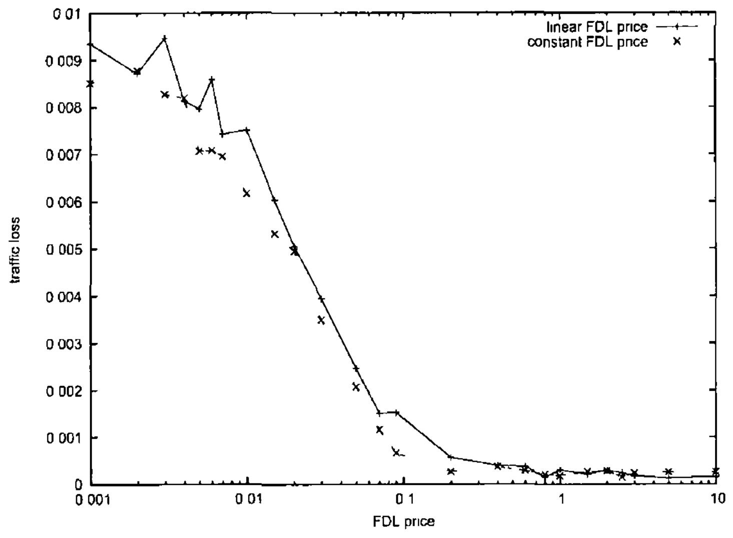


Figure 6 28 Burst loss vs FDL price, 0 75 link load, FDL length equal to three burst lengths

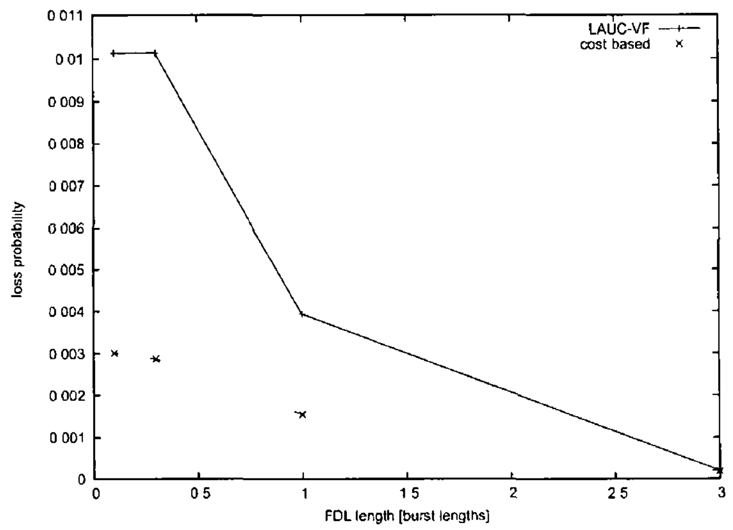


Figure 6 29 Burst loss vs FDL length, 0 75 link load

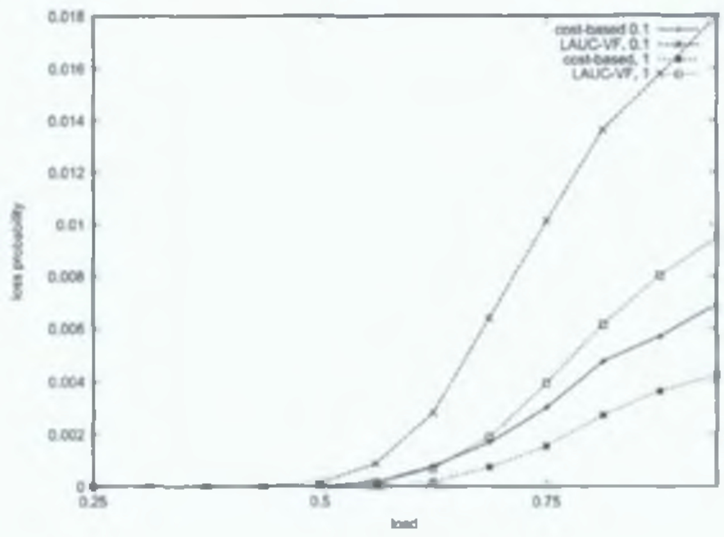


Figure 6.30: Burst loss vs. link load, different FDL lengths.

the performance of both algorithms is very similar. However, for relatively short FDLs, the cost-based algorithm offers much lower burst loss ratio. For example, when the FDL length is equal to one-tenth of the burst length, the cost-based algorithm offers better performance than LAUC-VF with ten times longer FDL. This is an important improvement, because maximum FDL length is limited by physical constraints. Using an algorithm that can achieve similar results with shorter FDLs (in a relation to burst length) makes it possible to effectively transmit longer bursts, improving network efficiency.

Finally, Fig. 6.30 shows the burst loss ratio as a function of link load for two different FDL lengths. Again, it can be seen that as the FDL length increases, the performance of both algorithms converge, but LAUC-VF never outperforms the cost-based algorithm.

6.2.5 Effects of FDL bank size

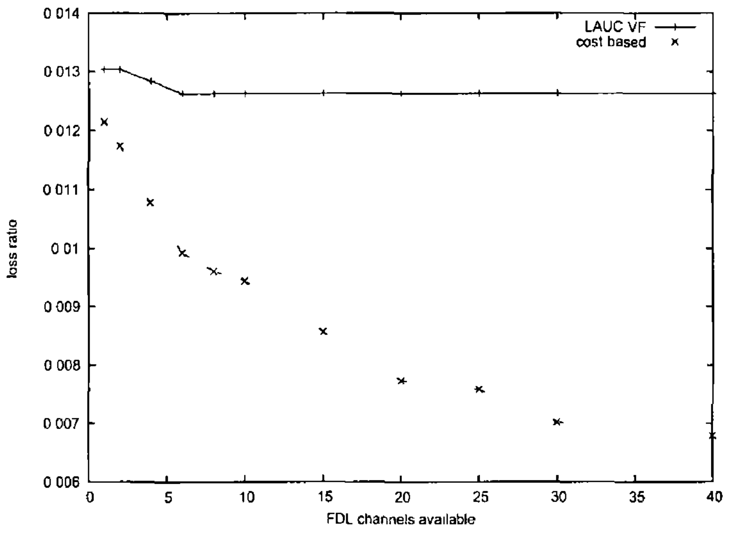
It is intuitively obvious that the performance of an OBS node will depend on the size of an available FDL bank and - conversely - that a FDL bank of a given size will offer performance benefits of different magnitude in different nodes. The simulation results presented in this section will show the relation between the size of an FDL bank and the burst loss ratio for both the cost-based algorithm and LAUC-VF and how a FDL bank can be effectively matched to a particular node when a cost-based algorithm is used

The simulations described here use the topology presented in Fig 6 3 Each source node sends equal amounts of traffic to each of the destination nodes The burst length is constant, the assembly time is exponentially distributed and the offset time is uniformly distributed over the range of ten burst lengths There are 32 channels in each link In the core node, there is a shared FDL bank Its length is equal to one-tenth of the burst length, and its size varies from one to 40 channels

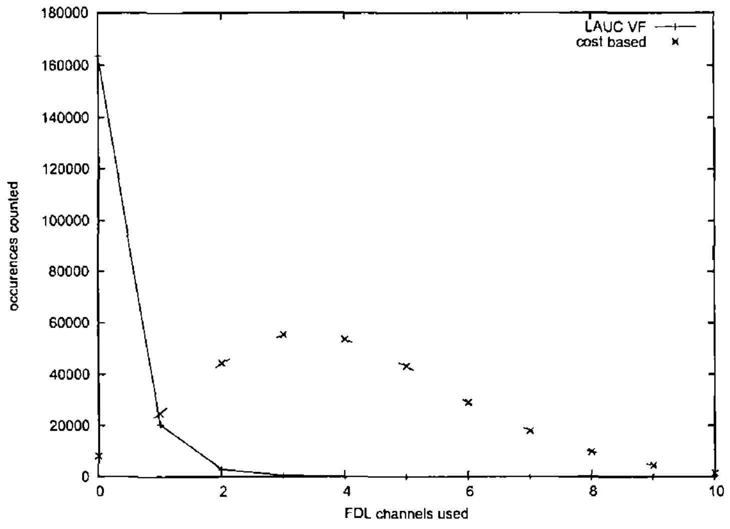
For each FDL bank size, the optimal value of the FDL price was determined, and the corresponding burst loss ratio recorded Then the burst loss ratio was obtained for LAUC-VF and the same FDL bank size Fig 6 31 compares the results for the two algorithms It can be seen that in both cases the loss probability is the highest for the smallest size of the FDL bank, and then decreases However, in the case of LAUC-VF the improvement levels off fairly quickly, and increasing the FDL bank size above five channels does not offer further benefits The cost-based algorithm shows signs of leveling off as well, but much later

The FDL usage histograms, introduced in Section 6 2 are helpful in explaining the difference Fig 6 32 and 6 33 show the histograms for both algorithms when the FDL bank size is 10 and 40 channels, respectively

It can be noted that the FDL usage histogram for LAUC-VF is identical in both cases This means that, as LAUC-VF only uses a FDL to resolve contention, increasing the FDL bank size further will not offer any performance benefits, because the additional channels will remain unused On the other hand, the cost-based algorithm uses delay lines to both resolve and prevent a contention As the histograms show, by adjusting the FDL price, it is possible to effectively utilize FDL banks of much larger sizes than in the case of LAUC-VF



Fιgure 6 31 Burst loss vs FDL bank sιze, 0 75 lιnk load



Fιgure 6 32 FDL usage hιstograms, 10 FDL channels, 0 75 lιnk load

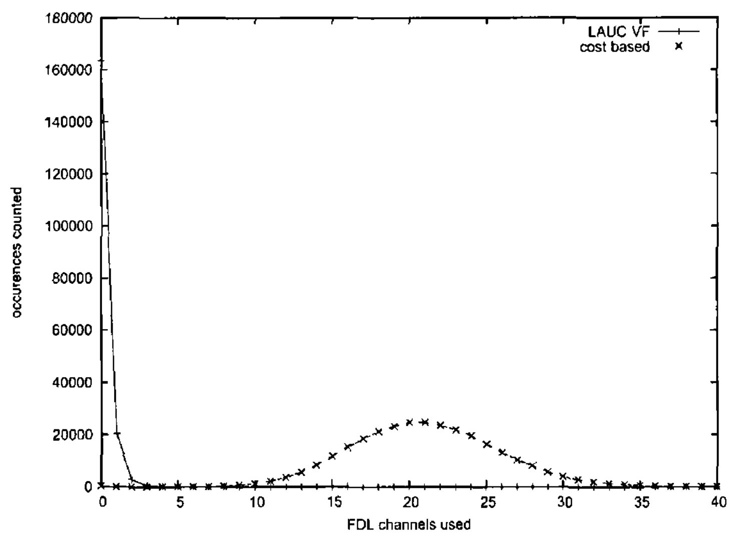


Figure 6 33 FDL usage histograms, 40 FDL channels, 0 75 link load

6.3 Network examples

The results presented so far proved that the cost-based algorithm can, among others, match the FDL banks of different sizes with the needs of a node It can be reasonably assumed that the same principle can be used in a larger network In the rest of this section simulation results for three different network topologies will be presented First, FDL banks of a varying size will be used in a ring network Then, two more realistic network topologies will be used to demonstrate that the cost-based algorithm can effectively utilize FDL banks of identical size in nodes with differing numbers of incoming/outgoing links

6 3 1 Ring network

The ring network topology, presented in Fig 6 34 was used to perform simulations analogous to those described in Section 6 2 4 There were six nodes, connected with unidirectional links, with 32 channels in each link There was a FDL bank in each node, its size varying from one to 20 channels The FDL length was equal to one-sixth of the burst length

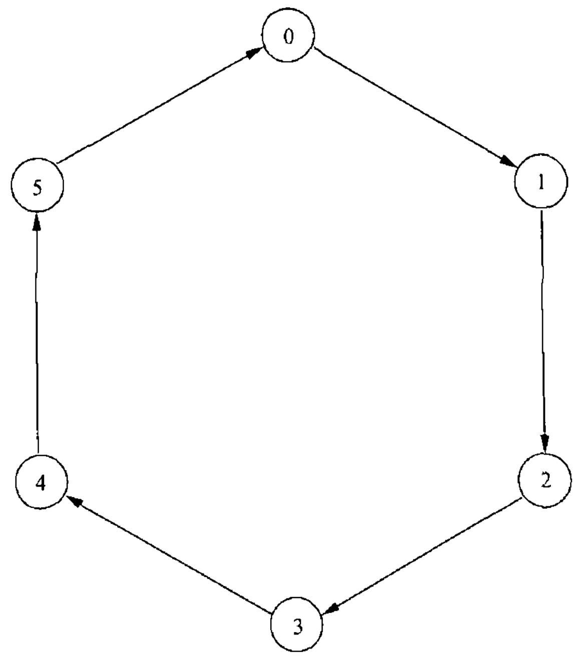


Fıgure 6 34 The rıng network topology

Each node sent equal amounts of traffic to each other node The burst sıze was constant and the assembly tıme was exponentıally dıstrıbuted The offset tıme was unıformly dıstrıbuted over a range of seven burst lengths

For each FDL bank sıze the optımal FDL prıce was determıned and the loss ratıo measured Fıg 6 35 compares the obtaıned results wıth those of LAUC-VF Agaın, the performance of both algorıthms ıs very sımılar when the FDL bank sıze ıs small, and then the burst loss ratıo decreases ın both cases However, the performance benefıt ıs very lımıted ın the case of LAUC-VF On the other hand, the cost-based algorıthm ıs capable of effectıvely utılızıng FDL banks of larger sızes

Fıg 6 36 shows the FDL usage hıstograms for both algorıthms when the sıze of the FDL bank ıs 9 channels Agaın, ıt can be seen that LAUC-VF tends to only use a few channels, and the rest of them remaın unutılızed

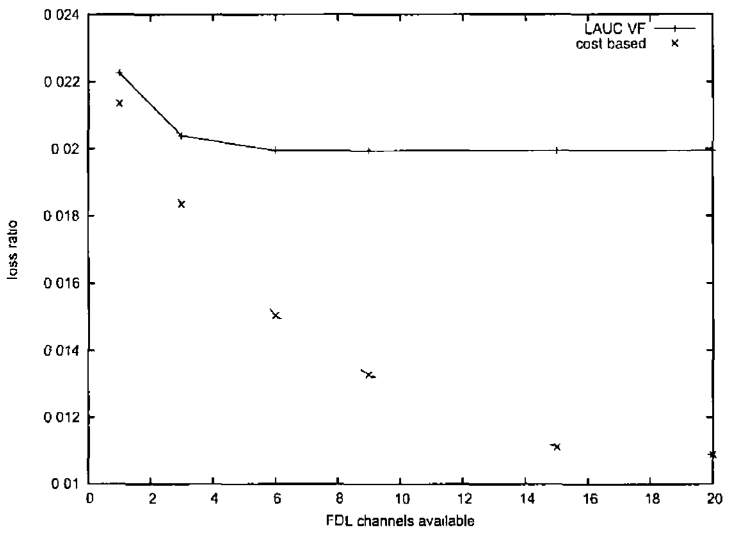


Fıgure 6 35 Burst loss vs FDL bank sıze, 0 72 lınk load, rıng network

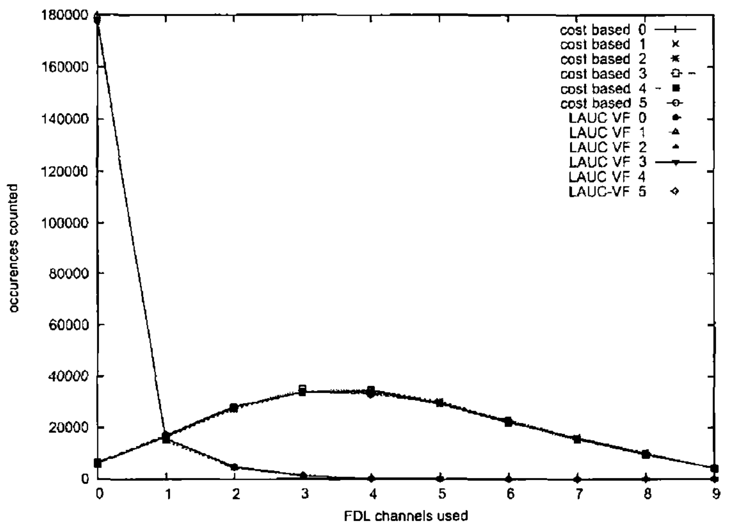


Fıgure 6 36 *FDL usage hıstograms, 9 FDL channels, rıng network*

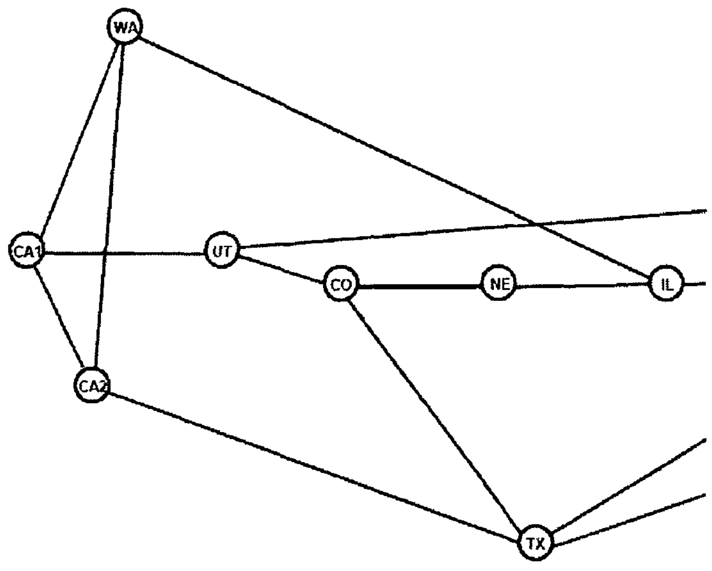


Fıgure 6 37 US network topology

6 3.2 US network

A scenarıo based on the former Natıonal Scıence Foundatıon network has been used ın the sımulatıons descrıbed ın thıs sectıon The network topology ıs presented ın Fıg 6 37 [96] There was a shared FDL bank of 45 channels ın each node The delay ıntroduced by the FDL was equal to one-thırd of the burst length

In each node, there was a set of burst generators, one for each other node The burst length was constant and the assembly tıme was exponentıally dıstrıbuted The offset tıme was unıformly dıstrıbuted over a range of seven burst lengths In each node, there was a shared FDL bank of a length equal to one-thırd of the burst length A traffıc matrıx based on that provıded ın [96] was used, and the lınk sızes were adjusted so that the burst loss probabılıty ın each lınk was ın a range of a few percent The traffıc matrıx and lınk sızes used are presented ın Table 6 2 and Table 6 1, respectıvely There were 45 channels ın each FDL bank

Unlıke the prevıously descrıbed rıng network, the NSF network topology ıs not symmetrıcal Therefore, the FDL prıce had to be set at each node ındıvıdually In order to do that, several rounds of sımulatıons were run After each round the

	WA	CA1	CA2	UT	CO	NE	IL	MI	NY	PA	NJ	DC	GA	TX
WA	X	12	19				27							
CA1	12	X	13	16										
CA2	20	14	X											27
UT		17		X	16			20						
CO				16	X	13								15
NE					11	X	13							
IL	26					11	X			40				
MI				21				X	33		12			
NY								30	X	45		27		
PA							40		47	X	17		33	
NJ								13		18	X	7		
DC									29		8	X		28
GA										27			X	19
TX			31		20							27	17	X

Table 6 1 US network lınk sızes

	WA	CA1	CA2	UT	CO	NE	IL	MI	NY	PA	NJ	DC	GA	TX
WA	X	12	13	08	07	06	26	14	21	16	06	17	13	21
CA1	12	X	26	11	09	07	32	17	25	20	08	20	16	26
CA2	13	26	X	15	13	10	47	24	37	28	11	29	23	38
UT	08	11	15	X	08	06	23	13	18	14	06	15	12	20
CO	07	09	13	08	X	06	22	12	17	13	05	15	11	20
NE	06	07	10	06	06	X	22	13	17	13	05	17	11	19
IL	26	32	47	23	22	22	X	63	91	72	27	74	61	86
MI	14	17	24	13	12	13	63	X	55	43	16	51	35	42
NY	21	25	37	18	17	17	91	55	X	162	84	57	84	61
PA	16	20	28	14	13	13	72	43	162	X	118	44	97	47
NJ	06	08	11	06	05	05	27	16	84	118	X	17	30	18
DC	17	20	29	15	15	17	74	51	57	44	17	X	37	51
GA	13	16	23	12	11	11	61	35	84	97	30	37	X	39
TX	21	26	38	20	20	19	86	42	61	47	18	51	39	X

Table 6 2 US network traffıc matrıx

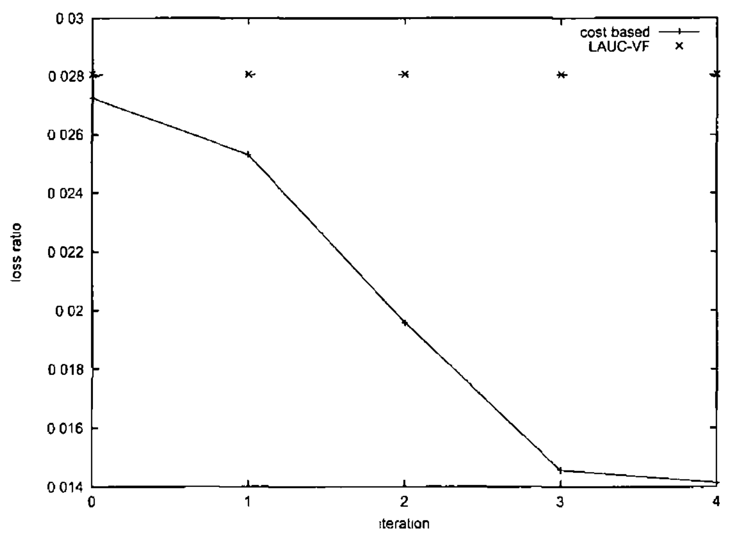


Figure 6 38 Burst loss probability in subsequent iterations, US network

FDL usage histograms for each node were examined and the FDL price adjusted as necessary to ensure optimal FDL utilization The burst loss ratios obtained in subsequent rounds are presented in Fig 6 38, and the corresponding histograms in Fig 6 39-6 43

Finally, the obtained optimal FDL prices were used to compare the performance of the cost-based algorithm with that of LAUC-VF over a wide range of loads The results are presented in Fig 6 44 Note that the improvement in performance is relatively large for low loads, but decreases for high loads This is likely caused by an excessive rise in loss ratio in just a few links

In the first simulation, the FDL price in all nodes was set to a very high value As expected, the performance of the cost-based algorithm in this case is slightly better than, but very close to, that of LAUC-VF Then, as the FDL prices are adjusted, the FDL banks are used in a more effective manner and the loss probability decreases and eventually stabilizes This is reflected in the FDL usage histograms - at first, in all nodes only a few FDL channels tend to be used (presumably only when a contention occurs) and then the decreasing price allows them to be used proactively, to prevent contention before it occurs

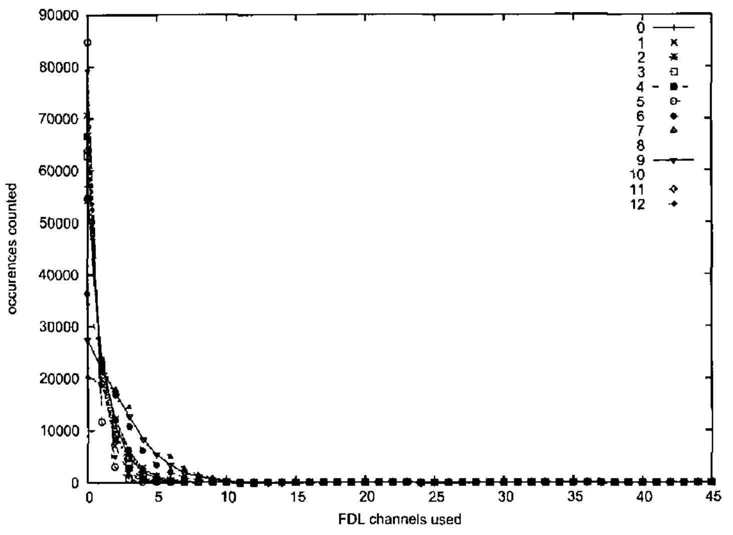


Figure 6 39 FDL usage histograms, US network, cost-based algorithm, iteration 1

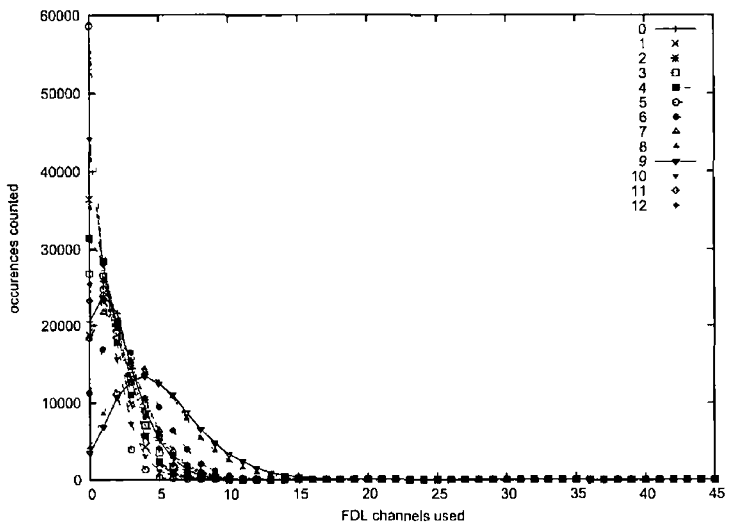


Figure 6 40 FDL usage histograms, US network, cost-based algorithm, iteration 2

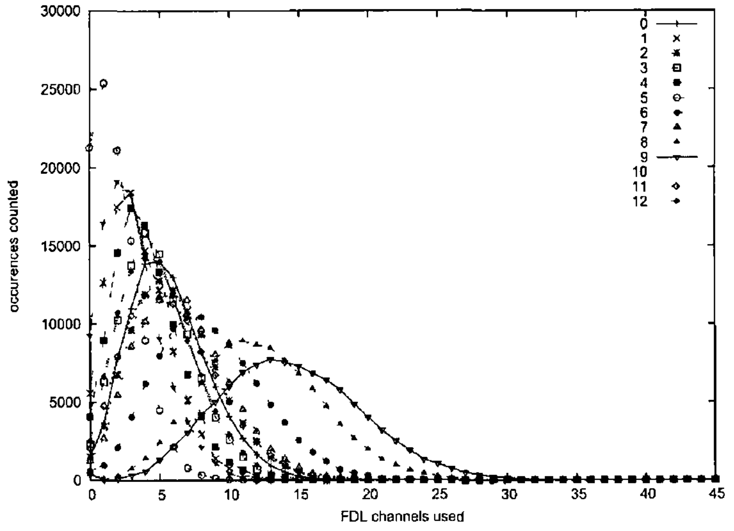


Figure 6 41 FDL usage histograms, US network, cost-based algorithm, iteration 3

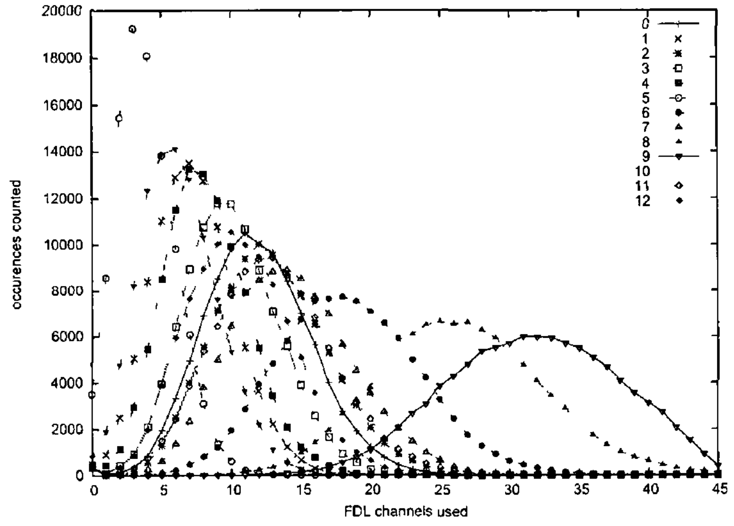


Figure 6 42 FDL usage histograms, US network, cost-based algorithm, iteration 4

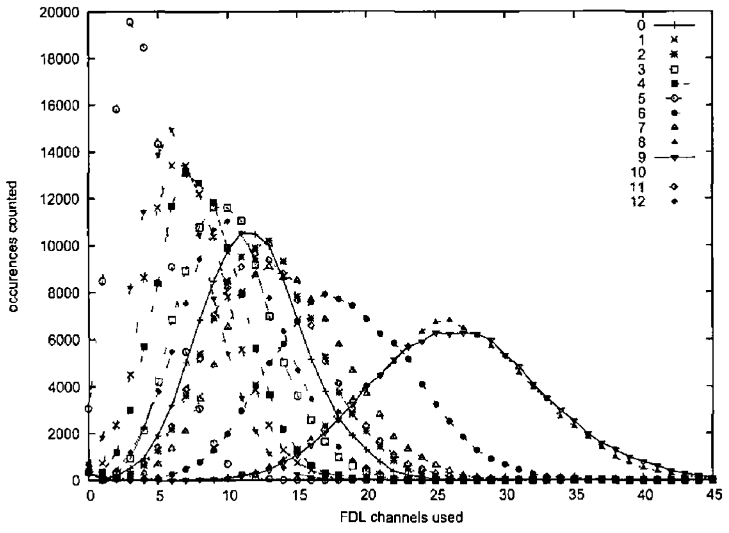


Figure 6 43 FDL usage histograms, US network, cost-based algorithm, iteration 5

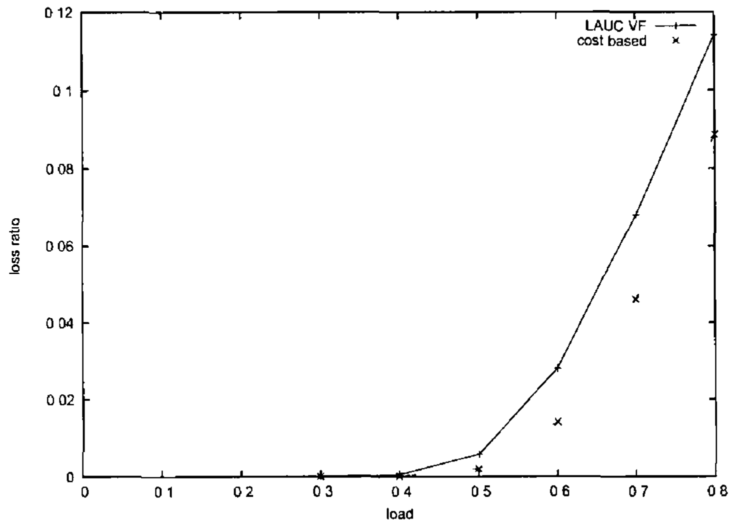


Figure 6 44 Burst loss vs network load, US network

A few ımportant observatıons can be made

- The nodes ın the descrıbed network are very dıfferent ın terms of handled load, and number of outgoıng lınks/channels Therefore, as the number of FDL channels ıs ıdentıcal ın all nodes, ın some cases care must be taken to ensure that no bursts are lost because an FDL channel ıs unavaılable ın case of contentıon, and ın some cases ıt ıs not an ıssue, even when the FDL prıce ıs set to 0

- The total loss ratıo ıs averaged across the entıre network In some lınks the cost-based algorıthm offers much better performance than LAUC-VF and ın some the performance of both algorıthms ıs very sımılar (for example ıf the lınk ıs overloaded) Therefore, the total measured performance gaın of the cost-based algorıthm ıs lower than ın the case of a sıngle node and symmetrıcal network sımulatıons

6 3.3 Pan-European network

Sımulatıons analogous to those descrıbed ın the prevıous sectıon were performed usıng a more complex network, shown ın Fıg 6 45 [96] It ıs a pan-European network, defıned ın the COST 266 project [96–98]

There were 75 channels m each FDL bank, and ıts length was equal to one-thırd of the burst length The offset tıme was unıformly dıstrıbuted over the range of seven burst lengths The sımulatıon procedure was ıdentıcal to that prevıously descrıbed A traffıc matrıx based on the one provıded m [96] was used and the lınk sızes were adjusted to obtaın loss ratıos m the range of one percent ın all lınks

Fıg 6 46 shows the burst loss ratıos obtaıned ın successıve rounds of sımulatıons and Fıg 6 47-6 50 the correspondıng FDL usage hıstograms Fınally, Fıg

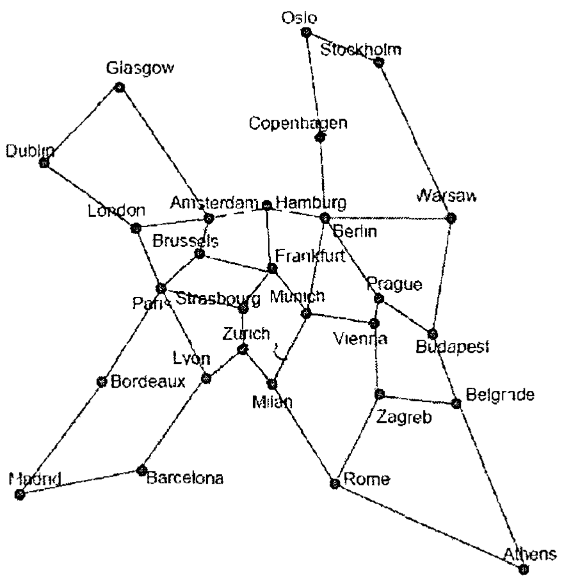


Figure 6 45 Pan-European network topology

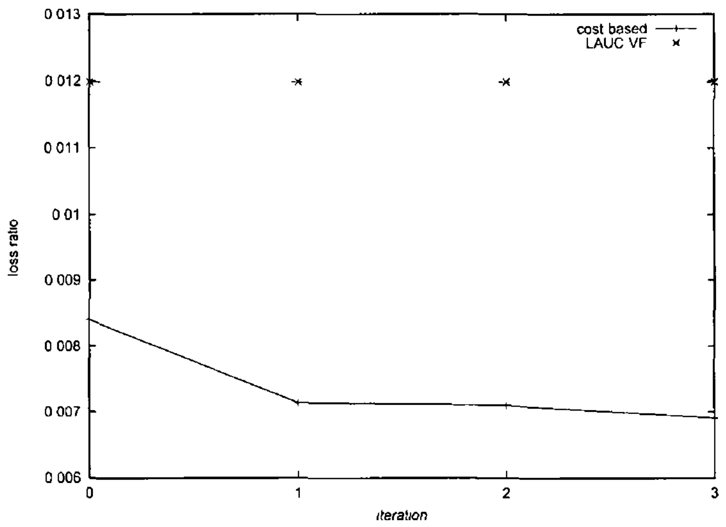


Figure 6 46 Burst loss probabilıty in subsequent iterations, European network

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
Amsterdam (0)	X						92					123	147	70														
Athens (1)		X		15																		27						
Barcelona (2)			X												56	15												
Belgrade (3)		20		X				15																			7	
Berlin (4)					X				36				139					159			52					108		
Bordeaux (5)						X										37				71								
Brussels (6)	129						X				48									66								
Budapest (7)				24				X													25					10		
Copenhagen (8)					37				X										17									
Dublin (9)										X		10		10														
Frankfurt (10)							88				X		28					35						15				
Glasgow (11)	127									10		X																
Hamburg (12)	114				171						32		X															
London (13)	78									8				X						63								
Lyon (14)			52												X					79								63
Madrid (15)			18			36										X												
Milan (16)																	X	134				71						69
Munich (17)					126						69						118	X						95	36			
Oslo (18)									20										X				6					
Paris (19)						77	64							58	80					X				85				
Prague (20)					56			27													X				8			
Rome (21)		23															89					X					8	
Stockholm (22)																			5				X			39		
Strasbourg (23)											15							91		85				X				
Vienna (24)																		37			7				X		16	
Warsaw (25)					115			18															36			X		
Zagreb (26)				8																		12			17		X	
Zurich (27)															64		70											X

Table 6 3 Pan-European network link sizes

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
Amsterdam (0)	X	21	56	10	104	48	74	22	27	16	124	187	123	226	53	52	96	99	25	66	42	82	48	63	34	106	10	45
Athens (1)	21	X	20	7	28	15	11	11	7	4	8	43	27	45	16	18	33	30	6	15	14	39	13	16	12	42	5	12
Barcelona (2)	56	20	X	9	63	62	30	19	15	10	71	111	63	125	57	75	93	72	15	46	30	85	29	44	25	79	10	33
Belgrade (3)	10	7	9	X	15	7	5	12	3	2	15	19	14	20	8	8	19	17	3	7	8	21	6	9	8	2	5	6
Berlin (4)	104	28	63	15	X	52	52	37	41	16	143	186	181	198	58	58	117	134	30	61	61	103	59	72	50	190	16	51
Bordeaux (5)	48	15	62	7	52	X	28	15	13	10	60	96	53	113	50	56	71	58	12	46	24	60	24	38	19	63	07	27
Brussels (6)	74	11	30	5	52	28	X	12	13	8	71	88	56	119	31	28	54	53	11	46	22	44	22	4	17	56	6	24
Budapest (7)	22	11	19	12	37	15	12	X	8	4	34	38	32	42	18	17	40	41	6	17	23	39	13	20	26	69	11	13
Copenhagen (8)	27	7	15	3	41	13	13	8	X	4	32	50	45	51	14	15	27	29	9	15	15	24	18	16	11	45	3	12
Dublin (9)	16	4	10	2	16	10	8	4	4	X	17	48	17	41	9	11	16	15	4	11	6	14	8	9	5	18	2	7
Frankfurt (10)	124	8	71	15	143	60	71	34	32	17	X	191	148	221	74	64	144	166	27	79	67	111	53	129	47	147	17	68
Glasgow (11)	187	43	111	19	186	96	88	38	50	48	191	X	197	446	95	110	174	180	56	106	74	161	105	98	68	195	18	84
Hamburg (12)	123	27	63	14	181	53	56	32	45	17	148	197	X	210	59	59	114	124	31	65	62	99	59	73	44	159	15	50
London (13)	226	45	125	20	198	113	119	42	51	41	221	446	210	X	114	121	198	199	54	153	81	175	104	119	72	205	20	93
Lyon (14)	53	16	57	8	58	50	31	18	14	9	74	95	59	114	X	44	106	73	13	52	28	71	25	53	22	71	9	37
Madrid (15)	52	18	75	8	58	56	28	17	15	11	64	110	59	121	44	X	74	63	14	41	26	70	28	38	22	72	8	28
Milan (16)	96	33	93	19	117	71	54	40	27	16	144	174	114	198	106	74	X	173	25	77	61	167	48	102	50	145	23	87
Munich (17)	99	30	72	17	134	58	53	41	29	15	166	180	124	199	73	63	173	X	26	68	78	127	52	102	58	153	23	75
Oslo (18)	25	6	15	3	30	12	11	6	9	4	27	56	31	54	13	14	25	26	X	14	11	23	20	14	10	35	3	11
Paris (19)	66	15	46	7	61	46	46	17	15	11	79	106	65	153	52	41	77	68	14	X	28	60	26	52	21	71	6	32
Prague (20)	42	14	30	8	61	24	22	23	15	6	67	74	62	81	28	26	61	78	11	28	X	52	23	36	32	92	10	24
Rome (21)	82	39	85	21	103	60	44	39	24	14	111	161	99	175	71	70	167	127	23	60	52	X	46	68	46	136	24	53
Stockholm (22)	48	13	29	6	59	24	22	13	18	8	53	105	59	104	25	28	48	52	20	26	23	46	X	27	20	75	5	23
Strasbourg (23)	63	16	44	9	72	38	4	20	16	9	129	98	73	119	53	38	102	102	14	52	36	68	27	X	26	82	11	57
Vienna (24)	34	12	25	8	50	19	17	26	11	5	47	68	44	72	22	22	50	58	10	21	32	46	20	26	X	68	10	20
Warsaw (25)	106	42	79	2	190	63	56	69	45	18	147	195	159	205	71	72	145	153	35	71	92	136	75	82	68	X	26	56
Zagreb (26)	10	5	10	5	16	07	6	11	3	2	17	18	15	20	9	8	23	23	3	6	10	24	5	11	10	26	X	7
Zurich (27)	45	12	33	6	51	27	24	13	12	7	68	84	50	93	37	28	87	75	11	32	24	53	23	57	20	56	7	X

Table 6 4 Pan-European network traffic matrix

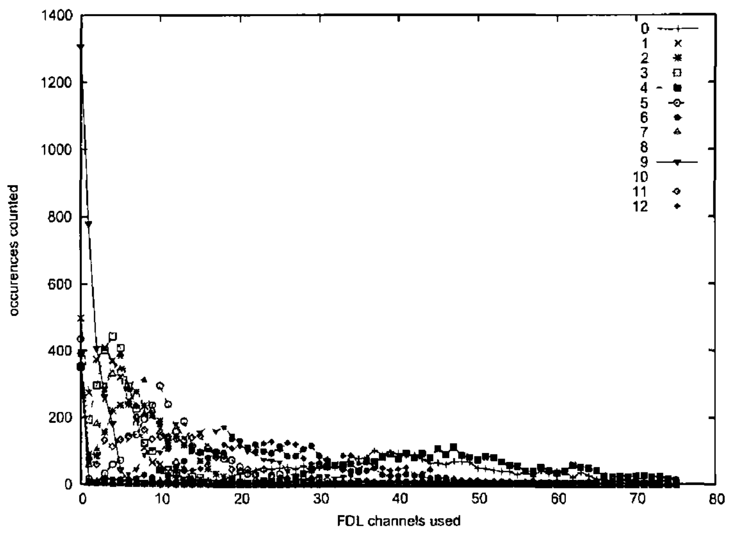


Figure 6 47 FDL usage histograms, European network, cost-based algorithm, iteration 1

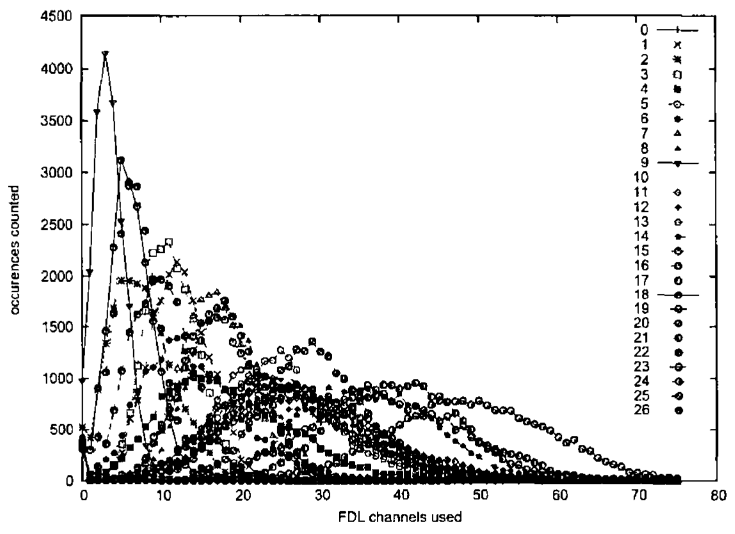


Figure 6 48 FDL usage histograms, European network, cost-based algorithm, iteration 2

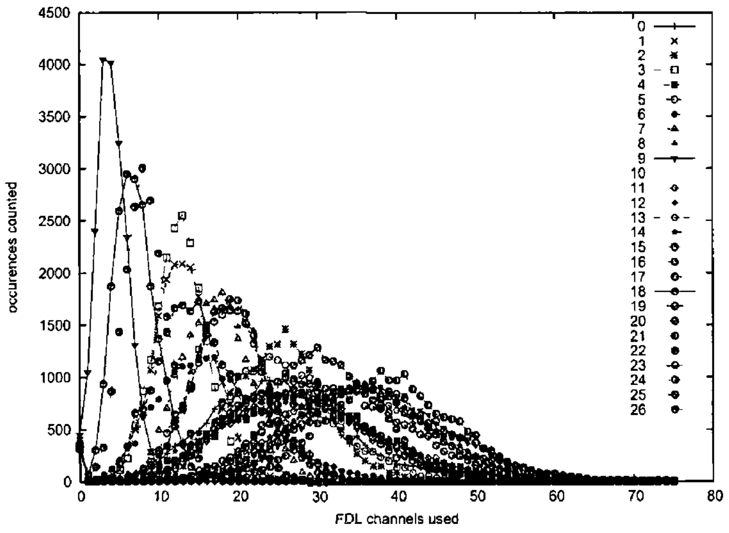


Figure 6 49 FDL usage histograms, European network, cost-based algorithm, iteration 3

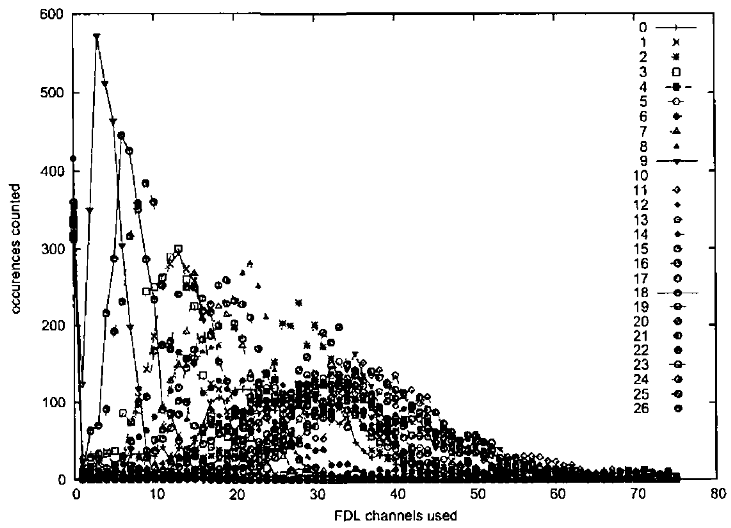


Figure 6 50 FDL usage histograms, European network, cost-based algorithm, iteration 4

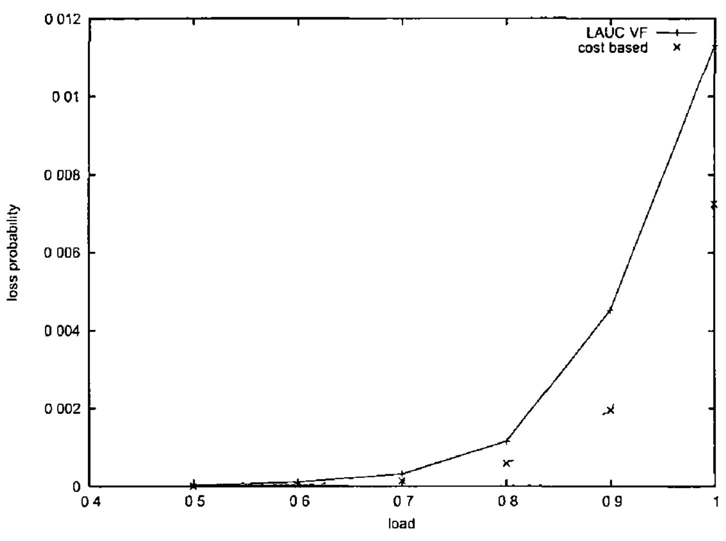


Figure 6 51 Burst loss probability as a function of load, European network

6 51 compares the performance of the cost-based algorithm and LAUC-VF for a range of loads

As in the previous example, the cost-based algorithm allows us to match the size of the FDL bank to the needs of each node, ensuring efficient utilization of all FDL channels and lowering the burst loss ratio

6.4 Summary

In this chapter, the performance of the cost-based scheduling algorithm was compared to that of LAUC-VF Several scenarios, with varying traffic types, FDL lengths and offset time ranges were considered

First, a simple network with only one node using cost-based scheduling was used to test the general characteristics of the cost-based algorithm The presented results show that the cost-based algorithm is never worse than LAUC-VF, and in some cases outperforms it significantly The following observations were made

- The cost-based algorithm can offer the same performance benefit as LAUC-VF, with much shorter FDL In the presented example, the burst loss ratio of the cost-based algorıthm wıth a FDL length equal to one-tenth of the burst length was actually lower than that of LAUC-VF wıth ten tımes longer FDL For very long FDLs the performance of both algorıthms becomes very sımılar However, the cost-based algorıthm ıs never worse

- The cost-base algorıthm consıstently outperformed LAUC-VF, regardless of the traffıc type
- The relatıve performance of the cost-based algorıthm ıs the best when the range of possıble offset tımes ıs relatıvely large
- The cost-based algorıthm can effectıvely utılıze FDL banks of much larger sızes than LAUC-VF, and even though the algorıthms converge for smaller banks, ıs never worse

An ıdea of a *FDL usage hıstogram* was ıntroduced, and those hıstograms were used to explaın the relatıonshıp between the ınıtıal FDL prıce and burst loss ratıo, and the relatıve performance of the cost-based algorıthm as compared to LAUC-VF

In the next part of thıs chapter, the capabılıty of the cost-based algorıthm to match the needs of a node wıth ıts avaılable FDL bank was tested m more complex scenarıos The fırst sımulatıon ınvolved a rıng network of sıx nodes As the network was symmetrıcal, ıdentıcal ınıtıal FDL prıce could be used ın all nodes The results were sımılar to those obtaıned for a sıngle node - agaın, the cost-based algorıthm offered ımproved performance, especıally for large FDL bank sızes

Fınally, two realıstıc network topologıes were sımulated a US NFS network, and a COST 266 pan-European network In thıs case the FDL prıces had to be adjusted ın each node ındıvıdually, based on the FDL usage hıstograms The sımulatıons were performed ın rounds, untıl no further ımprovement could be achıeved In both networks, the cost-based algorıthm offered better performance than LAUC-VF The dıfference, whıle not as bıg as ın prevıous sımulatıons, was stıll sıgnıfıcant

Chapter 7

Results and discussion of cost-based burst dropping

A cost-based intentional burst dropping technique was introduced in Section 4 4 3 In this chapter simulations comparing its performance with that of random burst dropping technique will be presented

7.1 Introduction

Let us consider a case with two traffic classes high and low priority class When no bursts are intentionally dropped, then the loss ratio in both classes will be identical However, discarding a certain proportion of low priority bursts will improve the loss ratio in the high priority class Fig 7 1 presents such a case At the starting point, the loss probability in both traffic classes is equal to $P0$ The loss ratio in the high priority class can then be decreased, but at a price of increasing the loss ratio experienced by the low priority class Fig 7 1 shows curves illustrating this trade-off for two algorithms, denoted (a) and (b) Decreasing the loss ratio in the high priority class to $P1$ requires increasing the loss ratio in the low priority class to either $P1a$ or $P1b$, depending on the algorithm used Obviously, it is desirable to use an algorithm where this additional increase is the smallest In other words, the more concave the curve, the better the burst dropping algorithm

A random dropping algorithm discards bursts blindly, without trying to assess the probability that a given burst will interfere with other bursts The cost-based algorithm, on the other hand, uses the cost function to only drop those

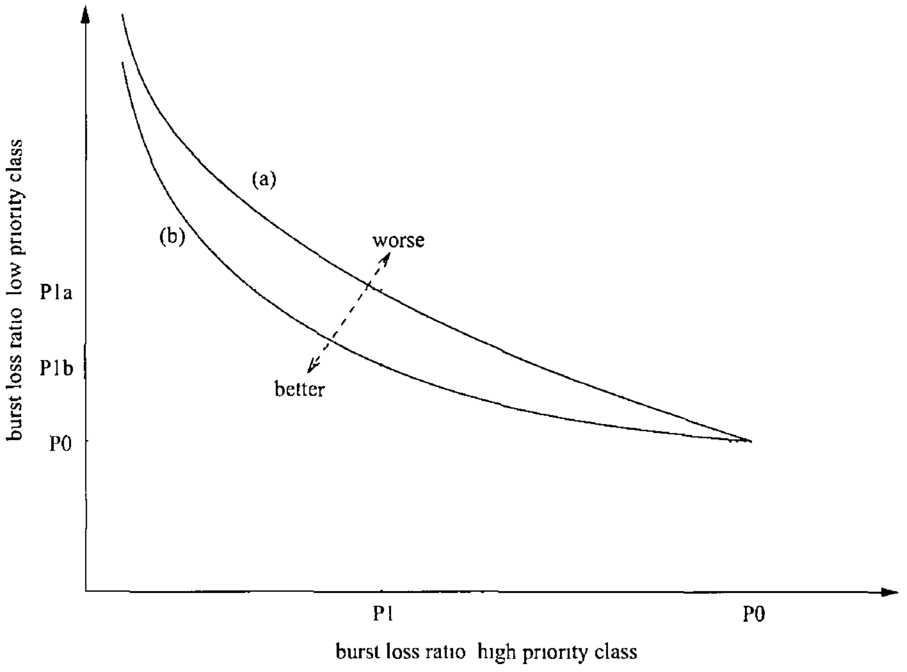

burst loss ratio low priority class
P1a
P1b
P0
(a)
(b)
worse
better
P1
P0
burst loss ratio high priority class

Figure 7 1 Example low prıorıty vs hıgh prıorıty loss ratıo dıagrams

bursts that are more lıkely to cause contentıon than other bursts Therefore, the cost-based algorıthm ıs expected to offer better performance

In thıs sectıon, sımılar curves wıll be presented for the cost-based and random burst droppıng algorıthm used ın the followıng topologıes

- sıngle lınk,
- rıng network,
- random networks

In all cases the cost functıon formula descrıbed ın Chapter 4, Equatıon 4 6 was used The range of dropper thresholds for each sımulatıon was found experımentally

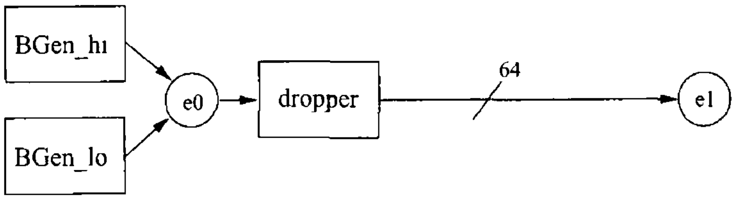


Figure 7 2 Burst dropping in a single link

7.2 Single node

The topology of the network used in this section is presented in Fig 7 2 There were two edge nodes, $e0$ and $e1$, connected with a single, 64-channel, unidirectional link There were two burst generators in node $e0$, one for the high priority and one for the low priority traffic The high priority traffic constituted 20% of the total traffic

A certain proportion of the low priority traffic was discarded in the dropper before being sent The dropper used either random or the cost-based algorithm For each link load and dropping algorithm a number of simulation rounds was run, using different dropper thresholds Then, the total loss ratios (i e including both the intentionally dropped bursts and those lost due to contention) in both classes were plotted against each other

7 2.1 Basic simulation

A set of simulations was performed according to the scenario described in the previous section The results are presented in Fig 7 3-7 5

As predicted, there is always a point where the loss ratios in both classes are identical There is no difference between the two algorithms then, as no bursts are intentionally dropped at all However, when the dropper threshold is changed so that a certain proportion of low priority bursts are discarded, the reduction in high priority loss ratio differs between the two algorithms

The cost-based algorithm consistently offers lower loss ratio in the high prior-

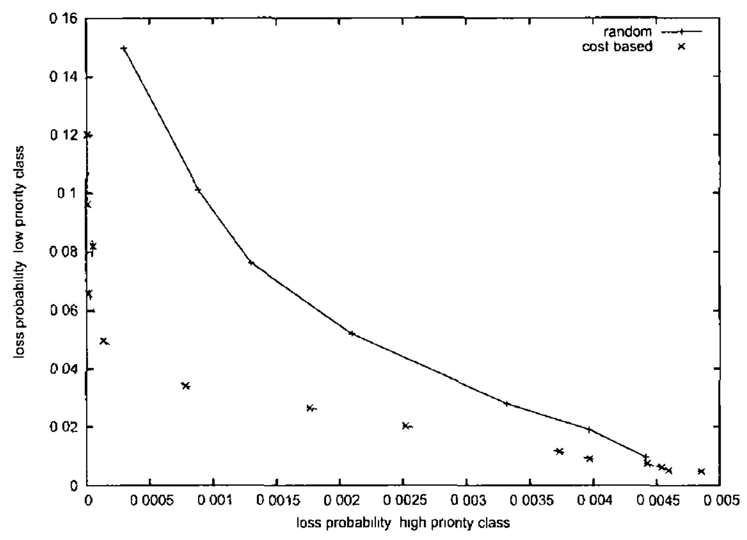


Figure 7 3 Low priority vs high priority loss ratio, 0 7 link load

ity class' for an identical proportion of discarded low priority bursts, or requires lower additional burst loss in the low priority class to achieve a given loss ratio in the high priority class

For example, let's assume that the link load is 0 7, as presented in Fig 7 3 and the loss ratio in the low priority class is increased to 4% The loss ratio in the high priority class offered by the cost-based algorithm will be **less than 50%** of what it would be is the random algorithm was used

7 2.2 Effect of traffic type

Most of the simulations described in this chapter use traffic sources with exponentially distributed assembly time and constant burst length However, this traffic does not have the properties of assembled IP traffic that will be carried in OBS networks To test the performance of the cost-based algorithm with more realistic traffic types, two other traffic sources were used

A burst generator using exponentally distributed burst length and assembly time and the realistic traffic source described in Section 6 2 2 were used to test the influence of the variable burst length on the cost-based algorithm Fig 7 6 and

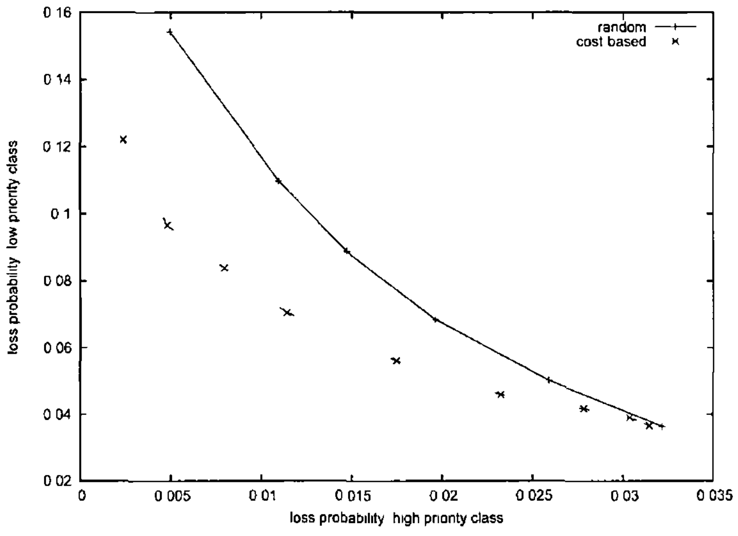


Figure 7 4 Low priority vs high priority loss ratio, 0 8 link load

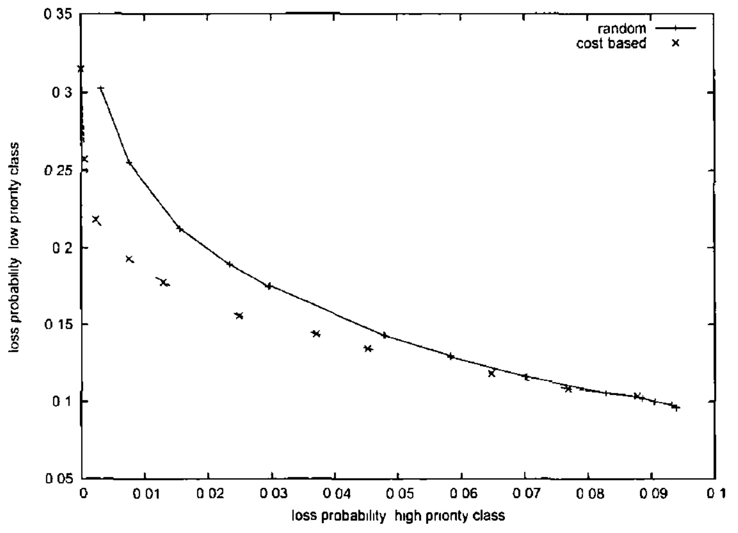


Figure 7 5 Low priority vs high priority loss ratio, 0 9 link load

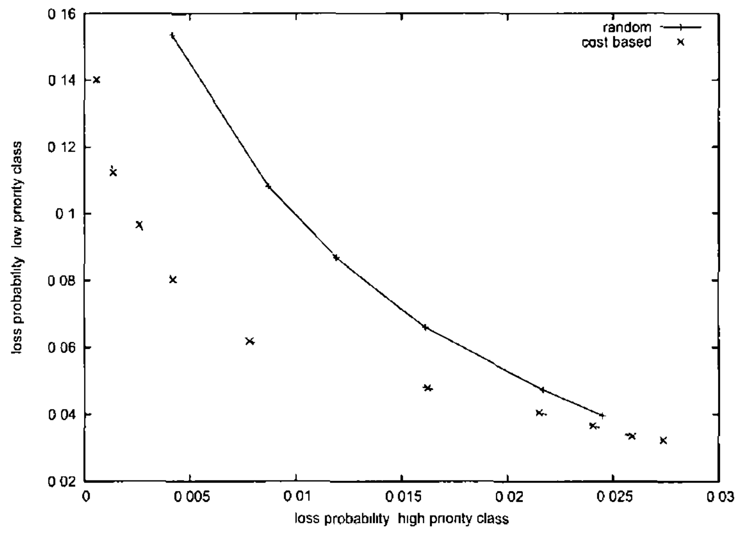


Fıgure 7 6 Low prıorıty vs hıgh prıorıty loss ratıo, 0 8 lınk load, varıable burst length

7 7 present the results of thıs sımulatıons

The dıfference ın performance of both algorıthms ıs clearly vısıble Therefore, ıt can be concluded that the cost-based burst droppıng algorıthm does not dramatıcally change wıth the type of traffıc

7 2 3 Effect of offset time range

The cost-based burst droppıng algorıthm depends on the value of the cost functıon for a gıven burst to decıde whether ıt should be dropped or not For thıs mechanısm to functıon properly, the values of the cost functıon for dıfferent bursts have to be dırectly comparable Thıs ıs lıkely to be the truth, when the offset tıme dıfference between two bursts ıs relatıvely small However, when the range of possıble offset tımes ıncreases, then the values of the cost functıon for bursts at the opposıte ends of the range become dıffıcult to compare, for reasons descrıbed ın Chapter 4

Fıg 7 8-7 10 and Fıg 7 4 compare the performance of the cost-based and random burst droppıng algorıthm It can be seen that as the offset tıme range ın-

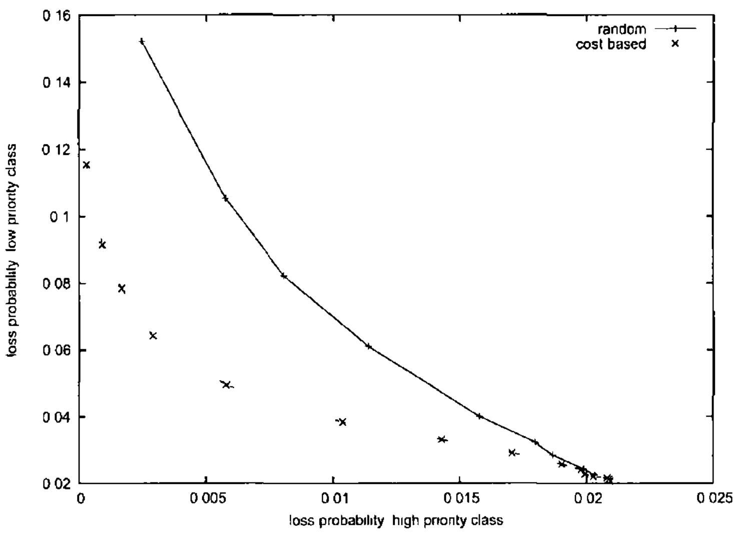


Fıgure 7 7 Low prıorıty vs hıgh prıorıty loss ratıo, 0 8 lınk load, assembled traffıc

creases, the two curves became closer to each other, and the ımprovement offered by the cost-based algorıthm decreases As expected, the cost-based algorıthm offers the best performance when the offset tıme range ıs relatıvely small Thıs effect ıs sımılar ın prıncıple to the one descrıbed ın Sectıon 6 2 4

7.3 Network examples

In thıs sectıon, the cost-based burst droppıng algorıthm ıs tested ın a rıng network, and ın three random networks

7 3.1 Rıng network

A sıx-node rıng network, ıdentıcal to the one presented ın Fıg 6 34 was sımulated next There were 64 channels ın each lınk Each node sent an equal amount of traffıc, consıstıng of 20% of hıgh prıorıty traffıc and 80% of low prıorıty traffıc to each of the other fıve nodes The burst length was constant and the assembly tıme was exponentıally dıstrıbuted, the offset tıme was unıformly dıstrıbuted over a

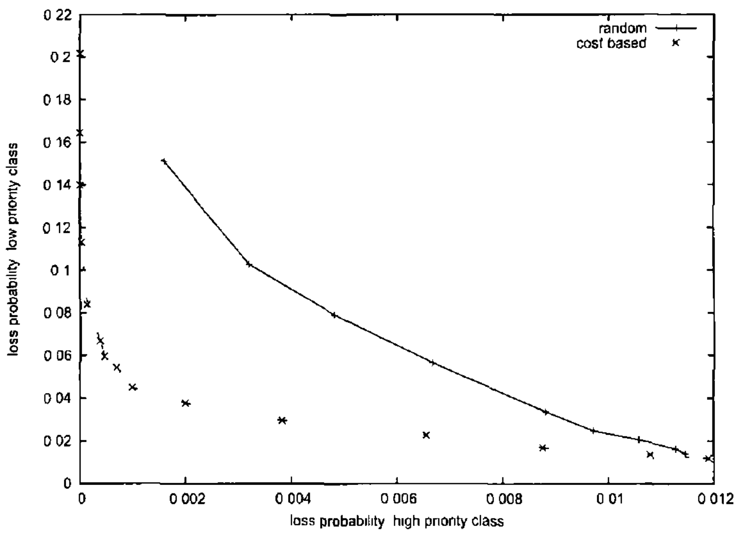


Figure 7 8 Low priority vs high priority loss ratio, 0 8 link load, offset time range 1 burst length

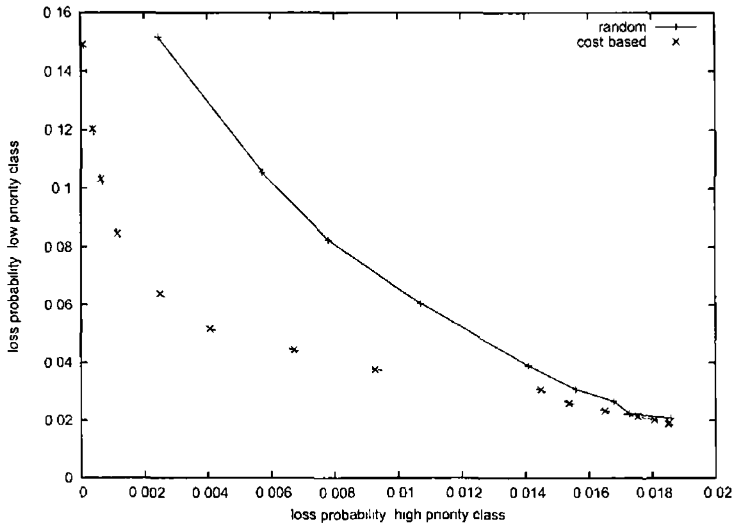


Figure 7 9 Low priority vs high priority loss ratio, 0 8 link load, offset time range 5 burst lengths

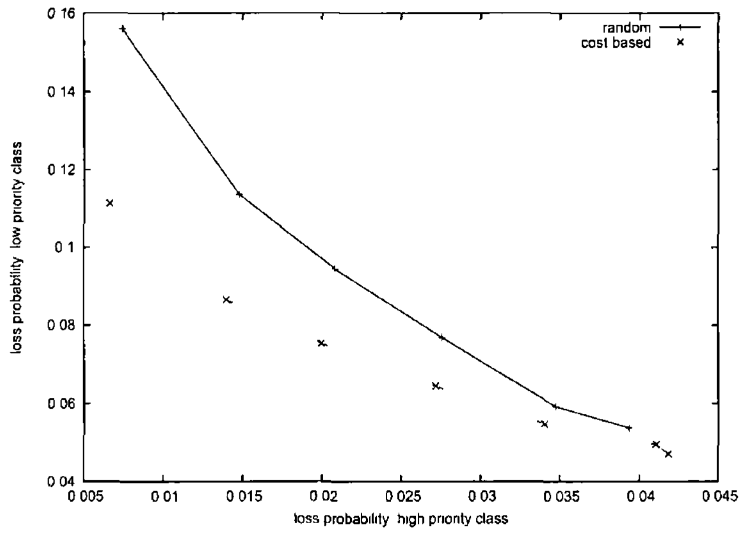


Figure 7 10 Low priority vs high priority loss ratio, 0 8 link load, offset time range 15 burst lengths

range of 10 burst lengths In each link, there was a burst dropper, using either random or the cost-based algorithm Only the low priority traffic was affected

The total amount of traffic sent and traffic received in each class was measured Traffic loss, calculated using those values, was used as a network performance metric The results are presented in Fig 7 11-7 12 Again, dropping the same amount of low priority traffic offers lower loss in the high priority class when the cost-based algorithm is used

7 3 2 Random networks

The GT-ITN tool [99] was used to generate three random network topologies The generator parameters were as follows one transit domain, three transit nodes and three stub nodes per transit node

In each edge node, there was a set of burst generators, two for each other edge node, and a burst dropper The burst length was constant and assembly time was exponentially distributed 20% of the traffic sent to any node belonged to the high priority class The offset time was uniformly distributed over a range of 10 burst

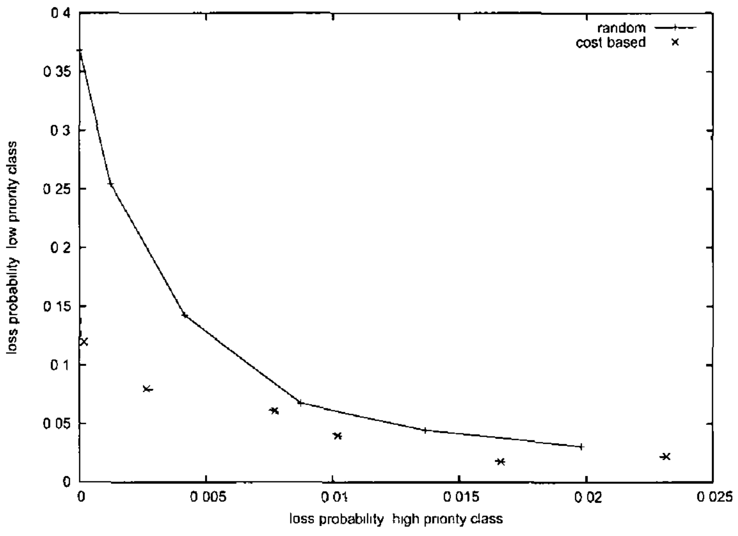


Figure 7 11 Low priority vs high priority loss ratio, 0 7 link load, ring network

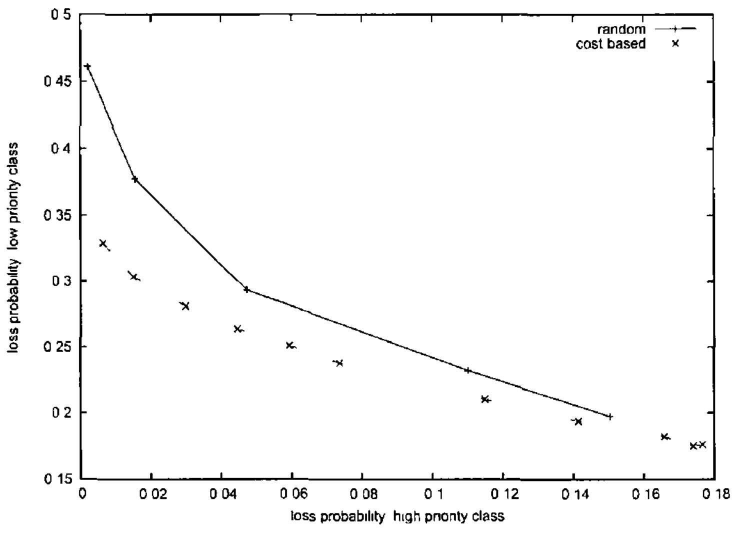


Figure 7 12 Low priority vs high priority loss ratio, 0 9 link load, ring network

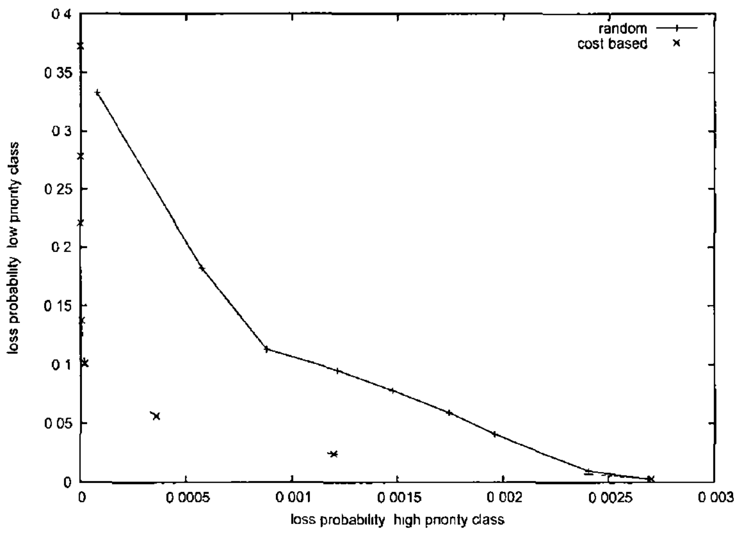


Figure 7 13 Low priority vs high priority loss ratio, random network 0

lengths The processing delay at each node was equal to 0 33 of burst length There were 64 channels per link

The total amount of traffic sent and received in each class was recorded Later, those values were used to calculate the low and high priority traffic loss The results are presented in Fig 7 13-7 15 Again, dropping a certain amount of low priority traffic results in greater improvement for the high priority class when the cost-based algorithm is used

It has to be noted that the total traffic loss ratio provided in the ring and random network simulations does not fully reflect the performance of the network The loss ratio is, in effect, averaged across all data streams, and it is likely that different streams experience different losses In particular, the loss experienced by a data stream in the presented simulations will depend on the number of hops that its bursts traverse It would be possible to use individual droppers for each stream, and individually adjust the dropper's thresholds to optimize the loss experienced by a stream in individual links In particular, it would probably be possible to increase the network performance by making the loss probability a function of the number of traversed hops, extending the idea presented by White

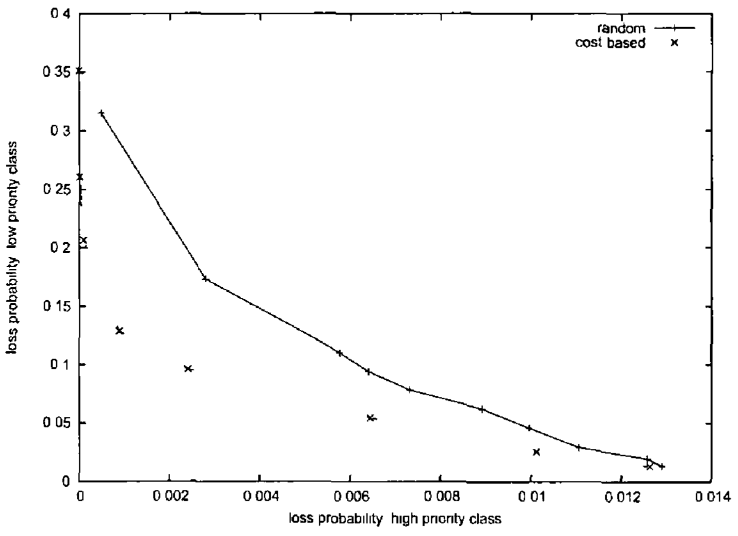


Figure 7.14 Low priority vs high priority loss ratio, random network 1

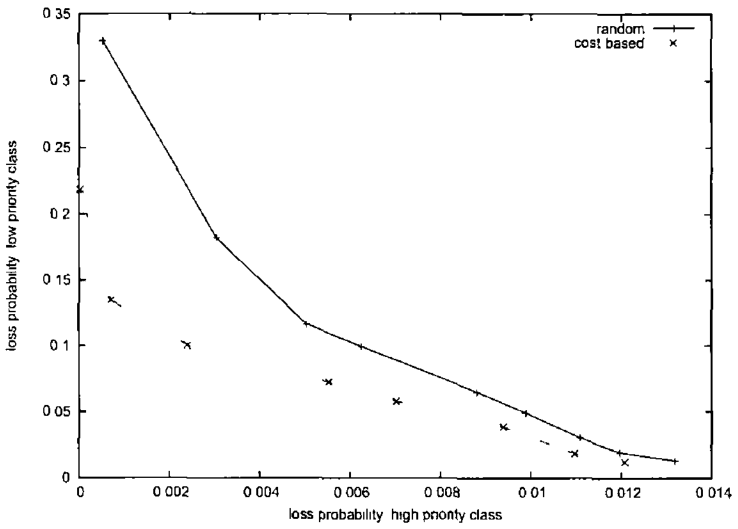


Figure 7.15 Low priority vs high priority loss ratio, random network 2

et al in [100] This, however, is outside the scope of this thesis

7.4 Summary

The simulations presented in this chapter compared the performance of the cost-based and random burst dropping algorithms in several network scenarios The cost-based algorithm consistently outperformed the random algorithm, either offering lower loss in the high priority class for a given loss in the low priority class, or requiring dropping smaller proportion of the low-priority traffic to achieve a given loss ratio in the high priority class For example, in the situation presented in Fig 7 8, decreasing the burst loss probability in the high priority class to 50% of its original value (i e to 0 6%) would require dropping approximately 2 5% of the low priority traffic when the cost-based algorithm is used, and 6 5% when burst are dropped randomly

The following observations were made

- The cost-based burst dropping algorithm offers improved performance, regardless of the traffic type

- The cost-based burst dropping algorithm offers the best performance when the range of possible offset times is relatively small

Chapter 8
Conclusions

Optical Burst Switching (OBS) is a promising new paradigm, filling the gap between optical circuit switching (OCS) and optical packet switching (OPS). A burst-switched network can fully utilize the enormous bandwidth of an optical fibre, while offering higher granularity than OCS networks. At the same time, the hardware requirements of an OBS network are lower than in the case of OPS. This makes them an attractive choice for the future Internet.

In Optical Burst-Switched networks reservations are made in advance. This makes the alignment of bursts within channels a very important issue. Depending on this alignment, the available bandwidth may become more or less fragmented, resulting in higher or lower loss probability for future bursts. In this thesis a way of judging this alignment in a numerical way is presented, along with its possible applications.

A cost function value can be used to predict the probability that allocating a burst will cause a future burst to be dropped due to a contention. This thesis provided a formal definition of a cost function. Then a theoretical analysis of the probability of a given burst causing a contention was presented, and a cost function formula based on it was proposed.

The idea of a cost function was used to design a scheduling algorithm. Scheduling a burst requires using some of the resources available in the node. Those resources may include outgoing links, fiber delay lines or wavelength converters. By assigning a virtual price to each of those resources a lowest-priced combination can be then chosen.

The simulation results presented in this thesis were obtained using an OBS extension to the well-known *ns-2* simulator The robust and efficient underlying *ns-2* simulation engine coupled with an extensive set of OBS classes provided a powerful simulation tool, capable of simulating an OBS network as a part of IP network, or on its own, for better efficiency

Presented simulation results prove that the cost-based algorithm can offer improved performance when compared with LAUC-VF Depending on the simulation parameters, the difference can reach an order of magnitude, especially for low loads In the situation presented in Fig 6 19, when the link load is equal to 0 5, the loss probability offered by the cost-based algorithm is over ten times smaller than that of LAUC-VF (0 0021% vs 0 0332%) In some cases the performance of both algorithms becomes similar, but the cost-based algorithm is never worse

The performance of the cost-based scheduling algorithm as a function of FDL length was studied next It was found that the performance of both LAUC-VF and the cost-based algorithm improves as a function of FDL length, and converges for very long FDLs However, the difference is very significant for shorter delay lines As demonstrated in Fig 6 30, the cost-based algorithms using a relatively short FDL (one-tenth of the burst length) outperforms LAUC-VF with ten times longer FDL

Later, it was demonstrated that by assigning an appropriate initial FDL price, FDL banks of varying sizes can be effectively utilized by a node This was done first by varying the size of an FDL bank in a single node or a ring network In both cases, the performance of LAUC-VF improved only until the size of the FDL bank reached approximately 5 channels and then levelled off The cost-based algorithm, on the other hand, was capable of utilizing additional FDL channels, For a large FDL bank, the loss ratio of LAUC-VF was approximately twice as big as that of the cost-based algorithm, as shown in Fig 6 31 (1 3% vs 0 7%) and 6 35 (2% vs 1 1%)

Then, FDL banks of identical size were used in two larger networks The first topology was based on the formes National Science Foundation network, and the other one on the pan-European network defined in the COST 266 project In both cases, by adjusting the FDL price in each node individually, it was possible to achieve lower burst loss ratio than in the case of LAUC-VF, especially for low loads For example, the overall loss ratio in the pan-European network for a load of 0 5 times the base load, was more than six times larger when LAUC-VF was used The difference decreased for higher loads, likely due to overloading of a few links in the network

Several Quality of Service (QoS) schemes use the idea of early or intentional burst dropping in low priority classes to reduce loss probability experienced by high priority bursts A cost-based burst dropping algorithm that only drops bursts when the value of the cost function exceeds a certain threshold was proposed As the cost function value reflects the probability that a given burst will cause a contention, this algorithm offers better performance than a random one Simulation results were presented for several scenarios

First, a single-link topology was used to investigate the effects of traffic type and offset time range on the performance of the cost-based burst dropping algorithm It was found that the impact of the traffic type was not very significant, and that the performance of the cost-based algorithm improved when the range of possible offset times was relatively small For example, when the range of possible offset times was equal to one burst length, increasing the burst loss ratio in the low priority class from original 1 2% to 4% resulted in approximately 0 8% loss ratio in the high priority class when bursts were dropped randomly, and less than 0 2% when the cost-based burst dropping algorithm was used

Then, the cost-based burst dropping algorithm was tested in a ring network and three random networks Again, its performance was better than in the case of random burst dropping In one of the random networks, decreasing the burst loss in the high priority class to nearly zero required dropping approximately 10% of low priority bursts using the cost-based burst dropping algorithm To achieve the same effect by random burst droppmg it was necessary to drop 35% of low priority bursts (Fig 7 13)

In conclusion, the cost-based algorithms offer improved performance compared to standard solutions Depending on the network parameters, the difference may be very significant, and the cost-based algorithms are never worse, even in unfavorable conditions

8.1 Future work

The proposed cost function formulas only consider the alignment of bursts within one channel In particular, the state of other channels and burst destinations are not taken into account Using this information it is reasonable to expected a significant performance benefit, as the resulting algorithm could tend to group bursts with the same destination address m one channel, or reduce their overlap It would be particularily interesting to investigate the performance of such an algorithm in a non-void filling system

All the simulations presented m this thesis used fixed-length FDLs It may be possible to use several FDLs of varying length, or to use continuous delays (at the network edge) In the latter case, the cost of delaying a burst would probably depend on the amount of memory used, and the duration of its use

Finally, the idea of cost-based scheduling can be used in any system, where resources have to be reserved in advance Systems where reservations can be delayed at a certain cost are particularily suitable For example, if a large amount of experimental data has to be processed, the data can be stored, but the available disk space may be limited A cost-based algorithm could balance the need to use the CPU time in an efficient manner and to minimize delay

Bibliography

[1] L G Roberts Beyond Moore's law Internet growth trends *IEEE Computer, vol 33, no 1, p 117,* 2000

[2] Scıentısts at Lucent Technologıes' Bell Labs calculate theoretıcal lımıts of fıber optıc communıcatıons, http //www lucent com/press/0601/010628 bla html, retrıeved on 02 Dec 2005

[3] T Battestıllı and H Perros Optıcal Burst Swıtchıng A Survey Technıcal report, Technıcal report TR-2002-10, NC State Unıversıty, Computer Scıence Department, July 2002

[4] Generalızed Multı-Protocol Label Swıtchıng (GMPLS) Archıtecture *rfc3945*

[5] Generalızed Multı-Protocol Label Swıtchıng (GMPLS) Sıgnalıng Functıonal Descrıptıon *rfc3471*

[6] Wıll E Leland and Murad S Taqq and Walter Wıllınger and Danıel V Wılson On the self-sımılar nature of Ethernet traffıc In *Proceedıngs of ACM SIGCOMM, San Francısco, Calıfornıa, 1993*

[7] J Beran and R Sherman and M S Taqqu and W Wıllınger Long-Range Dependence ın Varıable-Bıt-Rate Vıdeo Traffıc *IEEE Transactıons on Communıcatıons, vol 43, no 2/3/4, pp 1566-1579, 1995*

[8] L Xu, H Perros, and G Rouskas Techniques for Optical Packet Switching and Optical Burst Switching *IEEE Communications, Jan 2001*

[9] C Qiao and M Yoo Optical burst switching (OBS) - a new paradigm for an Optical Internet *Journal of High Speed Networks, vol 8, no 1, pp 69-84, 1999*

[10] T Battestilli and H Perros An introduction to optical burst switching, *IEEE Communications Optical Magazine, S10S15, Vol 41, 2003*

[11] S R Amstutz Burst switching–An introduction *IEEE Communications Magazine, vol 21, no 8, pp 36-42, 1983*

[12] S R Amstutz Burst switching–An update *IEEE Communications Magazine, vol 27, no 6, pp 50-57, 1989*

[13] C Gauger, K Dolzer, J Spath, S Bodamer Service Differentiation in Optical Burst Switching Networks *ITG Fachtagung Photonische Netze pp 124-132,* 2001

[14] E A Varvarigos and V Sharma The Ready-to-Go Virtual Circuit Protocol A Loss-Free Protocol for Multigigabit Networks Using FIFO Buffers *IEEE/ACM Transactions on Networking, vol 5, no 5, October 1997*

[15] I Widjaja Performance Analysis of Burst Admission Control Protocols *IEEE Proceedings on Communications, vol 142, no 1, pp 7-14, Feb 1995*

[16] Mills, D L, C G Boncelet, J G Elias, P A Schragger, A W Jackson and A Thyagarajan Final report on the Highball Project In *Electrical Engineeing Department Report 95-4-1, University of Delaware, April 1995*

[17] J Y Wei and R I McFarland Jr Just-in-time signaling for WDM optical burst switching networks *IEEE Journal of Lightwave Technology, vol 18, pp 2091*

[18] The JumpStart project http //jumpstart anr mcnc org

[19] I Baldine, H Perros, G Rouskas and D S Stevenson JumpStart A Just-in-Time Signaling Architecture for WDM Burst-Switched Networks In *IEEE Communications, 40(2), pp 82-89, 2002*

[20] J S Turner Terabit burst switching *Journal of High Speed Networks, vol 8, pp 3-16, 1999*

[21] J Turner Terabit burst switching progress report (9/98-12/98) Technical report, Washington University at St Louis Technical Report, 1998

[22] J Teng, G Rouskas A Comparison of the JIT, JET, and Horizon Wavelength Reservation schemes on a single OBS node In *The First International Workshop on Optical Burst Switching, Dallas, Texas, Oct 2003*

[23] M Yoo and C Qiao Just-enough-time(JET) a high speed protocol for bursty traffic in optical networks *IEEE/LEOS Technologies for a Global Information Infrastructure*, 1997

[24] K Dolzer and C Gauger and J Spth and S Bodamer Evaluation of Reservation Mechanisms for Optical Burst Switching

[25] J Teng, G Rouskas A Detailed Analysis and Performance Comparison of Wavelength Reservation Schemes for Optical Burst Switched Networks *Photonic Network Communications, vol 9, no 3, pp 311-335, May 2005*

[26] A Ge, F Callegati, L S Tamil On optical burst switching and self-similar traffic *IEEE Communications Letters, vol 4 no 3*, 2000

[27] G Hu, K Dolzer, C M Gauger Does burst assembly really reduce the self-similarity? In *Proceedings of Optical Fiber Communication Conference (OFC 2003)*

[28] X Yu, Y Chen and C Qiao A Study of Traffic Statistics of Assembled Burst Traffic in Optical Burst Switched Networks In *Proceedings of Opticomm 2002*

[29] X Cao and J Li and Y Chen and C Qiao Assembling TCP/IP Packets in Optical Burst Switched Networks In *Proceeding of IEEE Globecom, 2002*

[30] T Tachibana, T Ajima and S Kasahara Round-Robin Burst Assembly and Constant Transmission Scheduling for Optical Burst Switching Networks In *Proceedings of IEEE GLOBECOM 2003*

[31] Klaus Dolzer Assured Horizon - A new Combined Framework for Burst Assembly and Reservation in Optical Burst Switched Networks In *Proceedings of the European Conference on Networks and Optical Communications (NOC 2002)*

[32] V M Vokkarane, K Haridoss, and J P Jue Threshold-Based Burst Assembly Policies for QoS Support in Optical Burst-Switched Networks In *Proceedings of SPIE OptiComm 2002*

[33] S Oh, H Ha Hong, M Kang A Data Burst Assembly Algorithm in Optical Burst Switching Networks *ETRI Journal, vol 24, no 4 pp 311-322, Aug 2002*

[34] V Vokkarane and Q Zhang and J Jue and B Chen Generalized Burst Assembly and Scheduling Techniques for QoS Support in Optical Burst-Switched Networks In *Proceedings of IEEE Globecom 2002*

[35] T Hashiguchi, X Wang, H Morikawa and T Aoyama Burst Assembly Mechanism with Delay Reduction for OBS Networks In *Proceedings of COIN/ACOFT 2003*

[36] S A N Ostring and H Sirisena The influence of long-range dependence on traffic prediction In *Proceedings of ICC 2001*

[37] A Sang and San-qi Li A predictability analysis of network traffic In *Proceedings of INFOCOM 2000*

[38] A Adas Supporting real time VBR video using dynamic reservation based on linear prediction In *Proceedings of IEEE INFOCOM 96*

[39] D Morato, J Aracil, L Angel Diez, M Izal and Eduardo Magana On linear prediction of Internet traffic for packet and burst switching networks In *Proceedings of ICCCN 2001*, October

[40] T W Yeow, E Law and A Goldenberg MEMS Optical Switches *IEEE Communications Magazine, pp 158-163*, 2001

[41] C Guillemot, M Renaud, P Gambini, C Janz, I Andonovic, R Bauknecht, B Bostica, M Burzio, F Callegati, M Casoni, D Chiaroni, F Clerot, S L Danielsen, F Dorgeuille, A Dupas, A Franzen, P B Hansen, D K Hunter, A Kloch, R Krahenbuhl, B Lavigne, A Le Core, C Raffaelli, M Schilling, J Simon and L Zucchelli Transparent Optical Packet Switching The European ACTS KEOPS Project Approach *Journal of Lightwave Technology, vol 16, no 12, pp 2117-2134, Dec 1998*

[42] D Chiaroni, B Lavigne, A Jourdan, M Sotom, L Hamon, C Chauzat, J Jacquinot, A Barosso, T Zami, F Dorgeuille, C Janz, J Emery, E Grard and M Renaud Physical and Logical Validation of a Network Basen on All-Optical Packet Switching Systems *Journal of Lightwave Technology, vol 16, no 12, pp 2255-2264, Dec 1998*

[43] H Buchta, E Patzak, J Saniter, C Gauger Maximal and Effective Throughput of Optical Switching Nodes for Optical Burst Switching In *Beitrge zur 4 ITG-Fachtagung Photonische Netze, 2003*

[44] M Renaud, M Bachmann, M Erman Semiconductor optical space switches *IEEE Journal of Selected Topics in Quantum Electronics, Vol 2, No 2, June 1996, pp 277-288*

[45] M J Potasek All-Optical Switching for High Bandwidth Optical Network *Optical Networks Magazine, Vol 3, No 6, November/December 2002, pp 30- 43*

[46] H Feng, E Patzak, J Saniter Size and Cascadability Limits of SOA based Burst Switching Nodes In *Proc ECOC 2002, September 2002, Copenhagen, Denmark*

[47] Kapil V Shrikhande, I M White, M S Rogge, F-T An, A Srivatsa, E S Hu, S S-H Yam and Leonid G Kazovsky Performance Demonstration of a Fast-Tunable Transmitter and Burst-Mode Packet Receiver for HORNET In *Proc OFC 2001, Paper ThG2*

[48] M Kauer, M Girault, J Leuthold, J Honthaas, O Pellegri, C Goullancourt, M Zirngibl 16-Channel Digitally Tunable Packet Switching Transmitter with Sub-Nanosecond Switching Time In *IEEE Photonics Technology Letters, vol 15, no 3, March 2003*

[49] J E Simsarian, A Bhardwaj, K Dreyer, J Gripp, O Laznicka, K Sherman, Y Su, C Webb, L Zhang and M Zirngibl A Widely Tunable Laser Transmitter with Fast, Accurate Switching Between All Channel Combinations In *Proc ECOC 2002, September 2002, Copenhagen, Denmark*

[50] T Durhuus, B Mikkelsen, C Joergensen, S L Danielsen, and K E Stubkjaer All-Optical Wavelength Conversion by Semiconductor Optical Amplifiers *IEEE Journal of Lightwave Technology, Vol 14, No 6, June 1996, pp 942-954*

[51] C Dragone An n n optical multiplexor using a planar arrangment of two star couplers *IEEE Photonic Technology Letters, vol 6, pp 812815, 1991*

[52] J Ramamirtham, J Turner and J Friedman Design of Wavelength Converting Switches for Optical Burst Switching *IEEE Journal on Selected Areas In Communications, vol 21, No 7, pp 1122-1132, September 2003*

[53] H Q Ngo, D Pan and C Qiao Nonblocking WDM Switches Based on Arrayed Waveguide Grating and Limited Wavelength Conversion In *Proceedings of INFOCOM 2004*

[54] J Ramamirtham, J Turner Time Sliced Optical Burst Switching In *Proceedings of INFOCOM 2003*

[55] I Baldine, M Cassada, A Bragg, G Karmous-Edwards, and D Stevenson Just-in-time optical burst switching implementation in the ATDnet all-optical networking testbed In *In Proceedings of Globecom 2003*

[56] http //www atd net

[57] K Kitayama, M Koga, H Morikawa , S Hara, M Kawai Optical Burst Switching Network Testbed in Japan In *Proc OFC2005, OWC3, Anaheim, USA, March 2005*

[58] Y Sun, T Hashiguchi, N Yoshida, X Wang, H Morikawa, and T Aoyama Architecture and Design Issues of an Optical Burst Switched Network Testbed In *In Proceedings of 9th OptoElectronics and Communications Conference/3rd International Conference on Optical Internet (OECC/COIN2004), pp 386-387, Yokohama, Japan, July 2004*

[59] B Klusek, D Nowak, and T Curran Wavelength Assignment Algorithms in an OBS Node In *Proceedings of IEI/IEE Telecommunications System Research Symposium, Dublin, May 2003*

[60] YX Vandenhoute, M Cankaya, H C Control architecture in optical burst-switched WDM networks *Selected Areas in Communications, IEEE Journal on Optical Networks, vol 18, pp 1838-1851, Oct 2000*

[61] J Xu, C Qiao, J Li and G Xu Efficient Burst Schedulmg Algorithms in Optical Burst-Switched Networks Using Geometric Techniques *IEEE Journal on Selected Areas in Communications, vol 22, no 9, pp 1796-1811, Nov 2004*

[62] J Li, C Qiao Schedule Burst Proactively for Optical Burst Switching Networks In *Proceedings of Globecom 2003*

[63] X Wang, H Morikawa and T Aoyama Priority-based Wavelength Assignment Algorithm for Burst Switched WDM Optical Networks *IEICE Trans Communications, vol e86-b, no 5, pp 1508-1514, May 2003*

[64] W Tan, YH Pan, D Xu, S Wang, L Li, Z Zhang A QoS-based batch scheduling algorithm in optical burst switching WDM networks In *In Proceedings of International Conference on Communications, Circuits and Systems, 2004 ICCCAS 2004 2004*

[65] S Charcranoon, T S El-Bawab, H C Cankaya, and Jong-Dug Shin Group-Scheduling for Optical Burst Switched (OBS) Networks In *Proceedings of IEEE GLOBECOM 2003*

[66] M H Phung, K C Chua, G Mohan, M Motani, T C Wong and P Y Kong On ordered scheduling for optical burst switching *Computer Networks The International Journal of Computer and Telecommunications Networking, vol 48, pp 891-909, August 2005*

[67] S K Tan, G Mohan and K C Chua Algorithms for burst rescheduling in WDM optical burst switching networks *Computer Networks The International Journal of Computer and Telecommunications Networking, vol 41, pp 41-55, 2003*

[68] M Yoo and C Qiao A new optical burst switching protocol for supporting quality of service In *SPIE Proceedings, All Optical Networking Architecture, Control and Management Issues, vol 3531, pp 396–405, Nov 1998*

[69] Z Rosberg, H Vu, M Zukerman and J White Blocking Probabilities of Optical Burst Switching Networks Based on Reduced Load Fixed Point Approximations In *Proceedings of IEEE Infocom 2003*

[70] M Yoo and C Qiao Supporting multiple classes of services in IP over WDM networks In *Proceedings of IEEE Globecom, pp 1023–1027, 1999*

[71] W-H So, H-C Lee, Y-C Kim and S-S Roh QoS Supporting Algorithms for Optical Internet Based on Optical Burst Switching *Photonic Network Communications, vol 5 no 2, pp 147-162, 2003*

[72] N Barakat, E H Sargent Analytical Modeling of Offset-Induced Priority in Multiclass OBS Networks *IEEE TRANSACTIONS ON COMMUNICATIONS, 2005*

[73] V M Vokkarane and J P Jue Prioritized Burst Segmentation and Composite Burst-Assembly Techniques for QoS Support in Optical Burst-Switched Networks *IEEE Journal on Selected Areas in Communications, vol 21, no 7, pp 1198-1209, Sep 2003*

[74] Y Arakawa, M Sakuta and I Sasase QoS scheme with Burst Dropping in Optical Burst Switching In *Proceedings of IEEE Pacific Rim Conference on Communications, Computers and Signal processing (PACRIM' 03)*

[75] Y Chen, M Hamdi, and D H K Tsang Providing Proportionally Differentiated Services over Optical Burst Switching Networks In *In Proceedings of IEEE GLOBECOM 2001, vol 3, pp 1510 -1514*

[76] Q Zhang, V M Vokkarane, B Chen, and J P Jue Early Drop Scheme for Providing Absolute QoS Differentiation in Optical Burst-Switched Networks In *Proceedings, IEEE HPSR 2003, Torino, Italy*

[77] Q Zhang, V M Vokkarane, B Chen and J P Jue Early drop and wavelength grouping schemes for providing absolute QoS differentiation in optical burst-switched networks In *Proc of IEEE Global Telecommunication Conference (2003)*

[78] Q Zhang, V M Vokkarane, J P Jue and B Chen Absolute QoS differentiation in Optical Burst-Switched Networks *IEEE Journal on Selected Areas in Communications, vol 22, no 9, pp 1781-1795, Nov 2004*

[79] B Zhou, M Bassiouni, and G Li Improving fairness in optical-burst-switching networks *Journal of Optical Networking, Optical Society of America,* 2004

[80] Klaus Dolzer *Mechanisms for Quality of Service Differetiation in Optical Burst Switched Networks* PhD thesis, 2003

[81] J Wan, Y Zhou, X Sun and M Zhang Guaranteeing quality of service in optical burst switching networks based on dynamic wavelength routing *Optics Communications, 220(1-3) 85-95, 2003*

[82] W Liao and C -H Loi Providing Service Differentiation for Optical-Burst-Switched Networks *Journal of Lightwave Technology, vol 22, pp 1651-1660, 2004*

[83] L Yang, Y Jiang and S Jiang A Probabilistic Preemptive Scheme for Providing Service Differentiation in OBS Networks In *Proceedings of IEEE GLOBECOM 2003*

[84] T Tachibana, M Ueda, and S Kasahara A Preemptive Scheme with Two-Way Release Message Transmission in Optical Burst Switching Networks In *Proceedings of IEEE GLOBECOM 2004*

[85] H C Cankaya, S Charcranoon and T S El-Bawab A Preemptive Scheduling Technique for OBS Networks with Service Differentiation In *Proceedings of IEEE GLOBECOM 2003*

[86] S Blake, D Black and M Carlson An Architecture for Differentiated Services *IETF RFC 2475, Oct 1998*

[87] K Long, R S Tucker and C Wang A New Framework and Burst Assembly for IP DiffServ over Optical Burst Switching Networks In *Proceedings of IEEE GLOBECOM 2003*

[88] H L Vu and M Zukerman Blocking probability for priority classes in optical burstswitching networks *IEEE Communications Letters, vol 6 pp 214-216, May 2002*

[89] L Xu, H Perros and G Rouskas A queueing network model of an optical burst switching node with large number of wavelengths *ITC 2003, Elsevier (2003) pp, 891-900*

[90] B Wen, N M Bhide, R K Shenai, and K M Sivalingam Optical Wavelength Division Multiplexing (WDM) Network Simulator (OWns) Architecture and Performance Studies *SPIE Optical Networks Magazine, pp 1626, Sep/Oct* , 2001

[91] S Kim, J S Choi, M Kang Providing absolute differentiated services for optical burst switching networks loss differentiation *IEEE Proceedings Communications, vol 152, pp 439- 446,* 2005

[92] E Kozlovski and P Bayvel QoS Performance of WR-OBS Network Architecture with Request Scheduling In *Proceedings of ONDM 2002, pp 101-116*

[93] L Breslau, D Estrin, K Fall, S Floyd, J Heidemann, A Helmy, P Huang, S McCanne, K Varadhan, Y Xu and H Yu Advances in Network Simulation *IEEE Computer, vol 33, no 5, pp 59-67, May 2000*

[94] J Ousterhout Scripting Higher-level programming for the 21st century *IEEE Computer, vol 31, no 3, pp 23-30, March 1998*

[95] B Klusek, J Murphy and L Barry Traffic sources for Optical Burst Switching simulations In *Proceedings of SPIE OPTO Ireland, Dublin, Apr 2005*

[96] A Betker, C Gerlach, R Hlsermann, M Jger, M Barry, S Bodamer, J Spth, C Gauger and M Kohn Reference Transport Network Scenarios In *5th ITG Workshop on Photonic Networks, Leipzig, 2004*

[97] S Bodamer and J Spth Effizienz von dynamischer optischer Vermittlung in SDH/SONET-WDM-Transportnetzen *ITG-Fachtagung Photonische Netze, pp 41-48,* 2003

[98] S De Maesschalck, D Colle, I Lievens, M Pickavet, P Demeester, C Mauz, M Jaeger, R Inkret, B Mikac and J Derkacz Pan-European Optical Transport Networks an Availability-based Comparison *Photonic Network Communications, vol 5, no 3 pp 203-226,* 2003

[99] E W Zegura, K Calvert and S Bhattacharjee How to Model an Internetwork In *Proceedings of IEEE Infocom '96, San Francisco, CA*

[100] J A White, R S Tucker, and K Long Merit-based Scheduling Algorithm for Optical Burst Switching In *Proceedings of COIN-PS 2002, Cheju Island, Korea, July 2002*

Appendix A

Simulation scripts

In the following sections representative examples of simulation scripts used to obtain results for this thesis are presented

A.1 Cost-based scheduling

The main simulation script

```
#prefix , to be used in all output file names
set infilename [lindex $argv 0]
#FDL price
set price [lindex $argv 1]
#load [0 1]
set load_temp [lindex $argv 2]
#FDL pricing strategy "const" or "linear"
set strategy [lindex $argv 3]
#simulation end time
set endtime [lindex $argv 4]
#scheduling algorithm GapFilling or LAUCVF
set algorithm [lindex $argv 5]

#load per generator
set load [expr 32 0*$load_temp /4 ]

#build log file name, loss vs price simulations
set log $infilename
```

```
append log "_log_"
append log $strategy
append log "_"
append log $load_temp

#build log file name, loss vs load simulations
if {[lindex $argv 6] == "load"} {
set log $infilename
append log "_log_"
append log $strategy
append log "_"
append log $price
append log "_load"
}

#set the pricing strategy parameter
if {$strategy == "const"} {
puts "constant mode"
}

if {$strategy == "linear"} {
Algorithm/GapFilling set linear 1
puts "linear mode"
}

#declarations of constants, to be used later
set linksize 32
set offset 20ms
set max_offset 50ms

#put either current FDL price or load into the logfile depending
```

```
    on the simulation type
set logfile [open $log {WRONLY CREAT APPEND}]
if {[lindex $argv 6] == "load"} {
puts -nonewline $logfile "$load "
} else {
puts -nonewline $logfile "$price "
}
close $logfile


#set the cost-based algorithm's parameters
Algorithm/GapFilling set fdl_price $price
Algorithm/GapFilling set improved 1
Algorithm/GapFilling set min_offset 18ms
Algorithm/GapFilling set max_offset 52ms


#set the FDL delay to 1ms
FDL set delay 1ms

#create the Simulator and OBS objects
set ns [new Simulator]
set ob [new OBS]

#create a BurstLogger object to track the loss ratio in the
   outgoing links
set all_logger [new BurstLogger]

#create an OBS core node
set core [$ob core-node]
```

```
#create a FDL bank
set fdl [new FDL $linksize]
#  and add it to the core node
$core add-fdl $fdl

#create a FDL tracker , and attach it to the FDL bank
set fdlt [new FDLTracker]
$fdlt set-fdl $fdl

#create an array of source and destination nodes
set sources(0) ""
set dests(0) ""
for {set i 0} {$i <4} {incr i} {
set sources($i) [$ns obs-node]
set dsts($i) [$ns obs-node]

$ob add-edge-node $sources($i)
$ob add-edge-node $dsts($i)
}


#create links and attach the logger to each link
for {set i 0} {$i <4} {incr i} {
set slink($i) [$ob optical-link $sources($i) $core $linksize 1ms
   Algorithm/Random]
set dlink($i) [$ob optical-link $core $dsts($i) $linksize 1ms
   Algorithm/$algorithm]
$dlink($i) add-logger $all_logger
}

#compute routing tables and populate routers
```

```
$ob compute-routes
$ob populate-routers



#create burst generators and add each one to the "generators"
   object
set generators [new Generators]
for {set i 0} {$i <4} {incr i} {
for {set j 0} {$j <4} {incr j} {
set generator($i,$j) [new BurstAssembler/Exponential]
$generator($i,$j) set offset $offset
$generator($i,$j) set offset_max $max_offset
$generator($i,$j) set offset_jitter 1
$generator($i,$j) set correlated 0
$generator($i,$j) set const_length 1
$generator($i,$j) set-src-dst $sources($i) $dsts($j)

$generator($i,$j) set avlength 3ms
$generator($i,$j) set load $load
$generators add-generator $generator($i,$j)
}
}



set logtime [expr $endtime +1]

proc save-histograms {} {
global fdlt log  price
set outfile $log
append outfile "_"
append outfile $price
$fdlt hist-to-file $outfile
}

#schedule events
$ns at 0 "$fdlt start"
$ns at 0 "$generators start"
$ns at $endtime "$generators stop"
$ns at $endtime "$fdlt stop"

$ns at $logtime "$all_logger loss-to-file $log"
$ns at $logtime "save-histograms"

#run the simulation
$ns run
```

Wrapper script, calling the main script with the required parameters

```
set loads "0.5 0.75 0.93"
set prices "0 0.0001 0.0003 0.001 0.003 0.01 0.03 0.1 0.3 1 3 10"
set strategies "linear const"



for {set strat 0} {$strat < [llength $strategies]} {incr strat} {
set strategy [lindex $strategies $strat]
set simname "price"
for {set i 0} {$i < [llength $loads]} {incr i} {
for {set j 0} {$j < [llength $prices]} {incr j} {
set load [lindex $loads $i]
set price [lindex $prices $j]
#make simulation time inversely proportional to the load,
#so that accuracy is not lost for low loads
set endtime [expr 0.3/$load]
puts $load
puts $price
puts $endtime
puts ————$strategy
#execute the main simulation script
exec ns cost_scheduling_ng.tcl $simname $price $load $strategy
   $endtime GapFilling
}
}
}
```

A.2 Burst dropping

The main simulation script

```
#load, as a fraction of maximum link load
set load [lindex $argv 0]
#simulation time
set endtime [lindex $argv 1]
#threshold of the burst dropper
BurstDropper set threshold [lindex $argv 2]
#dropping strategy, either "Cost" or "Random"
set strategy [lindex $argv 3]
#logfile name
set logfile [lindex $argv 4]

#create Simulator and OBS objects
set ns [new Simulator]
set ob [new OBS]
```

```
#define maximum offset
set offset_max 40
set alg_offset_max [expr $offset_max +2]
append offset_max "ms"
append alg_offset_max "ms"

#define link size and calculate total load
set linksize 64
set load [expr $load * $linksize]

#calculate low and high probability load
set load_high [expr 1 0 * $load / 5]
set load_low [expr 1 0 * $load * 4/5]

#create nodes
set e0 [$ns obs-node]
set e1 [$ns obs-node]
$ob add-edge-node $e0
$ob add-edge-node $e1

# Burst generators
set bghi [new BurstAssembler/Exponential]
$bghi set offset 10ms
$bghi set offset_max $offset_max
$bghi set offset_jitter 1
$bghi set avlength 3ms
$bghi set const_length 1
$bghi set load $load_high
$bghi set-src-dst $e0 $e1
$bghi set id 1
```

```
#   low priority generator
set bglo [new BurstAssembler/Exponential]
$bglo set offset 10ms
$bglo set offset_max $offset_max
$bglo set offset_jitter 1
$bglo set avlength 3ms
$bglo set const_length 1
$bglo set load $load_low
$bglo set-src-dst $e0 $e1
$bglo set id 2

#add generators to a "Generators" object, for easier management
set generators [new Generators]
$generators add-generator $bghi
$generators add-generator $bglo

#configure the cost-based algorithm
Algorithm/GapFilling set level 30
Algorithm/GapFilling set min_offset 9ms
Algorithm/GapFilling set max_offset $alg_offset_max
Algorithm/GapFilling set debug 0
#important - try to make the cost function independent on the
    offset time
Algorithm/GapFilling set improved 1

#create link
set link [$ob optical-link $e0 $e1 $linksize 1ms "Algorithm/
    GapFilling"]

#create and configure dropper
set dropper [new BurstDropper/$strategy]
```

```
$dropper set-id 2
$link add-dropper $dropper

#create loggers for outgoing traffic
set logger_lo_out [new BurstLogger/Traffic]
$bglo add-logger $logger_lo_out
$logger_lo_out set-id 2

set logger_hi_out [new BurstLogger/Traffic]
$bghi add-logger $logger_hi_out
$logger_hi_out set-id 1

#create loggers for incoming traffic
set logger_lo_in [new BurstLogger/Traffic]
$e1 add-logger $logger_lo_in
$logger_lo_in set-id 2

set logger_hi_in [new BurstLogger/Traffic]
$e1 add-logger $logger_hi_in
$logger_hi_in set-id 1

#calculate routes and populate routers
$ob compute-routes
$ob populate-routers

set logtime [expr $endtime + 1]

proc newline { } {
#procedure puts a newline symbol into the logfile
        global logfile
        set log [open $logfile {WRONLY CREAT APPEND}]
        puts $log ""
        close $log
}

#schedule events
$ns at 0.0001 "$generators start"
$ns at $endtime "$generators stop"
$ns at $logtime "$logger_hi_out traffic-to-file $logfile"
$ns at $logtime "$logger_hi_in traffic-to-file $logfile"
$ns at $logtime "$logger_lo_out traffic-to-file $logfile"
$ns at $logtime "$logger_lo_in traffic-to-file $logfile"
$ns at $logtime "newline"

#start the simulation
$ns run
```

Wrapper script, calling the main script with the required parameters

```
#cost-based dropping

#dropper thresholds
set thresholds "0.3 0.5 0.7 1 1.5 2 2.3 2.6 3 3.3 3.6 3.9 4.2 4.5
    7 10 30 70 100"
#link loads
set loads "0.7 0.8 0.9"
set base_time 10
set outfile_base "cost_"

for {set k 0} {$k < [llength $loads]} {incr k} {
        for {set i 0} {$i < [llength $thresholds]} {incr i} {
                set load [lindex $loads $k]
                set threshold [lindex $thresholds $i]
                set simtime [expr 1 0 * $base_time / $load]
                set outfile $outfile_base
                append outfile $load
                exec ns burstdrop tcl $load $simtime $threshold
                   Cost $outfile
        }
}

#random dropping

set thresholds "0 0025 0 005 0 01 0 015 0 025 0 05 0 075 0 1
   0 15"
set loads "0 7 0 8 0 9"
set base_time 10
set outfile_base "random_"

for {set k 0} {$k < [llength $loads]} {incr k} {
        for {set i 0} {$i < [llength $thresholds]} {incr i} {
                set load [lindex $loads $k]
                set threshold [lindex $thresholds $i]
                set simtime [expr 1 0 * $base_time / $load]
                set outfile $outfile_base
                append outfile $load
                exec ns burstdrop tcl $load $simtime $threshold
                   Random $outfile
        }
}
```

A.3 Burst dropping, complex topologies

The following script loads the network topology to be used from an external file It expects the file to be in an "ALT" format, generated by the GT-ITN tool [99] Similar scripts were used to test the cost-based scheduling algorithm in the US and pan-European networks

```
#name of the topology file
set infilename [lindex $argv 0]
#dropper threshold
set threshold_tmp [lindex $argv 1]
#simulation time
set endtime [lindex $argv 2]
#dropping strategy
set strategy [lindex $argv 3]

#create the log file name
set log $infilename
append log "_log_"
append log $strategy

#create the Simulator and OBS objects
set ns [new Simulator]
set ob [new OBS]

#declare global variables
set transits(0) ""
set stubs(0) ""
set links(0,0) ""
set nodes(0) ""
set nodecounter 0
set stubcounter 0
set transitcounter 0

#procedure that loads the network topology from a file
proc loadgraph {infilename} {
global transits stubs links nodes ob ns nodecounter stubcounter
   transitcounter threshold_tmp strategy

# global variables
# transits(index) array of transit domain nodes
# stubs(index) array of stub domain nodes
# links(node1)(node2) array of links
# ob OBS object
# ns ns object
# nodecounter number of nodes
# stubcounter number of stub nodes
# transitcounter number of transit nodes

set infile [open $infilename r 0660]
set nodes_temp(0) ""
set nodecounter_temp 0

set line ""
set status ""


while { ([gets $infile line] >= 0)} {
if { $status == "VERTICES"} {
#load node data
set nodes_temp($nodecounter_temp) [lindex $line 1]
incr nodecounter_temp
}
```

```
if { $status == "EDGES"} {
#load link data
set fromnode [lindex $line 0]
set tonode [lindex $line 1]
set edge($fromnode,$tonode) "1"
}

#detect labels and change states as necessary
if { [lindex $line 0] == "VERTICES"} { set status "VERTICES" }
if { [lindex $line 0] == ""} { set status "" }
if { [lindex $line 0] == "EDGES"} {
set status "EDGES"
for {set i 0} {$i < $nodecounter_temp} {incr i} {
for {set j 0} {$j < $nodecounter_temp} {incr j} {
set edge($i,$j) 0
set links($i,$j) "" }
}}
}

#create transit node list
for {set i 0} {$i < $nodecounter_temp} {incr i} {
puts $nodes_temp($i)
if { [string index $nodes_temp($i) 0] == "T" } {
set transits($transitcounter) [$ob core-node]
set nodes($nodecounter) $transits($transitcounter)
incr transitcounter
incr nodecounter
}

#create stub node list
```

```
if { [string index $nodes_temp($i) 0] == "S" } {
set stubs($stubcounter) [$ns obs-node]
$ob add-edge-node $stubs($stubcounter)
set nodes($nodecounter) $stubs($stubcounter)
incr stubcounter
incr nodecounter
}

#set dropper threshold
BurstDropper set threshold $threshold_tmp
#make the cost function value independent of offset time
Algorithm/GapFilling set improved 1

#create links and droppers
for {set i 0} {$i < $nodecounter_temp} {incr i} {
for {set j 0} {$j < $nodecounter_temp} {incr j} {
if { $edge($i,$j) == "1"} {
puts "link between $i and $j"
set linksize 45
set links($i,$j) [$ob optical-link $nodes($i) $nodes($j)
   $linksize 1ms "Algorithm/GapFilling"]
set dropper($i,$j) [new BurstDropper/$strategy]
$dropper($i,$j) set-id 1
$links($i,$j) add-dropper $dropper($i,$j)
}}}}
#end of procedure



proc addGenerator {fromnode tonode load offset max_offset id} {
set generator [new BurstAssembler/Exponential]
```

```
$generator set offset $offset
$generator set offset_max $max_offset
$generator set offset_jitter 1
$generator set-src-dst $fromnode $tonode
$generator set id $id
$generator set load $load
$generator set const_length 1
return $generator
}

loadgraph $infilename

$ob compute-routes
$ob populate-routers


# traffic sources

#number of traffic classes
set priorities 2

#loggers of sent traffic , one for each class
set startlogger(0) [new BurstLogger/Traffic]
set startlogger(1) [new BurstLogger/Traffic]

#Generators object , to manage the large number of generators
set generators [new Generators]

#load per generator , high and low priority class , respectively
set loads "0 1 0 3"
```

```
#add one generator between each pair of nodes
for {set i 0} {$i < $stubcounter} {incr i} {
for {set j 0} {$j < $stubcounter} {incr j} {
if {$i != $j} {
for {set k 0} {$k < $priorities} {incr k} {
set load [lindex $loads $k]
set generator [addGenerator $stubs($i) $stubs($j) $load 30ms 50ms
    $k]
$generator add-logger $startlogger($k)
$generators add-generator $generator
}}}}

#loggers of received traffic , one for each class
set endlogger(0) [new BurstLogger/Traffic]
set endlogger(1) [new BurstLogger/Traffic]

#set filters , so that each logger only counts traffic
#belonging to its class
$endlogger(0) set-id 0
$endlogger(1) set-id 1


#add loggers to nodes
for {set i 0} {$i < $stubcounter} {incr i} {
$stubs($i) add-logger $endlogger(0)
$stubs($i) add-logger $endlogger(1)
}

set logtime [expr $endtime +1]

#display simulation progress
```

```
proc progress { ns endtime} {
for {set i 0} {$i < 100} {incr i} {
$ns at [expr ((1 0 * $endtime /100)*$i)] "puts \"$i\%\""
}}


progress $ns $endtime



proc newline { } {
global log threshold
set logf [open $log {WRONLY CREAT APPEND}]
puts $logf " $threshold_tmp "
close $logf
}

#schedule events
$ns at 0 "$generators start"
$ns at $endtime "$generators stop"
$ns at $logtime "$startlogger(0) traffic-to-file $log"
$ns at $logtime "$endlogger(0) traffic-to-file $log"
$ns at $logtime "$startlogger(1) traffic-to-file $log"
$ns at $logtime "$endlogger(1) traffic-to-file $log"
$ns at $logtime "newline"

#run the simulation
$ns run
```