Computer Science Department Technical Report
University of California
Los Angeles, CA 90024-1596

A PERFORMANCE MODEL OF DEFLECTION ROUTING AND ITS APPLICATION TO THE TOPOLOGICAL DESIGN OF MULTICHANNEL NETWORKS

Joseph Bannister
Flaminio Borgonovo
Luigi Fratta
Mario Gerla

January 1991
CSD-910002

A Performance Model of Deflection Routing and its Application to the Topological Design of Multichannel Networks[1]

Joseph Bannister
The Aerospace Corporation
El Segundo, California, USA

Flaminio Borgonovo
Istituto di Elettronica
Università di Catania, Italia

Luigi Fratta
Dipartimento di Elettronica
Politecnico di Milano, Italia

Mario Gerla
Computer Science Department
University of California at Los Angeles, USA

Abstract

Deflection routing, which can be used in networks with multiply connected stations and limited packet buffers, works by randomly diverting packets from their intended routes when two or more packets simultaneously contend for a common output port. The potentially unbounded number of routes that a given packet can take makes performance analysis of such networks difficult. In this paper we develop a high-fidelity performance model of deflection routing that can be applied to a network with any given topology and traffic matrix. The performance model is used to solve the virtual-topology design problem, which is to select the virtual topology of a wavelength-agile multichannel network that yields optimum performance under a given traffic loading. The optimized virtual topologies, found via simulated annealing, achieve significantly higher performance than unoptimized virtual topologies, such as ShuffleNet or the Manhattan Street Network.

Nomenclature

B_{ik}	service requirement for class-k packets at station i (random variable)
B_i	service requirement for packets at station i (random variable)
$B(z)$	z-transform of a discrete random variable B
γ_{st}	probability that the user at station s generates a packet destined for station t
γ_s	probability that the user at station s generates a packet
γ	probability that a packet is generated (offered traffic load)
$\delta_{ijk:t}$	probability of deflecting a t-packet away from internal link (i,j,k)
$\Delta(i,t)$	alternate route from station i to station t

[1]This work was partially supported by the Telecommunication Project of the CNR of Italy, US DARPA contract MDA903-87-C-0663, and State of California–Pacific Bell MICRO grant 4-541121-19907 KC7F.

$\mathbb{E}[\cdot]$	expectation operator
η_i	rate at which traffic departs from the user output port of station i
(i,j,k)	internal link from input port j to output port k of station i
$(i,j,k) \rightarrow (l,m,n)$	output port k of station i is tuned to input port m of station l
L_{ik}	number of packets in output queue k of station i (random variable)
$\lambda_{ijk:t}$	probability of finding a t-packet on internal link (i,j,k)
λ_{ijk}	probability of finding a packet on internal link (i,j,k)
N	number of stations in the network
M_i	number of packets in the user input queue at station i (random variable)
M	number of packets in the network (random variable)
μ_{ik}	service rate for class-k packets at the user input queue of station i
$\mathbb{P}[\cdot]$	probability of an event occurring
$\Pi(i,t)$	primary route from station i to station s
T	end-to-end packet delay (random variable)
$\phi_{ij:t}$	probability of finding a t-packet on the link coming into input port j of station i
ϕ_{ij}	probability of finding a packet on the link coming into input port j of station i
$\bar{x}$	logical complement of Boolean value x
X	random variable for generating the traffic matrix

1 Introduction

Lightwave technology, which continues to provide increasing bandwidth in a single-mode optical fiber, is the foundation for new network architectures designed to carry a large amount of traffic between many users. Wavelength-division multiplexing (WDM), which allows us to operate independent channels on an optical fiber, is the technological basis for the wavelength-division optical network (WON), a recently proposed architecture for high-speed networks [BFG89, Ban90, BG90, BFG90c, BFG90b]. Because WDM has the potential to support as many as 3000 1-gigabit/second (Gb/s) channels on a single optical fiber [VW89], the WON can concurrently transmit many messages to achieve high throughput.

With the introduction of a new network architecture—such as the WON—there arise novel problems in the analysis, design, and operation of this type of network. Our objectives are to develop techniques to evaluate analytically the performance of the WON and then to apply these techniques to design optimum topologies for the WON.

The design and analysis of the WON has been addressed in previous research. The first work on topological design of the WON was reported in [Ban90, BG90, BFG90c, BFG90b] where the problem of choosing a topology that minimizes delay is studied. That work, assuming ample packet buffers at each station and shortest-path routing, formulated a simple queueing-network model of the WON and derived a closed-form expression for delay; the expression was then used as the objective function of the topological-design problem. In [GG86, Aya89, ZA90, TB90, BC90a] the analyses of the WON under the assumption of limited packet buffers and deflection routing were undertaken, but these analyses were restricted to the case of uniform traffic and specific topologies. The analytical model presented in this paper goes beyond previous work by providing a method to evaluate the performance of the WON under an arbitrary traffic matrix. Also, our method is not restricted to certain classes of topologies, as are other approaches. The recent work reported in [BC90b] also attacks the problem of analyzing the performance of the Manhattan Street Network (MSN) with limited packet buffers and deflection routing under nonuniform traffic. Their work uses a different approach than the one that we offer in this paper.

This paper is organized as follows. In Section 2 we describe the architecture of the WON and the use of deflection routing in the WON. In Section 3 we develop an analytical model of the WON and use it for evaluating the WON's performance when deflection routing is used and stations possess only single output buffers. The model is then applied in Section 4 to the topological design of the WON; the optimization is done using simulated annealing with a cost function that incorporates the analytical model developed in the previous section. We also illustrate the techniques of topological design on two example 64-station WONs in Section 4. In Section 5 we conclude the paper by reviewing our results and discussing important research topics in this area.

2 Description of the Network

In this section we provide an overview of the architecture of the WON. We also describe the use of deflection routing in the WON and discuss the role of admission control in maintaining satisfactory levels of packet loss in the WON.

2.1 The Architecture of the Wavelength-Division Optical Network

The WON is a high-speed, multichannel, multihop network for use as a metropolitan area network or geographically dispersed local area network. The WON's point of attachment for its users is the station, which is an electronic store-and-forward packet switch with two optical receivers and transmitters, each of which can be independently tuned to a separate WDM channel. By appropriately tuning all stations' transceivers, a source station can send a packet to any destination station in a multihop manner via a sequence of intermediate WDM channels and stations.

The WON is similar to other multichannel networks that have been proposed in the past, e.g., Sytek's LocalNet [Bib81], but its implementation using lightwave technology distinguishes it from these earlier proposals, which were typically based on cable-television technology. Whereas cable-television systems can multiplex only tens of low-speed (e.g., 10 megabits/second) channels, lightwave systems can multiplex thousands of high-speed (e.g., 1 Gb/s) channels. Such a large collection of high-speed WDM channels can be used to interconnect many electronic stations via a shared optical medium. This type of network, first proposed by Acampora [Aca87, AKH87] and variously called the Multichannel Multihop Lightwave Network or ShuffleNet, uses a specific station tuning based on the perfect-shuffle directed graph. This network architecture, which can interconnect N dual-transceiver stations via $2N$ WDM channels, has since been extended and developed in several areas, especially by considering wavelength-agile transceivers that admit many different station interconnection patterns [Ban90, BG90, BFG90c, BFG90b, LA90]. In this paper we concern ourselves with WONs that have wavelength-agile transceivers. This wavelength agility permits the network administrator to specify a particular tuning, which results in a station interconnection pattern called the *virtual topology* of the WON. We mention that the virtual topology of the WON does not change dynamically, but rather remains fixed for an extended period of time, and changes only when the network administrator decides that a new virtual topology is required to better meet the needs of the users, e.g., because of changing traffic conditions.

To illustrate, we show in Figure 1 an example of an N-station, dual-transceiver WON implemented as a physical bus. As WON performance is extremely sensitive to the virtual topology used, the selection of the virtual topology is an important design decision. The virtual topology of the WON is established by tuning all stations' transmitters and receivers to effect a specific station interconnection pattern. The WDM channels can be shared by several stations or dedicated to a single pair of transmitting and receiving stations. In this paper we focus exclusively on dedicated-channel networks. An example of a virtual topology for a 16-station WON that is

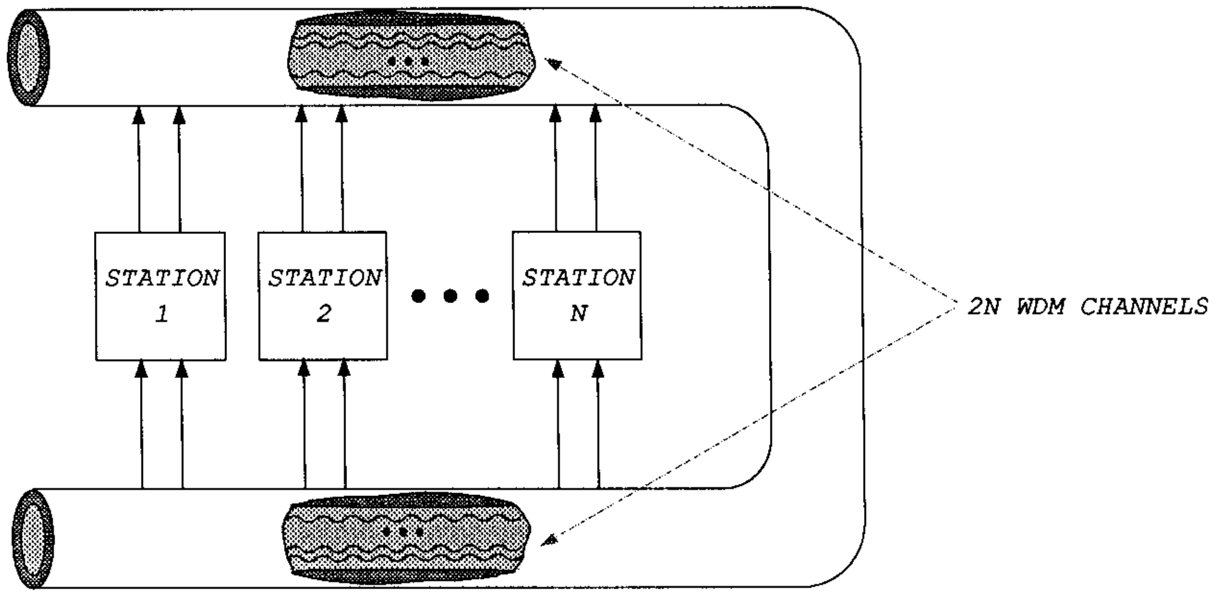


Figure 1: An Eight-Station WON Realized as a Tree.

based on the two-dimensional torus is shown in Figure 2. With the virtual topology of Figure 2, a packet that originates from station 2 would be delivered to station 6 by traversing the shortest route $2 \rightarrow 5 \rightarrow 3 \rightarrow 6$.

2.2 Deflection Routing

To reduce the cost and complexity of the station, we may configure each transmitter with as few as one packet buffer. Given that packets have a fixed size and that time is slotted so that a packet may be transmitted during a time slot, if two packets simultaneously arrive to the station, and both need to be transmitted via the same transmitter, then one packet can be rerouted via the other transmitter. Thus, both packets can be transmitted in the next time slot. We assume that each packet has a primary route to get to its destination, but if the packet is deflected from the primary route, the rerouted packet has an alternate path for reaching its destination; how this could be implemented will be discussed later. All routes are shortest paths with respect to the output port being used, and each packet has a single primary and alternate route from any given station to its destination, even if the station is a "don't care" node with equally short paths emitting from every output port. For example, in Figure 2 the primary route from station 0 to station 7 is $0 \rightarrow$

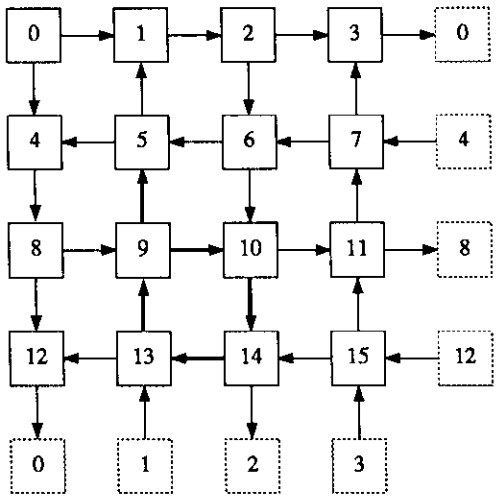


Figure 2: The Two-Dimensional Toroidal Virtual Topology.

4 → 7, and an alternate route is 0 → 1 → 2 → 6 → 10 → 11 → 7. This routing scheme, called *deflection* (or hot-potato) routing, has been proposed for the Manhattan Street Network (MSN) [Max85], which has a topology based on the two-dimensional torus shown in Figure 2. The MSN has the advantage that a deflected packet can simply make four additional hops to get back to its deflecting station, whence it can reattempt to continue along its intended path. Since packets are not normally queued within the network, they might have to be queued or possibly discarded (upon buffer overflow) at the entrance to the network if no free slots are available.

In a high-speed network such as the WON, mechanisms for routing should ideally be implemented in hardware so that these functions are performed in real time at rates comparable to the link speed. Upon arrival at a station the packet's header is quickly analyzed and switched to the output port that goes to the packet's next hop. The packet's next hop is specified by a preference vector, which lists for each possible destination the primary and alternate output ports for the packet. Unlike lower-speed packet-switch networks that rely on software to look up the next hop in a program-maintained routing table, switching in the WON would be performed entirely in hardware. Instead, the WON could provide hardware support for fast deflection routing with a small amount of additional circuitry. For example, each station could use a 16-kilobit by 3-bit random-access memory to store an encoding of the next hop of the primary and the alternate route (viz., output port 0, 1, or 2) for every possible destination (assuming that the network address space can be specified in 14 or fewer bits). Then, using the destination's network address to address the random-access memory, the station could rapidly read the output port number for both the primary and alternate routes. Since static random-access memories are built with access times as low as a few nanoseconds, the time to look up the next hop would be less than a few bit times on a 1-Gb/s channel. The decision logic for resolving contention and determining which of the contending packets to deflect could also be implemented easily in hardware. If packet switching were to require any more than a the equivalent of few bit-transmission times, the station would become a performance bottleneck.

Although originally proposed for the MSN, deflection routing can be used in the WON with any virtual topology, so long as every station is provided with two receivers and two transmitters. It is clear that deflection routing applies equally well when there are more than one buffer per transmitter. In that case deflection will occur only when there are more packets in need of service than there are buffers available. It is straightforward—though tedious—to extend the results of this paper to WONs with stations having an arbitrary number of transceivers, but we assume throughout the remainder of the paper that every station has two transceivers.

It should also be noted that deflection routing requires facilities for synchronizing the WON, so that time is divided into discrete slots, and this introduces additional complexity into the network. Normally, the arriving packets would be synchronized by means of an alignment function at the receivers of the station.

Although propagation delay in a high-speed lightwave metropolitan area network would comprise a significant component of the total packet delay, we choose to simplify our analysis by ignoring all propagation effects. Our analysis could be extended to account for propagation delay without difficulty, but the analytical results acquire a more complicated appearance. Equivalently, ignoring propagation effects can be viewed as using a network in which the propagation delays on all links are equal.

2.3 Admission Control

A price to be paid for deflection routing is that packets may be discarded at the user input port before they are ever admitted into the network. Thus, the problem of admission control also plays an important role. Several admission policies have been identified and studied in [BT89], including prerouting and postrouting access, in which the user's packets are admitted into the network before or after routing decisions are made. In postrouting access it is possible that a packet just admitted into the network can suffer an immediate deflection, whereas prerouting never deflects a packet until after it leaves its station of origin. We adopt here two simple policies that are similar in spirit to the prerouting access method, viz., fully queued access and independently queued access. In fully queued access there is one user input queue at which packets queue up in order of arrival, and during a time slot at most one packet is taken from the head of the queue when its transmitter is free. In independently queued access there are two independent user input queues, one for each output link. Packets queue up at a user input queue, depending on which output link is needed, and at most one packet is taken from the head of the queue when its transmitter is free. Our model could also accommodate WONs using the postrouting access method; we could utilize the analysis of [TB90], which is based on a bulk-service queueing model of the user input port. In this paper we assume that the user input port provides sufficient packet buffers to minimize the loss of packets entering the network. This assumption is not essential, and our models could be easily modified to accommodate limited user-input buffering.

As with admission policies, there are different options for routing and switching. Although in the MSN there are several shortest paths from a source to a destination, it has been claimed that specific types of shortest paths are desirable. For example, shortest paths that minimize the number of right-angle turns [GG86] or that follow the diagonal [BC87] have been proposed for the MSN. Since we do not restrict ourselves to any specific virtual topology, we do not require anything of a route other than it be a shortest path. When two packets arrive simultaneously to the station and contend for the same output port, the switching policy decides which packet will be deflected. Again, we do not consider policies based on topology-specific rules, such as the straight-through policy of [GG86, BC90a]. Nor do we consider policies that give preference to the packet with the greatest age or least distance from its destination, such as those discussed in [BC90a]. We assume that the decision of which of two contending packets to deflect is determined by a fair coin toss. This last assumption can also be relaxed, as we shall discuss in the concluding section.

3 A Performance Model of the Single-Buffer Deflection-Routing Network

In this section we formulate a queueing model of the WON intended to estimate the mean packet delay when deflection routing is used. We then propose a procedure for solving the model and demonstrate the fidelity and efficiency of the procedure.

Although simulation is a very accurate method to evaluate the performance of the WON, it is too expensive to be used in some applications. For example, the cost of simulation is prohibitive when we need to evaluate the performance of many different networks, as in the topological design problem to be presented in the next section. In such situations we need a fast algorithm to determine the performance of a given network. Because the use of simulation to evaluate the performance of the WON is relatively expensive, in this section we develop an analytical performance model of the WON that can be applied more efficiently than simulation. In the next section we employ the model to design optimum topologies for the WON.

3.1 Assumptions of the Model

Our mathematical model is derived for an N-station dedicated-channel WON in which every station has two input ports from the network and two output ports to the network, as shown in Figure 3. Each output port has a single packet buffer; thus, deflection routing is used to cope with packet contention. There are also a user input port, which is assumed to have sufficient packet buffers to accommodate the station's exogenous traffic, and a user output port, which has a single packet buffer. Time is assumed to be slotted and all transmissions are synchronized. A station is able to switch and transmit its incoming packets to the next station within a single time slot. When two packets simultaneously contend for the same output port, the priority arbiter decides to deflect one of the packets at random. A primary route is a shortest path from the source to the destination, and an alternate route is a shortest path from the port used by the deflected packet to its final destination. We do not address explicitly how the routing table is maintained, but postulate a route-updating algorithm that instantaneously discovers the shortest possible primary and alternate routes for all destinations. Immediately after packets from the input links have been switched to their appropriate output ports, the packet at the head of the user input buffer is examined by the priority arbiter and switched to the appropriate output port, if it is free. If the required output port is busy, then no packet from the user input port receives service in the current time slot and must reattempt access in the subsequent time slot. We assume that the packet at the head of the queue will wait until the output port on its primary route has a free slot. Consequently, even if the packet's primary output port is busy and its alternate output port is free, the packet will wait until its primary port becomes free. We show in Figure 3 the structural attributes of the station; in the following we assume that $Q = 1$ and $L = \infty$.

The performance of the WON is clearly influenced by the traffic offered to the network. In an N-station WON we denote by the $N \times N$ matrix (γ_{ij}) the average rate (in packets per time slot)

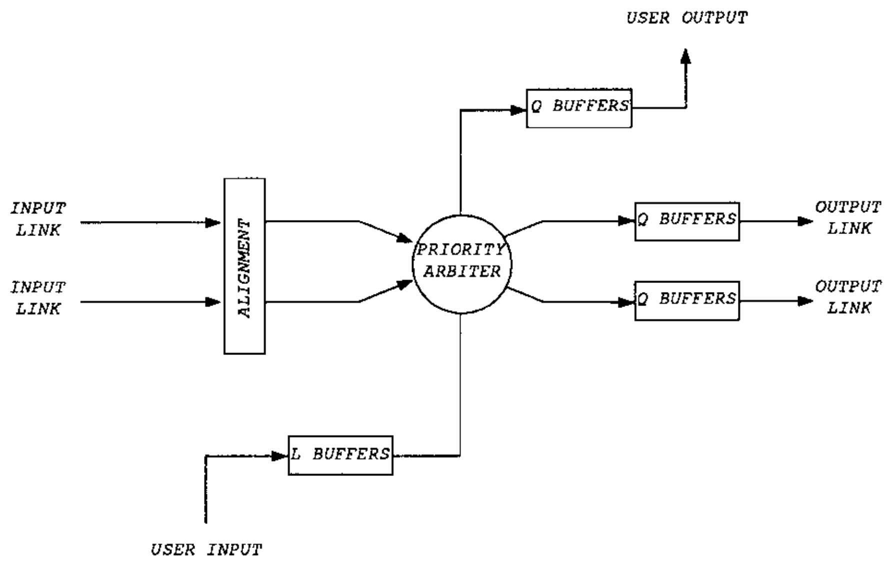


Figure 3: Structural Attributes of the Station.

at which traffic originating from source station i is sent to destination station j. During any time slot the user at station i may submit no more than one packet to the network, and the probability that this user submits a packet destined for station j is γ_{ij}. Hence, the probability that the user generates a packet is given by $\gamma_i \triangleq \sum_j \gamma_{ij}$. We assume that streams of packets between source–destination pairs are generated by users independently of each other and that the intervals between successively generated packets are independent of each other.

An *internal link* is a triple (i,j,k) that represents the logical channel from input port j to output port k within station i. Assuming that each station has exactly two input ports from the network and two output ports to the network,[2] we can list station i's internal links as $(i,0,0)$, $(i,0,1)$, $(i,0,2)$, $(i,1,0)$, $(i,1,1)$, $(i,1,2)$, $(i,2,0)$, $(i,2,1)$. [Note that we do not consider $(i,2,2)$ to be an internal link.] We define λ_{ijk} to be the rate (in packets per time slot) at which traffic flows over internal link (i,j,k) and ϕ_{ij} to be the rate (in packets per time slot) at which traffic flows into port j of station i. [We make the convention that $\lambda_{i22} = 0$ for all i.] These flows are illustrated in Figure 4. Since all traffic flowing into station i from the network must use one of the internal links,

[2]We use the convention that the (input and output) network ports are labeled 0 and 1, and the (input and output) user port is labeled 2.

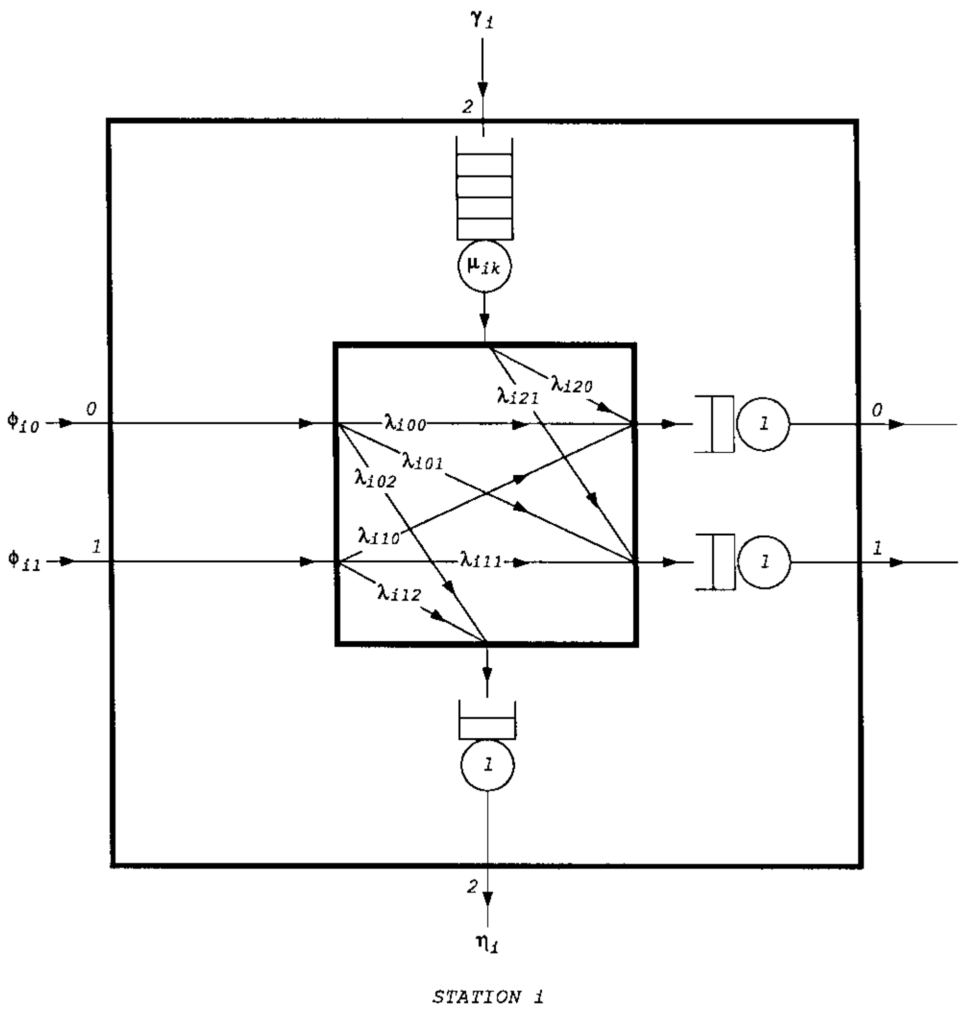


Figure 4: Link Flows and Queueing Centers in the Station.

we see immediately that

$$\phi_{ij} = \sum_{k=0}^{2} \lambda_{ijk} \tag{1}$$

where $1 \le i \le N$ and $0 \le j \le 1$.

Routes can be represented by the internal links that comprise the route. For a given virtual topology a path is completely specified by listing the internal links on the path, since each output port is connected to a unique input port. Formally, we define a path from station s to station t as a set of internal links $\{(i_1, j_1, k_1), (i_2, j_2, k_2), \ldots, (i_n, j_n, k_n)\}$ such that $i_1 = s$, $j_1 = 2$, $i_n = t$, $k_n = 2$, and for each m, $(i_m, j_m, k_m) \rightarrow (i_{m+1}, j_{m+1}, k_{m+1})$, i.e., output port k_m of station i_m is tuned to input port j_{m+1} of station i_{m+1}. A primary route from source station s to destination station t is the shortest path from s to t and can be formally described as the set—denoted $\Pi(s,t)$—of all internal links that comprise this shortest path. This set has the form $\Pi(s,t) = \{(s,2,i), \ldots, (t,j,2)\}$, where i and j are equal to 0 or 1, i.e., the path begins at s's user input port and ends at t's user output port. For the special case of $s = t$, we define $\Pi(s,t) = \emptyset$.

When a t-packet (i.e., one destined for station t) has been deflected at station r, it departs from its primary route and uses an alternate route, which is the shortest path from station r to destination t that does not use the same output port of station r as the primary path. If $(r,2,i) \in \Pi(r,t)$, then we represent the alternate route from r to t by $\Delta(r,t)$, the set of internal links $\{(r,2,i'), \ldots, (t,k,2)\}$ that forms the shortest path from output port i' of station r to station t such that $i' \neq i$.

The performance metrics of interest to us are delay and maximum throughput. Delay is a function of the number of hops that the typical packet makes and accounts for the queueing, transmission, propagation, and nodal-processing times of the packet. Maximum throughput is the total amount of traffic that the WON can handle without unbounded growth in delay. Given a traffic matrix (γ_{ij}), we figure maximum throughput by scaling up the matrix as much as possible: if the WON just saturates with offered traffic matrix $(\alpha\gamma_{ij})$ for some scale factor α, then the maximum throughput is equal to $\sum_{i,j} \alpha\gamma_{ij}$. In this paper we focus on the mean end-to-end packet delay, which is defined as the average time (in slots) that elapses from when a packet is generated by the user until it exits the WON. Since the virtual topology of the wavelength-agile WON can be reconfigured if it can not handle the offered traffic load, we conclude that the virtual topology should be optimized with respect to mean end-to-end packet delay, i.e., the chosen virtual topology should provide the lowest possible delay for the given traffic matrix and still be able to accommodate modest growth or fluctuation in the traffic.

We are therefore interested in determining analytically the value of $\mathbb{E}[T]$ for a given offered traffic load (γ_{ij}) and virtual topology $\mathcal{X}$, where the random variable T is the end-to-end packet delay. Since T depends upon the network's virtual topology $\mathcal{X}$ and the offered load (γ_{ij}), we sometimes write $T_{\mathcal{X},(\gamma_{ij})}$.

We have chosen to concentrate on end-to-end delay in our study of the WON, but we could also have used transport delay as the metric of interest, as in [BFG90a, BC90b]. Transport delay, defined as the elapsed time from a packet's admission into the network until its departure from the network, is included as a component of end-to-end delay. A complication of using transport delay would be that we must then take packet loss into account, because packets generated by a user can be discarded unless buffered by the user. Thus, as we scale up the traffic matrix, the throughput of the transport network increases steadily until a point is reached when all buffers of the transport network are continually occupied, at which point congestion sets in and throughput actually begins to decrease. With this model, however, mean packet transport delay is always bounded, because there is a finite number of buffers in the transport network. For this reason it is perhaps conceptually simplest to include unlimited user buffers in our model and consider end-to-end packet delay. Because of the unlimited packet buffering at the user input queues, throughput increases steadily as the traffic matrix is scaled up until a point of saturation is reached and throughput remains flat. At the point of saturation, however, mean end-to-end packet delay grows without bound. Since there are ample buffers in the network, no packets are ever discarded. It should be clear from the sequel that our model can be used to analyze transport as well as end-to-end performance.

To analyze the deflection-routing WON, we represent the WON as a network of queueing centers of the type depicted in Figure 4. The queueing network operates synchronously, so that all arrivals and departures occur only at the start of time slots. To analyze this queuing network we must be able to solve analytically for the rates of flow on all links and the mean numbers of packets in the various queues of the network. We address these tasks in the next two subsections.

3.2 The User Input Queue

Figure 4 also represents the queueing model of the station. At each of the three output ports there is a queue with space for exactly one packet and a server that can handle exactly one packet per time slot. At the user input port is a queue with unlimited space for buffering packets. A packet entering the user input queue will be switched to either output port 0 or 1, depending on its ultimate destination. (Notice that the packet enters the network only if it can use its primary route—the entering packet will never be placed on an alternate route.) Thus, we view the user input queue as a two-class queue in which the service rate depends on the class of the packet.

Calling a packet to be submitted from the user input queue to output port k a class-k packet, we observe that the service rate for a class-k packet will be $\mu_{ik} \triangleq 1 - \lambda_{i0k} - \lambda_{i1k}$, since a packet will only be switched from the user input port to output port k when there is no other packet waiting to be transmitted at output port k. In other words, during a given time slot the class-k packet in service will be placed on output link k with probability μ_{ik}. If we assume that the event that output port k will be free during a given time slot is independent of its being free during any other time slot, then this gives rise to a service time for class-k packets that is geometrically distributed with mean $(1-\mu_{ik})/\mu_{ik}$, i.e., the number of time slots that the packet must wait before being transferred to its output link is geometrically distributed. Therefore, the discrete random variable B_{ik}, which we define to be the amount of service (in time slots) required by a class-k packet, is characterized by the probability distribution

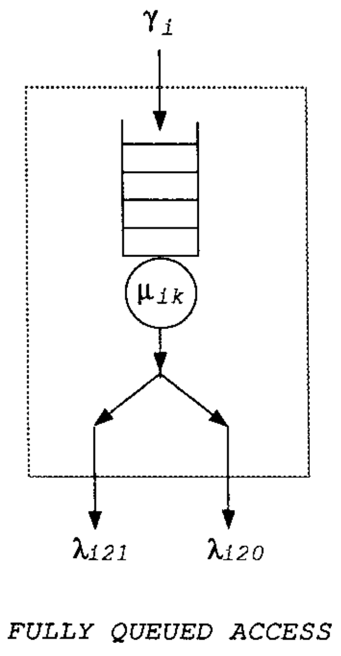
$$\mathbb{P}[B_{ik} = n] = \mu_{ik}\,(1-\mu_{ik})^n \tag{2}$$

for $n \geq 0$. The z-transform of the sequence specified by Equation (2) is

$$\begin{aligned} B_{ik}(z) &= \sum_{n=0}^{\infty} \mu_{ik}\,(1-\mu_{ik})^n\, z^n \\ &= \frac{\mu_{ik}}{1 - z + \mu_{ik} z} \end{aligned} \tag{3}$$

If $\lambda_{i2k} \geq \mu_{ik}$ then the user input queue can not adequately serve the user, and the queue will be unstable, so we assume henceforth that $\lambda_{i2k} < \mu_{ik}$.

In our analysis we consider two network-access strategies for the user input queue. The first access method to be considered is called fully queued access and the second is called independently queued access. In the fully queued access method, all packets from the user—regardless of their class—line up on the user input queue in first-come–first-served order. At the onset of each time slot, the server attempts to place the packet from the head of the queue onto its appropriate output link. If the output port is free (i.e., there is no other packet for it to transmit), then the server immediately places the packet on the output link and remains inactive until the next time slot. If the output port is busy (i.e., there is another packet for it to transmit), then the server becomes inactive until the next time slot, at which point it will again try to place the packet from the head of the queue onto the output link. In the independently queued access method, all packets from the user line up on the user input queue in first-come–first-served order, but there are two separate



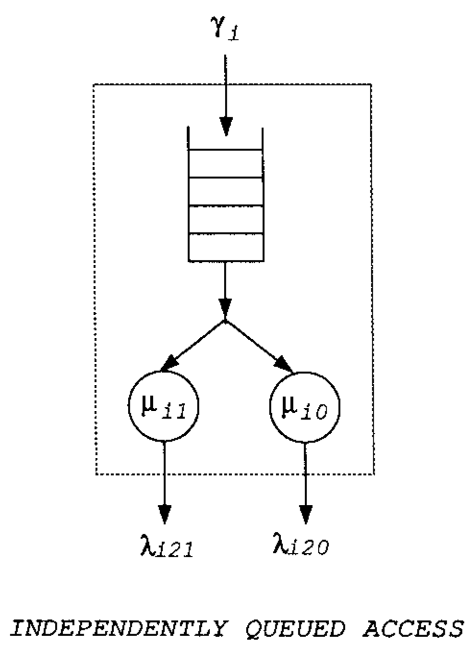


Figure 5: Two Access Methods for the User Input Queue: Fully Queued Access and Independently Queued Access.

servers, one for each packet class. At the onset of a time slot, server k removes the oldest class-k packet from the queue and attempts to place it onto output link k. Thus, there can be up to two packets in service, which increases the throughput of this queue, compared to one using the fully queued access method. We show in Figure 5 the structure of the user input queue employing both the fully and independently queued access disciplines.

An expression for the mean number of packets in the user input queue of station i depends on the access method used at the user input port. Regardless of the access method, we model the user input queue as a discrete-time queueing system in which packets arrive at the initiation of time slots. The probability that a packet is generated at station i is γ_i, and at most one packet is generated per user during any time slot. All transitions in the queueing system occur at the boundaries of time slots; hence, all arrivals and departures occur at the beginning of the time slot. The service-time requirement for a class-k packet, B_{ik}, is given by the probability mass function of Equation (2). We define the random variable M_i to denote the number of packets in the user input queue. The first moment of M_i is given by the Pollaczek–Khinchin mean-value formula for the Geom/G/1 discrete-time queue [Mei58]:

$$\mathbb{E}[M_i] = \gamma_i \,\mathbb{E}[B_i] + \frac{\gamma_i^2 \,\mathbb{E}[B_i(B_i-1)]}{2\,(1-\gamma_i\,\mathbb{E}[B_i])} \tag{4}$$

where B_i is the service time requirement of all packets in the user input queue.

Under the fully queued access discipline, the user input queue can be modeled as a Geom/H_2/1 queueing system, where H_2 represents the "hypergeometric" probability distribution. There are two classes of packets, and a clos-k packet has geometrically distributed service-time requirement B_{ik}. Thus the z-transform of all packets' service-time requirement is simply the weighted sum of $B_{i0}(z)$ and $B_{i1}(z)$. Using Equation (3) we calculate the z-transform of the service-time requirement B_i:

$$\begin{aligned} B_i(z) &= \frac{\lambda_{i20}\,B_{i0}(z) + \lambda_{i21}\,B_{i1}(z)}{\gamma_i} \\ &= \frac{\lambda_{i20}\,\mu_{i0}}{\gamma_i\,(1-z+\mu_{i0}z)} + \frac{\lambda_{i21}\,\mu_{i1}}{\gamma_i\,(1-z+\mu_{i1}z)} \end{aligned} \tag{5}$$

i.e., all class-k packets (which constitute λ_{i2k}/γ_i percent of the total flow from the user) entering the user input queue have their service times drawn from a geometric probability distribution with mean $(1-\mu_{ik})/\mu_{ik}$. Taking the first and second derivatives of Equation (5) and evaluating at $z=1$, we obtain the first two factorial moments

$$\begin{aligned} \mathbb{E}[B_i] &= \frac{\lambda_{i20}\,\mathbb{E}[B_{i0}] + \lambda_{i21}\,\mathbb{E}[B_{i1}]}{\gamma_i} \\ &= \frac{\lambda_{i20}\,(1-\mu_{i0})}{\gamma_i\,\mu_{i0}} + \frac{\lambda_{i21}\,(1-\mu_{i1})}{\gamma_i\,\mu_{i1}} \end{aligned} \tag{6}$$

and

$$\begin{aligned} \mathbb{E}[B_i(B_i-1)] &= \frac{\lambda_{i20}\,\mathbb{E}[B_{i0}(B_{i0}-1)] + \lambda_{i21}\,\mathbb{E}[B_{i1}(B_{i1}-1)]}{\gamma_i} \\ &= \frac{2\,\lambda_{i20}\,(1-\mu_{i0})^2}{\gamma_i\,\mu_{i0}^2} + \frac{2\,\lambda_{i21}\,(1-\mu_{i1})^2}{\gamma_i\,\mu_{i1}^2} \end{aligned} \tag{7}$$

We substitute Equations (6) and (7) into Equation (4) to obtain an expression for the mean number of packets in the user input queue operating under the fully queued access discipline:

$$\begin{aligned} \mathbb{E}[M_i] = {} & \frac{\lambda_{i20}\,(1-\mu_{i0})}{\mu_{i0}} + \frac{\gamma_i\,\lambda_{i20}\,(1-\mu_{i0})^2}{\mu_{i0}^2\,[1+\lambda_{i20}+\lambda_{i21}-\lambda_{i20}/\mu_{i0}-\lambda_{i21}/\mu_{i1}]} + \\ & \frac{\lambda_{i21}\,(1-\mu_{i1})}{\mu_{i1}} + \frac{\gamma_i\,\lambda_{i21}\,(1-\mu_{i1})^2}{\mu_{i1}^2\,[1+\lambda_{i20}+\lambda_{i21}-\lambda_{i20}/\mu_{i0}-\lambda_{i21}/\mu_{i1}]} \end{aligned} \tag{8}$$

Referring back to Figure 5, we see that the user input queue operating under the independently queued access discipline behaves as two distinct Geom/Geom/1 queues, the first with an input of intensity λ_{i20} and the second with λ_{i21}. Thus, for each of these queues we have geometrically distributed service times with z-transforms given by Equation (3). Again, taking the first and second derivatives of Equation (3) at $z = 1$, we obtain the following service-time factorial moments:

$$\mathbb{E}[B_{ik}] = \frac{1 - \mu_{ik}}{\mu_{ik}} \tag{9}$$

$$\mathbb{E}[B_{ik}(B_{ik} - 1)] = \frac{2\,(1 - \mu_{ik})^2}{\mu_{ik}^2} \tag{10}$$

Substituting Equations (9) and (10) back into Equation (4) for each k, we find that the combined mean number of packets in the user input queue operating under the independently queued access discipline is given by the following formula:

$$\mathbb{E}[M_i] = \frac{\lambda_{i20} - \lambda_{i20}\,\mu_{i0}}{\mu_{i0} - \lambda_{i20} + \lambda_{i20}\,\mu_{i0}} + \frac{\lambda_{i21} - \lambda_{i21}\,\mu_{i1}}{\mu_{i1} - \lambda_{i21} + \lambda_{i21}\,\mu_{i1}} \tag{11}$$

3.3 Derivation of the Internal-Link Flows

In [BFG89, Ban90] a queueing-network model was used to evaluate analytically the performance of the WON under the assumption of unlimited packet buffers at each station. Unfortunately, this approach does not extend easily to WONs that have limited packet buffers (and therefore must use deflection routing). The difficulty in calculating the mean end-to-end packet delay or throughput in WONs that use deflection routing derives from the fact that we can not easily compute the packet flow into each station, because the routing of a packet changes dynamically whenever deflections occur. This differs from models for infinite-buffer networks with static routing in which there exists a simple linear relationship between the traffic matrix and link flows.

We now present a model that can be used to approximate performance when deflection routing is used in the WON with single-buffer stations. The model is intended to estimate the internal-link flows λ_{ijk}, which we have defined to be the fraction of time slots in which there is a packet at input port j of station i being switched to output port k; restricting ourselves to two-transceiver stations, we see that $1 \le i \le N$, $0 \le j \le 2$, and $0 \le k \le 2$, where input and output ports number 2 are for the user. The quantity $\sum_j \lambda_{ijk}$ corresponds to the flow through output port k of station i, and therefore this sum must be less than 1, since each transmitter can transmit no more than a single packet per time slot.

The probability that a packet is found waiting at output port k is $\sum_j \lambda_{ijk}$ (where $0 \le j \le 2$). Define L_{ik} to be the number of packets in output queue k of station i (where $0 \le k \le 2$). Since each output queue has only one buffer, we can express the mean number of packets waiting a output queue k of station i as $\sum_j \lambda_{ijk}$, i.e.,

$$L_{ik} = \sum_{j=0}^{2} \lambda_{ijk} \tag{12}$$

Having derived expressions for the mean number of packets in the various queues of the network, we can now compute the total number of packets in the network. Defining the random variable M to be the number of packets in the WON and applying Equation (12), we find the mean number of packets in the WON by summing up the mean number of packets at each queue of each station:

$$\begin{aligned} \mathbb{E}[M] &= \sum_{i=1}^{N}\sum_{k=0}^{2} L_{ik} + \sum_{i=1}^{N} \mathbb{E}[M_i] \\ &= \sum_{i=1}^{N}\sum_{j=0}^{2}\sum_{k=0}^{2} \lambda_{ijk} + \sum_{i=1}^{N} \mathbb{E}[M_i] \end{aligned} \tag{13}$$

where the terms of the second summation are given by Equation (8) or (11), depending on the access method used at the user input queue. Assuming that the traffic load of $\gamma \triangleq \sum_{i,j}\gamma_{ij}$ packets per time slot can be offered to the WON without saturation, we apply Little's Result to derive the relationship

$$\mathbb{E}[T] = \frac{\mathbb{E}[M]}{\gamma} \tag{14}$$

Therefore, to compute the mean end-to-end packet delay in Equation (14), it is sufficient to calculate the value of $\mathbb{E}[M]$, which is completely determined by Equation (13) once all values of λ_{ijk} are known. If we further define $\lambda_{ijk:t}$ to be the fraction of slots during which internal link (i, j, k) is occupied by a t-packet, then it is obvious that $\lambda_{ijk} = \sum_t \lambda_{ijk:t}$. The remaining analysis will be aimed at determining the values of $\lambda_{ijk:t}$, given the virtual topology and offered traffic matrix (γ_{st}).

A simple formulation involves assuming that a packet attempts to queue at a station's output port independently of other events occurring at the station. We define $\phi_{ij:t}$ to be the component of ϕ_{ij} that is destined for station t; thus, $\phi_{ij} = \sum_t \phi_{ij:t}$. The deflection of a t-packet that is coming through input port j is dependent on the simultaneous occurrence of the following two events:

1. A t-packet arrives at input port j ready to be switched to output port k.

2. A packet arrives at input port $\bar{j}$ ready to be switched to output port k, where $\bar{j}$ represents the logical complement of j, i.e., $\bar{0} = 1$ and $\bar{1} = 0$.

The first event occurs with probability $\phi_{ij:t}$, and the second event occurs with probability

$$\sum_{(i,2,k)\in\Pi(i,s)} \phi_{i\bar{j}:s}$$

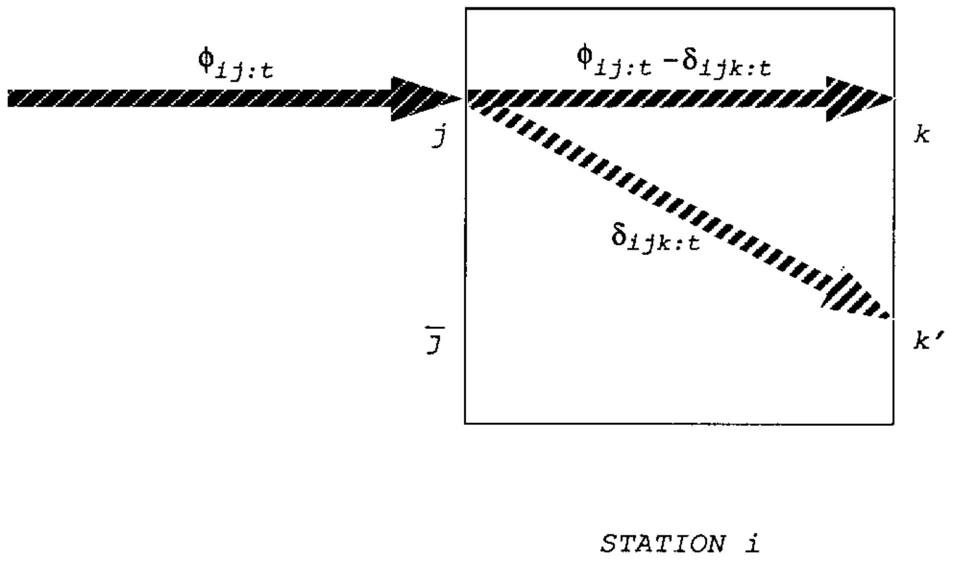


Figure 6: Deflection of t-Flow on Internal Links.

which is the sum total of all s-flows intended to use internal link $(i, \bar{j}, k)$. Given that these events are independent and that the t-packet will be randomly chosen for deflection, the probability of deflection is equal to one half the product of the probability of the two events described above. Now we can express the probability $\delta_{ijk:t}$ that a t-packet will be deflected away from internal link (i, j, k) because another packet simultaneously attempts to use output port k:

$$\delta_{ijk:t} = \sum_{(i,2,k) \in \Pi(i,s)} \frac{\phi_{ij:t}\,\phi_{i\bar{j}:s}}{2} \tag{15}$$

where $1 \leq i \leq N$, $0 \leq j \leq 1$, and $0 \leq k \leq 2$. We note that since internal link $(i, 2, k)$ can buffer the user's packets before they enter the network, they suffer no deflections (where would the packets be deflected to?).

Obviously, $\delta_{ijk:t}$ represents the fraction of t-packets that must be diverted away from internal link (i, j, k) and toward internal link (i, j, k'), where $(i, 2, k) \in \Pi(i, t)$ and $(i, 2, k') \in \Delta(i, t)$. This relationship between the link flow $\phi_{ij:t}$, the internal-link flow $\lambda_{ijk:t}$, and the deflected flow $\delta_{ijk:t}$, is graphically illustrated in Figure 6, and given by the following formulas:

$$\lambda_{ijk:t} = \phi_{ij:t} - \delta_{ijk:t} \tag{16}$$

and

$$\lambda_{ijk':t} = \delta_{ijk:t} \tag{17}$$

where $1 \leq i \leq N$, $0 \leq j \leq 1$, $1 \leq t \leq N$, $(i,2,k) \in \Pi(i,t)$, and $(i,2,k') \in \Delta(i,t)$.

It must also be true that the amount of traffic flowing into a station equals the amount flowing out. Thus, for $1 \leq i \leq N$, $1 \leq t \leq N$, $(i,2,0) \rightarrow (l,m,2)$, and $(i,2,1) \rightarrow (q,r,2)$, the conservation-of-flow conditions for the network are

$$\gamma_{it} + \phi_{i0:t} + \phi_{i1:t} = \phi_{lm:t} + \phi_{qr:t} \tag{18}$$

if $t \neq i$, and

$$\phi_{i0:t} + \phi_{i1:t} = \eta_i + \phi_{lm:t} + \phi_{qr:t} \tag{19}$$

if $t = i$.

Finally, we can use a variant of Equation (1) to express the relationship between link flows and internal-link flows:

$$\phi_{ij:t} = \sum_{k=0}^{2} \lambda_{ijk:t} \tag{20}$$

where $1 \leq i \leq N$ and $0 \leq j \leq 1$.

Together, Equations (16)–(20) form a collection of nonlinear equations, the solution of which is the values of $\lambda_{ijk:t}$. Although the variables $\phi_{ij:t}$ appear as unknowns in these equations, they are merely linear combinations of $\lambda_{ijk:t}$, as seen from Equation (20). Solving these equations for $\lambda_{ijk:t}$ permits us to evaluate $\mathbb{E}[T]$ using Equations (13) and (14). We display in Figure 7 the complete collection of equations that determine $\lambda_{ijk:t}$. To this collection we must add the conditions

$$0 \leq \sum_{j=0}^{2} \lambda_{ijk} \leq 1$$

for $1 \leq i \leq N$ and $0 \leq k \leq 2$; and

$$0 \leq \lambda_{i2k} \leq \mu_{ik} \leq 1$$

for $1 \leq i \leq N$ and $0 \leq k \leq 1$. The violation of either of these conditions signifies that the network is not capable of carrying the offered traffic load. Such a situation arises whenever the rate of traffic flow at an output queue exceeds one packet per time slot, or the arrival rate of packets to a user input queue exceeds the probability that the queue "sees" a free slot.

The internal-link–flow algorithm of Figure 8 permits us to compute the internal-link flows λ_{ijk}, given the network's virtual topology and offered traffic load. The algorithm begins by computing the routing tables for the given virtual topology, resulting in the creation of sets of primary and alternate routes, $\Pi(s,t)$ and $\Delta(s,t)$, for each source–destination pair in the network. We can execute step 1 using an all-pairs shortest-path algorithm such as the well-known Bellman–Ford,

- For $1 \leq i \leq N$, $0 \leq j \leq 1$, $0 \leq k \leq 2$, $1 \leq t \leq N$

$$\delta_{ijk:t} = \sum_{(i,2,k) \in \Pi(i,s)} \frac{\phi_{ij:t}\, \phi_{i\bar{j}:s}}{2}$$

- For $1 \leq i \leq N$, $0 \leq j \leq 1$, $1 \leq t \leq N$, $(i,2,k) \in \Pi(i,t)$

$$\lambda_{ijk:t} = \phi_{ij:t} - \delta_{ijk:t}$$

- For $1 \leq i \leq N$, $0 \leq j \leq 1$, $1 \leq t \leq N$, $(i,2,k) \in \Pi(i,t)$, $(i,2,k') \in \Delta(i,t)$

$$\lambda_{ijk':t} = \delta_{ijk:t}$$

- For $1 \leq i \leq N$, $(i,2,0) \rightarrow (l,m,2)$, $(i,2,0) \rightarrow (q,r,2)$, $1 \leq t \leq N$, $t \neq i$

$$\gamma_{it} + \phi_{i0:t} + \phi_{i1:t} = \phi_{lm:t} + \phi_{qr:t}$$

- For $1 \leq i \leq N$, $(i,2,0) \rightarrow (l,m,2)$, $(i,2,0) \rightarrow (q,r,2)$, $t = i$

$$\phi_{i0:t} + \phi_{i1:t} = \eta_i + \phi_{lm:t} + \phi_{qr:t}$$

- For $1 \leq i \leq N$, $0 \leq j \leq 1$, $1 \leq t \leq N$

$$\phi_{ij:t} = \sum_{k=0}^{2} \lambda_{ijk:t}$$

Figure 7: A Collection of Nonlinear Equations that Determine $\lambda_{ijk:t}$.

1. *Setup of Routing Tables.* For all s, t create primary and alternate routes, $\Pi(s,t)$ and $\Delta(s,t)$, respectively.

2. *Initialization of t-Flows.* Select a set of initial t-flows $\lambda_{ijk:t}$ based only on the primary routes, i.e., for all i, j, k, t assign

$$\lambda_{ijk:t} = \sum_{(i,j,k)\in\Pi(s,t)} \gamma_{st}$$

3. *Propagation of t-Flows.* For all i, j, t such that $(l,2,n) \rightarrow (i,j,2)$, assign

$$\phi_{ij:t} = \sum_{m=0}^{2} \lambda_{lmn:t}$$

4. *Calculation of Deflection Probabilities.* For all i, j, k, t assign

$$\delta_{ijk:t} = \sum_{(i,2,k)\in\Pi(i,s)} \frac{\phi_{ij:t}\,\phi_{i\bar{j}:s}}{2}$$

5. *Deflection of t-Flows.* For all i, j, t assign

$$\lambda_{ijk:t} = \phi_{ij:t} - \delta_{ijk:t} \quad \text{if } (i,2,k) \in \Pi(i,t)$$

$$\lambda_{ijk':t} = \delta_{ijk:t} \quad \text{if } (i,2,k') \in \Delta(i,t)$$

6. *Aggregation of t-Flows.* For all i, j, k assign

$$\lambda_{ijk} = \sum_{t} \lambda_{ijk:t}$$

7. *Calculation of Service Rate.* For all i, k assign

$$\mu_{ik} = 1 - \lambda_{i0k} - \lambda_{i1k}$$

8. *Calculation of End-to-End Delay.* Assign $\overline{T}_{\text{old}} = \overline{T}$ and

$$\overline{T} = \frac{1}{\gamma}\sum_{i=1}^{N}\left\{\left[\sum_{j=0}^{1}\sum_{k=0}^{2}\lambda_{ijk} + \sum_{k=0}^{1}\min(\lambda_{i2k},\,\mu_{ik})\right] + \overline{M_i}\right\}$$

9. *Convergence Test.* If $\left\|\overline{T} - \overline{T}_{\text{old}}\right\|$ is small enough then halt, else go to 3.

Figure 8: The Internal-Link–Flow Algorithm.

Floyd–Warshall, or Dijkstra algorithm. The initial t-flows are assigned in step 1 by routing all traffic from source station s to destination station t along the shortest path from s to t, without taking into account the deflections that would have occurred. The core of the internal-link–flow algorithm is steps 3–5, which route the t-flows $\lambda_{lmn:t}$ from station l's internal links to its outgoing links, resulting in the updating of the t-flows $\phi_{ij:t}$. Then the internal-link–t-flows $\lambda_{ijk:t}$ are updated according to the current deflection probabilities. The sequence of computations in steps 3–5 is as follows: update the $\phi_{ij:t}$ based on the $\lambda_{ijk:t}$, update the $\delta_{ijk:t}$ based on the $\phi_{ij:t}$, update the $\lambda_{ijk:t}$ based on the $\phi_{ij:t}$ and $\delta_{ijk:t}$. Steps 6–8 compute the mean end-to-end delay $\overline{T}$, and step 9 tests whether to reiterate or halt, based on the rate at which $\overline{T}$ is converging. In step 8 the calculation of $\overline{M_i}$ depends on the access method in use: we would invoke Equation (8) in the case of the fully queued access method and Equation (11) in the case of the independently queued access method. We apply Equation (14) to obtain the mean end-to-end packet delay for a given virtual topology and offered traffic load.

The use of the $\min(\cdot)$ function in step 8 is necessary because the resulting flow might be infeasible, i.e., the network might not be capable of handling the offered traffic load without one or more user input queues becoming unstable. A user input queue becomes unstable when the traffic rate from the user exceeds the rate at which the transport network can admit packets. In this case the mean end-to-end packet delay is unbounded.

The set of nonlinear equations in Figure 7 consists of $O(N^2)$ equations in $8N(N-1)$ unknowns. We compare this to the approach of [BC90b], which requires the solution of $O(N^3)$ equations in at least $O(N^2\sqrt{N})$ unknowns. Thus, we expect our approach to yield solutions more efficiently. Although we could solve the system of nonlinear equations displayed in Figure 7 using one of the well-known algorithms for solving sets of nonlinear equations, the internal-link–flow algorithm of Figure 8 is a more efficient method for solving this system. The algorithm does not have to concern itself with the conservation-of-flow condition in Equations (18) and (19), because we start in step 2 of the algorithm with a conservative flow and maintain conservative flows throughout each of steps 3–5. This saves computation steps and accelerates convergence in the internal-link–flow algorithm. In fact, each traversal of the main loop of the internal-link–flow algorithm uses $O(N^2)$ computation steps. The number of times that the main loop is traversed depends on several problem factors, including the tolerance used in the convergence test, the traffic load, and the virtual topology under evaluation.

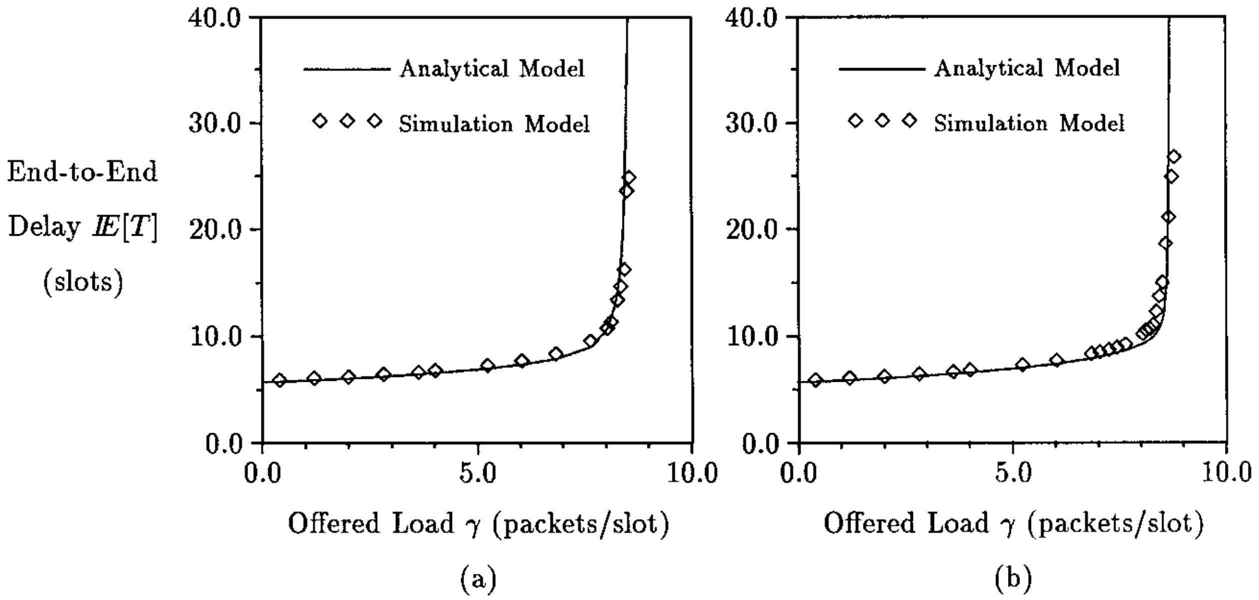


Figure 9: Comparison of Analytical and Simulation Models of a 64-Station WON with the ShuffleNet Virtual Topology under Uniform Traffic Loading. (a) Fully Queued Access Method. (b) Independently Queued Access Method.

3.4 An Assessment of the Internal-Link–Flow Algorithm

The internal-link–flow algorithm of Figure 8 allows us to determine the mean end-to-end packet time of a given WON in much less time than it would take to simulate the WON. Moreover, the accuracy of the algorithm is good, as we can see by comparing the analytically derived values of $\mathbb{E}[T]$ with those derived from simulating the WON. In the graph of Figure 9 we compare the performance predicted by the internal-link–flow algorithm with simulations of a 64-station WON that employs the ShuffleNet virtual topology defined in [Aca87]. The results are for a uniform traffic load, i.e.,

$$\gamma_{st} = \begin{cases} X & \text{if } s \neq t \\ 0 & \text{if } s = t \end{cases} \tag{21}$$

where the random variable X has a constant probability distribution, i.e., $\mathbb{P}[X = a] = 1$. Parts (a) and (b) of Figure 9 correspond to performance under the fully and independently queued access methods, respectively.

The second graph, shown in Figure 10, uses the 8×8 MSN virtual topology similar to the one shown in Figure 2. The curves of Figure 10 correspond to WON performance under nonuniform traffic load; in this case the traffic matrix is generated by choosing X in Equation (21) to be

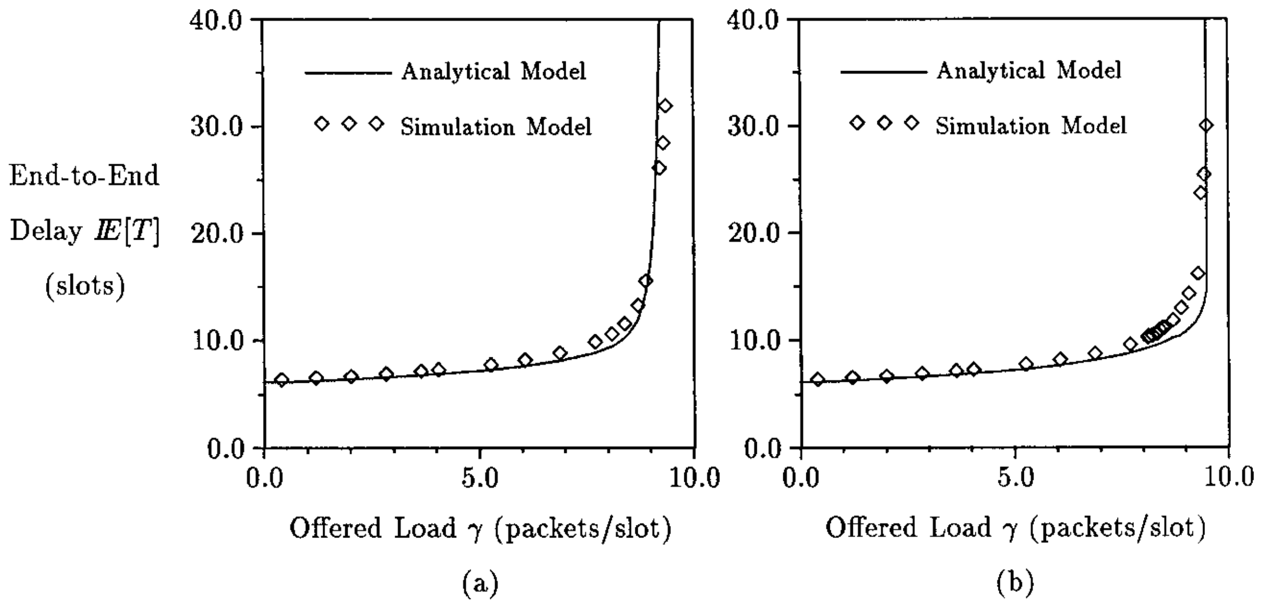


Figure 10: Comparison of Analytical and Simulation Models of a 64-Station WON with the MSN Virtual Topology under Nonuniform Traffic Loading. (a) Fully Queued Access Method. (b) Independently Queued Access Method.

uniformly distributed on the interval $(0, 2a)$, i.e. $\mathbb{P}[X \leq x] = x/2a$ for $0 < x < 2a$. Figure 10 also displays in parts (a) and (b) the performance to be expected when the fully and independently queued access methods, respectively, are used.

For both Figures 9 and 10 we note the slight performance advantage that the independently queued access method has over the fully queued access method. Although the low-load delays are essentially equal, the independently queued access method ultimately accommodates a heavier load than the fully queued access method before saturating. This advantage is, of course, intuitively obvious because the independently queued access method allows up to two packets to be simultaneously served from the user input queue during a time slot, whereas the fully queued access method can serve no more than one packet per time slot.

We also compare the analytical and simulation results for larger WONs. The graph of Figure 11 shows the predicted performance of a 196-station WON with the 14×14 MSN virtual topology. These results are also for nonuniform traffic, and the traffic matrix is generated by choosing the random variable X in Equation (21) to have the exponential probability distribution with mean a, i.e., $\mathbb{P}[X \leq x] = 1 - e^{-x/a}$.[3] The graph is only for networks using the fully queued access method.

[3]We actually truncate the tail of the exponential probability density function so that it does not generate a traffic

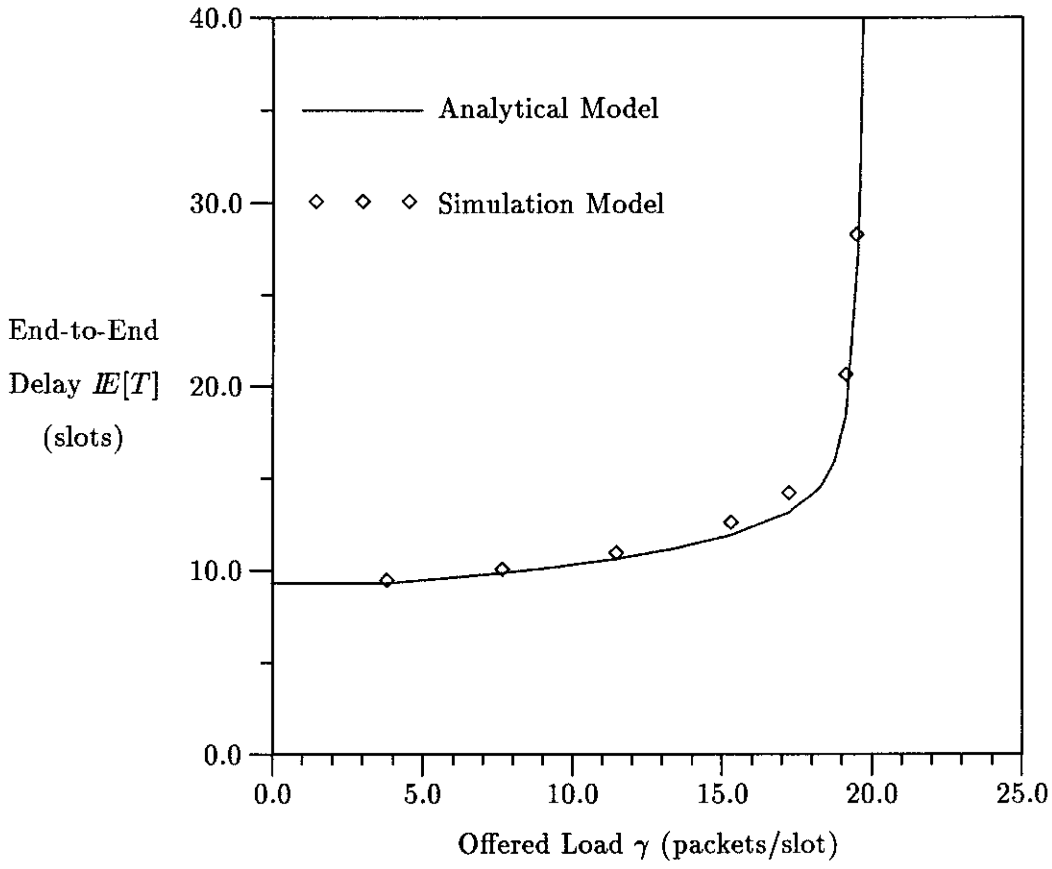


Figure 11: Comparison of Analytical and Simulation Models of a 196-Station WON with the MSN Virtual Topology Using the Fully Queued Access Method under Nonuniform Traffic Loading.

Table 1: Comparison of the Running Times of Simulation and the Internal-Link–Flow Algorithm.

Network Size and Type	Traffic Loading	Simulation Running Time (10000 Slots)	Algorithm Running Time (99-Percent Convergence)
64-Node	0.4 packets/s	327.6 s	3.6 s
Optimized	4.0 packets/s	329.0 s	5.1 s
Toplogy	10.1 packets/s	333.7 s	8.6 s
100-Node	1.0 packets/s	792.3 s	12.9 s
MSN	5.0 packets/s	793.1 s	14.4 s
Toplogy	10.0 packets/s	789.8 s	18.2 s
160-Node	2.5 packets/s	2869.3 s	48.0 s
ShuffleNet	7.6 packets/s	2870.5 s	59.7 s
Toplogy	10.2 packets/s	2863.7 s	69.9 s

As one can see, the agreement between the analytical and simulation models is quite close in all three cases considered. The analytical model tends to underestimate slightly the actual end-to-end packet delay. One explanation is that the independence assumptions required in the analysis tend to produce optimistic predictions. The first independence assumption is that the stream of free slots seen by the user input queue is a (discrete) memoryless process. This leads to our adoption of geometrically distributed service times at the user input queue. If, in fact, the free-slot arrival process is not memoryless, we could observe service times that have a higher degree of variation than the memoryless process, and this would cause the users' packets to have longer waiting times. A second source of inaccuracy might be the assumption that simultaneous contention for an output port is the outcome of two independent events. If the joint arrivals of contending packets are correlated, then the actual deflection probabilities could be higher than predicted. However, since the fidelity of the analysis appears to be extremely good, we should not be overly concerned about the inaccuracy of our independence assumptions.

Although the fidelity of the internal-link–flow algorithm is good, we should also examine the efficiency of the algorithm. We compare in Table 1 the running times of the internal-link–flow algorithm and a discrete-event simulation of the deflection-routing WON for several problem instances. The programs were run on a SPARC-based workstation with a 33-megahertz clock. The table shows data for three different WONs, i.e., a 64-station WON with a virtual topology optimized by the procedure to be described in the next section, a 10×10 MSN virtual topology, and a 160-station ShuffleNet virtual topology. For each type of network the table displays three different traffic load-

matrix in which a t-flow is greater than 1.

ings, and the traffic pattern is uniform in all cases. The algorithm was run until the estimated delay reached 99 percent of its actual value, and the simulation was run for 10000 time slots. It is clear from Table 1 that the algorithm executes much more quickly than the simulation—the algorithm runs from 41 to 91 times faster than the simulator.

Although the internal-link–flow algorithm is much faster than simulation of the WON, the sequential algorithm presented in Figure 8 could still benefit from speedup. The need for speeding up the algorithm is especially acute when we wish to evaluate the performance of a large number of WONs, as in the topological design problem to be presented in the next section. The algorithm can be parallelized in a straightforward way: steps 3–7, even though they must be evaluated in sequence, all consist of loops that can be individually parallelized. We parallelize each step by executing values of the loop index i concurrently, since there is no dependence between successive iterations of the loop. We must, however, provide a synchronizing barrier at the end of each of the steps 3–7, since a step uses the results of its preceding step. We note that we can also parallelize the computation in the initialization steps 1 and 2. The algorithm was parallelized in this way and run on the Sequent Symmetry shared-memory multiprocessor using the machine's parallel programming C-language library.

In general, the internal-link–flow algorithm of Figure 8 converges rapidly, but the rate of convergence is higher for lower traffic loads. Intuitively, the time that a deflection-routing network needs to reach a state of equilibrium after startup is dependent upon the likelihood of packet deflection. The more likely that packets are to be deflected, the longer an observer would have to wait to see the network flows stabilize to a steady state. We show in Figure 12 the convergence histories for a 64-station WON with the MSN virtual topology. The three curves of the figure are for three different traffic loads, but all three traffic matrices are uniform. The curves trace the estimated value of $\mathbb{E}[T]$ in the internal-link–flow algorithm, showing its value for each iteration of the loop in steps 3–8. The point at which the algorithm's estimate of $\mathbb{E}[T]$ reaches 99 percent of its actual value is also shown on each curve. Clearly, the number of iterations needed to reach the 99-percent point of convergence increases steadily as the total offered load is increased.

4 Topological Design

In this section we outline the virtual-topology design problem and discuss how to integrate the simulated-annealing and internal-link–flow algorithms to design optimum virtual topologies for the WON. We illustrate these techniques on two example 64-station WONs.

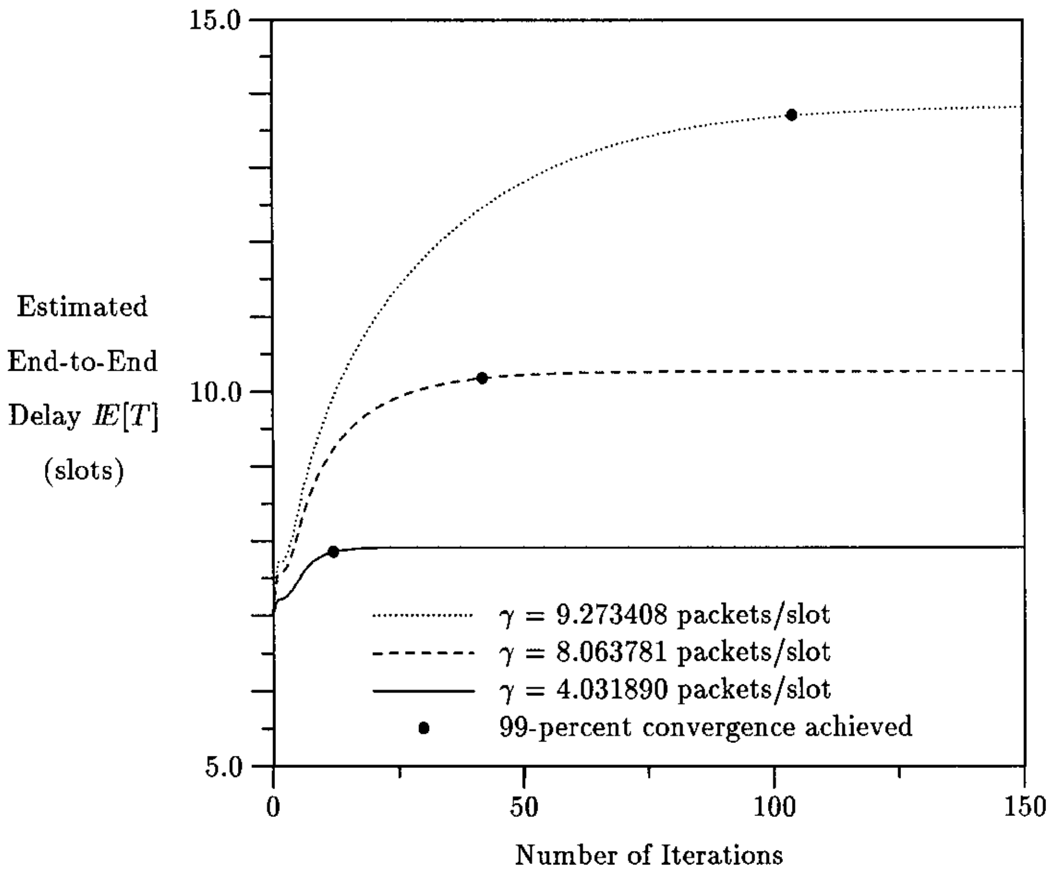


Figure 12: Convergence of the Internal-Link–Flow Algorithm for a 64-Station WON with the MSN Virtual Topology under Three Uniform Traffic Loadings.

4.1 The Virtual-Topology Design Problem

The virtual-topology design problem is to arrange the virtual topology of the WON in a such a way that the routing procedures—which incorporate both primary and alternate routes—can deliver packets in the least amount of time. The virtual-topology design problem requires us to choose the virtual topology and the set of primary and alternate routes that minimize the mean end-to-end packet delay $\mathbb{E}[T]$ for a given offered traffic load (γ_{ij}).

Given the use of deflection routing in the WON, the problem to be ultimately addressed in this paper is how to design a virtual topology that achieves superior performance. The toroidal virtual topology of the MSN is often advocated because no deflection incurs a penalty of more than four additional hops. Thus, alternate routes in the MSN have lengths comparable to primary routes. Intuitively, a virtual topology that does not permit short alternate routes for blocked packets will suffer from bad performance caused by excessive hopping. The key to designing a good virtual topology for deflection routing is to find a directed graph that combines small mean internode distance with short alternate routes.

A packet attempts to reach its source station via a primary route, which is the shortest path from the source station to the destination station. It is important to keep primary routes short, since deflections only add hops to the primary route. As the traffic load increases, so does the probability of deflecting a packet. The final route that a frequently deflected packet ultimately uses can differ radically from its intended primary route. Therefore, it is also crucial to keep alternate routes short.

To find the virtual topology that affords the best performance, we apply the simulated-annealing algorithm [KGV83], which is shown in Figure 13. We have previously obtained excellent results using the simulated-annealing algorithm to optimize the virtual topology of the unlimited-buffer WON (in which deflection routing is unnecessary) [BG90, BFG90c, BFG90b]. Obviously, the calculation of the cost function $\mathbb{E}[T]$ in step 3 of Figure 13 requires us to compute the primary and alternate routes for the given virtual topology, which involves the computation of shortest paths between all pairs of stations. We must also execute the internal-link–flow algorithm each time. Thus the evaluation of the cost function can be expensive.

In implementing the simulated-annealing algorithm, we represent the state of the system as a virtual topology. The virtual topology, which corresponds to a directed graph in which all nodes have two incoming and two outgoing arcs, is perturbed to generate a new virtual topology (step 3 of Figure 13) by swapping the targets of two randomly chosen arcs. This produces a new directed

1. *Initialization.* Select an initial temperature t and an initial virtual topology $\mathcal{X}$.
2. *Epoch Initiation.* Start a new epoch.
3. *Perturbation.* Randomly choose a virtual topology $\mathcal{Y}$ that is a nearest neighbor of $\mathcal{X}$. Assign $\Delta = \mathbb{E}[T_{\mathcal{X}}] - \mathbb{E}[T_{\mathcal{Y}}]$.
4. *Acceptance/Rejection.* Assign $\mathcal{X} = \mathcal{Y}$ with probability $\min(e^{-\Delta/t}, 1)$.
5. *Temperature Reduction.* If the epoch has reached steady state then assign $t = r\,t$, else go to 3.
6. *Convergence Test.* If no improvement has been observed for the last several epochs then halt, else go to 2.

Figure 13: The Simulated-Annealing Algorithm.

graph in which all nodes have exactly two incoming and two outgoing arcs, i.e., a legal virtual topology. The new directed graph can be viewed as a nearest neighbor of the original directed graph under the arc-swapping operation.

To ensure that we find a near-optimum solution, we perform a careful annealing, allowing each epoch to complete only when equilibrium is reached, and reducing the temperature by a small amount at the conclusion of the epoch. This is a computation-intensive algorithm, so we have chosen to speed up the algorithm by parallelizing the computation of the cost function $\mathbb{E}[T]$, as discussed previously in Section 3. Compared to the sequential version of the algorithm, the parallel algorithm has noticeably lower run time.

In evaluating the cost function $\mathbb{E}[T]$ the internal-link–flow algorithm must determine convergence by comparing whether the difference between two successive estimates is within a given tolerance. Therefore, the rate of converge depends upon the choice of the tolerance—the greater the tolerance, the faster the rate of convergence. When invoking the simulated-annealing algorithm, we can afford to relax the convergence test by using a tolerance that is somewhat larger than usual. Furthermore, the simulated-annealing algorithm only needs to rank the *relative* costs of the different virtual topologies, so the fidelity of the cost function is not critical. Thus, by specifying in the convergence test (step 9) of Figure 8 a tolerance that ensures rapid convergence, we can further trim valuable computation time from the simulated-annealing algorithm.

4.2 Examples: The Design of a 64-Station Network

This section describes experiments conducted to evaluate and compare the performance of deflection routing with various virtual topologies. In these experiments we compare the performance of WONs with optimized virtual topologies against that of the MSN and ShuffleNet.

The WON is optimized by simulated annealing using a traffic matrix that exerts a moderate traffic load on the network. The user input queue employs the independently queued access method. As its cost function the simulated-annealing algorithm uses Equations (11) and (14), which are evaluated by the internal-link–flow algorithm of Figure 8. The MSN's toroidal interconnection is used as the initial virtual topology in the optimizations.

We show in Figure 14 a comparison of mean end-to-end packet delay for the 64-station WON with the ShuffleNet, MSN, and optimized virtual topologies. The curves, which were produced using the internal-link–flow algorithm of Figure 8, depict performance under uniform traffic as the offered load is scaled up. Although the optimized virtual topology achieves lower delay than both the MSN and ShuffleNet, the improvement is small. The 30-percent improvement in maximum throughput, however, is significant, and this clearly justifies the use of virtual-topology optimization in the design of the WON.

Figure 15 also compares mean end-to-end packet delay for the 64-station WON with the ShuffleNet, MSN, and optimized virtual topologies. The curves, which were produced using the internal-link–flow algorithm of Figure 8, depict performance under nonuniform traffic as the offered load is scaled up. The traffic matrix was generated by choosing X in Equation (21) to have a Bernoulli distribution, i.e., $\mathbb{P}[X = 0] = 3/4$ and $\mathbb{P}[X = 1] = 1/4$, which corresponds to a scenario in which a given station exchanges traffic with about 1/4 of the other stations in the network. As in the uniform-traffic example, we observe the improved performance of the optimized virtual topology, compared to that of the MSN and ShuffleNet, and the improvement with nonuniform traffic is even more dramatic than in the uniform-traffic case. In addition to the modest reduction in delay, the optimized virtual topology provides at least 60 percent more throughput than either of the other two virtual topologies.

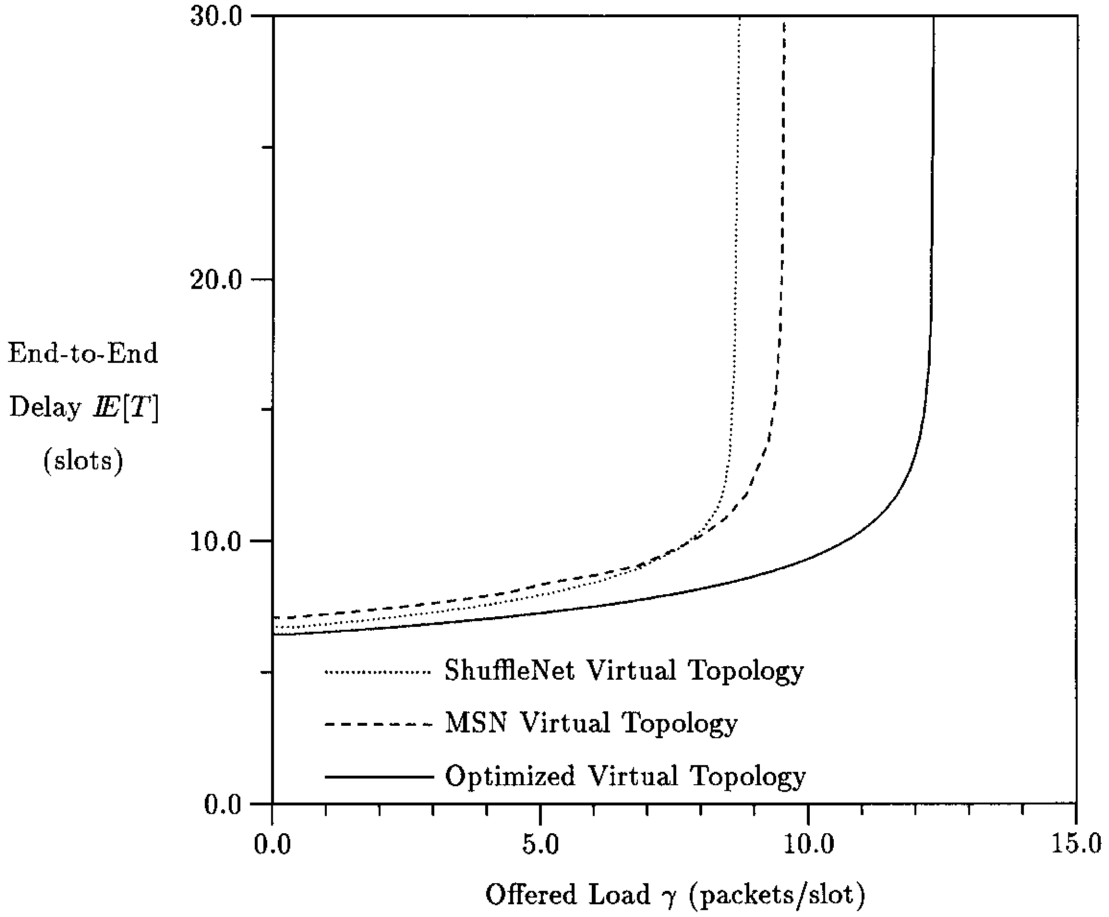


Figure 14: Comparison of Delays in 64-Station WONs with the MSN, ShuffleNet, and Optimized Virtual Topologies under Uniform Traffic.

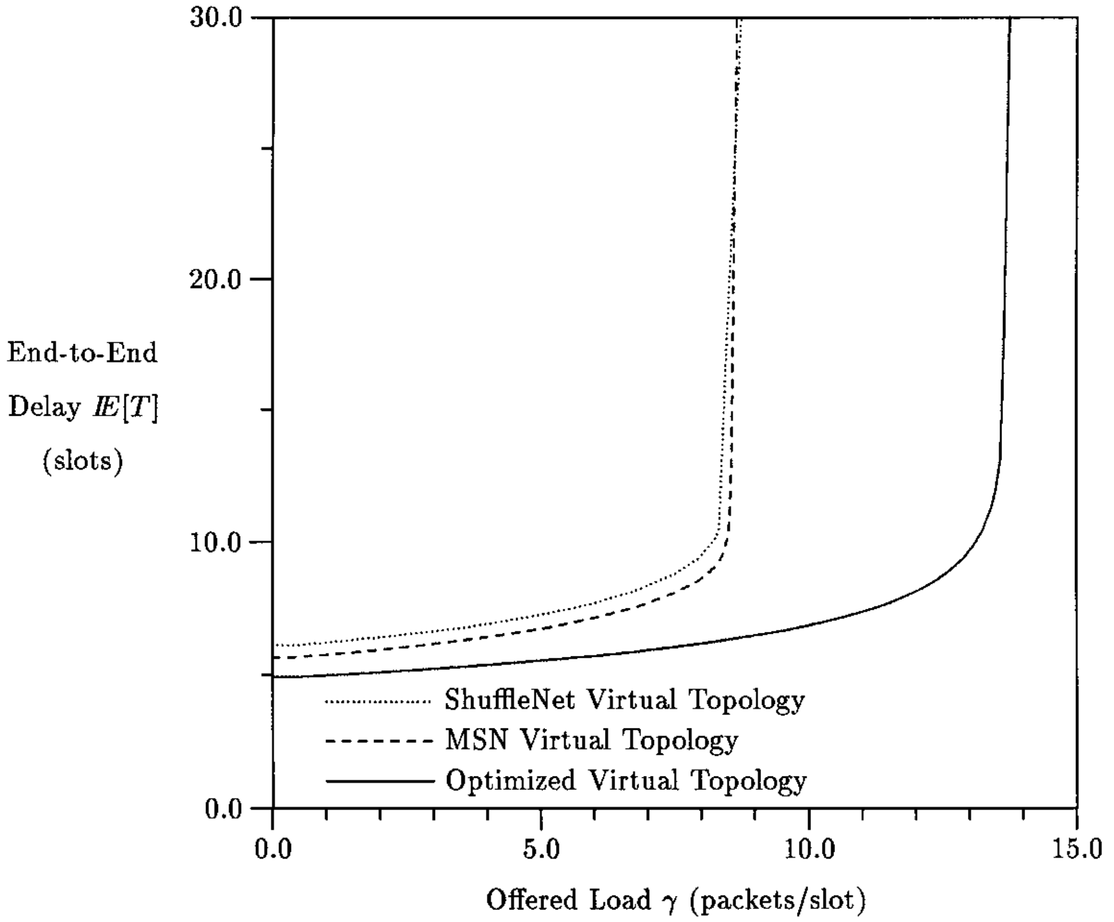


Figure 15: Comparison of Delays in the 64-Station WONs with the MSN, ShuffleNet, and Optimized Virtual Topologies under Nonuniform Traffic.

5 Conclusion

Deflection routing is made possible by the unique structure of virtual topologies that provide each station with exactly p dedicated input and p dedicated output links. It has several attractive features, including a straightforward implementation in hardware and a reduced requirement for packet buffers. Yet deflection routing has the obvious inefficiency of increasing the hop count of packets during periods of congestion. A good understanding of deflection routing is also hampered by the lack of flexible, general-purpose models for evaluating its performance.

Using a newly developed analytical model of link flows, we have proposed a method for designing the virtual topology of the WON so that performance is optimized when deflection routing is used. The simulated-annealing algorithm has provided good results by discovering virtual topologies that provide low delay and high maximum throughput for the specified traffic matrix. Compared to well-structured virtual topologies such as the MSN and ShuffleNet, the optimized virtual topologies excel in both delay and maximum throughput.

In the course of our research, many problems have been left unsolved. Our model has not incorporated the effects of propagation delay, which can be a significant component of overall delay. To model propagation delay between stations we can introduce additional infinite-server, fixed-delay queueing centers between connected output and input queues. Therefore, if $\tau_{ijk \to lmn}$ is the propagation delay (in time slots) on link $(i, j, k) \to (l, m, n)$, then we would add to Equation (13) the term $\sum_{l,m} \phi_{lm} \tau_{ijk \to lmn}$. Hence the mean end-to-end delay can be computed using Equation (14). Nor does our model consider postrouting network access. As noted earlier, we could enlist the results of [TB90] to handle this type of access. Finally, we have used a very simple model for resolving contention when two packets try simultaneously to access the same output port, i.e., the decision is made by a fair coin toss. However, more-sophisticated and better-performing schemes can be imagined, e.g., granting priority to packets from a given class of traffic, or to packets that are closest to their destinations, or to packets with the greatest age. Of these, our model could be adapted to deal with all but age-based priorities, since—unlike the formulation of [BC90b]—we maintain no information about the packet's age. For instance, to model a switching policy that gives priority to the packet with the closest destination, we could modify Equation (15) to reflect the fact that $\delta_{ijk:t}$ is influenced only by s-flows in which station s is closer to station i than is station t. Although superficially plausible, all of these extensions to the basic model would have to be validated by simulation.

Moreover, in light of the recent studies of Maxemchuk [Max90], we should keep in mind that switching and routing mechanisms that perform well in a statistical sense might still have problems with lockout, livelock, and congestion spreading. Although these problems occur only in pathological cases, they do remind us that the network must be designed in such a way as to operate fairly for all customers.

References

[Aca87] A. S. Acampora. A multichannel multihop local lightwave network. In *Proceedings of GLOBECOM '87*, pages 37.5.1–37.5.9, Tokyo, Japan, November 1987.

[AKH87] Anthony S. Acampora, Mark J. Karol, and Michael G. Hluchyj. Terabit lightwave networks: The multihop approach. *AT&T Technical Journal*, 66(6):21–34, November/December 1987.

[Aya89] Ender Ayanoğlu. Signal flow graphs for path enumeration and deflection routing analysis in multihop networks. In *Proceedings of GLOBECOM '89*, pages 1022–1029, Dallas, Texas, November 1989.

[Ban90] Joseph Anthony Bannister. *The Wavelength-Division Optical Network: Architectures, Topologies, and Protocols.* PhD thesis, Computer Science Department, University of California, Los Angeles, California, March 1990. Technical Report CSD-900007.

[BC87] Flaminio Borgonovo and Enrico Cadorin. HR^4-Net: A hierarchical random-routing reliable and reconfigurable network for metropolitan area. In *Proceedings of IEEE INFOCOM '87*, pages 320–326, March 1987.

[BC90a] Jack Brassil and Rene Cruz. An approximate analysis of packet transit delay for deflection routing in the Manhattan street network. *IEEE Transactions on Communications*, 1990. Submitted for publication.

[BC90b] Jack Brassil and Rene Cruz. Nonuniform traffic in the Manhattan street network. 1990. Submitted for publication.

[BFG89] Joseph A. Bannister, Luigi Fratta, and Mario Gerla. Designing metropolitan area networks for high-performance applications. In *Proceedings of the Seventh ITC Specialists' Seminar*, pages 3.5.1–3.5.8, Adelaide, Australia, September 1989.

[BFG90a] Joseph Bannister, Luigi Fratta, and Mario Gerla. Detour routing in high-speed multichannel networks. In M. Johnson, editor, *Proceedings of the Second IFIP WG6.1/WG6.4 International Workshop on Protocols for High-Speed Networks*, Palo Alto, California, November 1990. Elsevier Science Publishers.

[BFG90b] Joseph Bannister, Luigi Fratta, and Mario Gerla. Optimal topologies for the wavelength-division optical network. In *Proceedings of EFOC/LAN '90*, pages 53–57, Munich, Federal Republic of Germany, June 1990.

[BFG90c] Joseph A. Bannister, Luigi Fratta, and Mario Gerla. Topological design of the wavelength-division optical network. In *Proceedings of IEEE INFOCOM '90*, pages 1005–1013, San Francisco, California, June 1990.

[BG90] Joseph A. Bannister and Mario Gerla. Design of the wavelength-division optical network. In *Proceedings of ICC '90*, pages 962–967, Atlanta, Georgia, April 1990.

[Bib81] K. J. Biba. LocalNet: A digital communications network for broadband coaxial cable. In *Proceedings of COMPCON '81*, pages 59–63, Spring 1981.

[BT89] Allan M. Bignell and Terence D. Todd. SIGnet: A new ultra-high-speed lightwave network architecture. In *Proceedings of the IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, pages 40–43, June 1989.

[GG86] Albert G. Greenberg and Jonathan Goodman. Sharp approximate models of adaptive routing in mesh networks. In *Proceedings of the 1986 International Seminar on Teletraffic Analysis and Computer Performance Evaluation*, pages 255–270, Amsterdam, The Netherlands, June 1986.

[KGV83] S. Kirkpatrick, C. D. Gelatt, Jr., and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, May 1983.

[LA90] Jean-François P. Labourdette and Anthony S. Acampora. Wavelength agility in multihop lightwave networks. In *Proceedings of INFOCOM '90*, pages 1022–1029, San Francisco, California, June 1990.

[Max85] N. F. Maxemchuk. Regular mesh topologies in local and metropolitan area networks. *AT&T Technical Journal*, 64(7):1659–1685, September 1985.

[Max90] N. F. Maxemchuk. Problems arising from deflection routing: Live-lock, lockout, congestion and message reassembly. In *Proceedings of the NATO Advanced Research Workshop on Architecture and Performance Issues of High-Capacity Local and Metropolitan Area Networks*, Sophia Antipolis, France, June 1990.

[Mei58] Torben Meisling. Discrete-time queueing theory. *Operations Research*, 6(1):96–105, January–February 1958.

[TB90] Terence D. Todd and Allan M. Bignell. Performance modeling of the SIGnet MAN backbone. In *Proceedings of IEEE INFOCOM '90*, pages 192–199, June 1990.

[VW89] R. S. Vodhanel and R. E. Wagner. Multi-gigabit/sec coherent lightwave systems. In *Proceedings of ICC '89*, pages 14.4.1–14.4.6, Boston, Massachusetts, June 1989.

[ZA90] Zhensheng Zhang and Anthony S. Acampora. Analysis of multihop lightwave networks. Technical Report CU/CTR/TR-190-90-20, Center for Telecommunications Research, Columbia University, New York, New York, May 1990.