Architectures for Linear Lightwave Networks

by

Rajesh Kumar Pankaj

S.M. Massachusetts Institute of Technology (1988)
B.Tech. Indian Institute of Technology (1986)

Submitted to the Department of Electrical Engineering and Computer Science in partial fulfillment of the requirements for the degree of Doctor of Philosophy at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 1992

©Rajesh Kumar Pankaj

The author hereby grants to MIT permission to reproduce and to distribute copies of this thesis document in whole or in part.

Signature of Author

Department of Electrical Engineering and Computer Science September 1992

Certified by

Robert G. Gallager
Fujitsu Professor of Electrical Engineering and Computer Science Thesis Supervisor

Accepted by

Campbell L. Searle
Professor of Electrical Engineering and Computer Science
Architectures for Linear Lightwave Networks

by

Rajesh Kumar Pankaj

Submitted to the Department of Electrical Engineering and Computer Science on
August 1992, in partial fulfilment of the requirements for the degree of
Doctor of Philosophy

Abstract

In this work, we try to identify the functions that an all optical network should perform
and study how some of these functions can be performed in an efficient manner. Using
wavelength division multiple access (WDMA) on optical fibers, we can get a relatively
small number of high speed channels. In a wide area network, these channels must be
used efficiently to accommodate a large number of users. We focus on the issue of data
transport in the backbone of an optical network.

This work uses a version of the linear lightwave network model introduced by Stern
[Ste90]. We study the performance of the backbone network of an all optical network
based on this model. We first study the permutation problem and derive simple upper and
lower bounds on the number of wavelengths needed to solve it under certain nonblocking
conditions. The lower bounds are based on counting arguments. The upper bounds are
derived by example on specific topologies. Some specific graphs, particularly the perfect
shuffle network, the hypercube and the multi-Beneš network are discussed as possible
good topologies.

The analysis is extended to the m to m routing problem and the multicast problem.
Both these problems are of practical interest in designing a backbone network. Finally,
networks with wavelength converters are considered. As expected, the upper bounds
on the number of required wavelengths for the permutation and the m to m routing
problems are reduced when wavelength converters are allowed.

Keywords: Communications Networks, Optical Networks, Network Architectures

Thesis Supervisor: Robert G. Gallager

Title: Fujitsu Professor of Electrical Engineering and Computer Science
Acknowledgements

I thank my parents for their love and encouragement.

I thanks my advisor Prof. Robert Gallager for his patient guidance.

I thanks my thesis readers Dr. Steve Finn, Prof. Pierre Humblet and Prof. John Tsitsiklis for their helpful suggestions. I thank Prof. Sanjoy Mitter for substituting for Prof. Tsitsiklis at the thesis defense.

I thank my officemates Jane Simmons and Emre Telatar for numerous discussions throughout this work. I also thank Rick Barry, Tom Leighton, Kumar Sivarajan, David Tse and Manos Varvarigos for their help.

I thank the members of LIDS staff for their assistance.

I thank all my friends in the Boston area for making my stay at MIT enjoyable.

This work was supported in part by the Army Research Office through Grant No. ARO DAAL03-86-K-0171, the National Science Foundation through Grant No. NSF-8802991-NCR, the Lincoln Laboratories through Contract No. F19628-90-C-0002 and by the Fujitsu Chair.
Contents

1 Introduction
   1.1 Early Lightwave Networks ............................................. 8
       1.1.1 Star Networks .................................................... 9
       1.1.2 Multihop Star Networks ......................................... 9
       1.1.3 Other Networks .................................................. 11
   1.2 Wide Area Networks .................................................. 12
   1.3 Linear Lightwave Networks ........................................... 13
       1.3.1 Inseparability and Color Clash ................................ 17
       1.3.2 Multicasting and Multiplexing ................................ 18
   1.4 Limitations of the Linear Lightwave Network ................. 19
   1.5 Summary .................................................................... 20

2 Concept of the Network .................................................. 22
   2.1 Functions of the Network ............................................. 22
   2.2 Accessing the Channel ............................................... 26
       2.2.1 Circuit Switching on WDMA ..................................... 27
       2.2.2 Time Division Multiplexing on WDMA ....................... 28
   2.3 Some Routing Issues .................................................. 31
       2.3.1 A Simple Upper Bound ........................................... 32
       2.3.2 Nonblocking Networks ........................................... 33
       2.3.3 Topologies of Interest ........................................... 35

3 Lower Bounds on the Number of Wavelengths ..................... 37
   3.1 Oblivious Routing Algorithm ....................................... 37
   3.2 Counting the Switching States ...................................... 38
   3.3 Link Counting Bound ................................................ 39
       3.3.1 Bounds for an Arbitrary Network ............................ 40
       3.3.2 Uniform Networks .............................................. 42

4
3.3.3 Networks with Known Diameter ..................................... 43
3.4 Rectangular Grid Topology ............................................. 45
3.5 The Debruijn Graph ..................................................... 47

4 The Switching Networks ................................................... 52
  4.1 Rearrangeably Nonblocking Switch ..................................... 54
    4.1.1 The Perfect Shuffle Network .................................... 57
  4.2 Strict Sense Nonblocking Switch ..................................... 57
    4.2.1 Expander Graphs ............................................... 59

5 Upper Bounds on Number of Wavelengths ................................. 66
  5.1 Classes of Graphs with Constant Degree ............................. 67
    5.1.1 Wide Sense Nonblocking Network ................................ 67
    5.1.2 Rearrangeably Non-blocking Networks ....................... 72
    5.1.3 Beneš and other similar networks ....................... 73
    5.1.4 Limits of This Approach ................................... 76
  5.2 Classes of Graphs with $\Theta(\log N)$ Degree ................... 79
    5.2.1 Rearrangeably Nonblocking Routing .......................... 80
    5.2.2 Wide Sense Nonblocking Routing ............................ 83
  5.3 Wide Sense Nonblocking Networks Using Multibeneš ................ 83

6 Other Routing Problems ................................................ 85
  6.1 $m$ to $m$ Routing ................................................ 85
    6.1.1 Lower Bounds ............................................... 86
    6.1.2 Upper Bounds on the Number of Wavelengths ............... 90
  6.2 All to All Routing ............................................... 96
    6.2.1 Upper Bounds on the Number of Wavelengths ............... 96
  6.3 Multicasting Connections ........................................ 101
    6.3.1 Generalized Connectors .................................... 104
    6.3.2 Achievable Upper Bounds for the Multicast Problem ....... 106
7 Wavelength Conversion

7.1 Permutation Routing Problem ..................................... 111
7.2 Upper Bounds on the Number of Wavelengths ...................... 111
  7.2.1 Permutation Routing on Perfect Shuffle ..................... 112
  7.2.2 Permutation routing on a Hypercube ......................... 115
7.3 m to m routing ..................................................... 116
  7.3.1 m to m routing on a Debruijn network ....................... 116
  7.3.2 m to m routing on a hypercube ............................... 116
  7.3.3 Other Routing Problems ..................................... 117

8 Conclusions .................................................................. 118
  8.1 Implications of This Work ...................................... 118
  8.2 Summary .............................................................. 120
  8.3 Directions for Future Research ................................. 121

A Cantor Networks .......................................................... 123

B Oblivious Routing Algorithms ....................................... 126

C Strict Sense Nonblocking Switch Using Multi-Beneš ............. 129

D Generalized Connectors .................................................. 137

E Existence and Construction of Expander Graphs ................... 141
  E.1 Existence of Expander graphs .................................. 141
List of Figures

1 Logical Multihop Topology on a Star network .................. 11
2 A $3 \times 3$ LDC with 3 dividers and 3 combiners ............... 16
3 Color clash in LLN ........................................... 18
4 A simple two node network ................................... 30
5 Example of a uniform and a non-uniform network ............... 43
6 The rectangular grid and its rotated version .................. 46
7 The Debruijn graph .......................................... 48
8 The Clos Network ............................................ 53
9 A Beneš Network ............................................. 55
10 Some other rearrangeably nonblocking switches ............... 56
11 A Perfect Shuffle Switch ................................... 57
12 Cantor Network, a strict sense nonblocking switch ........... 58
13 An expander graph .......................................... 60
14 A multi-Beneš network being used as a switch ............... 62
15 The block diagram of a shuffle-Beneš network ............... 63
16 Two distinct paths from stage 1 to stage 2 ................... 70
17 Topological equivalence of baseline and inverse baseline networks ............... 75
18 A set of twelve nodes that can not be reached in one wavelength ............... 77
19 A two dimensional doubled hypercube ......................... 80
20 The back to back butterfly and the doubled hypercube networks ............... 82
21 A Cascaded Butterfly Network ................................ 97
22 Perfect Shuffle and Debruijn Graphs .......................... 114
23 An $(\alpha, \beta)$-expander graph .............................. 132
24 An $(\alpha', \beta')$-expander graph ............................. 133
1 Introduction

All optical networks show promise of providing us with very high speed communication networks. Since no electronics are involved, all optical networks are not limited by the speed of electronics. The optical fiber offers a large bandwidth. The low loss portion of the fiber bandwidth is tens of terahertz wide [Gre93]. We shall discuss the early attempts to make use of the large bandwidth offered by the optical fiber and then argue that a new approach must be taken to extend the use of the fiber to high speed wide area networks.

1.1 Early Lightwave Networks

In addition to having a large bandwidth, the optical fiber is also a low loss propagation medium. The low loss property of the fiber is used in present day networks to reduce the number of repeaters in long haul communication lines. In such networks, the fiber is simply a substitute for copper wire or other transmission media. The network itself is all electronic and the conversion to optical signals is made only at the input to the long haul transmission line. Such networks are not the topic of discussion in our work.

Even to be able to use fibers as a replacement of copper wire, a simple way to send signals over the fiber is needed. Frequency division multiple accessing or wavelength division multiple accessing (WDMA) is usually used to access the fiber channel. To send a signal on a wavelength, a laser with appropriate wavelength is used with desired modulation. To receive the signal from a wavelength channel, a filter is used to separate the desired wavelength which is then followed by either an energy detector, a heterodyne receiver or a homodyne receiver. The quality of current lasers, filters and fibers dictates that there should be a wide gap between two wavelength channels to avoid cross channel interference. The problem of cross channel interference becomes more severe as the length of the fiber increases. We shall later see how the inter-channel separation affects our ability to design the network. WDMA is not the only possible way to access the optical fiber channel and there is work going on in developing other, potentially more
efficient, ways of accessing the channel. The current technology, however, restricts us to WDMA.

1.1.1 Star Networks

The star network is an all optical network in the true sense. It is an optical broadcast network. There is a passive star coupler that connects all the nodes. It simply takes the power coming from an input and distributes it equally to all the outputs. The star coupler itself does not discriminate on the basis of wavelength or power. There is no electronics within the network itself. When node A wants to transmit to node B, it transmits its signal on an unused wavelength. The star coupler broadcasts node A's signal to all the nodes. All nodes except node B, disregard the signal. Only node B actually receives the signal.

It is easy to see that a number of issues must be resolved before such a communication system may work. Node A must have a way of selecting an unused wavelength. Node B must know that node A is sending a signal which it should receive. Node B must also know the identity of the wavelength channel on which node A is sending. Furthermore, the nodes must either have one transmitter for each wavelength that they may need to send on or one transmitter that can be tuned to the desired wavelength of transmission at will. The same must be true of the receivers at each node.

[Muk92] gives an overview of the various approaches taken to resolve these problems. Some of the work in resolving these issues can also be found in [Aca87, DGL+90]. It turns out that star networks are practical for local area networks.

1.1.2 Multihop Star Networks

In a star network, the nodes must either have separate transmitters and receivers for each wavelength channel or have transmitters and receivers that can be tuned to the desired wavelength. The option of having tunable transmitters and receivers is potentially a lower cost approach. But the current technology does not provide high tuning speed. Therefore
a significant latency time between sessions is involved when using a star network with tunable transmitters and receivers.

Multihop star networks are meant to correct the situation by using a relatively small, compared to the number of nodes, number of fixed wavelength transmitters and receivers at each node. Instead of direct connection from the source to the destination, a connection in a multihop star network may have to go through intermediate nodes. The intermediate nodes receive the message on one wavelength and forward it over another wavelength. The process of forwarding the message continues until the message appears on a wavelength on which the desired destination node can receive. These networks are not all optical because the intermediate nodes must convert the signal to electronics to be able to retransmit it on a different wavelength.

The basic idea in a multihop star network is to have a virtual multihop topology on a physical star network [Aca87, HK91]. A directed edge from node A to node B in the virtual topology is established by having a transmitter at node A and a receiver at node B that is tuned to the transmitter at node A. Figure 1 shows how a particular virtual topology is established. The picture on the left shows an actual star network. Each node transmits on a fixed wavelength. We assume that node i is transmitting at $\lambda_i$. Each node has two fixed wavelength receivers that receive on two different wavelengths. Nodes 1 and 4 receive on $\lambda_2$ and $\lambda_3$. Nodes 2 and 3 receive on $\lambda_1$ and $\lambda_4$. The resulting virtual topology is shown on the right. If node 1 needs to send a message to node 4, it must forward it via either node 2 or node 3. It should be noted that there are eight directed edges in the virtual network but there are only four transmitters, one for each node. Clearly, the transmitters are shared between edges. For example, when node 2 transmits, both nodes 1 and 4 receive the transmission. Typically, the transmission is meant for only one of the two listening nodes. There must be enough information in the message to specify the node it is meant for.
Node $i$ transmits on $\lambda_i$. Nodes receive on wavelengths specified next to them.

Figure 1: Logical Multihop Topology on a Star network

1.1.3 Other Networks

There have been other proposals for optical local area networks based on a physical star or other broadcast topology.

One proposal [AP91] is a variation of the multihop network. In this proposal, the transmitters at each node transmit at a fixed wavelength but the receivers can be tuned to any wavelength. Each node has more than one receiver. A connection uses a direct logical link from the origin to the destination if a receiver at the destination is free. Otherwise, a multihop path is used to get from the transmitter to the receiver. This way, the multihop topology is allowed to adapt itself to the traffic pattern in the network.

Another multihop proposal uses subcarrier modulation [RS91]. The messages for more than one user are collected at a concentrator. The concentrators are connected to each other with a broadcast star network. At each concentrator, there are only a few fixed wavelength receivers. The messages from a user share the wavelength channel with the other users connected to the same concentrator. The multiplexing is done using frequency division multiplexing which is also called subcarrier multiplexing. The basic advantage
of this proposals over the others is that the number of wavelengths required in this proposal is fewer than in other networks. A star network or a multihop network needs one wavelength for each node while the subcarrier based network requires one wavelength for each concentrator. The assumption is that a node does not produce enough traffic to require a full wavelength channel of its own.

All the proposals discussed in this section are for local area networks. In the next section we shall discuss why these networks cannot be easily extended to make a wide area optical network.

1.2 Wide Area Networks

The current state of optical technology is adequate for providing the components needed for the broadcast local area network proposals discussed above. Such simple network architecture is not feasible for wide area networks. The reasons are as follows:

Bandwidth: In a broadcast network, the bandwidth of the entire network is equal to the bandwidth of a single link. Since optical fibers offer a large bandwidth, this is not a major problem for local area networks in which the number of users is limited. In a wide area network, on the other hand, it is important to have the flexibility to expand. Even if a fiber can carry one terabit per second, with a million users the share of bandwidth per user is only one megabit per second. Furthermore, in a broadcast network, a user’s share of the bandwidth goes down as the network expands. This is clearly an unacceptable solution for wide area networks. To correct this situation, we must be able to reuse bandwidth in a wide area network.

Power: If there are $N$ users on a star network, a fraction $1/N$ of the transmitted power is sent out to each of the receivers. In a wide area network, the number of users, $N$, is likely to be very large making the received power very small. Optical amplifiers can compensate for some of the splitting loss but having to amplify a very weak signal means that the received signal to noise ratio will be small. The amplification
of the optical signal before splitting is limited by the nonlinearities of the amplifiers and the fiber channel. Having a device that allows us to channel the optical power to a limited region in the network will solve the problem of unnecessary splitting loss.

**Number of Wavelengths:** This problem is related to the lack of bandwidth discussed earlier. The constraint in this case comes from the state of the current technology. The lasers used as transmitters in a WDMA network do not have very stable frequencies. Hence, the guard band between wavelength channels must be large. One wavelength can carry on the order of gigabits per second or more but the number of lasers that can simultaneously use the fiber on a long distance fiber is relatively small. The number of available wavelengths is likely to remain a constraint in the future. Hence, we must be able to support large number of nodes on a small number of wavelengths in a wide area network if we want the network to be expandable.

The subcarrier based proposal [RS91] is one way to deal with this problem. However, this solution does not deal with the issues of power and bandwidth limitations discussed earlier. Clearly, the ability to reuse the wavelength channels is needed in order to make a scalable wide area network.

We would like to study networks that provides a mechanism to achieve channel reuse. We would also like to have a network model that is simple and uses components that are likely to be available in the near future. The Linear Lightwave Network (LLN) model, proposed by Stern [Ste90], has the desired properties.

### 1.3 Linear Lightwave Networks

A linear lightwave network is an all optical network in which linear operations are performed on the optical power at the intermediate nodes. The optical power can be amplified using optical amplifiers but there can be no signal regeneration. The model of linear lightwave networks we consider here also prohibits changing wavelength on a connection.
This means that a connection uses the same wavelength throughout its route in the network. Linear filtering of the signals is allowed. It is possible to separate signals that are on different wavelengths.

A crucial linear function is performed on the signals by a device called a *Linear Divider and Combiner* or LDC. Each node has one or more LDC. On wavelength \( w \) if \( i_1^w, i_2^w, \ldots, i_m^w \) are the powers at the \( m \) inputs of a node and \( o_1^w, o_2^w, \ldots, o_n^w \) are the powers at the \( n \) outputs of the node then they obey

\[
O^w = A^w I^w. \tag{1.1}
\]

The quantities \( O^w \) and \( I^w \) in equation 1.1 are

\[
O^w = \begin{bmatrix} o_1^w & o_2^w & \cdots & o_n^w \end{bmatrix}^T \tag{1.2}
\]

\[
I^w = \begin{bmatrix} i_1^w & i_2^w & \cdots & i_m^w \end{bmatrix}^T \tag{1.3}
\]

and \( A^w \) is an \( m \times n \) matrix characterizing the LDC for wavelength \( w \) at the node. For different wavelengths the matrix \( A^w \) can be different at the same node. We get different configurations of the LDC by changing the elements of the matrix \( A^w \). We assume throughout our discussion that the LDC configuration can be changed electronically.

We assume that the LDC is passive and lossless. This means that the elements of the matrix \( A^w \) are always non-negative and the columns sum to 1.

As discussed earlier, the LDC (Linear Divider and Combiner) is the most important component of an LLN. Conceptually, the LDC has two stages, the first consisting of a number of dividers and the second consisting of a number of combiners. A divider has a single input port and a number of output ports. If the input power to a divider is \( P_i \) then the output power at output \( j \),

\[
P_o^j = \alpha_j P_i.
\]
\[ 0 \leq \alpha_j \leq 1 \text{ and } \sum \alpha_j \leq 1. \]

A combiner is just the opposite of a divider. It has a number of input ports and a single output port. The output power is a weighted sum of the input powers. If \( P_i^j \) is the input power at port \( j \), the output power

\[ P_o = \sum_j \beta_j P_i^j, \]

where \( 0 \leq \beta_j \leq 1. \)

As shown in figure 2, the dividers and combiners together comprise the LDC. We assume that \( \alpha_j \) and \( \beta_j \) can be changed electronically. We also assume that there is a parallel electronic network to control the LDC's in the entire network. The mode in which the LLN operates depends upon the speed at which LDC's can be reconfigured and the amount of control information needed to establish a path in the LLN. If an LDC requires a long time to be reconfigured then the network may be used efficiently only in a circuit switched mode. If the amount of time needed to get the control information to the LDC's and to reconfigure the LDC's is short compared to the transmitting time of a packet then the LLN can be used in a packet switched mode. Since the packet transmitting time in a network is inversely proportional to the speed of the network, the LLN may have to use packet lengths of many thousands of bits to operate in a packet switched mode. In either case, we assume that the network requires relatively infrequent reconfigurations.

The connections on an LLN must satisfy the following two constraints:

**Channel Continuity:** A connection must use the same wavelength at every hop in its path. The model of LLN we are using here does not allow a connection to change wavelength in its route. We shall relax this constraint later while discussing wavelength converters.

**Distinct Channel Assignment:** If the signal from a connection reaches a link on which another connection is being routed then the two connections must be assigned
Figure 2: A $3 \times 3$ LDC with 3 dividers and 3 combiners
different wavelengths to avoid collision.

1.3.1 Inseparability and Color Clash

We have assumed that the matrix $A^w$ is different for different wavelengths at the same node. The original linear lightwave network proposal [Ste90] does not make this assumption. Different wavelengths may have to go through the same LDC setting. Such an LDC is probably easier to make but it creates difficulty in routing.

Suppose the LDC configuration for two different wavelengths is required to be the same. If two connections use the same link at any point in their paths on those two wavelengths, then the two signals cannot be separated at any point further down on their paths. Existence of one signal at any point after the common link implies the existence of the other signal as well. This leads to a problem called color clash as identified in [BSB91].

Color clash refers to a situation in which two existing non-interfering connections begin to interfere with each other after a new connection is routed on the network. Let us look at figure 3 for an example. Suppose connection 1 goes from node 2 to node 4 on wavelength 1 and another connection, say connection 2, goes from node 5 to node 7 also on wavelength 1. There is no interference between the two connections at this time. Suppose now, a new connection, connection 3, arrives from node 1 to node 8. We can assign wavelength 2 to it. The signals for connections 1 and 3 must use the same link to go from node 2 to node 3. If the LDC at node 3 must have the same configuration for wavelengths 1 and 2 then the signals for both the connections must go out to both the output links of node 3. This causes the signal from connection 1 to interfere with connection 2. Connections 2 must be assigned a new wavelength to resume collision free communication.

Color clash makes it difficult to do routing and channel assignment in linear lightwave networks. For simplicity, we shall assume at all times that the LDC can be configured independently for each wavelength at any node. In other words, the matrices $A^w$ for
Wavelength 1 is used to connect node 2 to node 4 and to connect node 5 to node 7. A new connection from node 1 to node 8 causes existing connections to interfere if independent routing for different wavelengths at node 3 is not possible.

Figure 3: Color clash in LLN

different wavelengths $w$ are allowed to be different from one another at the same node. This simplification allows us to focus on several other important issues in linear lightwave networks.

1.3.2 Multicasting and Multiplexing

The LDC as described above is capable of multicasting and multiplexing. Having more than one positive element in a column of the matrix $A^w$ allows input power coming into the LDC to go out on more than one output link. Similarly, by having more than one positive elements in a row, the LDC can send inputs from more than one link to the same output link.

For the sake of simplicity, we shall assume that no multicasting or multiplexing is allowed. The elements of matrix $A^w$ can only be 0's and 1's and there can be at most one 1 in any row or any column. The LDC in this case can be seen as a wavelength dependent switch. We believe that reducing the LDC to a switch provides a simpler model of a network that should be understood before introducing a more complex model that requires the LDC to have multiplexing and multicasting capabilities. We shall discuss one problem in Chapter 6 that requires relaxing this constraint.
1.4 Limitations of the Linear Lightwave Network

The linear lightwave model provides us with a set of devices that can be used to manipulate optical signals. Still the devices in the linear lightwave model are much less sophisticated than the devices available for electronic networks. Such limitations reflect the state of optical device technology. We discuss two limitations of the linear lightwave network model that play an important role in the rest of the work.

Wavelength Conversion: One option that a linear lightwave network does not allow is to change the carrier wavelength of a signal optically. If wavelengths can be changed within the optical network without electro optic conversion, routing can be simplified. In the circuit switched scenario, a connection can always be routed as long as there is capacity available in the network.

The difference between networks with wavelength converters and networks without wavelength converters can also be seen in the way they allow use of a link by different wavelength channels. A session in a network without wavelength converters must stay on the wavelength it originated on. If we have \( l \) wavelengths available, this effectively translates into having \( l \) copies of the network available to the session, each representing a separate wavelength channel, and the path for the session can be chosen on any one of the copies. The paths for each connection within the same copy must be edge disjoint to prevent collisions. These copies of the network do not interact with each other. The transmitters and receivers can be tuned to access any one copy of the network.

In networks with wavelength converters, each link can be replaced by \( l \) parallel links, each representing one wavelength channel. The routing problem now is to find edge disjoint paths in this new network with link dilation. This is because a session may use wavelength 1 on one hop and wavelength 2 on the next hop. This allows us to find a path for a new session if the link on hop 1 has wavelength 1 free and the link on hop 2 has wavelength 2 free and so on. In the network with
no wavelength converter, the same wavelength channel must be available on each hop for a path to be found. The number of wavelength channels is the same in both cases but networks with wavelength converters offer more flexibility in finding paths. In Chapter 7 we shall discuss networks that allow wavelength conversion.

**No Storage:** There is no random access optical storage available. This means that we cannot do time slot interchange at a switch. If the network is operating in a circuit switched mode, this does not pose a serious problem. But if the network is packet switched the timing issues become very important. We have discussed earlier that because of the relatively large delays involved in reconfiguring the LDC, the packets must be long (a thousand bytes or more) to keep the reconfiguration time overhead small. The lack of storage forces another issue in packet switched linear lightwave networks. If two packets arrive at a node simultaneously and both want to leave that node over the same link, only one may go through. The other packet must be dropped or misrouted. This is also true if the two packets do not arrive at the same time but the second one arrives before the first one finishes transmission. We shall discuss this issue in a little more detail in Chapter 2.

### 1.5 Summary

In the next chapter, the important functions of all optical networks are discussed. We also discuss the permutation routing problem and introduce the non-blocking criteria that are used in discussions later on. Chapter 3 discusses various bounds on the number of wavelengths needed for our routing problem. These bounds are obtained by simply counting the available network resources. Chapter 4 introduces results and concepts from switching theory that are needed for discussion in later chapters. Chapter 5 discusses networks based on perfect shuffle topology and derives lower bounds on the number of wavelengths needed for permutation routing under different nonblocking criteria. Chapter 6 discusses some interesting extensions of the permutation routing problem. We also discuss one problem that requires the LDC to have multicasting capability. Chapter 7
introduces wavelength changers. This is the only part of the work where we consider a device that allows information coming in on one wavelength to be transferred to a different wavelength.
2 Concept of the Network

We discussed the linear lightwave network in the previous section and described some of the restrictions imposed on the architecture of the network by the basic devices. It is important to look at the entire optical network and see what issues require our attention.

We have discussed earlier that the optical fiber offers large capacity channels. It is extremely difficult to have one outgoing 1 gigabit per second optical channel accessed by fifty incoming channels at 20 Megabits per second without using any electronics. This makes a lightwave network different from conventional networks. It is important in optical networks to combine small sessions to make one high bit rate super session which can use the large channels efficiently. Such aggregation can only be done electronically at present and it is important to have the capability of doing it optically so that electro-optic conversion can be avoided.

We have also pointed out that there is no optical random access storage available. We shall discuss how these constraints affect the design considerations in the network.

2.1 Functions of the Network

Let us divide the network conceptually into three parts. The first part deals with users and how they access the network. The second deals with aggregating the user sessions into super sessions that have large enough bit rates that they can use the optical channels efficiently. The last part deals with how to transport the super sessions efficiently over the optical network. The three parts naturally form a hierarchical structure but it may sometimes be convenient to break the hierarchy. For example, an architecture may allow high speed users to bypass the aggregation part of the network and feed their sessions directly to the transport function of the network. A low speed user in the same network may have to go through the aggregation process.

The challenge is to perform the functions of all three parts optically, if possible. If we must have electro-optic conversion, it should be at low speed points in the network
so that electronics does not become a bottleneck in the network.

**User Interface:** There are a number of very important issues that can be raised under this topic. How is the optical interface different from the electronic interface? How much flexibility can be provided to the user? There are a host of other issues that need to be understood.

One important issue is the diversity of the users. Some of the users may be high speed individual users and may require high speed access to the network. Other users may be local area networks. Conceptually, these local area networks can be seen as a single user by the all optical network but they consist of a number of smaller individual users. If a session from a local area network involves more than one individual user, it is likely to have multiple destinations. The all optical network must see it as either multiple point to point connections starting from one user or one multicast connection.

Although this work does not focus on the issues of user interface, we shall keep the diversity of users in mind in our future discussions.

**Aggregation:** Most of the user sessions are likely to use relatively small bit rates. Most of the current applications of communication networks, like electronic mail, are likely to stay important in the future. Only a small number of sessions will require high enough capacity to fill the entire bandwidth of one channel [Gal91]. However, in terms of use of bandwidth, the situation is likely to be the opposite. Most of the bandwidth will be consumed by a small number of high rate applications while the aggregation of a large number of low rate applications will probably use up only a small portion of bandwidth. These low rate applications must be supported by the optical network even though the optical channel is better suited for high rate applications. Services like electronic mail and voice will continue to be important to users and are likely to remain a source of revenue to the network.

As we have discussed earlier, the optical fiber does not easily divide into a large
number of low bandwidth channels, but can be easily used as a small number of high bandwidth channels. We cannot afford to use one high bandwidth channel for a low rate session, simply because we don't have enough channels. The issue of aggregating the low rate sessions so that the high bandwidth channels can be utilized efficiently becomes a critical one.

There are two issues in the aggregation problem. Once we have aggregated a number of point to point sessions into one super session, the super session is no longer point to point. The individual sessions may have different destinations and hence the super session is multicast. In the worst case, the number of intended destinations for a super session equals the number of individual sessions in the super session. Either the backbone network used to transport this super session must have multicasting capability (even though all the user sessions are point to point) or there must be a separate deaggregation network. In the latter case, the backbone can bring a super session to the deaggregation network which then distributes the individual sessions in the super session to their destinations. In either case, the issue of aggregation complicates the network design problem.

The second issue is multiplexing technology. We know how to multiplex electronic signals and hence allowing electro-optic conversion overcomes the technological problem of multiplexing. But electro-optic conversion increases the cost of the network and limits the speed of the sessions. Hence, we would like to develop multiplexing techniques that work in the optical domain and do not require electro-optic conversion.

We would like to get a better understanding of how various traditional multiplexing schemes can be used in optical networks. It is important to look at the problem of demultiplexing at the other end as well, since the two problems are closely related. The relative ease of broadcasting on optical networks may make the problem of demultiplexing the aggregated sessions easier. Still, it is important to understand how both multiplexing and demultiplexing can be done. Most of our work, however,
is devoted to the study of the final aspect of the backbone network function which we discuss next.

**Backbone Network:** The issue here is to use the fiber bandwidth in an efficient manner. As discussed before, optical fiber is best at providing high capacity channels. We have discussed linear lightwave networks in some detail. They can be used to make backbone networks for high speed data transfer. We would like to understand how well the backbone networks can perform using LLN. Linear lightwave networks do not use wavelength converters, which would allow a session to change wavelength in the middle of the network without going through electronics. Such devices are potentially expensive, and we would like to see what can be achieved without using wavelength converters.

Very little is understood about what the LLN can achieve. We make a number of simplifying assumptions in order to gain a basic understanding of the all optical network. We hope to get good bounds on network performance.

To get some understanding of how to design a backbone network, we draw an analogy between the backbone network and a switch. Geographically, a backbone network spans a large area and a switch is confined to a small space, but the two serve essentially similar functions. A backbone network must be able to find paths for a connection from the origin node to the destination node. A switch on the other hand must be able to find a path from an input to an output. The type of connections to be established depends upon the kind of applications they are being used for in each case.

If the cost of connecting links can be ignored, we can imagine having a backbone network that is small in terms of geographical spread. All the nodes generating traffic are connected to this small backbone network. The traffic must go through stages of aggregation if the small backbone network can serve as the backbone of a large network. In the extreme case we can think of shrinking this small backbone
network to an area comparable to the area occupied by a switch. In fact, the backbone can then be replaced by a switch that has the appropriate capacity and the capability to establish connections that must be established by the backbone network.

We would never want to replace a backbone network by a switch because the cost of connecting links to a central switch will be large for a large network. Furthermore, a single natural or man made disaster at the site of the switch could bring the whole network down. However, we can extend this scenario a little more and think of a switch that is distributed in space. This way, we can bring the access points closer to the actual nodes in the network and a disaster striking a limited geographical area will only affect a small portion of the network. With good network design we can build in enough redundancy to ensure that the network can be in operation if the affected area is small enough.

We shall study switching topologies to study how we can take advantage of this relationship between the backbone network and distributed switch. In spite of the conceptual analogy between switches and backbone networks, it is unreasonable to assume that in real life large backbone networks can follow regular switching topologies. However, the study of switching topologies can still give us backbone networks that are very efficient in transporting data. By analyzing these topologies, we can get bounds on the capabilities of real backbone networks.

2.2 Accessing the Channel

We have mentioned before that the fiber channel is accessed by wavelength division multiple accessing or (WDMA). This method of access divides the fiber channel into a relatively small number of large bandwidth channels. We shall look at two ways of using a WDMA network.
2.2.1 Circuit Switching on WDMA

The fiber channel is available in a small number of large bandwidth channels. Each channel is called a wavelength. The name wavelength simply signifies the wavelength and equivalently the frequency of the carrier optical signal. The optical carrier frequency of a signal cannot be changed using all optical processing under the linear lightwave network model. Hence, there is no conversion from one wavelength to another and hence a signal stays on the wavelength on which it originates. This effectively gives us one independent network for each wavelength. All these independent networks work in parallel with one another. We refer to these parallel networks as the layers of the network. We can assign paths on each layer for new connections without interfering with the connections on other layers. A user (transmitter or receiver) can access all the layers.

In a circuit switched mode, timing is not an issue. We assume that the life time of a circuit is much larger than the reconfiguration times of the devices in the network. In each layer of the network, a link can be used by only one connection. As long as an edge disjoint path from the source to the destination can be found in one of the layers of the network, the connection can be established by reconfiguring the devices appropriately. All the results that are derived under this assumption are also valid for any accessing scheme that effectively slices the network into identical independent layers.

Let us think of a network in which the propagation delay is zero. Suppose the network is time slotted and the time slots at each node are perfectly synchronized. If such a network has no data storage, a packet that appears on slot 1 at one node must also appear at slot 1 at every other node on its path. No signal from one slot can interfere with any signal on any other slot. Hence, each slot makes a separate layer. A packet can not move from one layer to another but the nodes can place their packets on any layer and can retrieve packets from any layer. A little thought shows that we can have such a system on special topologies even for non-zero propagation delay. However, the topologies must be such that the clocks at each node can be synchronized taking the propagation delay into account. For example, in a unidirectional bus the clocks can be
synchronized such that a packet that appears in the beginning of slot 1 of one node also appears in the beginning of slot 1 at all the subsequent nodes. We shall later see that it is not possible to synchronize the clock in this way for networks with arbitrary topologies.

We shall assume an access scheme that effectively divides the network into parallel layers that can be configured independently. For example, in one layer the network may be connecting node 1 to node 2 and in a different layer it may be connecting node 1 to node 3. In the context of linear lightwave model discussed earlier, the WDMA networks provide the necessary layering. We shall use the terms layers and wavelengths interchangeably throughout this work. However, it is understood that all arguments and results are valid for any access scheme that provides the layering and the corresponding network model that allows each layer to be configured independently.

### 2.2.2 Time Division Multiplexing on WDMA

To provide more flexibility in a WDMA network, we can time slot all or some of the wavelength channels. Since each wavelength channel is independent, it is possible to do slotting on selected wavelengths and use other wavelengths as dedicated channels for the duration of a connection.

Let us look at one wavelength channel that is time slotted. Suppose a frame of slots consists of \( T \) slots. Also suppose that it is possible to synchronize the clocks at each node. Then, as we discussed in the last subsection, we can think of this wavelength channel as being divided into \( T \) independent layers. So far, the system with time division multiplexing is conceptually no different from the system without time division multiplexing. Clearly, there is a difference in implementation of the layers between the two schemes but that is of no concern to us so far.

The real difference comes in when we take propagation delay into account. Let us define a network to be *partially ordered* if we can set time shifts of the clocks at each node compared to some reference clock in such a way that for any connection the signal that appears at the beginning (or at some other reference point) of slot 1 at the origin
also appears at the beginning (or at some other reference point) of slot 1 at each node on its path.

If a network is not partially ordered then there may not be a way to ensure perfect clock synchronization unless we use a global clock and leave enough empty time in a slot to accommodate the propagation delay over the longest possible path. In a high speed network the propagation time is large compared to the transmission time of a packet. Hence, the overhead in such a scheme would be too large for it to be practical.

Let us look at one way in which networks that are not partially ordered can be used with time division multiplexing on wavelength channels. Let us assume that the slots on a link are synchronized with the slots at their origin nodes. Clearly, with different propagation delays, the slots on two links arriving at a node may not be synchronized. We shall also assume that a frame consists of $T$ time slots at each node. Suppose we need to establish a connection from node $n_0$ to $n_m$ through nodes $n_1, n_2 \ldots n_{m-1}$. Suppose we assign slot $t_0$ on node $n_0$ to this connection and due to propagation delay it reaches node $n_1$ in the middle of slot $t_1$. If we assign both slots $t_i$ and $t_i + 1$ at node $n_1$ to this connection, it can go through node $n_1$ without any problem. In general, suppose the first signal of this connection reaches node $n_i$ in the middle of slot $t_i$. We assign slots $t_i$ and $t_i + 1$ for this connection. This way by assigning two slots instead of one to each connection we can get them through without requiring any storage and without any conflict in their path. It should be noted that this cannot be achieved by making slots twice as long since if a connection arrives at a node during the second half of a slot, it will be spilled over to the next slot and hence assigning one slot to each connection will not be sufficient.

Although we have a way of doing time division multiplexing in a wavelength channel on a network that is not partially ordered, it comes at a price. Using the network in the way just discussed does not break it up into independent layers. To illustrate this point let us look at a simple example. Suppose we have a two node network as shown in figure 4. The transmission time of a packet is exactly equal to the length of a slot.
The network shown in the figure shows why two slot assignment for each connection does not provide layering in a network with arbitrary propagation delay.

Figure 4: A simple two node network

Suppose also that the propagation delay from node 2 to node 1 is larger than two time slots. Now suppose we have a connection from node 1 to node 2 that occupies the slots 1 and 2 at both nodes 1 and 2. This can be achieved by simply renumbering the slots at the nodes. If this scheme could give us true layering then slots 1 and 2 by themselves would make a layer and any link that is not being used in this layer can be assigned to a new connection. Suppose we now wish to establish a connection from node 2 to node 1. At the moment nothing is leaving node 2 in slots 1 and 2 and nothing is reaching node 1 in slots 1 and 2 but we cannot assign slots 1 and 2 for the new connection. To see the reason let us try to establish the difference in the clocks at nodes 1 and 2. Since slots 1 and 2 are being used by a connection from node 1 to node 2, the start of slot 2 at node 2 cannot be earlier than the start of slot 1 at node 1. Hence, the start of slot 1 at node 2 is at most two slots before the start of slot 2 at node 1. If a signal leaves node 2 at the start of slot 1 then it will reach node 1 after slot 2 has already started at node 1. Since the packets are exactly one slot long, the end of the message will reach node 1 after slot 2 at node 1 has ended and hence it will spill over to slot 3.

It is easy to see that this argument can be carried out for more complicated networks as well. Also, the assumption that the length of a slot is equal to the length of a packet is not crucial to the argument. If we make the slot length larger than packet lengths, the delay required to create the same scenario may be different but the same basic argument is valid.

In this work we focus on channel access schemes that divide the network into independent layers. The connections can be assigned any link on a given layer without affecting
the link assignments on other layers. Hence, our discussion will not include access scheme described here for time division multiplexing on a network that is not partially ordered.

2.3 Some Routing Issues

We want to study the topologies suitable for the backbone of a linear lightwave network. To do an orderly study of the topologies, we must specify the kind of connections we want to route and the performance criteria we want to use to evaluate the topologies.

We first focus on one to one or permutation routing. In permutation routing, each node can be the origin of at most one connection and the destination for at most one connection. All the connections are point to point. This way, the total number of connections are upper bounded by the number of nodes $N$, and no two connections may ever have the same origin or destination. We want to compute the number of wavelengths needed to ensure that these connections can be routed on the LLN without violating the distinct channel assignment constraints listed in Section 1.3.

We discussed in Chapter 2 that the backbone must have multicast capability if sessions are aggregated into super sessions. Hence, permutation routing is not an appropriate model for the traffic pattern in the backbone of the network. Still, we concentrate on the permutation routing problem first because it is a simpler problem to understand. We shall use the understanding of the permutation routing problem to understand other, more practical routing problems. In particular, we shall look at the $m$ to $m$ routing problem, the all to all routing problem and the multicasting problem. They are discussed in Chapter 6.

The performance criterion we are looking for is the nonblocking property, i.e. the ability to route a given permutation routing problem such that no connection is blocked. The nonblocking property is defined more precisely in a later section.

We need to specify some assumptions about the description of the network.

- The network is represented by a directed graph.
• We assume that the directed graph representing the network is strongly connected. In other words, we can find a directed path from any origin node to any destination node. Without this assumption, we can have a permutation routing problem that cannot be solved.

• A session in the network can use any wavelength available to it in the network but the entire path for that connection must use the same wavelength. We think of our directed graph as having \( l \) distinct and disjoint replicas of itself where \( l \) is the number of wavelengths. Establishing a nonblocking path for any origin destination pair in the network is equivalent to finding an edge disjoint directed path on one of the replicas of the graph. The edge disjoint property must be assured only among other paths on the same replica and is completely independent of the paths on other replicas of the graph. We refer to the replicas as the layers of the network.

2.3.1 A Simple Upper Bound

A simple upper bound on the number of wavelengths needed for permutation routing in a nonblocking network can be derived. Suppose, in an \( N \) node network, we assign one wavelength to each node. We need \( N \) wavelengths to be able to do this. Now, whenever a session has to be routed, it uses the wavelength assigned to the origin. Since two sessions never use the same wavelength, there is no question of two paths using the same wavelength on the same edge. Also, since the network is strongly connected, a path for the session assigned to any given wavelength is guaranteed to be found. It is easy to see that this algorithm solves all permutation routing problems without blocking any session. Hence, \( N \) is an upper bound to the number of wavelengths needed for nonblocking permutation routing.

Let us introduce the \( O, \Omega \) and \( \Theta \) notations since they give us an easy way to represent the rate of growth of a function. By saying that a function \( g(N) \) is \( O(f(N)) \), we mean that for large \( N \), \( g(N) \) is upper bounded by a constant times \( f(N) \). Similarly, a function \( g(N) \) is \( \Omega(f(N)) \) if \( g(N) \) is lower bounded by a constant times \( f(N) \). A function \( g(N) \) is
said to be $\Theta(f(N))$ if $g(N)$ is both upper bounded by a constant times $f(N)$ and lower bounded by another constant times $f(N)$. More formally, a function $g(N)$ is $O(f(N))$ if for a given integer $N_0$, there exists a constant $c_1(N_0)$ such that $g(N) \leq c_1(N_0)f(N)$ for all $N \geq N_0$. Similarly, $g(N)$ is $\Omega(f(N))$ if for a given integer $N_0$, there exists a constant $c_2(N_0)$ such that $g(N) \geq c_2(N_0)f(N)$ for all $N \geq N_0$. A function $g(N)$ is $\Theta(f(N))$ if for a given integer $N_0$, there exist constants $c_1(N_0)$ and $c_2(N_0)$ such that $c_2(N_0)f(N) \leq g(N) \leq c_1(N_0)f(N)$. Clearly, if $g(N)$ is $\Theta(f(N))$, it is also $O(f(N))$ and $\Omega(f(N))$.

The upper bound on the number of wavelengths needed for nonblocking routing can be expressed in $O$ notation by stating that the number of required wavelength is $O(N)$.

2.3.2 Nonblocking Networks

We borrow the concept of nonblocking networks from switching theory to define the performance objectives for the network. The concept of nonblocking networks is dependent upon the arrival model and the routing algorithms. Let us first try to distinguish between the different ways in which a network can be nonblocking.

**Strict Sense Non-blocking Networks:** A strict sense non-blocking network ensures that regardless of how the previous connections are routed, a new connection can always be routed over the network.

**Wide Sense Non-blocking Network:** In a wide sense non-blocking network, there is an algorithm such that a path for a new connection can always be found by the algorithm as long as the previous paths are established using the same algorithm.

**Rearrangeably Non-blocking Network:** This is the weakest form of non-blocking property. In a rearrangeably non-blocking network, all the connections can be routed if they all arrive simultaneously. The same concept can also be stated as follows. Suppose there are a number of connections already established in the
network. A path for a new connection can be found if the existing connections are allowed to be rerouted.

We can look at the nonblocking property of the network as a way of guaranteeing services. For example, a network that is nonblocking for permutation connections is guaranteed to find an edge disjoint path for a connection that starts at a node with no other outgoing connections and ends at a node with no other incoming connections. In general, the network can establish other connections also but there is no guarantee that paths for such connection can be found and a connection that violates the permutation criterion may have to be terminated to find paths for connections that are within the permutation criterion.

As can be easily seen, the strict sense nonblocking criterion is the most stringent and rearrangeably nonblocking criterion is the least stringent among the three. The rearrangeably nonblocking criterion is not very useful for backbone networks since all the connections do not arrive simultaneously and it is undesirable to interrupt an existing connection when a new connection arrives. We next discuss why the strict sense nonblocking criterion is too strict for our purposes.

**Strict Sense Non-blocking** Can we improve upon the upper bound of $N$ wavelengths for a strict sense nonblocking network? We show that the answer is essentially no by showing how to route existing connections so as to make it very difficult to add a new connection.

Let us define the out-degree of a node to be the number of directed edges leaving the node. We similarly define the in-degree of a node as the number of directed edges coming into a node.

Let $n_o$ be the node with the smallest out-degree and suppose its out-degree is $d_o$. Consider a routing algorithm that routes each connection first through node $n_o$ and then finds a path from node $n_o$ to the destination. Hence, each connection must go through node $n_o$. Since the out-degree of $n_o$ is $d_o$, at most $d_o$ connections can be leaving $n_o$ in one
wavelength. If we try to route more than \( d_* \) connections through \( n_* \) then at least one outgoing edge must carry at least two connections and hence edge disjoint paths cannot be found in one wavelength.

Suppose the number of available wavelengths is \( \lfloor N/d_* \rfloor - 1 \) and there are \( d_*(\lfloor N/d_* \rfloor - 1) \) existing connections. Because of our routing algorithm, all output edges from node \( n_* \) are being used in all the available wavelengths. Suppose also that there is no connection originating from node \( n_* \) among the existing connections. Now, if a new connection arrives that originates at node \( n_* \), it cannot be routed on the network. Clearly, with \( \lfloor N/d_* \rfloor - 1 \) wavelengths, the network is not strict sense nonblocking. Hence, the minimum number of wavelengths needed for a network to be strict sense nonblocking is \( \lceil N/d_* \rceil \).

For a class of network with a given minimum out-degree \( d_* \), the number of wavelengths is \( \Omega(N) \).

Since the simple upper bound derived in Subsection 2.3.1 says that the required number of wavelengths is \( O(N) \), we conclude that the strict sense nonblocking criterion is too stringent to get any substantial improvement over the simple minded routing scheme used to derive the upper bound. To reduce the required number of wavelengths, we should focus on wide sense and rearrangeably nonblocking networks.

### 2.3.3 Topologies of Interest

It should be clear that the smallest number of wavelengths required to support a certain number of nodes is strongly dependent upon the graph. In particular, the degree of the graph is very important. For example, if we allow the degree of the graph to be \( N - 1 \), then \( N \) nodes can be supported using only one wavelength by making the graph a fully connected one. To bring some order to this situation, we consider two kinds of network classes, one in which the maximum node degree is independent of the number of nodes in the network and the other in which the maximum node degree grows as the logarithm of the number of nodes in the network.

Among the network classes with node degrees independent of the size of the network,
we study the perfect shuffle network, the Beneš network and variations of them. These networks can be constructed with node degree 2 or more. We only consider the networks with degree 2 for convenience. All the results from networks of degree 2 can be easily extended to networks with higher degree. We also study a number of network classes based on expander graphs. The node degree of these graphs is also independent of the size of the network, but typically the minimum node degree needed to construct these graphs is larger than 2.

Among the networks classes in which the node degree grows logarithmically with the number of nodes in the network, we consider the hypercube and a variation on the hypercube called the doubled hypercube. We shall define these networks more precisely in later chapters.
3 Lower Bounds on the Number of Wavelengths

We can derive some bounds on the number of wavelengths needed to support a certain number of nodes. These bounds depend upon the topology only through the degree. In all these cases, we assume that we need to find paths for a permutation routing problem under nonblocking criteria. These bounds are applicable for both wide sense and rearrangeably nonblocking criteria.

3.1 Oblivious Routing Algorithm

An oblivious routing algorithm is an algorithm that routes a connection from a given origin to a given destination over the same path no matter what other connections are present in the network.

Such an algorithm can be specified by $N(N-1)$ paths for the $N(N-1)$ possible origin destination pairs, where $N$ is the number of nodes in the network. We do not require an oblivious algorithm to specify the wavelength to be used for a particular connection but only the edges over which it is to be routed. In spite of the freedom to choose the wavelength, an oblivious routing algorithm is very restrictive, since in case of blocking, it does not allow a connection to be routed over a free alternative path. We borrow a theorem from packet routing [KKT90] to show that an oblivious routing algorithm requires $\Omega(\sqrt{N}/d)$ wavelengths where $n$ is the number of nodes and $d$ is the in-degree of the network. The $\Omega$ notation is explained on page 32.

Theorem 3.1 In any network with $N$ nodes and maximum in-degree $d$, the nonblocking permutation problem with oblivious routing requires at least $\sqrt{N}/\sqrt{2d}$ wavelengths.

The proof of this theorem follows the proof in [Lei92] and is presented in Appendix B.
3.2 Counting the Switching States

Let us try to count the number of distinct valid states, $S$, in which the network can be configured. Suppose there are $l$ wavelengths and the LDC switches at each node can be in one of $\Gamma$ states for each wavelength. There are $N$ switches and each operates on $l$ wavelengths. If the switches could be set independently for different wavelengths, then, for counting purposes, this is equivalent to having $Nl$ independent switches in the network. This is not quite the case since an LDC switch cannot be set completely independently on different wavelengths. A node can only receive on one wavelength and hence a switch setting that allows a node to receive on more than one wavelength is not valid. Similarly, a switch setting that allows a node to transmit simultaneously on more than one wavelength is not valid. By counting each switch as completely independent, we get an upper bound on the total number of allowable settings of the LDC switches. Hence, the total number of possible states for switches is upper bounded by $\Gamma^{Nl}$.

Each node has a receiver and a transmitter which can be tuned to one of the $l$ wavelengths and hence can be in one of $l$ possible states. However, if the network is in a valid state then the transmitter at a node is allowed to transmit at only one of the wavelengths. Similarly, a receiver can receive at only one of the wavelengths. Hence, the state of the switches in the network specify the wavelengths that the transmitters and the receivers must be tuned to.

There may be a number of these states that solve the same routing problem. For example, if we take a wavelength assignment in a routing problem and move all the wavelength 1 paths to wavelength 2 and wavelength 2 paths to wavelength 1, the resulting routing is the same as the original one. We do not compensate for this overcounting of the number of states because it does not change the bound very much.

There are $N$ nodes in the network and hence a total of $N!$ permutation routing problems are possible. If we do not allow connections from a node to itself, the number of possible permutations is less than $N!$. Still, the total number of allowable permutations
is greater than \((N - 1)!\). If we denote the number of allowable permutations by \(P\), then

\[
(N - 1)! < P < N!.
\]

A nonblocking network must be able to solve at least \(P\) different routing problems. Since in each state it can solve only one routing problem, we conclude \(P \leq S\) must be satisfied. Hence,

\[
(N - 1)! < \Gamma^N. \tag{3.1}
\]

\(\Gamma\) is dependent upon the degree of the network. The number of possible states for a switch in any node is \((d + 1)!\), where \(d\) is both the in-degree and the out-degree of the node. If the in-degree and out-degree are not equal, the expression for \(\Gamma\) will be a little different but it does not affect the final conclusion in any significant way.

We do not use the bound derived by counting the switching states in our future discussions. Instead, the link counting bound is used which we derive in the next section. We shall see that for permutation routing on a general graph, the link counting bound has the same rate of growth as the bound just derived, we use it because it appears in an easier form and can be easily adapted to problems other than permutation routing.

### 3.3 Link Counting Bound

Another lower bound for the number of wavelengths can be obtained by simply counting the number of links in a network and the number of links used by any given connection. Let \(H\) be the sum of the number of hops used by all the connections and \(E\) be the number of links in the network. If \(H > E\), some links in the network must be used by more than one connection. The number of times a link can be used is at most the number of available wavelengths, since on one wavelength, a link can only be used by one connection. If the number of wavelengths required to route the given set of connections is \(l\), then

\[
l \geq \frac{H}{E} \tag{3.2}
\]
In this section, we derive lower bounds on $H$ and upper bounds on $E$ to get lower bounds on the number of required wavelengths.

Suppose the maximum in-degree of a node in the network is $d_{in}$. The number of links in the network is then upper bounded by

$$E \leq N d_{in}. \quad (3.3)$$

The same bound can be expressed in terms of the maximum out-degree of the network as well. The choice of in-degree in expressing the bound is arbitrary.

### 3.3.1 Bounds for an Arbitrary Network

In a given network there are $N(N - 1)$ possible origin destination pairs. Suppose we have minimum hop paths for all possible pairs. The length of the longest path in the set of minimum hop paths is called the diameter of the network. The diameter of the network is denoted by $\Delta$.

In order to bound the link usage for a permutation problem in a network, we try to find a number of origin destination pairs such that the number of hops in a shortest path between each pair is large. In order to do that, we try to find a lower bound on the diameter $\Delta$.

Let us choose a node, say node $n$. Let $d_{in}$ be the maximum in-degree of the network. In the trivial case when $d_{in} = 1$ and the network is connected, the topology must be a one directional ring and hence $\Delta = N - 1$. We shall later see that tighter bounds can be derived for a one directional ring. For now, assume that $d_{in} \geq 2$.

Given any node $n$, at most $d_{in}$ nodes can have one hop paths to $n$, at most $d_{in}^2$ nodes can have two hop paths to $n$ and so on. If the diameter of the network is $\Delta$, then we must be able to reach node $n$ from every node in the network in at most $\Delta$ hops. Hence,

$$N \leq \sum_{i=0}^{\Delta} d_{in}^i \quad (3.4)$$

40
\[ = d_{in}^\Delta + \frac{d_{in}^\Delta - 1}{d_{in} - 1} \] \hspace{1cm} (3.5)

\[ \leq 2d_{in}^\Delta. \] \hspace{1cm} (3.6)

This means that,

\[ \Delta \geq \log_{d_{in}} \frac{N}{2}. \] \hspace{1cm} (3.7)

Now we know that the longest path to a given node \( n \) must be at least \( \log_{d_{in}} \frac{N}{2} \) long. We would like to find a permutation routing problem in which most of the connections require long paths.

Following the same reasoning as before, we see that the expression

\[ \sum_{i=0}^{\lfloor \log_{d_{in}} \frac{N}{2} \rfloor - 1} d_{in}^i \leq \frac{N}{2(d_{in} - 1)} \leq \frac{N}{2} \]

represents the number of nodes that have paths of length \( \lfloor \log_{d_{in}} \frac{N}{2} \rfloor - 1 \) or less to node \( n \). The last inequality is a consequence of \( d_{in} \geq 2 \). Since at most \( N/2 \) nodes have paths that are \( \lfloor \log_{d_{in}} \frac{N}{2} \rfloor - 1 \) hops or shorter to \( n \), we conclude that the shortest path from at least \( N/2 \) nodes in the network to node \( n \) has at least \( \lfloor \log_{d_{in}} \frac{N}{2} \rfloor \) hops.

Now let us take an arbitrary set of \( N/2 \) nodes \( \mathcal{N} \). If we take all the nodes in set \( \mathcal{N} \) as destinations in a particular routing problem, we can find distinct origins to pair up with each node in \( \mathcal{N} \) so that the path for each origin destination pair is at least \( \lfloor \log_{d_{in}} \frac{N}{2} \rfloor \) hops long. By pairing up the rest of the nodes arbitrarily, we have a permutation routing which requires at least

\[ \frac{N}{2} \lfloor \log_{d_{in}} \frac{N}{2} \rfloor \]

hops. Once again using equations 3.3 and 3.2, we conclude that

\[ l \geq \frac{H}{E} > \frac{\lfloor \log_{d_{in}} \frac{N}{2} \rfloor}{2d_{in}}. \]
By changing the base of the logarithm we get
\[ l > \frac{1}{2d_{in}} \left\lfloor \frac{\log \frac{N}{2}}{\log d_{in}} \right\rfloor . \] (3.8)

The bound derived here does not assume anything about the network except the number of nodes and the maximum in-degree. The same computation can be done for maximum out-degree as well. The choice of in-degree in the previous calculation is arbitrary.

Sometimes we have more information about the graph than just the degree and the number of nodes. The bound can be made tighter in those cases using the extra information. We discuss two such cases next.

3.3.2 Uniform Networks

Let us first discuss what is meant by uniform networks. Informally, a network is uniform if all its nodes see exactly the same thing. The graph shown in figure 5a shows a uniform network. Suppose we are given an arbitrarily labeled graph and identify an arbitrary node, say node 1. If we can relabel any other node as node 1 and then relabel the rest of the nodes in such a way that the resulting graph is topologically the same as the original one, then the network represented by the graph is defined to be uniform. Figure 5a shows a uniform network. Figure 5b shows an example of a non-uniform network. Only two nodes, node 1 and node 2, have three neighbors. If we relabel node 3, which has only one neighbor, as node 1, we cannot relabel the rest of the nodes in the graph in a way that after the new labeling, the graph is topologically equivalent to the original one.

The following lemma is true for all uniform networks with diameter \( \Delta \).

Lemma 3.2 In a uniform network with diameter \( \Delta \), we can find a permutation routing problem such that the minimum hop path for each origin destination pair is of length \( \Delta \).

Proof: Let us construct a bipartite graph \( T = (U, V, F) \). Both \( U \) and \( V \) have \( N \) nodes. There is an edge between node \( u_i \) and \( v_j \) in \( T \) if the minimum hop path from
node $i$ to node $j$ in the network is exactly $\Delta$ hops long. Clearly, the set of edges, $F$, is non-empty. Furthermore, since the network is uniform, all the nodes in $U$ must have the same degree. Similarly, all nodes in $V$ must have the same degree. We know from Hall's theorem [Ber91, Rys63] that in such a bipartite graph, we can always find a perfect matching. Hence, $T$ has a perfect matching. A perfect matching in $T$ identifies a permutation routing problem such that the minimum hop path for each origin destination pair is of length $\Delta$. This proves the lemma.

By choosing the permutation given by the lemma, we have a set of $N$ paths, each of which requires at least $\Delta$ hops. Hence, for a uniform network, $H$ satisfies the following inequality:

$$H \geq N\Delta.$$  

Using the expressions from equations 3.3 and 3.2 we conclude that

$$l \geq \frac{\Delta}{d_{in}}. \quad (3.9)$$

3.3.3 Networks with Known Diameter

In equation 3.7 we derived a lower bound on the diameter of a network in terms of the number of nodes $N$ and the maximum in-degree $d_{in}$. In this subsection we assume that
the diameter of the network is given along with the number of nodes and we want to find bounds on the required number of wavelengths. The bound we derive in this subsection is only useful for network with very large diameter.

Suppose the diameter of the network is given to be $\Delta$. Then there must be two nodes, say $n_0$ and $n_\Delta$, such that the shortest path from $n_0$ to $n_\Delta$ is $\Delta$ hops long. Let us suppose that a shortest path from node $n_0$ to $n_\Delta$ goes through nodes $n_1, n_2, \ldots, n_{\Delta-1}$ in that order.

The first thing to note is that the shortest path from node $n_i$ to $n_j$, $0 \leq i \leq j \leq \Delta$, must be $(j - i)$ hops long and a shortest path from $n_i$ to $n_j$ must be through nodes $n_{i+1}, n_{i+2}, \ldots, n_{j-1}$. If $\Delta$ is even, we take nodes $n_0, n_1, \ldots n_{\Delta/2}$ as origins and nodes $n_{\Delta/2}, n_{\Delta/2+1}, \ldots, n_\Delta$ as corresponding destinations. We have thus specified $(\Delta/2) + 1$ origin destination pairs, each requiring at least $\Delta/2$ hops. By assigning other nodes arbitrarily, we can be sure that the rest of the $N - \Delta/2 - 1$ origin destination pairs require one hop each. Now, we have a permutation routing problem in which the number of hops

$$H \geq \frac{\Delta}{2} \left( \frac{\Delta}{2} + 1 \right) + N - \frac{\Delta}{2} - 1$$

$$= \frac{\Delta^2}{4} + N - 1.$$

Similarly, if $\Delta$ is odd, we take nodes $n_0, n_1, \ldots n_{\Delta-1}$ as origins and nodes $n_{\Delta+1}, n_{\Delta+2}, \ldots n_\Delta$ as corresponding destinations. In this case, we have $(\Delta + 1)/2$ origin destination pairs, each requiring at least $(\Delta + 1)/2$ hops. Again, the other $N - (\Delta + 1)/2$ origin destination pairs must need one hop each. Hence

$$H \geq \frac{(\Delta + 1)^2}{2} + N - \frac{\Delta + 1}{2}$$

$$= \frac{\Delta^2 - 1}{4} + N.$$

44
In either case,

\[ H \geq \frac{\Delta^2}{4} + N - 1 \]  

and therefore the bound on the required number of wavelengths is

\[ l \geq \frac{1}{E} \left( \frac{\Delta^2}{4} + N - 1 \right). \]  

This calculation gives us a tighter bound than the one given by inequality 3.8 if the network diameter is larger than \( \sqrt{N \log N} \). In particular, this bound is tighter than inequality 3.8 if most of the nodes in a network are arranged in a line or ring with very few nodes away from it. It is easy to see that the diameter of such networks is approximately \( N \).

### 3.4 Rectangular Grid Topology

Let us look at the rectangular grid. Figure 6 shows a rectangular grid. We constrain the traffic to go either from left to right or from bottom to top. To ensure connectivity, the rectangular grid wraps around in such a way that the leftmost column of nodes is the same as the rightmost column and the topmost row of nodes is the same as the bottom row. It can be thought of as a rectangular grid on the surface of a doughnut.

We would like to show from very simple arguments that using a rectangular grid forces us to use \( \Omega(\sqrt{N}) \) wavelengths, where \( N \) is the total number of nodes in the network.

**Lemma 3.3** The longest path in a rectangular grid is of length at least \( 2\sqrt{N} \).

Suppose, the length and width of the network are \( \lambda \) and \( \omega \) respectively. Take any node, say \( n \), and look at the distance of the node which is one position below and one position to the left of the node, say \( n_{bl} \). In an undirected graph, the distance between the two nodes is two hops but in the directed case the distance from \( n \) to \( n_{bl} \) is \( \lambda + \omega \). Since \( N = \lambda \omega \), it is easy to check that \( \lambda + \omega \geq 2\sqrt{N} \). No two nodes in the network can be further apart. Hence the diameter of the rectangular grid is at least \( 2\sqrt{N} \).
Figure 6: The rectangular grid and its rotated version

The rectangular grid topology is uniform in the sense discussed in Subsection 3.3.2 and hence the bound given by equation 3.9 on the number of wavelengths translates to $l \geq \sqrt{N}$. The diameter of this network and the resulting lower bound on performance are too large for this topology to be satisfactory for our purposes. We would prefer to have a topology with a diameter of $O(\log N)$ and a number of required wavelengths close to the link counting bound of $\Delta/d_m$.

The reason for the large diameter of the network can be seen in the rotated version of the network as shown in figure 6. We can look at the rotated picture as a sequence of stages of nodes and the links always go from left to right or one stage to the next stage. It is easy to see that one node can only reach two nodes in the next stage, three nodes in the stage after that and so on. That is, in $h$ hops or less, a node can only reach $\frac{1}{2}(h+1)(h+2)$ nodes including itself. The number of nodes reached grows as the square of the number of hops. In networks with logarithmic diameters, this number should grow exponentially with $h$.

Another variation on the grid, known as the Manhattan grid, is to have alternating
flow directions. If a vertical line has traffic flowing from bottom to top, the next vertical line has traffic flowing from top to bottom and so on. Similarly, if one horizontal line has traffic flowing from left to right, the next one would have traffic flowing from right to left and so on. The diameter of this network is slightly smaller. Its diameter turns out to be $\sqrt{N}$ which is only smaller by a factor of two. Since the Manhattan grid is also uniform, the lower bound given by equation 3.9 says that we need at least $\sqrt{N}/2$ wavelengths for the Manhattan grid to be nonblocking. As we discussed before, we would like to find networks that can be nonblocking with fewer wavelengths.

### 3.5 The Debruijn Graph

We next consider a network based on the Debruijn graphs. The Debruijn graphs have small diameter and hence one might suspect that permutation routing can be done on these networks with a small number of wavelengths. We now show, however, that this suspicion is wrong.

Figure 7 shows an eight node Debruijn graph. A Debruijn graph has $2^k$ nodes. The nodes can be labeled by $k$ bit binary numbers. If the last $k - 1$ bits of node $a$ are the same as the first $k - 1$ bits of node $b$ then there is an edge from node $a$ to node $b$. From any given node we shall refer to the link that leads to a node with 0 as its last bit as the 0-link and the link that leads to a node with 1 as its last bit as the 1-link. Every node has in-degree and out-degree 2.

A $k + 1$ bit binary number denotes a link in a Debruijn graph. The first $k$ bits denote the origin node of the link and the last bit specifies whether it is a 0-link or a 1-link. Another way to look at it is to say that the $k$ bits beginning from the first bit specifies the origin node of the link and the $k$ bits beginning from the second bit specifies the destination node of that link. By extending this interpretation, we can write down a path of length $j$ by a binary number of $k + j$ bits. In a Debruijn graph, every $k + j$ bit number represents a $j$ hop path. The path starts from the node specified by the first $k$ bits and the $i$th node in the path is the $k$ bits starting from bit $i$. The destination node
Figure 7: The Debruijn graph
in the path is the last \( k \) bits in the \( k + j \) bits sequence. In each case, the bits are counted from the most significant bit to the least significant bit.

The diameter of a Debruijn network is \( k \). To see this, let us look at an arbitrary origin destination pair. By writing the \( k \) bits for the origin node followed by the \( k \) bits for the destination node, we get a sequence of \( 2k \) bits. This sequence specifies a \( k \) hop path from the origin to the destination. Hence, the diameter is at most \( k \).

Now, let us try to find a path from node \((0,0,\ldots,0)\) to node \((1,1,\ldots,1)\). At each step, the number of 1's in the node number can be increased by at most one. Hence, it will require at least \( k \) hops to reach node \((1,1,\ldots,1)\) from node \((0,0,\ldots,0)\).

We consider two easy ways to route connections in the Debruijn networks. One is shortest path routing and the other is called the \( k \) hop routing algorithm.

**The \( k \) Hop Routing Algorithm** The \( k \) hop routing algorithm is defined to be the algorithm that routes connections over the \( k \) hop path represented by the \( 2k \) bit binary number that is generated by concatenating the \( k \) bit origin node number and the \( k \) bit destination node number. Let \( R \) be the \( 2k \) bit binary number generated by concatenating the origin node number and destination node number.

Can the \( k \)-hop routing algorithm be used to make a wide sense non-blocking network for point to point connections using less than exponential (in \( k \)) wavelengths? The answer is no. We can create a permutation routing problem that needs at least \( \sqrt{N}/2 \) wavelengths using the \( k \) hop routing algorithm. The construction is as follows.

We want to be able to establish \( 2^k \) connections. The path of each connection can be represented by a \( 2k \) bit binary number as discussed above. Let us say that the \( 2k \) bit binary number \( R_i \) denotes connection number \( i \). Let us also assume that \( k \) is even. Let us partition the binary number \( R_i \) into four sets of consecutive \( k/2 \) bits. Fix the second and third parts of \( R_i \) to say, \( L = (l_{k-1} l_{k-2} \ldots l_0) \). All the paths represented by \( R_i \)'s that satisfy this constraint must go through node \( L \). But we can pick the first set of \( k/2 \) bits of \( R_i \) in \( 2^{k/2} \) different ways, each giving a path with a different origin. By picking the last set of \( k/2 \) bits in \( R_i \), we can provide a different destination for each path. This means
we can construct a set of point to point connections in which \(2^{k/2}\) paths go through the same node. Hence, at least \(2^{k/2-1}\) of the connections must use the same link. Since only one connection can be routed on one wavelength over a link, we need at least \(2^{k/2-1}\) or \(\sqrt{N}/2\) wavelengths (layers) to be able to establish a wide sense non blocking point to point network that uses the \(k\) hop routing algorithm in a Debruijn graph.

The same conclusion can be reached if \(k\) is odd. For this case, let us again partition \(R_i\) into four parts. The first part is the first \((k - 1)/2\) bits, the second part is the next \((k + 1)/2\) bits, the third part is the next \((k + 1)/2\) bits and the last part is the last \((k - 1)/2\) bits. Let us fix the second and third parts to some arbitrary sequence of bits. The fixed second and third parts make a \(k + 1\) bits sequence and hence specify a link. By varying the first and the last part we can get a set of \(2^{k-1}\) origin destination pairs that can be part of a permutation. All the \(2^{k-1}\) paths in this set must go through the same link and hence the permutation routing requires at least \(2^{k-1} > \sqrt{N}/2\) wavelengths.

**Shortest Path Routing**  Let us look at a way the shortest path can be computed in a Debruijn Network. Once again let \(R\) be the \(2k\) bit binary number formed by concatenating the origin node number and the destination node number. If the last \(l\) bits of the origin match the first \(l\) bits of the destination then the \(l\) bit matching sequence is repeated in \(R\). If we remove one of the repetitions of the \(l\) bit matching sequence, we are left with a sequence of \(2k - l\) bits, say \(R^* = (r^*_{2k-l-1}, r^*_{2k-l-2}, r^*_{2k-l-3}, \ldots, r^*_0)\). \(R^*\) denotes a \(k - l\) hop path from the origin to the destination. If \(l\) is the largest number such that the last \(l\) bits of the origin match the first \(l\) bits of the destination then the \(k - l\) hop path given by this algorithm is the shortest path from the origin to the destination.

How many wavelengths are needed if shortest path routing is used to make a wide sense non-blocking network for point to point connections in a Debruijn graph? Unfortunately the answer is the same as that for the \(k\) hop routing algorithm. The reason is that we can come up with a set of connections in which each shortest path requires \(k\) hops but the origin and destinations have the same property that was needed to show that the \(k\) hop routing algorithm needs at least \(2^{k/2-1}\) wavelengths. To see this, let us again assume
that $k$ is even. This time let us pick origins to be all the $2^{k/2}$ nodes which have node ID’s ending in $k/2$ 0’s and destinations to be all the $2^{k/2}$ nodes which have node ID’s beginning with $k/2$ 1’s. It should be easy to see that for every such origin destination pair $R = R_+$. In other words, the shortest path still has $k$ hops and each route must go through the node whose ID is $k/2$ 0’s followed by $k/2$ 1’s. Following the reasoning used for the $k$ hop routing, we see that at least $2^{k/2-1}$ or $\sqrt{N}/2$ wavelengths are needed to make such a network nonblocking. We can find a similar set of origin destination pairs if $k$ is odd by following the reasoning given in the $k$ hop routing case.

The algorithm discussed above did not present any improvement over the Manhattan topology in spite of the small diameter of the network. The problem this time is in the routing algorithms. The algorithms fix the route of a connection as soon as the origin and the destination are specified. They completely disregard the congestion caused by other connections. Both the routing algorithms we used are oblivious routing algorithms and we have already discussed that such routing algorithms require $\Omega(\sqrt{N}/d)$ wavelengths. We do not know if there is a routing algorithm that can be used on Debruijn graphs to do permutation routing using fewer wavelengths.
4 The Switching Networks

As we discussed in Chapter 2, the design of a backbone network can be seen as the same problem as designing a switch with additional constraints. In this chapter, we discuss some of the results from switch design that will be helpful to us. Part of the discussion in this chapter is taken from [Hui90]. The discussion on expander graphs is based on [LLK91].

Throughout this chapter we shall be discussing networks that can be used as switches. The concept of a node here is different from the concept of a node that we have discussed so far. In this chapter, we assume that the nodes do not generate their own traffic. All the networks discussed here have nodes arranged in stages. We can assume that there is a source attached to every input link of the nodes in the first stage. Similarly, there is a destination attached to every output link of the nodes in the last stage. There are no sources or destinations inside the network itself. The goal is to configure the switches in the nodes in such a way that desired connection patterns between the external sources and external destinations are realized. There are no cycles in the network. If $s$ is the number of stages in the network, the length of a path in the network is at most $s - 1$ hops. A successful path must reach from a source to a destination and hence the length of a successful path is always $s - 1$ hops, excluding the hop from the external source to the first stage and the hop from the last stage to the external destination.

The nonblocking properties of these switching networks can be defined in the same way that nonblocking properties are defined for a general network. We discuss results about rearrangeably nonblocking and strict sense nonblocking switching networks in this chapter. The problem with strict sense nonblocking discussed in Subsection 2.3.2 does not arise here because the switching networks have no directed cycles. In particular, in networks that are divided into $s$ stages, the number of hops in a path from an input to output is always $s - 1$ no matter what routing algorithm is used.

Let us look at a simple three stage network called a Clos network and get conditions that make it a rearrangeably nonblocking network. Figure 8 shows a Clos network.
Figure 8: The Clos Network

\[ m_1 = 2 \quad n_2 = r_3 = 2 \]
\[ r_1 = m_2 = 3 \quad n_1 = r_2 = m_3 = 5 \quad n_3 = 3 \]
As can be seen in the figure, there are three stages of nodes. The first stage has \( r_1 \) nodes and each node is an \( m_1 \times n_1 \) rearrangeably nonblocking switch. Similarly, in stage \( i \) there are \( r_i \) nodes and each node is an \( m_i \times n_i \) rearrangeably nonblocking switch. From each node in the first and second stages there is an edge to each of the nodes in the next stage. Thus, it is necessary that \( n_i = r_{i+1} \) and \( r_i = m_{i+1} \) for \( i = 1, 2 \). It should be kept in mind that this nonblocking property is for establishing permutation connections from the \( m_1r_1 \) inputs to the \( n_3r_3 \) outputs. The nodes in a Clos network do not act as sources or destinations.

### 4.1 Rearrangeably Nonblocking Switch

The following theorem, known as the Slepián-Duguid Theorem gives the condition for a Clos network to be a rearrangeably non-blocking switch. We state the theorem without proof. The proof can be found in [Hui90].

**Theorem 4.1** A three stage Clos network is rearrangeably nonblocking if and only if \( r_2 \geq \max(m_1, n_3) \). We can also state the condition as \( n_1 \geq m_1 \) and \( m_3 \geq n_3 \) which implies that each node in the first stage must have at least as many outputs as inputs and each node in the last stage must have at least as many inputs as outputs to make the Clos network rearrangeably nonblocking.

The interesting thing about the Clos network is that it suggests a recursive construction to make a large rearrangeably nonblocking switch using small switching nodes. An \( n \times n \) switch can be made by using \( n/2 \) \( 2 \times 2 \) switches as the first stage, \( n/2 \) \( 2 \times 2 \) switches as the last stage and two \( n/2 \times n/2 \) switches as the middle stage. The \( n/2 \times n/2 \) switches can in turn be made using \( 2 \times 2 \) and \( n/4 \times n/4 \) switches and so forth. If we start with \( n = 2^k \) for some integer \( k \), we end up with an \( n \times n \) rearrangeably nonblocking switch made only from \( 2 \times 2 \) switches. The resulting switching network is called a Beneš network. Figure 9 shows a Beneš network with 8 inputs and 8 outputs. It also shows the recursive structure of the Beneš network. A \( 2^{k+1} \times 2^{k+1} \) Beneš switch has \( 2^k \) nodes in each stage and \( 2k + 1 \) stages of nodes. Each node is a \( 2 \times 2 \) switch.
An $8 \times 8$ Beneš network. The boxes contain $4 \times 4$ Beneš networks.

Figure 9: A Beneš Network
It should be noted that the Beneš network is symmetrical about an axis going through the central stage of nodes. If we exchange the function of inputs and outputs, we get the same Beneš network again.

Figure 10 shows two of the topological equivalents of the Beneš network. They can also be used as rearrangeably nonblocking networks. They are all symmetrical about the middle stage of nodes, and hence exchanging the function of input and output does not change the network. We will refer to the second network shown in figure 10 as the back to back butterfly network. The name is given because half of the network is known as the butterfly network. We shall later talk about the connection between this network and a class of networks with $\Theta(\log N)$ degree.

We introduce the perfect shuffle network next.
4.1.1 The Perfect Shuffle Network

A perfect shuffle network is shown in Figure 11. There are \( N \) nodes in the network, arranged in \( s \) stages of \( 2^k \) nodes each. Figure 11 shows a network with \( k = 2 \) and \( s = 5 \). Each node has two incoming links and two outgoing links. The outgoing links of stage \( i \) connect to stage \( i + 1 \) for all \( i \neq s \).

To describe the connection pattern, let us number the nodes in each stage from 0 to \( 2^k - 1 \). We can express the node numbers as \( k \) bit binary numbers. The outgoing links from node \( b_k \ldots b_0 \) in stage \( i \) go to nodes \( b_k \ldots b_0 x \) in stage \( i + 1 \) where \( x \) can be either 0 or 1. This way, each node has links to two nodes in the next stage.

To make a \( 2^{k+1} \times 2^{k+1} \) rearrangeably nonblocking switch using perfect shuffle stages, the number of stages needed is \( 3k + 2 \) [WF81]. We later see that the properties of both the Beneš network and the perfect shuffle network can be used to our advantage.

4.2 Strict Sense Nonblocking Switch

A set of conditions can be derived to make a Clos network strict sense nonblocking. We discuss a different approach which is more useful for our purposes. It turns out that if we use a number of layers of Beneš networks in parallel, we can get a strict sense nonblocking
All layers are identical. They may be Beneš networks, some topological equivalent of it or \(2\log n - 1\) stages of perfect shuffle connection.

Figure 12: Cantor Network, a strict sense nonblocking switch

A Beneš network with \(2k+1\) stages of nodes has \(2^{k+1}\) inputs and \(2^{k+1}\) outputs. If we have \(k+1\) such Beneš networks in parallel and each source and destination can access each of the parallel layers then the whole layered network is strict sense nonblocking. We call such a parallel arrangement of Beneš networks a Cantor network. It turns out that the Beneš networks can be replaced by \(2k+1\) stage perfect shuffle networks and the resulting network is still strict sense nonblocking. The only constraint is that all parallel layers must be identical.

Figure 12 shows a Cantor network. The proof of the strict sense nonblocking property of Cantor networks is given in appendix A.
4.2.1 Expander Graphs

Let us think of a bipartite graph \((U, V, E)\). \(U\) and \(V\) are two disjoint sets of nodes and \(E\) is the set of edges each going from a node in \(U\) to a node in \(V\). Let us define \(v \in V\) to be a neighbor of \(u \in U\) if there is an edge between \(u\) and \(v\). Clearly, a node \(u\) has as many neighbors as the number of edges incident on it. The neighbor set of a set of nodes \(S \subseteq U\) is \(S' \subseteq V\) if each node in \(S'\) is a neighbor of at least one node in \(S\) and no node outside \(S'\) is a neighbor of a node in \(S\). In other words, \(S'\) is the set of all the nodes that are directly connected by edges to \(S\).

Let us look at special bipartite graphs in which the set of neighbors of each small enough subset of \(U\) is a large subset of \(V\). Such bipartite graphs are said to have the expansion property and are referred to as expander graphs. Definition 4.1 is a more precise statement of what we mean by a small enough subset of \(U\) and large subset of \(V\).

Bipartite graphs with the expansion property are of interest because a cascade of a number of special expander graphs in an appropriate manner can yield a strict sense non-blocking switch. The complexity of these switches made from graphs with the expansion property is smaller than that of the Cantor network.

**Definition 4.1** Given real numbers \(\alpha > 0\) and \(\beta > 1\), a bipartite graph \((U, V, E)\) is said to have the \((\alpha, \beta)\)-expansion property if, for every subset \(S\) of \(U\) with \(|S| \leq \alpha|U|\), \(S\) has edges to at least \(\beta|S|\) nodes in \(V\). \(\alpha\) and \(\beta\) must satisfy \(\alpha \beta|U| \leq |V|\). \(\beta\) is called the coefficient of expansion and \((U, V, E)\) is referred to as an \((\alpha, \beta)\)-expander graph.

It is easy to see that an \((\alpha, \beta)\)-expander graph is also an \((\alpha^*, \beta)\)-expander graph for all \(\alpha^* \leq \alpha\). An arbitrary bipartite graph \((U, V, E)\) is a trivial example of an \((\alpha, \beta)\)-expander graph with \(\alpha = |U|^{-1}\) and \(\beta = d\) where \(d\) is the smallest degree of a node in \(U\). For our purposes, we need classes of \((\alpha, \beta)\)-expander graphs with fixed \(\alpha\), \(\beta\) and degree but arbitrarily large \(|U|\) and \(|V|\).

We show an expander graph by a drawing as shown in figure 13. The two ellipses represent the sets of nodes \(U\) and \(V\). The straight lines connecting the two ellipses
represent the edges from $U$ to $V$. The smaller ellipses $S$ and $S'$ within $U$ and $V$ are shown to illustrate the expansion property. $S$ and $S'$ are subsets of $U$ and $V$. The dotted lines connecting the smaller ellipses denote that $S'$ is the set of neighbors of $S$. For $|S| \leq \alpha|U|$ we have $|S'| \geq \beta|S|$.

The figure shows the sets of nodes $U$ and $V$. $S$ is a subset of $U$ that has edges to $S'$ in $V$. $|S'| \geq \beta|S|$ if $|S| \leq \alpha|U|$.

Figure 13: An expander graph

Although we have defined expander graphs, we have not shown that there are any graphs with $\alpha|U| > 1$ that have the expansion property. In [LPS86], some explicit constructions for expander graphs are given. In [LLK91] a proof based on a random graph argument is given to show that classes of $(\alpha, \beta)$-expander graphs with constant $\alpha$, $\beta$ and degree but arbitrarily large $|U|$ and $|V|$ exist but no explicit construction is given. We shall present the random graph argument in appendix E. The argument constructs random graphs in which each node has the same degree and shows that at least some of the graphs constructed are $(\alpha, \beta)$-expander graphs. From this point on, we shall simply assume the existence of bipartite graphs in which each node in $U$ has the same degree, each node in $V$ has the same degree and and the graph has the desired expansion properties.

We next define a multi-Beneš network as a particular type of combination of bipartite graphs with the expansion property. A multi-Beneš network is a graph in which the nodes are arranged in $s$ stages where $s = 2k + 1$ for some integer $k$. Let us number the stages
from 0 to $2k$ from left to right. All the edges from stage $i$, $0 \leq i \leq 2k - 1$, go to stage $i + 1$. Let us suppose that the number of nodes in each stage is $T2^k$. In each stage the nodes are partitioned into subsets of equal size. We refer to these subsets as classes of nodes in each stage. The number of classes depends upon the stage. In stage $k$ the number of classes is 1. In stages $k + j$ and $k - j$ the number of classes is $2^j$, for all $0 \leq j \leq k$. For each stage $i$, we number the classes from 0 to $2^{k-i} - 1$ beginning from the top to the bottom.

A multi-Beneš network is shown in figure 14. For $0 \leq i \leq k - 1$, each class of nodes in stage $i$ together with a particular class of nodes in stage $i + 1$ forms an $(\alpha', \beta')$-expander graph. In particular, class $c$ of stage $i$ together with class $[c/2]$ of stage $i + 1$ makes an $(\alpha', \beta')$-expansion graph. It is easy to see that each class of nodes in stage $i + 1$ is involved in two $(\alpha', \beta')$-expander graphs with two different classes of nodes in stage $i$. Each $(\alpha', \beta')$-expander graph $(U, V, E)$ in these stages satisfies $|U| = |V|/2$. We know that any $(\alpha^*, \beta^*)$-expander is also an $(\alpha^{**}, \beta^*)$-expander graph for all positive $\alpha^{**} \leq \alpha^*$. We pick the value of $\alpha'$ such as $T\alpha'$ is an integer.

The connection is slightly different for stages $k$ to $2k - 1$. Let us look at stage $i$ for some $k \leq i \leq 2k - 1$. A class $c$ of nodes in stage $i$ forms two $(\alpha, \beta)$-expander graphs with the classes of nodes in stage $i + 1$. Once again the value of $\alpha$ chosen to represent the expansion property of these graphs is such that $T\alpha$ is an integer. One is with class $2c$ of the nodes in stage $i + 1$ and the other one is with class $2c + 1$. In each expander graph $(U, V, E)$, $|U| = 2|V|$. It should be noted that the expansion parameters for these expander graphs are different from the expansion parameters of the graphs formed by nodes in stages 0 through $k - 1$. We should be careful in interpreting figure 14. It seems symmetric about stage $k$. However, the network need not be symmetric about stage $k$. The important property of the network is that the individual bipartite graphs that constitute each stage are each $(\alpha', \beta')$-expander graphs in the first half of the stages and are $(\alpha, \beta)$-expander graphs in the second half of the stages.

It is possible to make a multi-Beneš network such that each node has in-degree and
out-degree $d$ for some $d > 2$ (see Appendix E). Clearly, the expansion factor for each expander graph depends upon the actual value of $d$. We shall assume in our future discussions of multi-Beneš networks that the in-degree and out-degree of each node is $d$ where $d$ is some integer greater than 2.

**Theorem 4.2** If each input $i$ is connected to each of the $T$ nodes of class $i$ in stage 0 and each output $j$ is connected to each of the $T$ nodes of class $j$ in stage $2k$ as shown in figure 14 then a multi-Beneš network is a strict sense nonblocking switch for permutation routing between $2^k$ inputs and $2^k$ outputs provided that $\alpha' \geq 2\alpha$, $\beta' > 2$, $\beta > 1$ and

$$T > \max \left\{ \frac{1}{2\alpha(\beta - 1)}, \frac{2}{\alpha' (\beta' - 2)} \right\}. \quad (4.1)$$

The proof of Theorem 4.2 is given in appendix C. The proof also provides a simple routing algorithm to find node disjoint paths.

The number of inputs and outputs in a strict sense nonblocking multi-Beneš switch is
different from the number of nodes in each stage of the network. We shall see in Chapter 5 that it is better for our purposes if the number of inputs and outputs equal the number of nodes in each stage. To achieve that, let us construct another network as follows. Suppose \( T = 2^\tau \) for some integer \( \tau \). We take a multi-Beneš network and after each class of \( T \) nodes in stage \( 2k \), we add \( \log T \) or \( \tau \) stages of perfect shuffle network. Similarly, before each class of \( T \) nodes in stage 0, we add \( \tau \) stages of perfect shuffle network. Now we have a network with \( 2(k + \tau) + 1 \) stages, each stage having \( 2^{k+\tau} \) nodes. The first \( \tau \) stages are \( 2^k \) parallel perfect shuffle networks, each with \( 2^\tau \) or \( T \) nodes per stage. The next \( 2k + 1 \) stages form a multi-Beneš network. And the last \( \tau \) stages are \( 2^k \) parallel perfect shuffle networks, each with \( T \) nodes per stage. Let us call this augmented network a shuffle-Beneš network. Figure 15 shows a block diagram of a shuffle-Beneš network. We refer to the \( 2^k \) perfect shuffle networks in the first \( \tau \) stages as perfect shuffle input networks and the \( 2^k \) perfect shuffle networks in the last \( \tau \) stages as perfect shuffle output networks. We number the stages from \(-\tau\) to \(2k+\tau\) so that the numbering of multi-Beneš stages is consistent with earlier discussion.

**Corollary 4.3** A network with \( 2T - 1 \) parallel layers of shuffle-Beneš networks is a wide sense nonblocking switch with \( T2^k \) inputs and \( T2^k \) outputs if each input and each output has access to all the layers and \( \alpha, \beta, \alpha', \beta' \) and \( T \) satisfy the conditions given in Theorem
4.2.

Proof We should notice that the number of nodes in each stage is \( T2^k \). Hence, each node in the input stage is connected to only one external input. The same is true in the output stage as well. This is different from our discussion of Beneš and perfect shuffle switches in which each node in the first stage connects to two inputs and each node in the last stage connects to two outputs. It is also different from the multi-Beneš network in which each input is connected to \( T \) nodes and each output is connected to \( T \) nodes.

Each perfect shuffle input network gives us a \( T \times T \) connector from external inputs to one class of nodes in stage 0 of the multi-Beneš. This connector is not nonblocking for permutation connections but it can establish one connection from an arbitrary input to an arbitrary node within its class of nodes in stage 0 of the multi-Beneš. Similarly, each perfect shuffle output network can establish one connection from an arbitrary node within one class of nodes in stage \( 2k \) of multi-Beneš to any one of the \( T \) external outputs connected to it.

Suppose a new connection arrives from a free input \( i_0 \) to a free output \( o_0 \). Let input \( i_0 \) be connected to class \( c_{i_0} \) of stage 0 and output \( o_0 \) be connected to class \( c_{o_0} \) of stage \( 2k \). The algorithm that is used to route connections on a shuffle-Beneš network has the property that the new connection is assigned to a layer that has no connections going through any node of class \( c_{i_0} \) in stage 0 and no connection going through any node of class \( c_{o_0} \) in stage \( 2k \). To show that the shuffle-Beneš network is wide sense nonblocking, we need to show that for permutation routing problems, a layer with this property can always be found. We also need to show that a path can be found in the layer assigned to a new connection.

We have \( 2T - 1 \) layers. Only \( T \) inputs can reach nodes in class \( c_{i_0} \). Since the new connection originates from a free input, at most \( T - 1 \) existing connections can be going through class \( c_{i_0} \). Each of these connections are occupying one layer and hence there is a connection going through a node of class \( c_{i_0} \) of stage 0 for at most \( T - 1 \) layers. By the same argument, there is a connection going through a node in class \( c_{o_0} \) for at most
$T - 1$ layers. Hence, there are at most $2T - 2$ layers on which either there is a connection going through a node in class $c_{t_0}$ of stage 0 or there is a connection going through a node in class $c_{o_0}$ of stage $2k$. There must be at least one layer in which there is no connection through class $c_{t_0}$ of stage 0 or through class $c_{o_0}$ of stage $2k$.

In the layer assigned to the new connection, there are no connections through class $c_{t_0}$ of stage 0. Hence, none of the inputs that share the input perfect shuffle network with input $i_0$ are using this layer. Similarly, none of the outputs that share the output perfect shuffle network with $o_0$ are using this layer. Moreover, if the same algorithm has established all previous connections then the assigned layer is being used by at most one input in each of the other input perfect shuffle networks. Similarly, the layer is being used by at most one output in each of the other output perfect shuffle networks. Effectively, this layer is being used as a $2^k \times 2^k$ switch. Each active input can reach all the nodes of one class in stage 0 through the input perfect shuffle network and each active output can reach all the nodes of one class in stage $2k$ through the output perfect shuffle network. The nodes from stage 0 to stage $2k$ is a multi-Beneš network. We have seen in Theorem 4.2 that a multi-Beneš network can establish such connections in a strict sense nonblocking manner. Hence, $2T - 1$ layers of shuffle-Beneš networks form a $T2^k \times T2^k$ wide sense nonblocking switch.

In the next chapter we use the properties of the switching networks to get upper bounds on the number of wavelengths needed for permutation routing on a linear lightwave network.
5 Upper Bounds on Number of Wavelengths

In this chapter we derive some upper bounds on the number of wavelengths required for permutation routing. So far, we have only one upper bound which states that the number of wavelengths required is at most $N$ for an $N$ node network. The bounds in this chapter are derived by example. We specify a network and by showing a routing algorithm or using existing results, we derive the number of wavelengths required for permutation routing on those networks. Other topologies may not be able to achieve as small a number of wavelengths as the ones obtained here but the results in this section give us an idea of what is achievable, especially if we are free to choose the topology within the restrictions on the degree.

As discussed earlier, having $l$ wavelengths on a network is like having $l$ copies of the same network. The copies do not interact with each other since there is no wavelength conversion. We call the copies of the network layers of the network. We use the term layers and wavelengths interchangeably. A connection from one node to another can be routed through any layer but it must stay in the same layer in which it originated.

Unlike the switching networks discussed in the last chapter, the nodes in the networks discussed in this chapter are both sources and destinations of traffic. Throughout this chapter, we assume that each node has a source and a destination attached to it. If the graph shows a node with in degree $d_{in}$ and out degree $d_{out}$, the LDC at that node must have $d_{in} + 1$ inputs and $d_{out} + 1$ outputs. The extra input is connected to the source residing at the node and the extra output is connected to the destination residing at the node.

We must put some constraints on the network. Different classes of networks are likely to have different results regarding the number of wavelengths required. We discuss some classes for which the degree of the graph is independent of the number of nodes. We also discuss classes of graphs for which the degree goes up as the logarithm of the number of nodes in the network. We treat the two classes separately.
5.1 Classes of Graphs with Constant Degree

Earlier, we discussed a number of graphs with constant degree. Classes of graphs like Beneš networks, perfect shuffle networks and Debruijn networks fall in this category. These graphs can be arbitrarily large but the in-degree and out-degree of each node in each of these graphs is 2.

We focus on the perfect shuffle network first. The Beneš network and other such networks are discussed later. As before, we denote the number of nodes in the network by \( N \). It is assumed that there are \( s \) stages in the network. The perfect shuffle connection pattern is used to connect each successive pair of stages. The first stage is regarded as the stage after the last stage. It is as if the entire network is on a cylinder and hence there is no distinction between the first stage of nodes and any other stage. The number of nodes in each stage is \( 2^k \) and hence \( N = s2^k \). We number the stages from 0 to \( s - 1 \) starting from some arbitrary stage for ease of reference. The numbering is done initially and remains fixed for the rest of the discussion.

5.1.1 Wide Sense Nonblocking Network

Can we make a wide sense non-blocking network with the perfect shuffle topology by using a number of wavelengths that is polylogarithmic in \( N \)?

The answer to this question is yes under certain conditions. Let us suppose that \( s \geq 4k \). Each stage still has \( 2^k \) nodes. We describe an algorithm for setting up connections from nodes of stage \( i \) to nodes of stage \( j \) using a set of \( k + 1 \) layers dedicated to the connections from stage \( i \) to stage \( j \).

Suppose \((j - i)(\mod s) \geq 2k\). This means that there are at least \( 2k + 1 \) stages of nodes between stage \( i \) and stage \( j \) including the end stages. If \( 2k + 1 \) stages of \( 2^k \) nodes are available in \( k + 1 \) layers, we know we have a Cantor network that can be used as a \( 2^{k+1} \times 2^{k+1} \) nonblocking switch and hence can route all the connections starting from the nodes of stage \( i \) and destined to the nodes of stage \( j \). Hence, using only \( k + 1 \) wavelengths we can route all the connections that originate in stage \( i \) and are destined to stage \( j \).
If \((j - i) \mod 8 < 2k\), the problem is somewhat more complex. Let us break the problem into three steps. The first step is to route the connection from stage \(i\) to some node in stage \(j\) which is not necessarily the final destination. The second step is to route it from the intermediate node in stage \(j\) to an appropriate node in stage \(i\). Obviously, this node in stage \(i\) is different from the origin node. Otherwise, we don't gain anything. The third step is to route the connection from the current stage \(i\) node to the correct destination in stage \(j\). The first and third steps of routing must be done over the same stages and hence it is important to make sure that we can find edge disjoint paths for these steps. To ensure that, we use an algorithm that gives us little choice in which intermediate nodes are reached. However, the second step ensures that no matter what intermediate nodes in stage \(j\) and stage \(i\) are chosen by the first and the third routing steps, an appropriate path can be found to complete the connection.

**Lemma 5.1** In a perfect shuffle graph with \(2^k\) nodes in each stage, it is possible to find \(2^{k+1}\) edge disjoint paths from stage \(i\) to stage \(j\) in which each node in stage \(i\) is the origin of two paths and each node in stage \(j\) is destination for two paths.

It should be noted that the lemma doesn't say anything about the origin destination pairs except the stages to which the origins and destinations belong.

Each stage of links in a perfect shuffle graph, combined with the two stages of nodes, is a bipartite graph. Each node in this bipartite graph has degree 2. From Hall's theorem [Ber91, Rys63] we conclude that we can partition the set of links in any given stage into two perfect matchings. The partition is not unique but we are only interested in one such partition.

The property discussed in Lemma 5.1 is not unique to the perfect shuffle graphs. Let us suppose we have a graph with \(s\) stages. The edges from each stage connect to the next stage. The edges from the last stage connect back to the first stage. Each stage has the same number of nodes. Suppose, in the bipartite graph formed by the nodes of stages \(\sigma\) and \(\sigma + 1\) and the edges between them, all the nodes have degree \(d_\sigma\). The we can find \(\min_\sigma(d_\sigma)\) edge disjoint paths from the nodes of an arbitrary stage \(i\) to some other stage.
This is because the edges between stage \( \sigma \) and stage \( \sigma + 1 \) can be partitioned into \( d_\sigma \) perfect matchings due to Hall's theorem. For our purposes we require only that the set of edges between two consecutive stages of nodes contain two disjoint perfect matchings.

Let us first look at the links that go from nodes of stage \( \nu \) to stage \( \nu + 1 \). We first identify two matchings and label them \( M_{\nu,0} \) and \( M_{\nu,1} \). We can do the same thing for each stage of links.

Now, to go from a node in stage \( i \) to a node in stage \( j \) we can follow the links in \( M_{i,0}, M_{i+1,0}, \ldots, M_{j-1,0} \). This provides us with a set of edge disjoint paths from nodes of stage \( i \) to the nodes of stage \( j \). We can find another set of paths using the links in \( M_{i,1}, M_{i+1,1}, \ldots, M_{j-1,1} \). This way, we have identified two sets of edge disjoint paths from each node of stage \( i \) to the nodes of stage \( j \). It should be noted that for each node in stage \( i \), we guarantee two paths from that node each going to some node in stage \( j \) but not to any particular nodes of stage \( j \).

Figure 16 shows an example of how the links of a stage can be partitioned. The links numbered 0 form \( M_{1,0} \) and the links labeled 1 form \( M_{1,1} \). It should be noted that this lemma is valid for Beneš networks and all of the equivalent networks as well. The only requirement is that out-degree of each node in a stage be the same and equal the in-degree of each node in the next stage and that the in-degree and out-degree of each node be at least 2.

Let us suppose we have identified the matchings \( M_{\nu,0} \) and \( M_{\nu,1} \) for all \( 0 \leq \nu \leq 2s - 1 \) as suggested in the proof of Lemma 5.1. We can define two functions from the set of nodes in stage \( i \) to the set of nodes in stage \( j \). Let \( \phi_{ij}(m_i) \) denote the node in stage \( j \) which is reached from node \( m_i \) of stage \( i \) by using edges from the matchings \( M_{\nu,1} \), and let \( \phi_{0ij}(m_i) \) denote the node in stage \( j \) which is reached from node \( m_i \) of stage \( i \) if edges from matching \( M_{\nu,0} \) are followed. These functions depend upon the choice of partition. We assume that the partition is chosen beforehand and hence the functions \( \phi_{0ij} \) and \( \phi_{1ij} \) are uniquely defined. Since these functions are one to one, we also talk about their inverses \( \phi_{1ij}^{-1}(m_i) \) and \( \phi_{0ij}^{-1}(m_i) \) with obvious meanings.
Figure 16: Two distinct paths from stage 1 to stage 2

Now we are ready to route all connections from stage $i$ to stage $j$. Suppose, we want to route a connection from node $m^i_1$ of stage $i$ to node $m^j_2$ of stage $j$. We first find nodes $\phi^{-1}_{1ij}(m^i_1)$ and $\phi^{-1}_{0ij}(m^j_2)$. If we can find a path from $\phi^{-1}_{0ij}(m^j_2)$ to $\phi_{1ij}(m^i_1)$ we are done. Now we know that $(i - j)(\text{mod } s) \geq 2k$ because $s \geq 4k$. Hence, with $k + 1$ wavelengths we have a Cantor network which can be used as a $2^{k+1} \times 2^{k+1}$ strict sense nonblocking switch. We only need a $2^k \times 2^k$ nonblocking switch to ensure that we can find a path from $\phi^{-1}_{0ij}(m^j_2)$ to $\phi_{1ij}(m^i_1)$. Since we have already identified a path from node $m^i_1$ to $\phi_{1ij}(m^i_1)$ and from $\phi^{-1}_{0ij}(m^j_2)$ to $m^j_2$, we have a path from $m^i_1$ to $m^j_2$ which can always be established as long as all the connections from stage $i$ to stage $j$ use the same algorithm.

This proves that with $k + 1$ wavelengths or layers, we can route all connections from stage $i$ to stage $j$ in a wide sense nonblocking manner. Since there are only $s^2$ such pairs of stages, the perfect shuffle topology gives us a wide sense nonblocking network in $s^2(k + 1)$ wavelengths where $s \geq 4k$ and $N = s2^k$. Clearly, for $s = 4k$, the number of wavelengths is $16k^2(k + 1)$ which in terms of asymptotic growth is $O((\log N)^3)$.

An improvement of almost a factor of 2 can be achieved in the number of wavelengths. Let us look first at the set of connections that go from nodes of stage $i$ to nodes of stage $j$ and second the set of connections going from stage $j$ to stage $i$ for some $i$ and $j$. We
have assigned two sets of \( k + 1 \) wavelengths to route the two sets but they can be routed in one wavelength. To do so, we notice that having \( 2k + 1 \) stages of \( 2^k \) nodes with \( k + 1 \) layers gives us a \( 2^{k+1} \times 2^{k+1} \) strict sense nonblocking Cantor network. But in both cases we used only half that capacity.

Let us assume that \((j - i)(\text{mod} s) \geq 2k\) and hence we have a \( 2^{k+1} \times 2^{k+1} \) switch from stage \( i \) to stage \( j \). To route all the connections from stage \( j \) to stage \( i \) we use the three step procedure described above. After this routing is done, we have one path from each node in stage \( i \) and one path coming to each node in stage \( j \). The Cantor network allows us to establish a set of connections in which two paths originate from each node in stage \( i \) and two paths terminate in each node in stage \( j \). Furthermore, these connections can be established in a strict sense nonblocking manner. Since the network from stage \( i \) to stage \( j \) is a Cantor network, there is still enough capacity left to route a set of connections such that each node in stage \( j \) is the origin of one connection and each node in stage \( j \) is the destination of one connection. Hence, all the connections from stage \( i \) to stage \( j \) can be routed on the same set of \((k + 1)\) wavelengths. This can be done for all \( i \neq j \). The wavelengths assigned to the connections going from a stage to itself cannot be shared by any other group. This reduces the total number of wavelengths required to \( s(s+1)(k+1)/2 \) which for \( s = 4k \) is \( 2k(4k+1)(k+1) \).

The argument given above uses the fact that half of the stages in the network give us a switch. We can write down the conclusion in a more general form as follows.

**Theorem 5.2** A network of \( N \) nodes, divided into \( s \) stages of \( N/s \) nodes each, is a wide sense nonblocking network in \( s^2 \Lambda \) wavelengths if the following conditions are satisfied:

1. For each stage, all the outgoing edges from that stage go to the next stage; the outgoing edges from the last stage go to the first stage. Any two consecutive stages along with the connecting edges form a bipartite graph in which two disjoint perfect matchings can be found.

2. From any arbitrary stage in the network, if we take the next \( s/2 \) stages then the
s/2 + 1 stages make a wide sense or strict sense nonblocking end to end N/s × N/s switch with Λ wavelengths.

If the resulting switch in condition 2 is a 2N' × 2N' switch then the required number of wavelengths is s(s + 1)Λ/2.

We shall apply Theorem 5.2 to Beneš and other networks.

5.1.2 Rearrangeably Non-blocking Networks

Can we do a lot better if we are looking only for rearrangeably non-blocking networks?

Once again we use the perfect shuffle network but we assume that the number of stages satisfies s ≥ 6k + 2.

If (j − i)(mod s) ≥ 3k + 1, we have a rearrangeably nonblocking switch between stages i and j. This switch can be used to route the connections in one layer.

If (j − i)(mod s) < 3k + 1, we know that (i − j)(mod s) ≥ 3k + 1. We first partition all the edges into two sets of matchings as discussed in the proof of Lemma 5.1. Once again, for each connection that originates from node m^i_j in stage i and is destined to node m^j_i in stage j, we do the routing in three steps. We first find the node φ_{ij}^{-1}(m^i_j) in stage j which is reached from m^i_j by following links numbered 1 and the node φ_{ij}^{-1}(m^j_i) in stage i from where node m^j_i is reached by following links numbered 0. Now the problem boils down to routing the connection from φ_{ij}^{-1}(m^i_j) in stage j to φ_{ij}^{-1}(m^j_i) in stage i in a rearrangeably nonblocking manner. But we already have a rearrangeably nonblocking switch from stage j to stage i and hence we can do this routing in one wavelength. Clearly, the first and the last steps of the routing can also be done in one wavelength.

One wavelength is needed to route all connections from a given stage i to another stage j. Since the total number of such stage pairs is s^2, we can make a rearrangeably nonblocking network with s^2 wavelengths. If s = 6k + 2 then N = (6k + 2)2^k and the total number of wavelengths required for a rearrangeably nonblocking network is (6k + 2)^2. A factor of close to 2 can be gained by routing all connections from stage i to stage j on the
same wavelength that is reserved for connections from stage \( j \) to stage \( i \). The argument given in the wide sense nonblocking case works in this case also. With the reduction, the required number of wavelength is \((6k + 2)(6k + 3)/2\). In terms of rate of growth, the number of required wavelengths is \(O((\log N)^2)\).

The general form of this result can be expressed as follows.

**Theorem 5.3** A network of \( N \) nodes, divided into \( s \) stages of \( N/s \) nodes each, is a rearrangeably nonblocking network in \( s^2 \Lambda \) wavelengths if the following conditions are satisfied:

1. For each stage, all the outgoing edges from that stage go to the next stage; the outgoing edges from the last stage go to the first stage. Any two consecutive stages along with the connecting edges form a bipartite graph in which two disjoint perfect matchings can be found.

2. From any arbitrary stage in the network, if we take the next \( s/2 \) stages then the \( s/2 + 1 \) stages make a rearrangeably nonblocking end to end \( N/s \times N/s \) switch with \( \Lambda \) wavelengths.

*If the resulting switch in condition 2 is a \( 2N/s \times 2N/s \) switch then the required number of wavelengths is \( s(s + 1)\Lambda/2 \).*

**5.1.3 Beneš and other similar networks**

A perfect shuffle graph with the last and first stages merged is a uniform topology in the sense that any node can be made to take the place of any other node by rearranging the nodes and links differently. This is not true for Beneš and other similar networks, though they have some other interesting and useful properties. In fact, the results for Beneš and similar networks are very similar to the ones we derived for the perfect shuffle graphs. Let us first look at one important property of Beneš networks.

The first half of the Beneš network is also known as a baseline network, and therefore the second half can be called an inverse baseline network as it is a mirror image of the baseline network. Figure 17 shows that by renumbering the nodes of a baseline network
appropriately, we can get an inverse baseline network. Hence, the baseline network and the inverse baseline network are topologically equivalent. The corollary is that the network in which an inverse baseline network is followed by a baseline network must also be a topological equivalent of Beneš network and hence be a rearrangeably nonblocking switch.

Wide Sense Nonblocking Networks We continue to assume that there are \( s \) stages of nodes, each stage consisting of \( 2^k \) nodes. Suppose, we cascade three Beneš networks and merge the last stage of nodes with the first stage of nodes. Since the interfacing node stages can be merged, the total number of stages is \( s = 6k \). If we take any \( 3k + 1 \) consecutive stages of nodes out of the \( 6k \) that we started out with, there is either a Beneš network contained in it or an inverse baseline network followed by a baseline network. If we have \( k + 1 \) layers available, we have a \( 2^{k+1} \times 2^{k+1} \) strict sense nonblocking switch. Clearly, the new network also satisfies Lemma 5.1. Hence from Theorem 5.2 we conclude that this network is wide sense nonblocking with \( s(s + 1)(k + 1)/2 \) wavelengths. The number of wavelengths can also be written as \( 3k(6k + 1)(k + 1) \) wavelengths which is \( O((\log N)^3) \). The number of wavelengths required for the Beneš network is higher by a constant factor compared to the perfect shuffle network which requires only \( 8k^2(k + 1)/2 \) wavelengths. The asymptotic growth of the number of wavelengths is the same in both cases.

Rearrangeably Nonblocking networks In the cascaded Beneš network, \( 3k + 1 \) stages beginning at any stage contains a Beneš network or an inverse baseline network followed by a baseline network. In either case, \( 3k + 1 \) stages make a \( 2^{k+1} \times 2^{k+1} \) rearrangeably nonblocking switch with one wavelength. We already know that the network satisfies the other condition of Theorem 5.3. Hence, the network is rearrangeably nonblocking with \( s(s + 1)/2 \) wavelengths. The number of wavelengths can also be expressed as \( 3k(6k + 1) \) which is asymptotically \( O((\log N)^2) \). The required number of wavelengths in this case is slightly smaller than the perfect shuffle network which requires \( (6k + 2)(6k + 3)/2 \)
Figure 17: Topological equivalence of baseline and inverse baseline networks
wavelengths.

5.1.4 Limits of This Approach

It should be noted that the link counting lower bound states that the number of wavelengths needed must be $\Omega(\log N)$ for constant degree graphs. The algorithms we have discussed for perfect shuffle networks do not achieve the lower bound. We do not know if the link counting lower bound is achievable.

In our discussions so far, we partition the nodes into stages and then group the connections based on their origin and destination stage. This gives a partition for the connections into $s^2$ subsets. In the case of rearrangeably nonblocking networks, each subset is assigned one wavelength and in case of wide sense nonblocking networks, each subset is assigned $k + 1$ wavelengths, and we have shown that this is sufficient for nonblocking routing of permutation routing problems.

Suppose we try to extend this approach as follows. Instead of partitioning on the basis of both origins and destinations, let us partition the connections only on the basis of their origins. One possible partition would be according to the stage in which the origin nodes belong. This is an $s$ way partition of connections. There can be a number of other ways to partition the set of nodes. Each partition of nodes gives us a different way of partitioning the set of connections based on which subset of nodes a connection originates from. After partitioning the set of connections, we assign one wavelength to each subset. We would like to be able to do nonblocking routing based on this wavelength assignment. We might expect to get nonblocking routing with fewer wavelengths under this strategy. The following example shows that this is not the case.

If we partition the connections on the basis of their origin nodes, we have no control over the destinations. Suppose we have a partition of the set of nodes into $P$ subsets. This gives us a partition of the set of connections into $P$ subsets. We assign one wavelength to route each subset of connections. We show that in a perfect shuffle network $P$ must be $\Omega(N)$ for this wavelength assignment to give us a nonblocking network.
Figure 18: A set of twelve nodes that cannot be reached in one wavelength
Let us look at the example shown in figure 18. It shows a perfect shuffle network with sixteen nodes in each stage. The figure shows only three stages of nodes but there can be more stages of nodes in the network. Twelve nodes in the figure are shaded. A look at the network shows that all the paths going to the shaded nodes must go through the eight numbered links in the first stage that are also shown by darker lines. Clearly, we can not find edge disjoint paths to the twelve nodes in one wavelength. Now let us suppose, we have divided the set of nodes into $P$ subsets but each subset contains more than twelve nodes. The connections with their origins in the same subset are routed on the same wavelength. We ask an adversary to choose the destinations for each subset of origins. One subset of origins can be assigned the destinations shown in the figure and we can be sure that nonblocking routing is impossible for that set of connections in one wavelength. The interesting property of the example shown in figure 18 is that we can cover most of the nodes by sets of twelve nodes such that no single set can be reached in one wavelength. The only requirement is that each stage must have sixteen or more nodes. Let us see how this can be done.

Let us first consider the case when each stage has exactly sixteen nodes. Let us also assume that nodes in each stage are numbered from 0 to 15 and are identified by their four bit binary representation. Let us pick three consecutive stages $i$, $i - 1$ and $i - 2$. For a given two bit sequence $b_1b_2$, we pick four nodes in stage $i$ with numbers $b_1b_2xx$ where $x$ can be either 0 or 1. We also pick four nodes from stage $i - 1$ with numbers $xb_1b_2x$ and four nodes from stage $i - 2$ with numbers $b_1b_2xx$. To reach any of the twelve selected nodes we must go through one of eight links coming into stage $i - 2$ to nodes $xxb_1b_2$. For each $b_1b_2$, we get a different set of twelve nodes. By varying $b_1b_2$ over all possible combinations, we get four sets of twelve nodes that cover stage $i$, $i - 1$ and $i - 2$. This way each three consecutive stages of nodes can be partitioned into four sets of twelve nodes. If the number of stages is a multiple of 3, we can cover all the nodes with these sets of twelve nodes. If not, we can cover all except at most two stages of nodes with these sets.
To extend the above construction of a set of twelve nodes to networks in which a stage has $2^k$ nodes for some $k \geq 4$ is simple. Once again we number the nodes in each stage from 0 to $2^k - 1$ and identify them by their $k$ bit binary representation. We pick consecutive stages $i$, $i - 1$ and $i - 2$. In stage $i$ we pick nodes of the form $b_1 b_2 \ldots b_{k-2} x x$, in stage $i - 1$ we pick nodes of the form $x b_1 b_2 \ldots b_{k-2} x$ and in stage $i - 2$ we pick nodes $x x b_1 b_2 \ldots b_{k-2}$. For any given sequence $b_1 b_2 \ldots b_{k-2}$ we have a set of twelve nodes and in each case the paths through the twelve nodes must go through the eight links coming into stage $i$ to nodes $x x b_1 b_2 b_3 \ldots b_{k-2}$.

This tells us that if we want to partition the connections only on the basis of their origins, each set of origins must have less than twelve nodes and hence the number of wavelength required is $\Omega(N)$. Partitioning the connections on the basis of destination has the same problem. In fact, if we use the set of twelve nodes discussed above as origins, the path from them must go through one of the eight links leaving stage $i$ from nodes $b_1 b_2 \ldots b_{k-2} x x$. We can make the same argument for Beneš networks and all networks that are topologically equivalent to a Beneš network. We should note that this example does not tell us anything about what happens when each group is assigned multiple wavelengths.

5.2 Classes of Graphs with $\Theta(\log N)$ Degree

We consider the hypercube network in the class of networks with $\Theta(\log N)$ degree. We represent the number of dimensions by $k$. The number of nodes $N$ equals $2^k$ in a hypercube. The nodes can be identified by a $k$ bit binary number. There is an undirected edge between two nodes if their numbers are different in exactly one bit position.

Let us look at a modified version of the hypercube. Let us assume that each single undirected edge of the hypercube is replaced by four directed links, two in each direction. Let us call this graph the doubled hypercube. The diameter of this graph is $k$, same as the original hypercube, but the in-degree and the out-degree of each node in this graph is $2k$ instead of $k$ for the original hypercube. We shall see that a doubled hypercube is
a rearrangeably nonblocking network because of its relationship with the back to back butterfly network which in turn is a topological equivalent of the Beneš network.

Figure 19 shows a two dimensional doubled hypercube. It is easy to see that the doubled hypercube graph is uniform in the sense discussed in Subsection 3.3.2. Equation 3.9 gives a lower bound on the number of wavelengths for uniform networks as $\Delta/d_{in}$ where $\Delta$ is the diameter and $d_{in}$ is the maximum in-degree of the network. Substituting the numbers, we get a lower bound of 1/2 wavelength for the doubled hypercube to be nonblocking. Since, the number of wavelengths must be a positive integer, we need at least one wavelength to make the doubled hypercube nonblocking. This is a trivial lower bound since there cannot be any communication without at least one wavelength.

We now try to find algorithms to route the connections over the doubled hypercube. We look at both the rearrangeably nonblocking and wide sense nonblocking case. Let us first look at the rearrangeably nonblocking case.

### 5.2.1 Rearrangeably Nonblocking Routing

First look at the back to back butterfly network shown in figure 20a. The graph is a topological equivalent of the Beneš network as we have already seen in Section 4.1. Hence, it is a rearrangeably nonblocking switch. If we want to do permutation routing from nodes of the first stage to the nodes of the last stage, we can do it in one wavelength. We next show that with appropriate modifications, the algorithm used in finding edge
disjoint paths for connections over a back to back butterfly can be used to find paths for connections over a doubled hypercube.

We number the nodes in each stage from 0 to $2^k - 1$. The node numbers can be written as $k$ bit binary numbers as shown in figure 20a. We notice that in each stage half of the edges connect nodes with the same number. The connection of the other half of the edges is as follows. The edges that go from stage 1 to stage 2 are between nodes that differ in their least significant bits. The edges that go from stage 2 to stage 3 are between nodes that differ in the second least significant bit. It can be verified that in general the edges from stage $i$ to stage $i + 1$ are between nodes that differ in the $i$th least significant bit for all $i \leq k$. Similarly, for $i > k$ the edges from stage $i$ to stage $i + 1$ are between two nodes that differ in the $(2k - i + 1)$th least significant bit. For each bit, say the $j$th least significant bit, there are exactly two set of edges, one from stage $j$ to stage $j + 1$ and the second from stage $(2k - j + 1)$ to stage $(2k - j + 2)$, in which the edges are between nodes that differ in the $j$th least significant bit position.

Recall that in the definition of a doubled hypercube with $2^k$ nodes, there are two directed links in each direction between two nodes which differ in exactly one bit position. The same is true in the back to back butterfly network as well. Let us look at figure 20a again. All nodes with the same number are enclosed in a dotted box. If we consider each dotted box as a node, the resulting graph is a doubled hypercube with $2k$ self loops at each node as shown in figure 20b.

We know that we can find edge disjoint paths for any permutation routing problem that connects nodes in stage 1 to nodes in stage $2k + 1$ on the back to back butterfly network in only one wavelength. From the equivalence discussed above, we can find edge disjoint paths for any permutation routing problem in the doubled hypercube with self loops in one wavelength under the rearrangeably nonblocking condition. But a self loop can be removed from the paths without affecting their edge disjoint property. Hence, rearrangeably nonblocking routing for permutation routing problems can be done on a doubled hypercube in only one wavelength.
a: The butterfly network

b: A 2-dimensional doubled hypercube with self loops

Figure 20: The back to back butterfly and the doubled hypercube networks
5.2.2 Wide Sense Nonblocking Routing

We have shown that the back to back butterfly network with $1 + \log N$ wavelengths is a strict sense nonblocking network for permutation routing problems if the origins are all in the first stage and the destinations are all in the last stage.

By the equivalence between the back to back butterfly network and the doubled hypercube with self loops, we conclude that the doubled hypercube with self loops is a wide sense nonblocking network with $1 + \log N$ layers or wavelengths. The number of wavelengths required in this case is $1 + \log N$. Once again the self loops can be removed from all the paths without affecting their edge disjoint property. Hence, the doubled hypercube is a wide sense nonblocking network with $1 + \log N$ wavelengths. The asymptotic growth of the number of wavelengths can be written as $O(\log N)$.

We cannot conclude that the doubled hypercube is a strict sense as opposed to wide sense nonblocking network because nonblocking routing can be done on the doubled hypercube only if we map the paths from the back to back butterfly network. If we use some other algorithm to route connections on the doubled hypercube with $1 + \log N$ layers, we may not be able to do nonblocking routing of connections.

5.3 Wide Sense Nonblocking Networks Using Multibeneš

We introduced the shuffle-Beneš network in Chapter 4 and showed that a $2T - 1$ layered shuffle-Beneš network can be used as a $T2^k \times T2^k$ strict sense nonblocking switch. We assume that $T = 2^\tau$ for some integer $\tau$.

Let us look at a network which consists of four shuffle-Beneš networks cascaded and wrapped around. The cascading is done such that the last stage of nodes of the first shuffle-Beneš is also the first stage of the second shuffle-Beneš and so on. By wrap around we mean that the last stage of the fourth shuffle-Beneš is also the first stage of the first shuffle-Beneš. We call this network a cascaded shuffle-Beneš network. The cascaded shuffle-Beneš can be viewed as if it is mounted on a cylinder.

The number of nodes in each stage is assumed to be $T2^k$. The number of stages
in one shuffle-Beneš is $2(k + \tau) + 1$, and hence the number of stages in the cascaded shuffle-Beneš network is $s = 8(k + \tau)$ and the total number of nodes in the network is $N = 8(k + \tau)2^{k+\tau}$. The cascaded shuffle-Beneš can also be seen as an alternating pattern of perfect shuffle networks and multi-Beneš networks. Let us assume that each node in the multi-Beneš parts of the shuffle-Beneš network has in-degree and out-degree $d$ where $d$ is some integer greater than 2. We know that the in-degree and out-degree of the nodes in the perfect shuffle part of the network is 2.

In the bipartite graph formed by two consecutive stages and their connecting edges each node has the same degree and this degree is at least two. Hence, from Hall’s theorem [Ber91, Rys63], we can find two disjoint matchings in each stage. This satisfies the first criterion listed in Theorem 5.2. As discussed in Chapter 4, the shuffle-Beneš is a nonblocking switch with $2T - 1$ layers, where $T$ satisfies equation 4.1. Since any $4(k + \tau)$ stages in the cascaded shuffle-Beneš network contain a shuffle-Beneš switch, the second criterion listed in 5.2 is satisfied. Hence, a cascaded shuffle-Beneš network is a wide sense nonblocking network with $(2T - 1)[8(k + \tau)]^2$ wavelengths. For a constant $T$, the number of wavelengths is $O((\log N)^2)$. We have seen in Theorem 4.2 that the value of $T$ depends only upon $\alpha, \beta, \alpha'$ and $\beta'$. We show in Appendix E that for a given degree $d$ that depends only upon $\alpha$ and $\beta$ we can make $(\alpha, \beta)$-expander graphs of arbitrary size. Hence, it is correct to assume that $T$ does not grow with the size of the network.

In terms of asymptotic growth, the cascaded shuffle-Beneš network is an improvement over the perfect shuffle network and the Beneš network. The asymptotic rate of growth of the number of required wavelength in perfect shuffle and Beneš networks is $O((\log N)^3)$.  

84
6 Other Routing Problems

We have concentrated on the problem of permutation routing so far. Although the permutation problem is important, there are other problems that need attention. We observed in Chapter 2 that a node in the network may be the access point for a large number of individual users. A session originating from a node is typically the aggregation of a large number of individual sessions. While the individual sessions are point to point, the aggregation of a number of them has multiple destinations. Hence, a node in the network must be able to establish connections with more than one other node at a time. Clearly, the permutation problem is not sufficient to model this situation.

A node can have connections with multiple destinations in two different ways. One way is to simply have independent connections on disjoint paths to each destination. The other way is to have a multicast connection. A session originating from a node is sent out on a tree that includes all the desired destinations. The tree need not span the entire network. Such a scheme requires that the LDC's in each node have multicast capability. Furthermore, all the nodes in the tree receive the same information from the session.

We shall discuss the all to all routing problem, the \( m \) to \( m \) routing problem and the multicast problem. They illustrate the two different approaches of connecting one origin node to several destination nodes. We shall use the techniques developed in analyzing the permutation routing problem to get good bounds for these problems.

6.1 \( m \) to \( m \) Routing

In this section we look at the \( m \) to \( m \) routing problem. This is a simple generalization of the permutation problem. In the permutation routing problem, each node can be the origin of at most one and the destination of at most one connection. In the \( m \) to \( m \) routing problem, each node can be the origin of at most \( m \) connections and the destination of at most \( m \) connections. At most \( mN \) connections can exist in the network at any given time. We must find edge disjoint paths for each of the connections. Since each node
participates in a number of conversations, each node must be able to transmit at more than one wavelength and receive at more than one wavelength at the same time. We shall later see that the routing algorithm allows one node to have only one conversation per wavelength. Hence, each node must have $m$ transmitters and $m$ receivers.

We first discuss some lower bounds on the number of wavelengths needed to find edge disjoint paths for the $m$ to $m$ routing problem. The lower bounds are based on the link counting argument.

### 6.1.1 Lower Bounds

To compute the link counting lower bound on the number of wavelengths needed to solve the $m$ to $m$ routing problem on a network, we first find a lower bound on the link usage in a worst case instance of the problem. We then find an upper bound on the number of links in the network. Since each link can be used at most once on any wavelength, we get a lower bound on the number of required wavelengths by dividing the lower bound on the link usage by the upper bound on the number of available links.

In Subsection 3.3.1, we showed that in any graph, the path from at least $N/2$ nodes to a given node $a$ must be at least $\lceil \log_{d_{ia}} \frac{N}{2} \rceil$ hops long. $N$ is the number of nodes in the network and $d_{ia}$ is the maximum in-degree of the network. Let us consider one node in the graph, say node $a$. We find a set of $N/2$ nodes such that the path from each node in the set to node $a$ is at least $\lceil \log_{d_{ia}} \frac{N}{2} \rceil$ hops long. We name this set $F_a$. Now, let $R = \{r_1, r_2, \ldots, r_{N/2}\}$ be an arbitrary set of $N/2$ nodes. For each $r_i \in R$, there is a set $F_{r_i}$ of $N/2$ nodes such that each of the nodes in $F_{r_i}$ require at least $\lceil \log_{d_{ia}} \frac{N}{2} \rceil$ hops to reach $r_i$. Let us look at two cases:

If $m \geq N/2$, we take the nodes in $R$ as destinations and for each destination $r_i$, take the nodes of $F_{r_i}$ as origins. These are the only connections we want to establish. This way, we have found a set of $(N/2)^2$ connections such that each node is the destination of at most $m$ connections and each node is the origin of at most $m$ connections. The path for each connection in this set is at least $\lceil \log_{d_{ia}} \frac{N}{2} \rceil$ hops
long. It follows that for \( m \geq N/2 \), the sum of the total number of hops for all connections is at least

\[
\left( \frac{N}{2} \right)^2 \log_{\log_{\log_2 N}} \frac{N}{2}.
\]

Next suppose \( m < N/2 \). We again take the nodes in \( R \) as destinations and for each destination \( r_i \), we want to mark \( m \) nodes in \( F_{r_i} \) such that each node is marked at most \( m \) times. Now, we use the marked nodes in \( F_{r_i} \) as origins for the destination \( r_i \). This way, we have identified \( mN/2 \) connections such that each node is the origin of at most \( m \) connections and the destination of at most \( m \) connections. We need to show that we can indeed mark \( m \) nodes in each \( F_{r_i} \), such that no node is marked more than \( m \) times.

To show that, we represent the situation in the form of a bipartite graph \( B = (U, V, E) \) with \( |U| = N/2 \) and \( |V| = N \). An edge connects node \( u_i \in U \) to \( v_j \in V \) if node \( j \) in the original graph is in set \( F_i \) associated with destination \( i \). Clearly, the degree of each node in \( U \) is \( N/2 \) and the degree of each node in \( V \) is at most \( N/2 \). The degree of a node in \( V \) can be 0. We shall prove the following theorem.

**Theorem 6.1** Let \( B = (U, V, E) \) be a bipartite graph with \( |U| = N/2 \). If each node in \( U \) has degree \( N/2 \) and each node in \( V \) has degree at most \( N/2 \), then we can find a set of edges \( E_m \subset E \) such that each node in \( U \) has exactly \( m \) edges from \( E_m \) incident on it and each node in \( V \) has at most \( m \) edges from \( E_m \) incident on it.

From the definition of bipartite graph \( B \), it should be clear that Theorem 6.1 implies that the desired set of \( mN/2 \) connections can be identified in the original graph.

We shall prove the following lemma first.

**Lemma 6.2** Let \( B' = (U', V', E') \) be a bipartite graph such that each node in \( U' \) has degree \( k \) and each node in \( V' \) has degree at most \( k \). Then we can find a set of edges \( E_1' \subset E' \) such that each node in \( U' \) has exactly one edge from \( E_1' \) incident on
it, each node in $V'$ has at most one edge from $E'_1$ incident on it and each node in $V'$ with degree $k$ has exactly one edge from $E'_1$ incident on it.

**Proof:** If we remove the last condition that each node in $V'$ with degree $k$ must have one edge from $E'_1$ incident on it, then we simply have the problem of finding a matching for $U'$ in $E$. A matching of $U'$ in $E'$ is simply a subset of $E'$ such that each node in $U'$ has exactly one edge from the subset incident on it and no two edges in the subset are incident on the same node. The necessary and sufficient condition for such a matching to exist is that each subset $U'_i \subset U'$ must have at least $|U'_i|$ neighbors [Rys63]. As we discussed in Subsection 4.2.1, $v$ is a neighbor of $u$ if there is an edge between $u$ and $v$.

Let us first take an arbitrary $U'_i \subset U'$ and say, $|U'_i| = i$. The number of edges incident on the nodes of $|U'_i|$ is $i k$. The set of neighbors of $U'_i$ is the set of nodes in $V'$ on which these $i k$ edges are incident. Suppose that set is $V'_{U'_i}$. Since at most $k$ edges are incident on each node of $V'_{U'_i}$, and the total number of edges incident on $V'_{U'_i}$ is at least $i k$, $|V'_{U'_i}| \geq i$. Hence, we can find a matching for $U'$ in $E'$. That is, we can find a set of edges $E'_{1U'_i}$ such that each node in $U'$ has exactly one edge in $E'_{1U'_i}$ incident on it and each node in $V'$ has at most one edge in $E'_{1U'_i}$ incident on it. But there is no guarantee that each node in $V'$ with degree $k$ has exactly one edge in $E'_{1U'_i}$ incident on it.

Let us look at the set of all the nodes in $V'$ with degree $k$. Let us call this set $V^{k'}$. Let $U^{k'} \subset U'$ be the set of neighbors of $V^{k'}$. Let us look at the subgraph $B^{k'}$ of $B'$ that consists of the nodes in $U^{k'}$, the nodes in $V^{k'}$ and the edges that go between those nodes. It is easy to see that in $B^{k'}$ the nodes in $V^{k'}$ have degree $k$ and the nodes in $U^{k'}$ have degree at most $k$. Hence, from the argument in the last paragraph, we can find a set of edges in $B'$ such that each node in $V^{k'}$ has exactly one edge from this set incident on it and each node in $U^{k'}$ has at most one edge incident on it. Let us call this set of edges $M_k$. 

88
Suppose no edge in \( E^{1'} \) is incident on a node \( v_{j_1} \in V^{k'} \). Clearly, in graph \( (U', V' - \{v_{j_1}\}, E' - E'_{v_{j_1}}) \), we can find a matching for \( U' \) in \( E' - E'_{v_{j_1}} \). In fact, \( E^{1'} \) is one such matching. Let us pick edge \( e_{v_{j_1}} \subseteq M_k \) that is incident on \( u_{i_1} \). Suppose, \( e_{v_{j_1}} = (u_{i_1}, v_{j_1}) \). There is exactly one edge in \( E^{1'}_{v_{j_1}} \) that is incident on \( u_{i_1} \). Let this edge be \( e^{1'}_{v_{j_1}} \). Now, let us look at the graph \( (U' - \{u_{i_1}\}, V' - \{v_{j_1}\}, E' - E'_{v_{j_1}} - E'_{u_{i_1}}) \), where \( E'_{u_{i_1}} \) is the set of edges in \( E' \) that are incident on \( u_{i_1} \). It should be noted that \( E' - E'_{v_{j_1}} - E'_{u_{i_1}} \) contains all the edges of \( M_k \) except \( e_{v_{j_1}} \). In this graph, we can find a matching for \( U' - \{u_{i_1}\} \) in \( E' - E'_{v_{j_1}} - E'_{u_{i_1}} \). In fact, \( E^{2'} = E^{1'} - e^{1'}_{v_{j_1}} \) is one such matching. Suppose there is another node \( v_{j_2} \in V^{k'} \) that has no edge in \( E^{1'} \) incident on it, we can again pick an edge \( e_{v_{j_2}} \subseteq M_k \) and repeat the process on a smaller graph. It is guaranteed that each smaller graph contains all the edges in \( M_k \) that are incident on the remaining nodes of \( V^{k'} \). Hence, each time a node in \( V^{k'} \) has no edge incident on it, we can find an edge in \( M_k \) that is incident on that node.

We continue this until we are left with a set of \( k - \gamma \) edges \( E^{\gamma'} \) in step number \( \gamma \) such that all the nodes in \( V' \) with degree \( k \) left at the step have exactly one edge from \( E^{\gamma'} \) incident on them. At this point, we look at the set of edges \( E'_1 = E^{\gamma'} \cup \{e_{v_{j_1}}, e_{v_{j_2}}, \ldots, e_{v_{j_{\gamma-1}}}\} \). The set \( E'_1 \) has exactly one edge incident on each node in \( U' \), at most one edge incident on each node in \( V' \) and exactly one edge incident on each node in \( V' \) that has degree \( k \).

**Proof of Theorem 6.1:** It is easy to see that bipartite graph \( B \) satisfies all the assumptions of Lemma 6.2 with \( k = N/2 \). Hence, we can find a set of edges \( E_1^1 \subseteq E \) such that exactly one edge in \( E_1^1 \) is incident on each node in \( U' \), at most one edge is incident on each node in \( V' \) and exactly one edge is incident on each node in \( V' \) with degree \( N/2 \).

Let us now look at the graph \( (U, V, E - E_1^1) \). Once again, this graph satisfies all the assumptions of Lemma 6.2 with \( k = \frac{N}{2} - 1 \). We can go ahead and find a set of
edges $E_1^2$ now with one edge in $E_1^2$ incident on each node in $U$, at most one edge incident on each node in $V$ and exactly one edge incident on each node in $V$ with degree $\frac{N}{2} - 1$ in the new graph. It is clear, that this process can be continued $m$ times for any $m \leq N/2$. The set of edges

$$E_m = \cup_{i=1}^{m} E_1^i$$

is a subset of $E$ with exactly $m$ edges in $E_m$ incident on each node in $U$ and at most $m$ edges incident on each node in $V$. ■

It follows that for $m < N/2$, we can find an instance of the $m$ to $m$ routing problem such that the sum of the total number of hops for all connections is at least

$$\frac{mN}{2} \left\lfloor \log_{d_{in}} \frac{N}{2} \right\rfloor .$$

Combining the case $m \geq \frac{N}{2}$ with $m < \frac{N}{2}$ we conclude that we can find an $m$ to $m$ routing problem such that the sum of the number of hops required for each connection exceeds

$$\min\{m, N/2\} \frac{N}{2} \left\lfloor \log_{d_{in}} \frac{N}{2} \right\rfloor .$$

The number of links in the network can be at most $Nd_{in}$. Hence the number of wavelengths required for $m$ to $m$ routing in any network is at least

$$\frac{\min\{m, N/2\} \left\lfloor \log_{d_{in}} \frac{N}{2} \right\rfloor}{2d_{in}} .$$

(6.1)

6.1.2 Upper Bounds on the Number of Wavelengths

In this subsection we discuss routing strategies on multistage perfect shuffle networks, Beneš networks, doubled hypercube networks and multi-Beneš networks that lead to establishing edge disjoint paths for the $m$ to $m$ routing problem with a small number
of wavelengths. Since the connections can arrive and depart, we have the notion of rearrangeably nonblocking routing and wide sense nonblocking routing. We discuss both.

To find routing algorithms for the \( m \) to \( m \) routing problem, we first partition the problem into a set of permutation routing problems. The number of wavelengths required for the \( m \) to \( m \) routing problem equals the number of wavelengths required for the corresponding permutation problem times the number of partitions.

**Rearrangeably Nonblocking Routing**

**Lemma 6.3** An \( m \) to \( m \) routing problem can be partitioned into \( m \) permutation routing problems.

Let us represent the \( m \) to \( m \) routing problem by a bipartite graph \((U, V, E)\). \( U \) has \( N \) nodes and it represents the origins, \( V \) has \( N \) nodes and it represents the destinations and \( E \) is the set of edges representing the connections. There is an edge connecting \( u_i \in U \) and \( v_j \in V \) if there is a desired connection from node \( i \) to node \( j \) in the original network. For any \( m \) to \( m \) routing problem, it is clear that the degree of each node in \( U \) and \( V \) in the corresponding bipartite graph is at most \( m \). Hence, from Hall’s theorem [Rys63], we can always partition \( E \) into \( m \) matchings. A matching is defined to be a set of edges in which no two edges are incident on the same node in either \( U \) or \( V \). A set of edges in which each node has exactly one edge incident to it is called a perfect matching. A matching in \((U, V, E)\) corresponds to a permutation problem in the original network. It should be noted that Hall’s theorem requires that each node has the same degree and that the partitions are perfect matchings. In our case, the degree of each node is at most \( m \) and different nodes may have different degrees. However, we can always add fictitious edges in the bipartite graph to make sure that each node has degree \( m \) and then do the partitioning. It is possible that we may have to add one or more fictitious edges between nodes that already have an edge between them. Afterwards, the fictitious edges can be removed from all the partitions and we are again left with a set of matchings.

Since we are talking about rearrangeably nonblocking networks, we assume that all
the desired connection of an \( m \) to \( m \) routing problem are given in advance. We know from Lemma 6.3 that we can partition the given \( m \) to \( m \) routing problem into a collection of \( m \) permutation routing problems. Hence, the number of wavelengths needed to find paths for the given \( m \) to \( m \) routing problem on any network is at most \( m \) times the number needed for permutation routing on that network. We have already studied the permutation routing problem in detail in Chapter 5. Here we simply state the results as they apply to the \( m \) to \( m \) routing problem.

We shall talk about three different networks in the context of rearrangeably nonblocking routing of the \( m \) to \( m \) routing problem. The first two networks have nodes divided into stages with edges going from one stage to the next. The degree of the first two networks is independent of the size of the network. The first one is a multistage perfect shuffle network with \( 2^k \) nodes in each stage and \( 6k + 2 \) stages. The second network is a cascaded Beneš network constructed by cascading three Beneš networks. The cascaded Beneš network also has \( 2^k \) nodes in each stage but has a total of \( 6k \) stages of nodes. These two networks are described in Chapter 5. Both the networks are wrapped around as if on the surface of a cylinder by identifying the last stage of nodes in a one to one fashion with the first stage of nodes. The third network is the doubled hypercube described in Chapter 5. The doubled hypercube has \( 2^k \) nodes. The in-degree and out-degree of each node is \( 2k \). In each case, we partition the \( m \) to \( m \) routing problem into \( m \) permutation routing problems. Hence, the number of wavelengths needed to route the \( m \) to \( m \) routing problem is exactly \( m \) times the number of wavelengths needed to route the permutation problem.

The number of wavelengths required to route the \( m \) to \( m \) routing problem under the rearrangeably nonblocking criterion on these networks is as follows:

- On a multistage perfect shuffle network, the number of wavelengths required is
  \[ m(6k + 2)^2 \]
  in a network with \( N = (6k + 2)2^k \). As \( N \) grows, the required number of wavelength approaches \( 36m(\log N)^2 \).

- On a cascaded Beneš network with \( N = 6k2^k \), the number of wavelengths required
is $36mk^2$. Once again the number of wavelength approaches $36m(\log N)^2$ as $N$ grows.

- On a doubled hypercube with $N = 2^k$, the number of wavelengths required is $m$. The required number of wavelengths remain constant but in-degree and out-degree of the nodes in this network grow as the logarithm of the size of the network.

In each case, we have just multiplied the number of wavelengths needed to route a permutation problem on the network by the factor $m$. Next, we look at the wide sense nonblocking condition.

Wide Sense Nonblocking Routing

**Lemma 6.4** If a network is wide sense nonblocking for the permutation routing problem with $l$ wavelengths then it is wide sense nonblocking for the $m$ to $m$ routing problem with $l(2m - 1)$ wavelengths.

**Proof:** Let us suppose we have a network that is wide sense nonblocking for the permutation problem with $l$ wavelengths. We now have the same network with $l(2m - 1)$ wavelengths available on it. Let us partition the set of wavelengths into classes of $l$ wavelengths. Clearly, there are $2m - 1$ classes of wavelengths.

We assert that the algorithm used to route connections on the network makes sure that the connections assigned to a particular class of wavelengths form a permutation problem or a partial permutation problem. We show that if such an algorithm is followed to route all the previous connections, a new connection can also be routed in a way that after the routing, each class of wavelengths is assigned a perfect or partial permutation problem.

Suppose that a set of $m$ to $m$ routing connections are in progress and a new connection arrives from node $i$ to node $j$. We look for a class of wavelengths which is not currently routing any connections from node $i$ and or any connections to node $j$. Since, the arrival of the new connection does not violate the $m$ to $m$ constraint by assumption,
there are at most \( m - 1 \) existing connections from node \( i \) and at most \( m - 1 \) existing connections to node \( j \). The maximum number of classes of wavelengths that these existing connections can occupy is \( 2m - 2 \) and hence there is a class of \( l \) wavelengths that has no connection from node \( i \) and no connection from node \( j \). Now, let us look at this class of \( l \) wavelengths. The existing connections being routed over the wavelengths in this class form a permutation problem. The addition of the new connection does not violate the permutation problem constraint. The network with \( l \) wavelengths is a wide sense nonblocking network by assumption. Hence, if the prescribed algorithm has been followed for routing the existing connection, a path for the new connection can be found.

This proves that on the network with \( l(2m - 1) \) wavelengths, there is an algorithm to route connections as they arrive as long as the new connection does not violate the \( m \) to \( m \) constraint. In other words, the network with \( l(2m - 1) \) wavelengths is a wide sense nonblocking network for the \( m \) to \( m \) routing problem.

In Chapter 5 we discussed how to route a permutation problem under the wide sense nonblocking condition on a number of networks. We shall state the results here with appropriate modifications for the \( m \) to \( m \) routing problem.

We shall talk about four different networks in the context of wide sense nonblocking routing of the \( m \) to \( m \) routing problem. The first three networks have nodes divided into stages with edges going from one stage to the next one. The degree of each of the first three networks is independent of the size of the network. The first is a multistage perfect shuffle network with \( 2^k \) nodes in each stage and \( 4k \) stages. The second network is a cascaded Beneš network constructed by cascading three Beneš networks. The cascaded Beneš network also has \( 2^k \) nodes in each stage but a total of \( 6k \) stages of nodes. These two networks are described in Chapter 5. The third network is the cascaded shuffle-Beneš network discussed in Section 5.3. A cascaded shuffle-Beneš network consists of four shuffle-Beneš networks. We have described the shuffle-Beneš network in Chapter 4. The cascaded shuffle-Beneš network also has \( 2^k \) nodes in each stage but the number of
stages is $8k$. It should be noted that the notation here for the number of nodes in each stage of a cascaded shuffle-Beneš is slightly different from the notation used in Section 5.3. All three networks are wrapped around as if on the surface of a cylinder by identifying the last stage of nodes in a one to one fashion with the first stage of nodes. The fourth network is the doubled hypercube described in Chapter 5. The doubled hypercube has $2^k$ nodes. The in-degree and out-degree of each node is $2k$. In each case, we partition the $m$ to $m$ routing problem into $2m - 1$ permutation routing problems. Hence, the number of wavelengths needed to route the $m$ to $m$ routing problem is exactly $2m - 1$ times the number of wavelengths needed to route the permutation problem.

The number of wavelengths required to route the $m$ to $m$ routing problem under wide sense nonblocking criterion on these networks is as follows:

- In a multistage perfect shuffle network the number of nodes is $N = 4k2^k$. The number of wavelengths required is $16k^2(k + 1)(2m - 1)$ which approaches $16(2m - 1)(\log N)^3$ for large $N$.

- On a cascaded Beneš network, where the number of nodes is $6k2^k$, the number of wavelengths required is $36k^2(k + 1)(2m - 1)$. The number of required wavelengths approaches $36(2m - 1)(\log N)^3$ for large $N$.

- In a cascaded shuffle-Beneš network, the number of nodes is $8k2^k$. The number of wavelengths required is $64k^2(2m - 1)(2T - 1)$ which is less than $64(2m - 1)(2T - 1)(\log N)^2$. The reduction in the number of wavelengths for large $N$ comes at the expense of a larger in-degree and out-degree for each node. The in-degree and out-degree are still independent of the size of the network.

- In a doubled hypercube the number of nodes is $2^k$. The number of wavelengths required for the doubled hypercube is $(2m - 1)(k + 1)$ which approaches $(2m - 1)\log N$ for large $N$. The in-degree and out-degree of the nodes in this network grow as the logarithm of the size of the network.

95
6.2 All to All Routing

The all to all routing problem is a special version of the $m$ to $m$ routing problem. In this case $m = N$. In other words, each node wants to talk to all the other nodes simultaneously. However, we want a distinct disjoint path between every pair of nodes. A node can thus be simultaneously involved in separate conversations with all the other nodes of the network. The number of edge disjoint paths we must find is $N(N - 1)$.

The all to all routing problem is static in nature. There are no new paths added or old paths taken away. All the $N(N - 1)$ paths can be established in advance and the network may never need reconfiguration. All the origin destination pairs are specified before edge disjoint paths for each connection are computed. Effectively, the situation here is the same as in the rearrangeably nonblocking case discussed in Section 2.3. Unlike the permutation routing problem or the $m$ to $m$ routing problem, there is no concept of wide sense nonblocking or strict sense nonblocking in this problem, since the connections never have to be taken down and there are no new connections for which paths must be found. Because of the static nature and the symmetry of the all to all routing problem, it is possible to do nonblocking routing on simpler networks.

We use the lower bound for the all to all routing problem that was derived for the $m$ to $m$ routing problem in Subsection 6.1.1 and substitute $m = N$. The bound is a link counting bound. No nontrivial bounds can be obtained by counting the number of states in this case since there is only one state in which the network must remain.

6.2.1 Upper Bounds on the Number of Wavelengths

Since the all to all routing problem is a special case of the $m$ to $m$ routing problem, all the results for the $m$ to $m$ routing problem are valid for the all to all routing problem as well. Fortunately, there is more symmetry in the all to all routing problem. This symmetry allows us to reduce the difference between the lower bound and the achievable upper bound on the number of required wavelengths. We shall also see that we can do all to all routing with relatively few wavelengths on simpler networks than the ones used
in the discussion of the $m$ to $m$ routing problem.

To exploit the symmetry in the all to all routing problem, we take a slightly different approach in routing these connections. Instead of looking at the perfect shuffle network, we shall look at a butterfly network. Having two butterfly networks back to back makes a Beneš network. A butterfly itself is topologically equivalent to a perfect shuffle network with $k$ stages. The butterfly network is chosen because it is easier to describe the routing algorithm on it.

The network of our choice is the cascaded butterfly which is simply two butterflies cascaded. This is different from the Beneš network because in the Beneš network the back to back butterflies are mirror images of each other. The cascaded butterfly has $s = 2k$ stages and $2^k$ nodes in each stage. The total number of nodes in the network is $N = s2^k$. Figure 21 shows a cascaded butterfly network with $k = 2$.

Let us number the stages from 0 to $2k - 1$ and the nodes in each stage from 0 to $2^k - 1$. We can represent the node number as a $k$ bit binary number. It can be seen from the figure that there are two kinds of edges in the network. One set of edges connect two nodes with the same node number in consecutive stages. We call such edges straight
edges. The other set of edges connect two nodes that differ in one bit. We call such edges cross edges. In other words, in going from one stage to the next on a cross edge, one bit of the node number is complemented. In any given stage, all the cross edges complement the same bit. The position of the bit complemented by the cross edges rotates to the right as we move from one stage to the next. Clearly, to complement all bits one pass on the cross edges through $k$ successive stages is enough.

Suppose we are given a $k$ bit number $C$ and we need to connect node $n_0$ of stage $i$ to node $n_0 \oplus C$ in stage $i + k \pmod{s}$, where $\oplus$ is the bitwise exclusive or operation. Since the path is exactly $k$ hops long, by choosing a straight or a cross edge at each stage we can complement whichever bits we choose of $n_0$. By choosing to complement the bits such that the end result is $n_0 \oplus C$, we can find the desired path. Now suppose, we find a path from each node $n$, for $0 \leq n \leq 2^k - 1$, in stage $i$, to node $n \oplus C$ in stage $i + k$ in the same way. Each path starts out in stage $i$ and traverses the edges one stage after next. If one path takes a cross edge, all other paths also must take cross edges. Similarly, if one path takes a straight edge, all other paths must take the straight edge also. This is because all paths need to complement the exact same bits. But since each path starts out on a separate node, the only way two paths can meet at a node is if one path takes a straight edge and the other one takes a cross edge, which never happens. Hence, the paths found for each $n$ are node disjoint. We know that all node disjoint paths are also edge disjoint. Hence, we have the following lemma.

**Lemma 6.5** Given a stage $i$ in the cascaded butterfly and a $k$ bit number $C$, we can find edge disjoint path from each node $n$, $0 \leq n 2^k - 1$, of stage $i$ to node $n \oplus C$ in stage $i + k \pmod{s}$.

Let us now try to solve the all to all routing problem on the cascaded butterfly. There are $N(N - 1)$ connections that need to be established. Let us also add one extra connection for each node in the set of all connections. These connections go from a node to itself. Now we have a set of $N^2$ connections. First, we take all the connections and partition them into $s^2$ sets. A set is identified by the ordered pair $(p, q)$ where
$0 \leq p, q \leq s - 1$. The set $(p, q)$ contains all the connections that start in stage $p$ and end in stage $q$. Then each set $(p, q)$ of connections is partitioned into $2^k$ groups. Each group is identified by a $k$ bit binary number. In group $C$ of set $(p, q)$ we have all the connections that go from node $n$ of stage $p$ to node $n \oplus C$ of stage $q$, for $0 \leq n \leq 2^k - 1$. Each group has $2^k$ nodes and there are $s^2 2^k$ groups. The total number of connections in all groups is $s^2 2^k$ which equals $N^2$. It is easy to check that no two groups contain the same connection and all connections in each group are distinct.

Now, we assign one wavelength to each group. We claim that all the connections in a group can be routed in one wavelength.

Let us look at two cases again. In each case we are looking at a group $C$ in set $(p, q)$.

Suppose there are at least $k$ stages from stage $p$ to stage $q$. That means going from stage $p$ to stage $p + k \pmod{s}$, we do not go past stage $q$. In that case, from each node $n$ in stage $p$ we can find disjoint paths to node $n \oplus C$ in stage $p + k \pmod{s}$ according to Lemma 6.5. Since $n \oplus C$ is also the number of the destination node in stage $q$, we can simply take straight edges from stage $p + k \pmod{s}$ to stage $q$ to finish routing.

Suppose there are less than $k$ stages from stage $p$ to stage $q$. That means going from stage $p$ to stage $p + k \pmod{s}$, we go past stage $q$. In that case, from each node $n$ in stage $p$ we take cross edges until we reach stage $q$. The nodes reached are numbered $n \oplus D$ where $D$ is the address of the node in stage $q$ reached after following cross edges from node 0 of stage $p$. If $C \neq D$, the nodes reached in stage $q$ are not the correct destinations. Let $C' = C \oplus D$. From each node $n \oplus D$, we can reach node $n \oplus D \oplus C'$ in stage $q + k \pmod{s}$ according to Lemma 6.5. Since $s = 2k$, we would not cross stage $p$ before reaching stage $q + k \pmod{s}$. The nodes reached in $q + k \pmod{s}$ have the node number $n \oplus C$ which is the desired destination's node number but it is in the wrong stage. We notice that the straight edges from stage $q + k \pmod{s}$ to $p$ and then from stage $p$ to stage $q$ are unused. By using them, we reach the correct destinations in stage $q$.  

99
This way we can route each group in one wavelength. Since the total number of groups is \( s^2 2^k \), the number of wavelengths required to route the all to all routing problem is \( s^2 2^k \). Since \( s = 2k \), the number of wavelengths is \( 4k^2 2^k \) which is less than \( 2N \log N \). This is within a factor of 4 of the lower bound.

It can be seen that a wrapped around multistage perfect shuffle network with \( 2k \) stages and \( 2^k \) nodes in each stage is topologically equivalent to the cascaded butterfly we discussed above. Hence the same result is valid for the multistage perfect shuffle also.

**Hypercube Networks** A hypercube is usually viewed as an undirected graph. The hypercube we discuss here is a directed graph obtained by replacing each undirected link in the undirected hypercube by one directed link in each direction. The number of nodes in a hypercube is \( N = 2^k \) and the in-degree and out-degree of each node is \( k \). Each node can be given a node number from 0 to \( 2^k - 1 \) that can be expressed as a \( k \) bit binary number. The nodes are numbered in such a way that each link connects two nodes whose numbers differ in exactly one bit. An edge that connects two nodes differing in the \( i \)th bit is said to be in the \( i \)th dimension. There are \( k \) dimensions of edges and each dimension contains \( N \) directed edges. Each node has one edge coming into it in each dimension and each node has one edge going out of it in each dimension. We number the dimensions from 0 to \( k - 1 \) arbitrarily.

Suppose we want to find a path from each node \( n \) to node \( n \oplus C \) where \( C \) is an arbitrary \( k \) bit number \((c_0, c_1, \ldots, c_{k-1})\). Let us suppose \( l \) bits of \( C \) are 1 and they are \((c_{i_0}, c_{i_1}, \ldots, c_{i_{l-1}})\) where \( i_0 < i_1 < \ldots < i_{l-1} \). Then from each node, we can follow the edges in dimension \( i_0 \) then dimension \( i_1 \) and so on to reach the destination. It is easy to see that all paths are edge disjoint since in each step the edges used for different paths are different and in two different steps edges of two different dimensions are used. Hence we get the following lemma.

**Lemma 6.6** *In a hypercube, from each node \( n \), we can find disjoint paths to nodes \( n \oplus C \) where \( C \) is some \( k \) bit binary number.*
Let us now look at the all to all problem. There are $N(N - 1)$ connections that need to be established. We can partition these connections into $N - 1$ or $2^k - 1$ groups. We number the groups from 1 to $2^k - 1$. In group $C$ we include all connections that go from a node $n$ to node $n \oplus C$. We do not include $C = 0$ because that only gives us connections that go from a node to itself and these are not included in the set of of connections. It is easy to see from Lemma 6.6 that we can find edge disjoint paths for each group of connections in one wavelength. Hence, the number of wavelengths required for routing the all to all problem on a hypercube is $2^k - 1$ or $N - 1$. We can save almost a factor of 2 by noticing that we can route group $C$ and $C'$, where $C'$ is the bitwise complement of $C$, on one wavelength. The number of wavelengths required is reduced to $N/2$.

The lower bound derived states that the number of wavelengths required for the doubled hypercube is $\Omega(N/\log N)$.

### 6.3 Multicasting Connections

In our discussions so far, we have assumed that the LDC at each node can connect only one output to each input. In this section, we shall assume that the LDC's have multicasting capability. Each output of an LDC can still be connected to only one input but each input can be connected to an arbitrary number of outputs. In terms of the matrix $A^w$ of an LDC for a given wavelength $w$ (see Section 1.3), the number of non zero elements in a row can be at most one while the number of non zero elements in a column can be more than one.

The routing problem that motivates the change in the LDC model is the problem of multicasting. So far we have assumed that each node either wants to have a connection with only one other node, or if it wants to have connections with multiple nodes, it sets up independent connections over edge disjoint paths with each of them. What if a node wants to send the same message to $m$ other nodes? We can still establish $m$ separate connections from the origin node to each of the desired destinations. But we can also use the multicasting capability of the LDC to achieve the same objective more efficiently.
The particular routing problem we shall try to solve is as follows. As before, there are \( N \) nodes in the network. We want to establish multicast connections. Each node can be the destination of at most one multicast connection. There is no limit on the multiplicity, i.e. the number of destinations, for each multicast connection except the obvious limit that the multiplicity of a multicast connection cannot be more than \( N \). It should be noted that some of the nodes may not be origins of any connection because a node cannot be the destination for two connections. We would like to know if such a routing problem can be solved in relatively few wavelengths. Although, the problem itself does not require a node to be able to transmit on multiple wavelengths at any given time, we shall assume that the nodes have more than one transmitter and they can transmit on more than one wavelength at the same time. The reason is that if a tree connecting the origin to all the destination nodes cannot be established on one wavelength, we want to be able to partition the set of destinations into smaller subsets and establish smaller trees, each on a separate wavelength, each connecting the origin to the subset of destinations. This will require that the origin has as many transmitters as the number of subtrees so that it can transmit the same message on all the subtrees simultaneously.

Apart from the need for a node to establish multicast connections with more than one other node, there are other reasons why the capability to establish such connections is important. Let us first think of a situation in which one node of a backbone network generates just enough traffic to fill up one wavelength channel. The traffic generated at this node is the aggregation of a number of individual connections and hence parts of the traffic are supposed to go to different destinations. In such a case, we can establish a multicast connection from the origin node to all the destinations and transmit all the information on the multicast connection using a time or frequency multiplexing scheme. Only part of the information reaching a particular destination is meant for it. The destination nodes can simply extract the information they need and ignore the rest. There is some amount of wasted bandwidth associated with this scheme but if the multicast connections can be established efficiently, the amount of wastage may be smaller than the
wastage associated with having separate connections from the origin to each destination. The reason behind this intuition is easy to see. To reach \( n \) destination nodes from an origin on a multicast connection, we can use a binary tree. If all the destinations are \( \log n \) hops away from the origin and we can find a binary tree with depth \( \log n \) which has all the destinations as its leaves, the number of links used in this tree is only \( 2n - 1 \). If, on the other hand, we establish separate paths from the origin to the destinations, we need to establish \( n \) disjoint paths and each one must use at least \( \log n \) links. The number of links used in this case is \( n \log n \) which is much larger.

The second scenario in which the ability to do multicasting is useful is if the reconfiguration time for the LDC is much larger than the tuning time for transmitters and receivers. With the ability to have multicast connections, we can treat the LDC in a quasi static state. In this case, we would want to start with a generalized model of multicasting problem in which each node can be the destination of up to \( m \) multicast connections, each on a different wavelength. A destination node can receive from different multicast connections by simply tuning its receiver to different wavelengths. In this configuration, the origin sends messages for different destinations over a multicast connection using time division multiplexing. The destinations receive by tuning their receiver to the right wavelength in the right time slot. The extreme case of such a scenario is that the LDC’s never reconfigure. In this case, they are equivalent to what are known as "wavelength routers." [Bar92b] discusses some issues pertaining to networks with wavelength routers.

We shall only study the multicast problem in which each node is the destination of at most one multicast connection. The permutation routing problem is a special case of the multicast problem in which the multiplicity of each connection is 1. Hence, the lower bounds on the number of wavelengths derived for the permutation routing problem are also valid for the multicast problem as defined here.
6.3.1 Generalized Connectors

A generalized connection is simply a set of multicast connections from a set of inputs to a set of outputs such that each output is the destination of at most one connection. A network that can establish such connections in a nonblocking manner is called a generalized connector. In such a connector, the inputs and outputs are external to the network itself.

We are interested in wide sense nonblocking generalized connectors. We define two kinds of wide sense nonblocking generalized connectors:

**Weakly wide sense nonblocking generalized connector:** In a weakly wide sense nonblocking generalized connector, there is an algorithm such that an edge disjoint set of trees for a new multicast connection can be found by the algorithm as long as all the outputs of the multicast connection are specified upon the arrival of the connection and the previous connections are routed using the same algorithm.

As mentioned before, a single multicast connection may have to be routed on more than one tree, each on a different layer, if one tree connecting the input to the all outputs can not be found within one layer.

**Strongly wide sense nonblocking generalized connector:** In a strongly wide sense nonblocking generalized connector, there is an algorithm such that an edge disjoint set of trees for a new multicast connection can be found by the algorithm as long as all the previous connections are routed using the same algorithm. In this case, an output can be added or deleted to an existing multicast connection using the same algorithm.

It turns out that by using graphs with the expansion property, we can make generalized connectors. We introduced the multi-Beneš network in Chapter 4. In appendix C it is shown that a multi-Beneš network can be used as a weakly wide sense nonblocking generalized connector. As we observed before, the number of nodes in each stage of the multi-Beneš is larger than the number of inputs and outputs it can connect. For our
purposes, we need connectors in which each stage has the same number of nodes as the number of inputs and outputs. We again use the shuffle-Beneš network which is also a weakly wide sense nonblocking generalized connector in $2T - 1$ layers. Recall that the inputs and outputs in a shuffle-Beneš network are grouped into classes of $T$ inputs and $T$ outputs and only one input or output from a class can use a given layer for a connection. Because of the structure of the shuffle-Beneš, more than one output of one class can be reached by a single output. To understand this let us recall that a shuffle-Beneš network has $\tau$ or $\log T$ stages of input perfect shuffle networks followed by a multi-Beneš network followed by another $\tau$ or $\log T$ stages of output perfect shuffle networks. A node in the last stage of the multi-Beneš part of the network can reach all the outputs of its class using its output perfect shuffle network. Still, we may not be able to find the tree reaching all the outputs of a multicast connection in one layer. Suppose a multicast connection needs to reach outputs $o_1$ and $o_2$ belonging to two different classes, say $c_1$ and $c_2$ respectively. Output $o_1$ can be reached in only those layers which are not being used by other outputs of class $c_1 - 1$. Similarly, output $o_2$ can be reached in only those layers which are not being used by other outputs of class $c_2$. Also, the input of the connection can only use those layers that are not being used by other inputs of its class. Hence, there may be no layer on which both outputs $o_1$ and $o_2$ can be reached and is also usable by the input. Since we have $2T - 1$ layers available, the input can reach at least $T$ layers and the set of outputs can be partitioned into at most $T$ subsets such that each subset can be reached by the input in one layer.

We now want to look at a network that is a strongly wide sense nonblocking generalized connector. Let us introduce the copy-Beneš network. The nodes in a copy-Beneš network are arranged in stages of $2^k$ nodes each for some integer $k$. All the edges from the nodes of one stage go to the nodes of the next stage. The number of stages is $3k + 1$. The last $2k + 1$ stages of the network form a shuffle-Beneš network with values of $\alpha, \alpha', \beta, \beta'$ and $T$ such that inequality 4.1 is satisfied. Hence, from Corollary 4.3, $2T - 1$ layers of the shuffle-Beneš network make a $2^k \times 2^k$ strict sense nonblocking switch. The bipartite
graph formed by the nodes of stage \(i\) and \(i + 1\), for \(i \leq k\), and the edges between them form an \((\alpha'', 3)\)-expander graph. The parameter \(\tau\) in this case is the same as the \(\tau\) or \(\log T\) in the discussion of multi-Beneš networks in Subsection 4.2.1. The parameter \(k\) in this network is equivalent to \(k + \tau\) in the discussion of multi-Beneš networks in Subsection 4.2.1.

**Theorem 6.7** The copy-Beneš network is a strongly wide sense nonblocking generalized connector for \(2^k\) external inputs and \(2^k\) external outputs with \(2(2T - 1)/\alpha''\) layers.

We assume that each node in the first stage of the copy-Beneš network is connected to one input and each node in the last stage is connected to one output. We also assume that each external input can simultaneously transmit on more than one layer. The outputs must be able to access all the layers but they are only required to be able to receive on one layer at a time. We outline the proof of Theorem 6.7 in appendix D. As can be seen there, it may not be possible to find an edge disjoint tree from an input to all the desired outputs in one layer. In that case, the set of outputs to be reached is partitioned into a number of subsets and each subset is assigned a layer. The input must transmit on all the layers simultaneously to be able to reach all the desired outputs.

We also notice that in the generalized nonblocking connector, the bipartite graph formed by any two consecutive stages and the edges between them is a constant degree bipartite graph and hence its edges can be partitioned into a number of perfect matchings. The number of perfect matchings equals the degree of each node in the bipartite graph. We shall later use this property of the network to route connection on the cascaded copy-Beneš network that is introduced in the next subsection.

### 6.3.2 Achievable Upper Bounds for the Multicast Problem

The routing problem we want to solve is a little different from the one solved by a generalized connector. The sources and destinations in a generalized connector are external to the network itself while in the networks we want to discuss, each node can be a source
and a destination. The essence of the problem remains the same. We need to find edge disjoint trees for multicast connections. Each node can be the destination of only one connection. We have a number of wavelengths available. Each wavelength gives us access to an independent layer of the network. Two trees can use the same edge on different layers of the network but threes on the same layer must be disjoint. The goal is to solve this routing problem using as few wavelengths or layers of the network as possible.

We shall look at the wide sense nonblocking routing of multicast connections. Just as in generalized connectors, the multicast routing problem can be solved under weakly or strongly wide sense nonblocking criteria. The distinction between the two is the same as in the case of generalized connectors. In the weakly wide sense nonblocking criterion, all the destinations of a multicast connection must be specified when a connection arrives to ensure nonblocking routing. On the other hand, in the strongly nonblocking criterion, a new destination can be added to or removed from a multicast connection while the connection is in progress.

We look at the two nonblocking criteria separately.

Weakly wide sense nonblocking multicast networks We consider the cascaded shuffle-Beneš we introduced in Section 5.3. The cascaded shuffle-Beneš network consists of four shuffle-Beneš networks cascaded and wrapped around. There are $2^k$ nodes in each stage. The notation here is slightly different from the one used in Section 5.3. The number of stages is $s = 8k$ and the total number of nodes is $N = 8k2^k$. The stages are numbered from 0 to $8k - 1$. We have shown that the cascaded shuffle-Beneš is a wide sense nonblocking network for permutation routing with $s^2(2T - 1)$ or $64k^2(2T - 1)$ wavelengths. We shall show that the same number of wavelengths is sufficient for a cascaded shuffle-Beneš to be weakly wide sense nonblocking for multicast connections.

The idea is to follow the same line of reasoning that was used to prove that cascaded shuffle-Beneš networks are wide sense nonblocking for permutation connections. We note that if we take an arbitrary pair of stages $i$ and $j$ on the cascaded shuffle-Beneš, either there is one complete shuffle-Beneš network contained in the stages from stage $i$ to $j$ or
there is one in the stages from $j$ to $i$. We take advantage of the fact that the shuffle-Beneš network is a weakly wide sense nonblocking generalized connector.

We have $s^2(2T - 1)$ wavelengths. We partition the set of wavelengths into $s^2$ sets of $2T - 1$ each. Each set is numbered by a pair of integers $(i, j)$ where $0 \leq i, j \leq s - 1$. A set numbered $(i, j)$ is dedicated to connections with origins in stage $i$ and destinations in stage $j$. When we need to route a new multicast connection, we partition the destinations into sets such that all destinations in a set belong to the same stage. If the origin is in stage $i$ then the multicast connection to all the destinations in stage $j$ is assigned to the set of $2T - 1$ wavelengths numbered $(i, j)$. We route the multicast connections from the origin to each set of destinations separately. Let us first concentrate on the set of destinations assigned to the set of wavelengths numbered $(i, j)$.

Each stage of nodes in the cascaded shuffle-Beneš, together with the next stage and the edges between them forms a bipartite graph in which each node has the same degree. Also, the in-degree and out-degree of each node in the network is at least 2. Let us first look at the edges that go from nodes of stage $\nu$ to stage $\nu + 1$. We first identify two matchings and label them $M_{\nu,0}$ and $M_{\nu,1}$. We can do the same thing for each stage of edges. Now, to go from node $i$ to node $j$ we can follow the edges in $M_{i,0}, M_{i+1,0}, \ldots, M_{j-1,0}$. This provides us with a set of edge disjoint paths from nodes of stage $i$ to the nodes of stage $j$. We can find another set of paths using the edges in $M_{i,1}, M_{i+1,1}, \ldots, M_{j-1,1}$. This way, we have identified two sets of edge disjoint paths from each node of stage $i$ to the nodes of stage $j$. It should be noted that for each node in stage $i$, we guarantee two paths from that node each going to some node in stage $j$ but not to any particular nodes of stage $j$.

We will look at two cases.

**Suppose, there is a shuffle-Beneš in the stages from $i$ to $j$.** In that case, we follow the edges from matchings $M_{i,0}, M_{i+1,0}, M_{i+2,0}, \ldots$ from the input in stage $i$ until the first stage of the shuffle-Beneš network is reached. Then from the destinations in stage $j$ we follow the edges from matchings $M_{j-1,1}, M_{j-2,1}, M_{j-3,1} \ldots$ until the
last stage in the shuffle-Beneš network is reached. Now, we have a multicast connection problem on the shuffle-Beneš network in which an external input needs to be connected to external outputs. We know that with $2T - 1$ layers, the shuffle-Beneš can solve this problem. Since the number of wavelengths in set $(i,j)$ is exactly $2T - 1$, we know that this set of multicast connections can be solved. It should be noted that all the destinations in a multicast connection that are assigned to set $(i,j)$ need not be reached in the same wavelength but they are guaranteed to be routed on one of the $2T - 1$ wavelengths in the set.

Suppose, there is no shuffle-Beneš in the stages from $i$ to $j$. In that case there must be a shuffle-Beneš network in the stages from stage $j$ to stage $i$. Let us identify the first and last stages of this shuffle-Beneš network as $b_f$ and $b_l$ respectively. Once again we follow the edges from matchings $M_{r,0}, M_{r+1,0}, M_{r+2,0}, \ldots$ from the input until stage $b_f$ is reached. We follow the edges from matchings $M_{j-1,1}, M_{j-2,1}, M_{j-3,1} \ldots$ until stage $b_l$ is reached. We know that matchings $M_{x,0}$ and $M_{x,1}$ are disjoint so that the nodes picked so far do not create any problem and we have a node in the input stage of the shuffle-Beneš and a set of nodes in the output stage of the shuffle-Beneš which can be connected to establish the desired multicast connection. But we know that with $2T - 1$ wavelengths the shuffle-Beneš is capable of establishing this multicast connection.

Hence, the set of wavelengths $(i,j)$ is sufficient for establishing any multicast connection with origin in stage $i$ and destinations in stage $j$. We also know that any multicast connection can be partitioned into smaller multicast connections such that all the destinations of the smaller multicast connection belong to the same stage. Hence, having $s^2$ sets of $2T - 1$ wavelengths each ensures that the cascaded shuffle-Beneš network is a weakly wide sense nonblocking network for multicast connections. Thus the number of required wavelengths is $s^2(2T - 1)$ or $64k^2(2T - 1)$. In $O$ notation, the number of wavelengths can be written as $O((\log N)^2)$. 

109
Strongly wide sense nonblocking multicast networks  The argument used to show that the cascaded shuffle-Beneš is a weakly wide sense nonblocking network for multicast connection hinges only on two facts:

1. We can find two disjoint matchings $M_{\nu,0}$ and $M_{\nu,1}$ between the nodes of stages $\nu$ and $\nu + 1$.

2. Between any two stages $i$ and $j$ there is either a weakly wide sense nonblocking generalized connector contained in the stages from $i$ to $j$ or there is one in the stages from $j$ to $i$.

If we change the second condition to say that there is a strongly wide sense nonblocking generalized connector contained either in the stages from stage $i$ to stage $j$ or in the stages from stage $j$ to stage $i$, then the network can be shown to be strongly wide sense nonblocking for multicast connection with arbitrary origins and destinations.

Let us look at the following network. We cascade four copies of the copy-Beneš network and wrap around. We call this network a cascaded copy-Beneš network. The number of stages in this network is $s = 12k$, each with $2^k$ nodes. The total number of nodes is $N = 12k2^k$. In the bipartite graph made by the nodes of stages $\nu$ and $\nu + 1$ and the edges between them, each node has the same degree and hence two disjoint matchings $M(\nu, 0)$ and $M(\nu, 1)$ can be found. Also, given two arbitrary stages $i$ and $j$, we can always find a full copy-Beneš network contained either in the stages from $i$ to $j$ or from $j$ to $i$. But we know that the copy-Beneš network is a strongly wide sense nonblocking generalized connector. Hence, following the same algorithm as the one described to show that the cascaded shuffle-Beneš is weakly wide sense nonblocking for multicast connections, we can show that the cascaded copy-Beneš network is strongly wide sense nonblocking for multicast connections.
7 Wavelength Conversion

The linear lightwave network model requires that a connection must stay on the same wavelength over its path in the network. We relax this constraint for this chapter. We assume that there are devices called wavelength changers in the network. A wavelength changer has one input and one output. It takes a signal modulated on a particular wavelength and outputs the same signal modulated on a different carrier wavelength. We assume that the wavelength changers are integrated with the LDC in such a way that an input signal coming in on a wavelength can be directed to any unused output on any wavelength. In this chapter, we study how the extra flexibility due to wavelength changers affect our ability to solve the permutation routing problem and the m to m routing problem.

7.1 Permutation Routing Problem

We derived some lower bounds on the number of wavelengths needed to solve a permutation routing problem in Chapter 3 and the bounds in Chapter 5. In this section we shall discuss those bounds with the assumption that wavelength changers are available in the network.

All the arguments used in deriving the link counting lower bounds in Section 3.3 remain unchanged if we add wavelength changers to the network. Hence, all the lower bounds derived in Section 3.3 remain valid. We focus our attention on the changes in the upper bounds on the number of wavelengths needed for permutation routing.

Once again, we look at both rearrangeable and wide sense nonblocking cases. As before, we consider graphs with constant degree and graphs with log N degree.

7.2 Upper Bounds on the Number of Wavelengths

By simply not using the option of changing wavelengths, a network with wavelength changers can simulate a network without changers. Hence, all the upper bounds derived
for networks without wavelength changers are also valid for networks with wavelength changers. We would like to see if we can get some extra benefit from having wavelength changers in the network.

We have seen before that having $l$ wavelengths in a network without wavelength changers is like having $l$ parallel and independent layers or copies of the network. A node can transmit on or receive from any of the $l$ layers of the network, but a connection cannot jump from one layer of the network to another. Having wavelength changers on the other hand allows the connections to jump layers from hop to hop. Another way to look at a network with wavelength changers is to think that each link is replaced by $l$ parallel links. This point of view leads to the conclusion that in a network which has multiple links between nodes, we can replace each set of multiple links by a single link as long as the number of available wavelengths is larger than the multiplicity of the link being replaced. The wavelength changers can simulate the original network on the new network by using different wavelength to simulate each parallel link. This leads to simplification of network topologies on which nonblocking routing can be done.

We look at the perfect shuffle and the hypercube networks for permutation routing.

### 7.2.1 Permutation Routing on Perfect Shuffle

We have seen in Chapter 4 that a perfect shuffle network with $2^k$ nodes in each stage and $3k + 2$ stages is a rearrangeably nonblocking switch if the inputs are in the first stage and the outputs are in the last stage.

Let us look at the top figure in figure 22. It shows a multistage perfect shuffle network with $s$ stages of nodes. Each stage has $2^k$ nodes. The figure is not wrapped around and hence the number of edge stages is only $s - 1$. The nodes in each stage are labeled by a $k$ bit binary number. Suppose we group all the nodes with the same label together as shown by the boxes and label the boxes by the label of the nodes contained in them. Now, if we treat each box as one node but let each edge be distinct, we get the $2^k$ node graph shown at the bottom of figure 22 where each thick line represents $s - 1$ parallel
edges. It is interesting to note that had each thick line in the graph represented only one edge, the graph would be isomorphic to the Debruijn graph shown in figure 7. We shall refer to this graph as the Debruijn graph with multiple edges for obvious reasons. The number of parallel edges represented by each thick line in a Debruijn graph with multiple edges is referred to as the multiplicity of the edges.

Let us first talk about the original multistage perfect shuffle network. We know that with \( s = 3k + 2 \), the multistage perfect shuffle network can route an arbitrary permutation problem under the rearrangeably nonblocking criterion if all the origins are in the first stage and all the destination nodes are in the last stage. Simply by the correspondence between the multistage perfect shuffle graph and the Debruijn graph with multiple edges, we conclude that the Debruijn graph with multiple edges can route any permutation routing problem under the rearrangeably nonblocking criterion if the multiplicity of the edges is at least \( 3k + 1 \). But we know that with wavelength changers, multiple edges between two nodes in a graph can be replaced by a single edge if the number of available wavelengths equals the multiplicity of the edges in the original graph. Hence, a simple Debruijn graph can do permutation routing under the rearrangeably nonblocking criterion with \( 3k + 1 \) wavelengths. Since the number of nodes in a Debruijn graph is \( 2^k \), the number of wavelengths needed for rearrangeably nonblocking routing of permutation problems can also be written as \( 3 \log N + 1 \). In the \( O \) notation, it can be represented as \( O(\log N) \). For comparison, the lower bound given by inequality 3.8 for any network with in-degree 2 is \( \frac{1}{4} \log(N/2) \).

Similarly, we can talk about wide sense nonblocking routing. Let us suppose we have a perfect shuffle network with \( s = 2k \). We know that \( k + 1 \) layers of the perfect shuffle network make a strict sense nonblocking switch for permutation connections. In this case, the origins are in the first stage and the destinations are in the last stage. But we know that the perfect shuffle network with \( 2k \) stages is equivalent to the Debruijn graph with multiple edges with edge multiplicity \( 2k - 1 \). Hence, \( k + 1 \) layers of the Debruijn graph with multiple edges with edge multiplicity \( 2k - 1 \) make a wide sense nonblocking
The top figure is a multistage perfect shuffle network. The dotted boxes show how the nodes are grouped. The bottom figure is the one stage equivalent of the top figure. The thick lines represent $s-1$ parallel edges.

Figure 22: Perfect Shuffle and Debruijn Graphs
network for the permutation routing problem. Once again with the help of wavelength changers each layer of the Debruijn graph with multiple edges can be simulated by a simple Debruijn graph and $2k - 1$ wavelengths. Hence, with a total of $(2k - 1)(k + 1)$ wavelengths a Debruijn graph is a wide sense nonblocking network for the permutation routing problem if wavelength changers are available. The number of needed wavelengths can also be written as $(2 \log N - 1)(\log N + 1)$. In the $O$ notation the number of wavelength is $O((\log N)^2)$.

7.2.2 Permutation routing on a Hypercube

We have introduced the hypercube in Section 5.2. It has $N = 2^k$ nodes, each represented by a $k$ bit binary number. Each node has in-degree $k$ and out-degree $k$. The directed version of the hypercube has two directed edges, one in each direction, between two nodes that differ in one bit position.

In section 5.2 we discussed a network called the doubled hypercube. A doubled hypercube is the same as a hypercube except each directed edge of the hypercube is replaced by two parallel edges in the doubled hypercube. Hence, a doubled hypercube can be simulated by a hypercube network with wavelength changers and two wavelengths.

We have seen in section 5.2 that a doubled hypercube can solve the permutation routing problem under rearrangeably nonblocking constraint in one wavelength. Hence, a hypercube with wavelength changers can solve the permutation routing problem under rearrangeably nonblocking constraint in two wavelengths.

We have also seen that the doubled hypercube can solve the permutation problem under wide sense nonblocking constraint in $\log N + 1$ wavelengths. Hence, a hypercube with wavelength changers can solve the permutation routing problem under wide sense nonblocking constraint under $2\log N + 2$ wavelengths.
7.3 m to m routing

We extended the results from permutation routing to results in m to m routing for networks without wavelength changers in Chapter 6. The same can be done for networks with wavelength changers.

7.3.1 m to m routing on a Debruijn network

From Lemma 6.3 we know that an m to m rearrangeably nonblocking routing problem can be partitioned into m permutation problems. In Subsection 7.2.1 we saw that a permutation routing problem can be solved under the rearrangeably nonblocking constraint on $3\log N + 1$ layers of the Debruijn graph with wavelength changers. An m to m routing problem can be partitioned into m permutations each requiring $3\log N + 1$ wavelengths. Hence, the total number of wavelengths required to solve an m to m routing problem under the rearrangeably nonblocking constraint is $m(3\log N + 1)$. In $O$ notation, the required number of wavelengths is $O(\log N)$.

From Lemma 6.4 we know that if $l$ layers of a network are wide sense nonblocking for the permutation routing problem, $l(2m - 1)$ layers of the same network make a wide sense nonblocking network for the m to m routing problem. We know from Subsection 7.2.1 that $(2\log N - 1)(\log N + 1)$ layers of a Debruijn graph with wavelength changers make a wide sense nonblocking network for permutation routing. Hence, the number of layers or wavelengths needed to make the Debruijn network with wavelength changers a wide sense nonblocking network for the m to m routing problem is $(2m - 1)(2\log N - 1)(\log N + 1)$. In $O$ notation, this number is $O((\log N)^2)$.

7.3.2 m to m routing on a hypercube

Once again we can discuss the rearrangeably nonblocking and wide sense nonblocking cases separately.

We know from Subsection 7.2.2 that with 2 wavelengths, the permutation problem can be solved on a hypercube under the rearrangeably nonblocking criterion. We also know
from Lemma 6.3 that an \( m \) to \( m \) routing problem can be partitioned into \( m \) permutation routing problems. By partitioning the \( m \) to \( m \) routing problem into \( m \) permutation problems and assigning 2 wavelengths to each permutation problem, we can solve it in \( 2m \) wavelengths under the rearrangeably nonblocking constraint.

Similarly, we know from Subsection 7.2.2 that with \( 2 \log N + 2 \) wavelengths, the permutation problem can be solved on a hypercube under wide sense nonblocking criteria. We also know from Lemma 6.4 that if a network is wide sense nonblocking for the permutation routing problem with \( l \) wavelengths, then it is wide sense nonblocking for the \( m \) to \( m \) routing problem with \( l(2m - 1) \) wavelengths. Hence, we conclude that with \( (2m - 1)(2 \log N + 2) \) wavelengths, we can solve the \( m \) to \( m \) routing problem on a hypercube under the wide sense nonblocking constraint.

### 7.3.3 Other Routing Problems

For networks without wavelength changers, we had discussed two more routing problems. The first problem was the all to all routing problem. As discussed earlier, the all to all routing problem is a special case of the \( m \) to \( m \) routing problem. Also, the all to all routing problem is static in nature. Hence, we can simply apply the upper bound from the \( m \) to \( m \) routing under the rearrangeably nonblocking criterion to the all to all routing problem.

The second problem was the multicast problem. Unfortunately, the networks used to derive the upper bound on the number of wavelengths for the multicast problem cannot be transformed in the same way as the multi stage perfect shuffle network for derivation of a simpler network that can be used with wavelength changers. The main reason why perfect shuffle networks are easy to transform is that each stage has identical structure. The same is not true for the cascaded shuffle-Benes network or the cascaded copy-Benes network used for multicast connections. We do not know if having wavelength changers allows us to reduce the upper bound on the number of wavelengths significantly or to use simpler topologies to solve multicasting problems.
8 Conclusions

8.1 Implications of This Work

This work is intended to be a first step towards the design of a wide area lightwave network. The focus is on the scalability of the architecture and hence we worked with topologies for which the required number of wavelengths grows slowly with the size of the network. We studied the permutation routing problem in detail. We shall discuss the implication of the results from the permutation routing problem. We looked at both the rearrangeably nonblocking and wide sense nonblocking constraints. In a network, the connections arrive over time and a connection should not be disturbed during its lifetime. Hence, the wide sense nonblocking constraint is the most practical one.

The implication of the results in Chapter 3 are easy to see. We found that for a given degree the lower bounds on the number of wavelengths grow linearly with the diameter. For a given diameter, the number of wavelengths goes down as the degree goes up. Since the degree of a node is restricted by the cost of fiber and the size of the LDC's, a network with large degree is likely to be more expensive to build. Hence, we should look for topologies that have small diameter for a given degree. The other implication of the discussions in Chapter 3 relates to the routing algorithm. A good routing algorithm must be non-oblivious. We discovered that an oblivious routing algorithm needs at least $\sqrt{N}/(2d)$ wavelengths to solve the permutation routing problem. The number of wavelengths grows too fast with the size of the network in this case.

In Chapter 5, we have given the rates of growth of the required number of wavelengths for the permutation routing problem but no numerical values are discussed. We shall look at some numbers here. We only consider the wide sense nonblocking constraint. Let us look at the upper bounds derived from the perfect shuffle networks. For a network with $N = 4k2^k$, the number of required wavelengths is $2k(4k + 1)(k + 1)$. For $k = 10$, the number of nodes in the network is 40,960 and the number of wavelengths is 9020. For $k = 5$, the number of nodes is 640 and the number of wavelengths is 1260. We
know that we can solve the permutation routing problem on an arbitrary network if we have as many wavelengths as the number of nodes. Hence, for $k = 5$, the upper bound derived for the perfect shuffle network is not very useful. But for $k = 10$, the reduction in the number of wavelengths is substantial. The required number of wavelengths becomes smaller compared to the size of the network as the network size grows. Also, the number of required wavelengths is heavily dependent upon the degree of the network. For example, if we have a perfect shuffle structure with in-degree and out-degree 10, the number of wavelengths for a network with 800 nodes is 108 and for a network with 12,000 nodes is only 312. The required number of wavelengths for the cascaded shuffle-Beneš network has a slower rate of growth but because of the large constant factor, these networks are useful only for very large number of nodes. It should be noted, however, that the results in Chapter 5 are achievable upper bounds. It is possible that there are better topologies that can solve the permutation problem with fewer layers. By looking at the particular topologies, we wanted to get an understanding of what kind of structures can be used as topologies of linear lightwave networks.

In all cases, the number of required wavelengths is large compared to what we believe can be achieved today for long distance communication on optical fibers. However, this work is based on the assumption that the optical device technology will advance to accommodate enough wavelengths for long distance communication. As we pointed out in Chapter 2, the results in this work are relevant to any network with a channel access mechanism that effectively divides the network into a number of independent layers. Although having different wavelengths is one obvious way of dividing a network into independent layers, the optical technology may give us some other channel accessing scheme that can provide the required number of layers in a network. Even with current technology, we can get a large number of layers by physically increasing the number of fibers. Instead of one fiber, we can use a large number of fibers and each fiber can be used as a number of layers.
8.2 Summary

The basic motivation behind this work was to get a better understanding of the possible ways in which wide area optical networks can be built. In Chapter 2 we described three functions of the network, i.e., user interface, aggregation and transport. Most of this work concentrated on solving various transport problems using the linear lightwave network model.

One major aspect of the transport problem in an optical network is the bandwidth access mechanism. Current technology restricts us to access the fiber-optic bandwidth as a small number of high capacity channels. Hence, we must use each channel as efficiently as possible to be able to support a large number of nodes in the network. The other problem is the speed of reconfiguration of the switches in the network. Since the reconfiguration time for the devices in a linear lightwave network is expected to be relatively large, we focused on circuit switching or packet switching with large packet lengths as the appropriate modes in which the network can operate. The question we then wanted to ask was how many channels or wavelengths are needed to route a particular set of connections over the network.

We first focused on the permutation routing problem. For a backbone of a wide area optical network, the permutation problem is not a very practical problem but our interest in this problem was mainly due to the understanding it can provide in other routing problems. We first discussed the nonblocking criteria. The wide sense nonblocking criterion turns out to be the most appropriate for the networks of interest. We also discussed rearrangeably nonblocking criterion in some detail.

The first step was to find a lower bound on the number of channels needed to route the permutation problem on networks with some given constraint on the degree, i.e., the number of outgoing and incoming links at each node. We established the lower bounds using simple counting arguments.

Next, we considered the situation in which we were given full freedom to choose the topology. The task was to find a topology that could solve the permutation routing prob-
lem using as few channels as possible. We looked at network topologies that are derived from switching topologies and showed that they come close (in order of magnitude) to achieving the lower bounds on the number of channels that were established earlier. It may be difficult to have enough control over the topology of a wide area network to get it to match a switching topology exactly, but by examples we showed what can be done under most favorable circumstances. This study had two goals in mind. First, it provided a reference for the performance of a real network and allowed us to get a measure of how close the real network is to achieving what we know can be achieved. Second, if we do have control over the topology of the network, we know good candidate topologies.

We next looked at other routing problems. We considered the problem in which each node wants to talk a set of other nodes in the network. We also considered the multicasting problem. Both these problems are of practical importance in the backbone of a wide area network.

Finally, we considered the impact of wavelength changers on the design of the network. The ability to go from one wavelength to another allows the best performing topologies to be simpler. The added flexibility also closed the gap between the lower bounds on the number of channels and the achievable upper bounds for the permutation routing problem and the \( m \) to \( m \) routing problem.

There are a number of important issues that we raised but did not formulate quantitatively. Some of the issues are recapitulated in the next section. By understanding these issues we can get a better understanding of the potentials of optical wide area networks in general and linear lightwave networks in particular.

### 8.3 Directions for Future Research

In this work, we focused most of our attention on the data transport function. Within that function, we only considered the linear lightwave network model and some variations on it. We have indicated that other network models are being studied. One example is the wavelength router model studied in [Bar92a].
The questions of aggregation becomes very important with the emergence of ATM networks as the basis for an integrated services network. The ATM cells are only 53 bytes long. We have seen that optical networks are better at handling large and steady flow of data. How to aggregate the ATM cells so that the traffic presented to the optical network is appropriate for efficient transportation remains a challenging problem. Similarly, deaggregating highly aggregated sessions into individual sessions is a challenging problem. The networks discussed in connection with the multicast routing problem in Chapter 6 may be helpful in efficient deaggregation.

The architectures studied in this work are not hierarchical. The rationale behind looking at a non-hierarchical architecture is that even in a hierarchical structure, each level of the hierarchy must be studied and designed well. Since each individual level of a hierarchical network is in itself non-hierarchical, study of non-hierarchical architecture should help us design a hierarchical architecture. It is now time, however, to look at the problems of designing a hierarchical architecture.
A Cantor Networks

Having $k + 1$ identical layers of a network of $2k + 1$ stages of $2^k$ nodes, each connected in a shuffle exchange fashion makes a $2^{k+1} \times 2^{k+1}$ Cantor switch. We have asserted that as a switch that connects inputs in the first stage to outputs in the last stage, this switch is strict sense nonblocking for the permutation switching problem. In this section we present the proof. This proof is taken from [Hui90].

**Theorem A.1** With $k + 1$ layers the above network is a strict sense nonblocking switch.

The number of nodes in each stage is $2^k$ and each node in the first stage has two inputs. Similarly, each node in the last stage has two outputs. We have a system with $2^{k+1}$ inputs and $2^{k+1}$ outputs. Let us assume that there are at most $2^{k+1} - 1$ active connections. We do not know how the active connections were routed. Since we are only considering permutation problems, at most $2^{k+1} - 1$ active connections imply that at most $2^{k+1} - 1$ inputs and at most $2^{k+1} - 1$ outputs are busy. Now, we want to be able to connect an idle input to an idle output. If the Cantor network is strict sense nonblocking, we should be able to find a path from the idle input to the idle output without using any of the edges already in use by one of the active connections.

Denote the number of nodes the idle input can reach in stage $i$ by $A(i)$. In the case of a Cantor network, $A(1) = k + 1$ because an input is connected to one node in each layer and all the links from the input to the nodes in the first stage are free.

In the second stage, the number of nodes the idle input can reach is doubled but the link to one of them might be occupied by the other input with which it shares the first node. A look at figure 11 makes this point clear. Hence, the number of nodes the idle input can reach in the second stage, given the connections from all the other inputs, is

$$A(2) \geq 2A(1) - 1.$$
links leading to these nodes and hence the number of nodes that an input can reach is bounded by

\[ A(3) \geq 2A(2) - 2. \]

Similarly,

\[ A(i) \geq 2A(i - 1) - 2^{i-2}. \]

This relation is true for all \( i \leq k \). Carrying out the recursion, we get

\[ A(i) \geq 2^{i-1} A(1) - (i - 1)2^{i-2}. \]

If \( A(k + 1) \) is greater than half the nodes in stage \( k + 1 \) then by similar calculations starting in stage \( 2k + 1 \) and then moving backwards to stage \( k \), we conclude that the idle output can reach more than half the nodes in stage \( k + 1 \). Hence, there is at least one node in stage \( k + 1 \) which can be reached by both the idle input and the idle output and hence the desired path can be found.

\( A(1) \) equals the number of layers available and hence the number of nodes in any stage equals \( A(1)2^k \) when all the layers are counted. Hence, the sufficient condition for Cantor network to be nonblocking is

\[ 2^k A(1) - k2^{k-1} > 2^{k-1} A(1). \]

By putting \( A(1) = k + 1 \), it can be verified that this inequality is satisfied.

It should be noted that the argument takes advantage of the following properties of the network:

1. An input has access to all the layers.
2. All the layers have the same connections
3. The number of nodes that can be reached by an input at any stage is double the number of nodes that can be reached in the previous stage. This doubling
continues until all the nodes of the stage can be reached. The same thing is true going backwards from the outputs also.

The theorem and the proof above apply to any network that satisfies these criteria. Fortunately, a Beneš network also has all the properties needed for this proof to be valid. Hence, we deduce the following corollary.

**Corollary A.2** A Beneš network with \(2^{k+1}\) inputs and outputs and \(k + 1\) layers is a strict sense nonblocking switch for permutation switching.
B Oblivious Routing Algorithms

If a path found by a routing algorithm for a given connection depends only on the origin and the destination of that connection then the routing algorithm is called an oblivious routing algorithm. An oblivious routing algorithm in an \( N \) node network can be completely specified by a table of \( N(N - 1) \) paths, one for each possible origin destination pair. These algorithms do not adapt to the congestion in the network and hence, in the worst permutation problem, there may be some links which must carry a large number of connections. As a result, the number of wavelengths needed to make a network non-blocking while using an oblivious routing algorithm may be large. The following theorem formalizes this intuition.

**Theorem B.1** A network with \( N \) nodes and degree \( d \) uses at least \( \lceil \sqrt{N}/\sqrt{2d_in} \rceil \) wavelengths for nonblocking routing using any oblivious algorithm.

This theorem is based on a result in [KKT90] for undirected graphs and packet routing problems. The proof presented here is the same as the proof in [Lei92] with minor modifications.

Let \( G = (V, E) \) be an \( N \) node graph with maximum in-degree \( d_{in} \). Let \( A \) be the oblivious algorithm being used to route connections on \( G \).

Let us consider a path from node \( u \) to node \( v \). Let us call this path \( P_{u,v} \). Since \( A \) is oblivious \( P_{u,v} \) is well defined. We are interested in finding a large set of origin destination pair \((u_1,v_1),(u_2,v_2)\ldots\) with \( u_i \neq u_j \) and \( v_i \neq v_j \) for \( i \neq j \), such that \( P_{u_1,v_1}, P_{u_2,v_2} \ldots \) contain the same edge \( e \in E \).

For any node \( v \), let us consider \( P_v \), the set of \( N - 1 \) paths \( P_{u,v} \) that end in \( v \). For any integer \( k \), let \( S_k(v) \) be the set of all edges that carry \( k \) or more paths in \( P_v \). Let, \( S_k^*(v) \) be the set of all nodes which have at least one incoming edge or one out going edge in \( S_k(v) \). It is easy to see that \( 2|S_k(v)| \geq |S_k^*(v)| \).

Consider \( k \leq (N - 1)/d_{in} \). Since there are \( N - 1 \) paths coming to \( v \) over one of \( d_{in} \) links, \( v \in S_k^*(v) \). For a node \( u \notin S_k^*(v) \) let us look at path \( P_{u,v} \). Starting from \( u \), if
we follow $P_{u,v}$, we are bound to reach a node in $S_k^*(v)$. Suppose $w \in S_k^*(v)$ is the first such node and $w'$ is the node immediately before $w$ in $P_{u,v}$. Clearly, $w' \notin S_k^*(v)$ and $(w', w) \notin S_k(v)$. For every node $w \in S_k^*(v)$ let us try to compute an upper bound on the number of paths for which $w$ is the first node in $S_k^*(v)$. Each link carrying such paths coming into $w$ can bring at most $k - 1$ paths, for otherwise $w$ could not be the first node in $S_k^*(v)$. The number of links coming into $w$ is at most $d_{in}$. Hence, the total number of paths for which $w$ is the first node in $S_k^*(v)$ is at most $d_{in}(k - 1)$. But all such paths start from a node that is not in $S_k^*(v)$ and the first node on the path that is an element of $S_k^*(v)$ is unique. Hence,

$$|V - S_k^*(v)| \leq d_{in}(k - 1)|S_k^*(v)|.$$  

Now,

$$N = |V - S_k^*(v)| + |S_k^*(v)| \leq ((k - 1)d_{in} + 1)|S_k^*(v)| \leq kd_{in}|S_k^*(v)| \quad \text{if } d_{in} \geq 1. \quad \text{(B.1)}$$

The condition $d_{in} \geq 1$ is satisfied by any non-trivial network.

Therefore,

$$|S_k(v)| \geq \frac{1}{2}|S_k^*(v)| \geq \frac{N}{2kd_{in}}.$$  

Summing $|S_k(v)|$ over all $v$ we get

$$\sum_v |S_k(v)| \geq \frac{N^2}{2kd_{in}}.$$ 

But $|E| \leq N d_{in}$, and therefore at least one edge $e \in E$ must be in $S_k(v)$ for at least $N/(2kd_{in})$ destinations.

If $N \geq 2$, $\sqrt{N}/\sqrt{2d_{in}} \leq (N - 1)/d_{in}$ and hence the assumption $k \leq (N - 1)d_{in}$ is valid for all $k \leq \sqrt{N}/\sqrt{2d_{in}}$. Putting $k = \lceil \sqrt{N}/\sqrt{2d_{in}} \rceil$ we get that at least one edge $e$ is on $k$
paths to at least $k$ destinations. This allows us to choose a set of one to one connections in which at least $k$ paths must go through $e$. Hence, for any oblivious routing algorithm on a network with two or more nodes and $d_{in} \geq 1$, we can find a permutation routing problem which requires one edge to carry at least $\lfloor \sqrt{N}/\sqrt{2d_{in}} \rfloor$ connections. Hence the number of wavelength required to route such a permutation routing problem is at least $\lfloor \sqrt{N}/\sqrt{2d_{in}} \rfloor$. 
C Strict Sense Nonblocking Switch Using Multi-Beneš

We prove that the multi-Beneš network is a strict sense nonblocking switch. The proof is taken from [ALM90]. We have discussed multi-Beneš networks in subsection 4.2.1. They are made up of bipartite graphs with the expansion property. A multi-Beneš network is shown in figure 14.

As can be seen in the figure, the multi-Beneš network has nodes divided into $2k + 1$ stages. Each stage has $T2^k$ nodes. We can number the stages from 0 to $2k$. The edges from nodes of stage $i$ go to the nodes in stage $i + 1$. The in-degree and out-degree of each node is $d$. The nodes of each stage are divided into smaller sets. We call these sets the classes of nodes. The number of classes in stage $k$ is 1. There are $2^j$ classes in stages $k - j$ and $k + j$. Let us also number the classes from 0 to $2^{k-j} - 1$ in stage $i$ going from the top to the bottom. It can be seen from the figure that in stages 0 through $k - 1$, all the edges from the nodes of class $c$ of stage $i$ go to class $[c/2]$ of stage $i + 1$. The bipartite graph formed by class $c$ of stage $i$ and class $[c/2]$ of stage $i + 1$ is an $(\alpha', \beta')$-expander graph. The connection pattern is different in stage $k$ through $2k - 1$. The edges coming out of a node of class $c$ in stage $i$, for $k \leq i \leq 2k - 1$, are split into two groups. One group of $d/2$ edges lead to nodes of class $2c$ in stage $i + 1$ and the other group of $d/2$ edges lead to nodes of class $2c + 1$ in stage $i + 1$. The bipartite graph formed due to class $c$ of stage $i$ and class $2c$ or $2c + 1$ of stage $i + 1$ is an $(\alpha, \beta)$-expansion graph. The values of $\alpha$ and $\alpha'$ chosen to represent the expansion properties of the graph are such that $T\alpha$ and $T\alpha'$ are both integers. We shall determine the required relationship between $\alpha$, $\alpha'$, $\beta$ and $\beta'$ later.

The number of classes of nodes in stage 0 is $2^k$. The number of classes in stage $2k$ is also $2^k$. Each class in both stage 0 and stage $2k$ has $T$ nodes. To use the multi-Beneš network as a $2^k \times 2^k$ nonblocking switch, we connect each input to all $T$ nodes of one class in stage 0. Similarly, we connect all $T$ nodes of a class in stage $2k$ to one output. We
label the inputs and outputs with the number of the class of nodes they are connected to. For example, input \( i \) is connected to all the nodes of class \( i \) in stage 0 and output \( j \) is connected to all the nodes of class \( j \) in stage \( 2k \).

We outline a strategy to search for an edge disjoint path from the free input to the free output. In fact, we will find node disjoint paths for each of our connections. Since node disjoint paths are also edge disjoint, finding node disjoint paths is sufficient for our purposes. The main idea is to treat each node in the network that has one connection going through it as faulty and try to route new connections around it.

We define a node to be busy if a connection is going through it. We also define the concept of blocked and unblocked nodes. In a multi-Beneš network, the degree of each node is \( d \). Let us number the stages from 0 to \( 2k \). Let us look at a particular class \( c \) of nodes in stage \( i \) for \( k \leq i \leq 2k - 1 \). For convenience, we refer to all the edges that go to class \( 2c \) of stage \( i + 1 \) as the up output edges and all the edges that go to class \( 2c + 1 \) of stage \( i + 1 \) as the down output edges. We define a blocked node recursively going back from stage \( 2k \). All the nodes in stage \( 2k \) are by definition unblocked. A node in stage \( k \) through \( 2k - 1 \) is defined to be blocked if either all its up output edges lead to busy or blocked nodes or all its down output edges lead to busy or blocked nodes. A node in stage 0 to \( k - 1 \) is defined to be blocked if each of its edges leads to busy or blocked nodes. From the definition, it should be clear that a node can be busy and still be unblocked.

The concept of blocked and unblocked nodes is important because from an unblocked node in stages 0 through \( k \), we can reach all the outputs without ever having to go through a busy node. From a node in stage \( i \), for \( k + 1 \leq i \leq 2k \), paths exist only to \( 2^{2k-i} \) outputs, and an unblocked node can reach all these outputs without ever having to go through a busy node. Hence, to show that a multi Beneš network is strict sense nonblocking, we only need to show that if a number of permutation connections are in progress, then for an unused input \( i \) there is at least one unblocked node in class \( i \) of stage 0. This will ensure that a new connection from an unused input to an unused output can always be routed.
Lemma C.1 Suppose a number of permutation connections are in progress through a multi-Beneš. If $T > (2\alpha(\beta - 1))^{-1}$, at most a fraction $2\alpha$ of the nodes of any class in stage $k$ through $2k - 1$ of the multi-Beneš are blocked.

Proof The proof is by induction. In stage $2k$ none of the nodes are blocked. Hence the induction hypothesis is trivially true for stage $2k$. Suppose the induction hypothesis is also true for stages $i + 1$ through $2k - 1$ for $k \leq i \leq 2k - 1$.

We shall focus our attention on one $(\alpha, \beta)$-expander graph made by class $c$ of stage $i$ and class $2c$ of stage $i + 1$. Figure 23 shows one such expander graph. Class $c$ of nodes in stage $i$ has $M = T2^{2k-i}$ nodes. To establish a contradiction, suppose more than $\alpha M$ of these nodes are blocked due to the upper output edges. Since the upper output edges from class $c$ of stage $i$ lead to class $2c$ of stage $i + 1$, this is equivalent to supposing that more than $\alpha M$ nodes are blocked due to the nodes in class $2c$ of stage $i + 1$ being blocked or busy. Let $W$ be the set of all blocked or busy nodes in class $2c$ of stage $i + 1$. From the expansion property of the bipartite graph formed by class $c$ of stage $i$ and class $2c$ of stage $i + 1$ we conclude that

$$|W| \geq \alpha \beta M. \quad (C.1)$$

It should be noted that if $\alpha M$ is not an integer we can only assert that $|W| \geq \beta \lfloor \alpha M \rfloor$ which is not sufficient for our proof. $\alpha M$ is an integer because $M$ is an integer multiple of $T$ and $\alpha T$ is an integer by assumption.

From figure 14, it can be seen that the upper outputs of all nodes in one class in stage $i$, for $k \leq i \leq 2k - 1$, can reach only $2^{2k-i-1}$ or $M/(2T)$ classes in stage $2k$ and hence can reach only $M/(2T)$ outputs. But since at most one connection can exist to each output, the number of busy nodes in class $2c$ of stage $i + 1$ is at most $M/(2T)$. By the induction hypothesis, the number of blocked nodes in class $2c$ of stage $i + 1$ is at most $2\alpha M/2$ or $\alpha M$. Therefore, $|W| \leq \alpha M + M/(2T)$. But since $T > (2\alpha(\beta - 1))^{-1}$, we get

$$|W| < \alpha \beta M. \quad (C.2)$$

131
Figure 23: An \((\alpha, \beta)\)-expander graph

We have a contradiction since equations C.1 and C.2 cannot both be true. Hence, the assumption that the number of blocked nodes in class \(c\) of stage \(i\) due to its upper output edges is more than \(\alpha M\) must be incorrect. The same argument can be repeated for the lower output edges to conclude that the number of nodes blocked due to the lower output edges in class \(c\) of stage \(i\) is at most \(\alpha M\). Hence, the total number of blocked nodes in class \(c\) of stage \(i\) is at most \(2\alpha M\). The choice of class \(c\) was arbitrary and the argument can be applied to any class in stage \(i\). This proves the lemma.

Let us now look at the stages 0 through \(k\). These stages are made of \((\alpha', \beta')\)-expander graphs. In each of these expander graphs \((U, V, E)\) the number of nodes in \(V\) is twice the number of nodes in \(U\). Let us also assume that \(\alpha' \geq 2\alpha\). The following lemma is true for these stages.

**Lemma C.2** Suppose a number of permutation connections are being routed through the multi-Beneš network. At most a fraction \(\alpha'\) of the nodes of any class in stage 0 through \(k\) of the multi-Beneš are blocked if the fraction of blocked nodes in stage \(k\) is no more than \(\alpha'\) and \(T > 2(\alpha'/(\beta' - 2))^{-1}\).

**Proof** Once again we use induction to prove this. The base case of stage \(k\) is true.
by assumption. Let us assume that the fraction of blocked nodes in any class of nodes is at most $\alpha'$ for all stages from $i + 1$ to $k - 1$. Let us look at class $2c$ in stage $i$. This forms an $(\alpha', \beta')$-expander graph with class $c$ in stage $i$. The number of nodes in class $2c$ of stage $i$ is $M = T^{2i}$. The number of nodes in class c of stage $i + 1$ must be $2M$. Figure 24 shows one such expander graph.

Suppose the fraction of nodes blocked in class $2c$ of stage $i$ is greater than $\alpha'$. Let $B$ denote the set of all the blocked nodes in this class. Also, let $W$ denote the set of all the blocked or busy nodes in class $c$ of stage $i + 1$. Since $|B| > \alpha'M$, and the neighbor set of $B$ must be a subset of $W$, we get

$$|W| \geq \alpha'\beta'M. \quad \text{(C.3)}$$

Once again, the assumption that $\alpha'T$ is an integer is necessary for equation C.3 to be true. Now let us count the number of blocked or busy nodes in class $c$ of stage $i + 1$. The number of busy nodes in class $c$ of stage $i + 1$ can be at most the number of inputs that can reach at least one node in that class. It is clear from figure 14 that this number cannot exceed $2M/T$. The number of blocked nodes is at most $2\alpha'M$ from the induction
hypothesis. Hence, \( |W| \leq 2\alpha'M + 2M/T \). But since \( T > 2(\alpha'(\beta' - 2))^{-1} \), we get

\[
|W| < \alpha'\beta'M. \tag{C.4}
\]

We have a contradiction since equations C.3 and C.4 cannot be true at the same time. Hence the assumption that more than \( \alpha' \) fraction of the nodes in class 2c of stage i are blocked must be incorrect. It is easy to see that the choice of class 2c is arbitrary and the argument presented above is valid for each class of nodes in stage i. This proves the lemma.

These two lemmas help us prove theorem 4.2 from subsection 4.2.1.

**Theorem 4.2** If \( \alpha' < 1 \), \( \alpha' \geq 2\alpha \), \( \beta' > 2 \), \( \beta > 1 \) and

\[
T > \max \left\{ \frac{1}{2\alpha (\beta - 1)}, \frac{2}{\alpha' (\beta' - 2)} \right\}
\]

then a multi-Beneš network with \( 2k + 1 \) stages and \( T2^k \) nodes in each stage can be used as a \( 2^k \times 2^k \) strict sense nonblocking switch for permutation routing from \( 2^k \) external inputs to \( 2^k \) external outputs.

**Proof** Let us suppose we have a number of permutation connections between the inputs and the outputs and a new connection arrives from a free input to a free output. Since

\[
T > \max \left\{ \frac{1}{2\alpha (\beta - 1)}, \frac{2}{\alpha' (\beta' - 2)} \right\}
\]

and \( \alpha' \geq 2\alpha \), the conditions for both lemmas C.1 and C.2 are satisfied. Hence, in any free class of nodes in stage 0, the number of blocked nodes is at most \( \alpha'T \). Since the number of blocked nodes must be an integer it can be at most \( \lfloor \alpha'T \rfloor \) and hence the number of unblocked nodes is at least \( \lceil (1 - \alpha')T \rceil \). Since \( T > 2(\alpha'(\beta' - 2))^{-1} \), the number
of unblocked nodes is at least
\[
\left\lceil \frac{2}{\beta' - 2\left(\frac{1}{\alpha'} - 1\right)} \right\rceil
\]
which is at least 1 since \(\alpha' < 1\). Hence, at least one node in each free class of nodes in stage 0 is not blocked. Any unblocked node can reach all the free outputs and hence a new connection from an unused input to an unused output can always be routed. In other words, the multi-Beneš network described here is a strict sense nonblocking switch.

It is easy to derive a routing algorithm from the proof of the theorem. Let us number the outputs from 0 to \(2^k - 1\) from top to bottom. Let us express this number as a \(k\) bit binary number. Starting from a node in stage \(k\), a path to an output can be found very easily. We simply trace the \(k\) bit output address from the most significant bit to the least and determine the edge that should be followed. Each time a 0 is encountered, the path follows an up output edge to the next stage and each time a 1 is encountered, the path follows a down output edge. From figure 14 it is easy to see that this algorithm finds a path from a node in stage \(k\) to an output. At each stage there are a number of choices either among the up output edges or the down output edges. We will next describe how to choose the appropriate edge in each case.

Before routing connections, we first determine which nodes are blocked or busy. Each node can do this computation locally after receiving the information from the nodes of the next stage. This information can also be locally updated as new connections are added or deleted. Each node that is neither blocked nor busy in stage \(k\) to \(2k - 1\) identifies at least one up output edge and at least one down output edge each of which leads to a node in the next stage that is neither blocked nor busy. In stages 0 to \(k - 1\), each node that is neither blocked nor busy identifies at least one output edge that leads to a node in the next stage that is neither blocked nor busy. Now, to find a path from a free input to a free output, we find an unblocked node in stage 0 that is attached to the free input. Theorem 4.2 tells us that such a node must exist. The path simply follows an edge that
leads to a node in the next stage that is neither busy nor blocked until stage $k$ is reached. From this point on an up output edge or a down output edge is chosen depending upon the output address. Each time an edge is followed that leads to another node that is neither blocked nor busy.

It should be noted that if a free input wants to establish a multicast connection to a number of free outputs, the same routing algorithm can achieve that as well. In subsection 6.3.1, we defined a generalized nonblocking connector. A multi-Beneš network is a generalized wide sense nonblocking connector under the constraint that all the outputs of a multicast connection are identified at the same time. Once a multicast connection has started from an input to a set of outputs, no new outputs are allowed to join the multicast connection at a later point. The routing algorithm for a new connection from a free input to a set of free outputs has three steps. First, we identify the nodes that are neither busy nor blocked. Then a node in stage 0 is found that is connected to the input and is neither blocked nor busy and a path from that node to a node in stage $k$ if found that goes through nodes that are neither blocked nor busy. Finally, paths from that node in stage $k$ to all the desired outputs is found. Once again these paths go through nodes that are neither blocked nor busy. If this algorithm is followed then all the conditions needed to prove theorem 4.2 remain valid and hence a free input is guaranteed to have access to a node in stage 0 that is neither blocked nor busy.

We write the result as a corollary.

**Corollary C.3** A multi-Beneš network satisfying all the assumptions of theorem 4.2 is a wide sense nonblocking generalized connector with the condition that all the outputs of a multicast connection be identified at the time the multicast connection arrives.
D Generalized Connectors

In Chapter 6, we referred to generalized nonblocking connectors without giving much detail. Here we show how a generalized nonblocking connector can be constructed using graphs with the expansion property. Most of this discussion is based on [FFP88]. We shall construct strongly wide sense nonblocking generalized connectors.

As we discussed before, a generalized wide sense nonblocking connector allows a set of inputs to establish multicast connections to a set of outputs as long as each output is connected to only one input and the total number of outputs is at most a given constant. There may also be a constraint on the multiplicity of each multicast connection. Since we are discussing strongly wide sense nonblocking generalized connectors, if an input decides that it wants to include another unused output in its multicast connection or a free input wants to set up a new multicast connection then the generalized connector should be able to find paths for the new connections without disturbing the existing connections.

To make our discussion more concrete we shall define a $(\nu, \mu)$-limited generalized connector. An $(\nu, \mu)$-limited generalized connector can connect a given set of inputs to a specified set of outputs with multicast connections if the total number of outputs reached is at most $\nu$ and each input is connected to at most $\mu$ outputs. We specify which outputs are to be connected to a given input.

To make a generalized nonblocking connector, we first make a copy network and then follow it by a wide sense nonblocking switch. A copy network simply connects each input to a set of outputs. We can specify the number of outputs an input must be connected to but not the identity of the outputs. If the number of outputs each input is connected to equals the multiplicity of the multicast connection starting at that input, we can cascade a nonblocking switch after the copy network, use the nonblocking switch to connect the copies to the desired output and thus establish the desired multicast connection.

We can also define a $(\nu, \mu)$-limited copy network analogous to a $(\nu, \mu)$-limited generalized connector. A $(\nu, \mu)$-limited copy network connects a set of inputs to a set of outputs with multicast connections as long as the number of outputs reached is at most $\nu$ and
each node is connected to at most $\mu$ outputs.

The crucial lemma in [FFP88] relates the copying capability of a bipartite graph with its expansion property. Let $B = (U, V, E)$ be a bipartite graph with $|U| = |V| = n$ and $(\alpha'', 3)$-expansion property.

**Lemma D.1** The bipartite graph $B$ described above is a $(\frac{\alpha''}{2} n, 2)$-limited wide sense non-blocking copy network if each node in $U$ is an input and each node in $V$ is an output.

The proof of this lemma can be found in [FFP88]. We shall not present the proof here since it is not very illustrative. The paths found in this case are node disjoint.

Now, let us see what happens when we put two copies of $B$ in cascade. This is a three stage network. The nodes in the first stage are inputs and the nodes in the last stage are outputs. The middle stage together with the first stage makes the first copy of $B$ and the middle stage with the last stage makes the second copy of $B$. We claim that such a network is an $(\frac{\alpha''}{4} n, 4)$-limited wide sense nonblocking copy network. Let us see why this is so. In an $(\frac{\alpha''}{4} n, 4)$-limited copy network, an input needs to be connected to at most four outputs. If an input needs to be connected to two or more outputs, it can reach two nodes in the middle stage and then the two nodes can each reach one or two outputs to achieve the desired number of output copies for the input. If an input only needs to reach one output, it reaches one node in the middle stage and that node reaches one of the output nodes. A little thought shows that breaking up the connections into two stages like this ensures that each copy of $B$ is required to act as an $(\frac{\alpha''}{2} n, 2)$-limited copy network. But we know from Lemma D.1 that the network $B$ is in fact a $(\frac{\alpha''}{2} n, 2)$-limited copy network.

A slight generalization of this argument shows that if we cascade a $(\nu, \mu_1)$-limited copy network and a $(\nu, \mu_2)$-limited copy network, the resulting network is a $(\nu, \mu_1 \mu_2)$-limited copy network. Hence, by cascading $\lceil \log(\frac{\alpha''}{4} n) \rceil$ copies of $B$ one after the other, we can get a $(\frac{\alpha''}{2} n, \frac{\alpha''}{2} n)$-limited copy network. This way, we can get up to $\frac{\alpha''}{2} n$ copies of each input at the last stage and the total number of copies can be at most $\frac{\alpha''}{2} n$. In a generalized network we may need up to $n$ copies. One way to achieve that is by having more than
one layer. We shall later compute the number of layers needed for our purposes. Now, we need to cascade a wide sense nonblocking switch after the copy network to get a generalized connector.

The wide sense nonblocking switch we use here is the shuffle-Beneš network discussed in Chapter 4. We know from Corollary 4.3 that $2T - 1$ layers of a shuffle-Beneš network is a strict sense nonblocking switch with parameters $\alpha, \beta, \alpha', \beta'$ and $T$ if $T$ satisfies Inequality 4.1. Since a strict sense nonblocking switch is also a wide sense nonblocking switch, the shuffle-Beneš network is sufficient. By cascading the shuffle-Beneš network with the copy network we get the generalized connector. The new network is nothing more than the copy-Beneš network introduced in Subsection 6.3.1.

Let us see how the copy-Beneš network can be used as a strongly wide sense nonblocking generalized connector. The number of nodes in each stage of a copy-Beneš network is $n = 2^k$. The first $k + 1$ stages form a $(\frac{\alpha'}{\alpha} 2^k, \frac{\alpha'}{2} 2^k)$-limited copy network. We shall refer to the first $k + 1$ stages as the copy part of the network and the last $2k + 1$ stages as the switch part of the network due to obvious reasons. One stage is common between the two parts of the network. In a generalized connector, an input may want to connect to all the outputs and hence we may need to make $2^k$ copies of an output. The number of layers needed to do this is $\frac{2^k}{\alpha'}$. Also, the sum of the number of copies needed equals the total number of destinations reached by all the multicast connections combined. Hence, it is at most $2^k$ and having $\frac{2^k}{\alpha'}$ layers makes sure that the copy part of the copy-Beneš network will be able to make enough copies. Clearly, not all copies can be made on one layer. Suppose now that the copies of the inputs are available at the end of the copy part of the network. We have some freedom in choosing exactly which copy should be connected to which output if more than one output is reached by a multicast connection. The number of copies of an input is the same as the number of outputs that the input wants to reach. We do an arbitrary one to one mapping between the copies and the outputs to determine exactly which copy should be connected to which output. Now, we use the switch part of the network to establish these connections. Unfortunately, the
shuffle-Beneš network requires $2T - 1$ layers to become a strict sense nonblocking switch. Hence, we need $2T - 1$ layers of shuffle-Beneš for each layer of copies. Therefore, by having $2(2T - 1)/\alpha''$ layers, we are guaranteed to have enough copies and switch capacity to have a strongly wide sense nonblocking generalized connector.

It should be noted that one inputs multicast connection may be established over a number of layers. Although we are looking at the situation in which each destination connected to an input receives the same information over the channel, the input may send different information on each layer. We shall not use this feature of the copy-Beneš network in our work.
E  Existence and Construction of Expander Graphs

In this chapter we shall show that for given $\alpha$ and $\beta$, $(\alpha, \beta)$-expander graphs of arbitrary size can be constructed. Throughout the discussion in this chapter, we assume that we have a bipartite graph $B = (U, V, E)$ with $|U| = |V| = n$, where $n$ is called the size of the network.

E.1  Existence of Expander graphs

The material in this section is taken from [LLK91].

**Theorem E.1** For a given $\alpha > 0$ and $\beta > 1$ with $\alpha \beta < 1$, there is a $d$ such that for all $n$ there exists a bipartite graph $B$ with degree $d$ that is an $(\alpha, \beta)$-expander graph.

**Proof:** We shall construct a random bipartite graph of degree $d$. Let us first make a different graph. We construct two sets of $nd$ nodes called $U'$ and $V'$. For each node $u_0$ in $U$ there are $d$ nodes in $U'$ labeled $u^1_0, u^2_0, u^3_0, \ldots, u^d_0$. Similarly for each node $v_0$ in $V$ there are $d$ nodes in $V'$, labeled $v^1_0, v^2_0, v^3_0, \ldots, v^d_0$ respectively. Now we find a random perfect matching between the nodes of $U'$ and $V'$, i.e., a perfect matching is picked from the set of all perfect matchings each with probability $(dn)!^{-1}$.

We now have a bipartite graph of size $nd$ in which each node has degree 1. We add edges between the nodes of $U$ and $V$ in the following manner. If $u^j_0 \in U'$ has an edge to $v^i_0$ for $1 \leq i, j \leq d$ then we add an edge from node $u_0 \in U$ to node $v_0 \in V$. The set of edges constructed this way is named $E$. Now we have a bipartite graph $B = (U, V, E)$ with $|U| = |V| = n$. Each node in graph $B$ has degree $d$. It should be noted that there may be multiple edges between two nodes in this graph. For a given $\alpha > 0$ and $\beta > 1$ such that $\alpha \beta < 1$, we want to compute the minimum value of $d$ for which the graph $B$ has a positive probability of being an $(\alpha, \beta)$-expander graph. If the $d$ found in this way is independent of $n$, the proof of the theorem is complete.

Let us pick a set $S \subset U$ with $|S| \leq \alpha n$ and a set $R \subset V$ with $|R| = \beta |S|$. If the neighbor set of $S$ is a proper subset of $R$ then graph $B$ is not an $(\alpha, \beta)$-expander graph.
graph. The probability that the neighbor set of \( S \) is a proper subset of \( R \) is less than
the probability that the neighbor set of \( S \) is a subset of \( R \) which is the same as the
probability that all the \( d|S| \) nodes in \( U' \) that correspond to the nodes in \( S \) get matched
to a subset of the \( \beta d|S| \) nodes in \( V' \) that correspond to the nodes in \( R \). The probability
of that event is

\[
\frac{(\beta d|S|)! (dn - d|S|)!}{(\beta d|S| - d|S|)! (dn)!}.
\]

By summing this over all the sets \( S \) and \( R \) we get that \( p_n \), the probability that graph \( B \)
is not an \((\alpha, \beta)\)-expander graph, is upper bounded by

\[
p_n \leq \sum_{|S|=1}^{\alpha n} \left( \begin{array}{c} n \\ |S| \end{array} \right) \left( \begin{array}{c} n \\ \beta |S| \end{array} \right) \frac{(\beta d|S|)! (dn - d|S|)!}{(\beta d|S| - d|S|)! (dn)!}.
\] (E.1)

By replacing \(|S|\) by a dummy variable \( k \) and multiplying both the numerator and the
denominator by \((dk)!\), we can write the equation as

\[
p_n \leq \sum_{k=1}^{\alpha n} \left( \begin{array}{c} n \\ k \end{array} \right) \left( \begin{array}{c} n \\ \beta k \end{array} \right) \frac{\beta dk! (dn - dk)!}{dk! (dn)!}.
\] (E.2)

We use the following inequalities to get an upper bound on \( p_n \).

\[
\sqrt{2\pi N} N^N e^{-N} \leq N! \leq e\sqrt{N} N^N e^{-N} \text{ for all } N \geq 1.
\] (E.3)

This inequality is taken from [Gal92]. We now find bounds on \( \left( \begin{array}{c} b \\ a \end{array} \right) \).

\[
\left( \begin{array}{c} b \\ a \end{array} \right) \leq \frac{b^a}{a!}
\] (E.4)

\[
\leq \frac{b^a}{\sqrt{2\pi a^a} e^{-a}}
\] (E.5)

\[
= \frac{1}{\sqrt{2\pi a}} \left( \frac{be}{a} \right)^a
\] (E.6)
We used Inequality E.3 to go from E.4 to E.5. We want to find another bound on \( \begin{pmatrix} b \\ a \end{pmatrix} \) but this time we write \( b \) as \( \gamma a \).

\[
\begin{pmatrix} 
\gamma a \\
\ a
\end{pmatrix} = \frac{\gamma a!}{((\gamma - 1)a)!a!} \tag{E.7}
\]

\[
\leq \frac{e\sqrt{\gamma a \left( \frac{\gamma a}{e} \right)^{\gamma a}}}{\sqrt{2\pi (\gamma - 1)a \left( \frac{1}{e} \right)^{(\gamma - 1)a}}} \frac{\sqrt{2\pi a \left( \frac{a}{e} \right)^a}}{\sqrt{\gamma (\gamma - 1)^{\gamma a}}} \tag{E.8}
\]

\[
\leq \frac{\sqrt{\gamma (\gamma - 1)^{\gamma a}}}{\sqrt{\gamma - 1)(\gamma - 1)^{(\gamma - 1)a}}} \sqrt{a} \tag{E.9}
\]

\[
\leq C \frac{(\gamma)^{\gamma a}}{\sqrt{a} (\gamma - 1)^{(\gamma - 1)a}} \tag{E.10}
\]

\[
= C \frac{1}{\sqrt{a}} \left( \frac{\gamma}{(1 - \frac{1}{\gamma})^{(\gamma - 1)}} \right)^a \tag{E.11}
\]

where \( C \) is a constant larger than \( \frac{1}{\sqrt{\gamma - 1}} \). We used Inequality E.3 to go from E.7 to E.8. The step from E.8 to E.9 is because \( \frac{e}{2\pi} < 1 \).

From Inequalities E.3, E.6 and E.11 we get

\[
\begin{pmatrix} 
n \\
bk
\end{pmatrix} \leq \frac{1}{\sqrt{2\pi n}} \left( \frac{ne}{k} \right)^k \tag{E.12}
\]

\[
\begin{pmatrix} 
n \\
\beta k
\end{pmatrix} \leq \frac{1}{\sqrt{2\pi n}} \left( \frac{ne}{\beta k} \right)^{\beta k} \tag{E.13}
\]

\[
\begin{pmatrix} 
\beta dk \\
dk
\end{pmatrix} \leq C \frac{1}{\sqrt{dk}} \left( \frac{\beta}{(1 - \frac{1}{\beta})^{(\beta - 1)}} \right)^{dk} \tag{E.14}
\]

\[
(dk)! \leq e\sqrt{dk} \left( \frac{dk}{e} \right)^{dk} \tag{E.15}
\]

\[
((dn - dk))! \leq e\sqrt{(dn - dk)} \left( \frac{dn - dk}{e} \right)^{(dn - dk)} \tag{E.16}
\]
(dn)! \geq \sqrt{2\pi dn} \left(\frac{dn}{e}\right)^{dn}. \quad (E.17)

C in E.14 is $\sqrt{\beta/\beta - 1}$.

Substituting E.12 through E.17 in E.2, we get

\[ p_n \leq C \sum_{k=1}^{\infty} \frac{\frac{1}{\sqrt{2\pi k}} \left(\frac{ne}{k}\right)^k \frac{1}{\sqrt{2\pi \beta k}} \left(\frac{ne}{\beta k}\right)^k \frac{1}{\sqrt{dk}} \left(\frac{\beta}{(1 - \frac{1}{\beta})^{\beta - 1}}\right)^{dk} e^{\sqrt{dk} \left(\frac{dk}{e}\right)^{dk}} e^{\sqrt{dn - dk} \left(\frac{dn - dk}{e}\right)^{dn - dk}}}{\sqrt{2\pi dn} \left(\frac{dn}{e}\right)^{dn}}. \quad (E.18)\]

\[ \leq C \sum_{k=1}^{\infty} \frac{\left(\frac{ne}{k}\right)^k \left(\frac{ne}{\beta k}\right)^k \left(\frac{\beta}{(1 - \frac{1}{\beta})^{\beta - 1}}\right)^{dk} \left(\frac{dk}{dn}\right)^{dk} \left(\frac{dn - dk}{dn}\right)^{dn - dk}}{(dn)^{dn}}. \quad (E.19)\]

\[ = C \sum_{k=1}^{\infty} \left(\frac{ne}{k}\right)^{k-1} \left(\frac{ne}{\beta k}\right)^{k} \left(\frac{\beta}{(1 - \frac{1}{\beta})^{\beta - 1}}\right)^{dk} \left(\frac{dk}{dn}\right)^{dk} \left(\frac{dn - dk}{dn}\right)^{dn - dk}. \quad (E.20)\]

\[ = C \sum_{k=1}^{\infty} \left(\frac{k}{n}\right)^{d-1-\beta} e^{1+\beta d-\beta} \left(\frac{\beta}{1 - \frac{1}{\beta}}^{\beta - 1}\right)^{d} \left(1 - \frac{k}{n}\right)^{(\frac{d}{\beta}-1)d} \left(\frac{1 - \frac{k}{n}}{1 - \frac{1}{\beta}}^{\beta - 1}\right) \quad (E.21)\]

\[ \leq C \sum_{k=1}^{\infty} \left(\frac{k}{n}\right)^{d-1-\beta} e^{1+\beta d-\beta} \left(\frac{1 - \frac{k}{n}}{1 - \frac{1}{\beta}}^{\beta - 1}\right)^{d} \quad \text{for } d > 1 + \beta. \quad (E.22)\]

E.19 follows from E.18 because

\[ \frac{e^2}{(2\pi)^{\frac{1}{2}}} \frac{1}{\sqrt{k}} \frac{1}{\sqrt{\beta k}} \sqrt{\frac{dn - dk}{dn}} \leq 1. \]

E.22 follows from E.21 because $\frac{k}{n} \leq \alpha$. C in E.18 through E.22 is $\sqrt{\beta/(\beta - 1)}$.

It can be seen that

\[ f(x) = (1 - x)^{\frac{1}{\beta}-1}. \]
is an increasing function of $x$ for $x \in (0,1)$. Since $\alpha \beta < 1$ and $\frac{x}{\alpha} \leq \alpha, \frac{1}{\beta} > \frac{k}{\alpha}$ and therefore the term
\[
\left(\frac{1 - \frac{x}{\alpha}}{(1 - \frac{1}{\beta})^{\beta - 1}}\right)^d
\]
in E.22 is less than 1. Replacing this term by 1, we get the following bound on $p_n$,
\[
p_n \leq C \sum_{k=1}^{\infty} \left((\alpha \beta)^{d - 1 - \beta} \beta e^{1+\beta}\right)^k \quad (E.23)
\]
\[
\leq C \sum_{k=1}^{\infty} \left((\alpha \beta)^{d - 1 - \beta} e^{1+\beta}\right)^k. \quad (E.24)
\]
Since $\alpha \beta < 1$, for large enough $d$, the right hand side of Inequality E.24 is a decreasing geometric series. We want to determine the values of $d$ for which it is not only a decreasing geometric series but also sums to less than 1. To ensure that the right hand side of E.24 is less than 1 the following inequality is sufficient.
\[
(C + 1) \left((\alpha \beta)^{d - 1 - \beta} e^{1+\beta}\right) < 1 \quad (E.25)
\]
By taking logarithm of both sides we get
\[
d > 1 + \beta - \frac{1 + \beta + \ln \beta + \ln (C + 1)}{\ln (\alpha \beta)}. \quad (E.26)
\]
It is easy to see that the right hand side of E.26 is independent of $n$. This proves the theorem.

[LPS86] gives an explicit construction for expander graphs with $\alpha = 0.5$ and a given $\beta$. [FFP88] modifies the construction of [LPS86] to construct expander graphs with different $(\alpha, \beta)$. The degree needed to achieve a given $(\alpha, \beta)$ in these constructions are much larger than the degree shown by the existence proof described above. For example, to construct a $(\frac{1}{9}, 3)$-expander graph the explicit construction given in [FFP88] requires
that each node must have degree 54. The degree given by the random graph argument is only 10 for the same expansion.
References


[Lie88] Soung Chang Liew. Capacity Assignment in Non-Switching Multichannel Networks. Technical Report LIDS-TH-1778, Laboratory for Information and


