DISSERTATION

WDM RING NETWORKS BASED ON INTEGRATED GROUP-SEARCH WAVELENGTH ADD/DROP MULTIPLEXERS

Submitted by

Abdel-Rahman Mohamed Abdalla

Department of Electrical and Computer Engineering

In partial fulfillment of the requirements

For the Degree of Doctor of Philosophy

Colorado State University

Fort Collins, Colorado

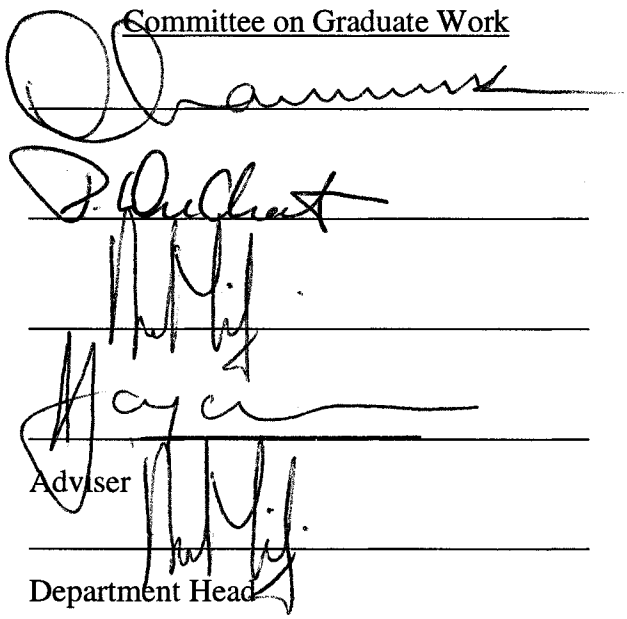
Summer 2006

COLORADO STATE UNIVERSITY

August 5, 1998

WE HEREBY RECOMMEND THAT THE DISSERTATION PREPARED UNDER OUR SUPERVISION BY ABDEL-RAHMAN MOHAMED ABDALLA ENTITLED: WDM RING NETWORKS BASED ON INTEGRATED GROUP-SEARCH WAVELENGTH ADD/DROP MULTIPLEXERS BE ACCEPTED AS FULFILLING IN PART REQUIREMENTS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY.

Committee on Graduate Work



Adviser

Department Head

ABSTRACT OF DISSERTATION

WDM RING NETWORKS BASED ON INTEGRATED

GROUP-SEARCH WAVELENGTH ADD/DROP MULTIPLEXERS

Wavelength division multiplexing (WDM) has emerged as the most promising technique for exploiting the terabit capacity optical networks. The acceptability of WDM networks depends heavily on the availability and cost of optical components. A wavelength add-drop multiplexer is one of the key network components, which is used for selectively dropping and inserting optical signals into the WDM network. Reconfigurable add-drop multiplexers offer great flexibility in the design of new network architectures.

Ring network architectures, utilizing WDM transmission, present an efficient means for implementing optical, high speed local and metropolitan area communication. Its implementation appears to be feasible with state of the art components.

In this dissertation, three new architectures of WDM ring networks are introduced and discussed in detail. These architectures are based on an especially designed add/drop multiplexer; namely the Integrated Group-Search Wavelength Add/Drop Multiplexer (IGSWADM). First, a WDM ring with a physical topology based on a double-fiber ring is presented. Each network station is connected to the ring via an IGSWADM device and equipped with two transmitters and two receivers. A medium access protocol for fast

circuit switched applications is proposed and the performance is evaluated. The second WDM ring architecture has the same physical topology, but has different node architecture. Each node is equipped with only one tunable transmitter and one fixed receiver. A medium access protocol with timeout mechanism is proposed for fast circuit switched applications and evaluated. The third WDM ring architecture considers the case where the number of channels is less than number of nodes. A medium access protocol with timeout mechanism is proposed for fast circuit switched applications. All the simulation models are built and executed using a software tool called UltraSAN. The results of the first architecture showed that the proposed ring has high performance at high arrival rates and long message lengths. For the second and third architectures, the results are affected heavily by the timeout duration. For a given set of network parameters, there exists an optimal timeout duration, which will maximize the network performance.

Abdel-Rahman Mohamed Abdalla
Electrical and Computer Engineering Department
Colorado State University
Fort Collins, CO 80523
Summer 2006

ACKNOWLEDGMENTS

I would like to express my great appreciation to my adviser Professor Anura Jayasumana, who helped and support me during the process of completing this research. I am indebted to professor Jayasumana for his encouragement, advice, nice personality and for all kinds of support he has given me.

I am grateful also to all my committee members without whom I could not complete my degree. All my committee members have been cooperative and kind. Also, I want to thank all the members of the Electrical and Computer Engineering Department. Special thanks to Elisabeth Wadman for her support and patience.

Thanks are also due to the Defense Advanced Research Projects Agency (DARPA) and the Microelectronics Technology Office (MTO), for supporting in part this WDM research.

Finally, thanks to my country Egypt that provided me with all kinds of support to get this degree. Also, I want to thank my wife and my kids who helped me in many ways and had been patient with me during the courses and the preparation of this dissertation.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

The 1960's invention of laser, the 1970's development of the low-loss optical fibers, the 1980's demonstration of long-lived semiconductor laser diodes, and the 1990's development of practical optical amplifiers have introduced a new era for optical fiber communication [40]. In view of the potential and recent advances in optical communication networks, optical networks are considered a promising candidate for the generation of networking to follow the emerging ATM generation. The motivations for this arise from the abilities of fiber to satisfy the growing demand for bandwidth per user. Optical networks are simple in structure and make heavy use of passive components [43].

Though impressive capacities have been achieved, today's systems utilize only a small fraction of the transmission capacity offered by the optical fiber. The main reason is that the opto-electronic components at the input and the output ends of the fiber are not capable of operating at speeds commensurate with the fiber bandwidth. It should be desirable to share the available fiber bandwidth among several communication nodes by allowing multiple accesses to the same resource. Therefore, efforts are now turned to a new generation of fiber systems that called multi-access optical-fiber networks. In these networks, the entire path between several end nodes is passive and optical without any conversions except at the ends of a connection.

The available technology makes it possible to fit a very large number of simultaneous connections within the large available bandwidth of a single fiber [12]. One can build a multiple-access network in which dynamically available connections between nodes use different optical wavelengths (Wavelength Division Multiaccess - WDMA), different timeslots (Time Division Multiaccess - TDMA), or by different signal waveforms (Code Division Multiaccess - CDMA). For TDMA and CDMA, one thing that must be taken into account is the fact that the speed of the electronic and photonic components must be much higher than the bit rate of one connection, where this is not true for WDMA [18]. The basic need of TDMA and CDMA to have nodes synchronized to within one time slot (for TDMA) and one chip time (for CDMA) make them relatively less attractive than WDMA. Also, WDMA employs the mostly existing technologies associated with the intensity modulation direct detection systems, and it is the current favorite since all of the end-user equipment need to operate only at the bit rate of a WDM channel, which can be chosen to be the peek electronic processing speed (a few Gb/s). That is, a single fiber can accommodate up to 10^4 electronic channels [17].

Theoretically, WDM technique is capable of exploiting the large bandwidth of the optical fiber. However, there are some problems facing WDM networks. One of the most important constrains in the multiaccess optical networks is the power budget limitation. The limitation of the power budget restricts the maximum number of users that can access the optical medium far below the theoretical capacity limit. Also, the selection of the network topology that efficiently deals with the power budget problem is very important [24], [25].

There are different topologies for building multiaccess optical-fiber networks: the ring, the bus, and the star. Recently, Ring networks using the wavelength division multiplexing technique have become an attractive solution for optical networks. Rings are expandable and use simple control software and interfaces. Ring topologies have proven to be efficient and survivable structures, they provide fault tolerance, they allow simple acknowledgment mechanism, and its implementation appears to be feasible with state of the art components [24], [28], [29].

The acceptability of WDM networks relies heavily on the available optical components. A wavelength add-drop multiplexer (WADM) is one of the key network components which is used for selectively dropping and inserting optical signals into the WDM network [56]. The ability to add and drop or pass through wavelengths is very useful for controlling the data flow on and off the network and achieving successful routing of wavelengths. Also, reconfigurable ADM's offer great flexibility in the design of new network architectures [58], [61].

In the literature, there are several WDM ring architectures that have been proposed and evaluated. A scalable multihop ring architecture which exploits the multihop principle is presented in [41]. A multi-channel ring system with time slotted access is discussed in [52]. An optical ring based on add/drop multiplexers and a central control node is presented in [24]. A WDM ring network employing a centralized multiwavelength light source is developed in [54].

This dissertation proposes three new architectures for WDM ring networks based on an Integrated Group-Search Wavelength Add/Drop Multiplexer (IGSWADM). The first WDM ring, named WDM Ring I, physical topology is based on a double-fiber ring.

Each network node is connected to the ring via an IGSWADM device and equipped with two transmitters and two receivers. A medium access protocol for fast circuit switched application is proposed and the performance is evaluated. The second WDM ring architecture, named WDM Ring II, has the same physical topology, but has different node architecture. Each node is equipped with only one tunable transmitter and one fixed receiver. A medium access protocol with timeout mechanism is proposed for fast circuit switched application and the performance is evaluated. The third WDM ring architecture, named WDM Ring III, considers the case where the number of channels is less than number of nodes. A medium access protocol with timeout mechanism is proposed for fast circuit switched application, and the performance is evaluated. All the simulation models are built and executed using a software tool called UltraSAN [9], [31], [32], [35].

The main goals of this dissertation are as follows:

- Design and develop different WDM ring architectures based on the Integrated Group-Search Wavelength Add/Drop Multiplexers.
- Design different access protocols that can efficiently control the function of the proposed WDM rings.
- Evaluate the performance of the proposed WDM ring architectures.

The dissertation is organized as follows. Chapter two introduces the WDM networks, its classifications, and reviews some of the existing WDM ring architectures. The third chapter presents the WDM Ring I architecture based on the IGSWADM device, and the medium access protocol that controls the function of this network for fast circuit switched applications. The performance of the network is evaluated. In chapter four, the design of the WDM ring network is modified to be simpler. The WDM Ring II is

presented, a medium access protocol is introduced, and the performance is evaluated. Chapter five presents WDM Ring III which consider number of channels less than the number of nodes. A medium access protocol for controlling the function of this network is introduced. In chapter six, the simulation models, using UltraSAN, for the designed WDM rings is presented. Chapter seven concludes the dissertation and highlights some of the future work.

CHAPTER 2

MULTIACCESS OPTICAL NETWORKS

2.1 Introduction

Optical fiber is a highly low-loss propagating medium. Losses can approach 0.15 dB/km. In comparison, conventional coaxial cables will lose half the electrical power in just a few hundred meters. In addition to the lower losses, packaging and volume benefits are also realized by using fiber [1], [49]. Light propagating in an optical fiber is subject to attenuation. The attenuation varies as a function of the wavelength. The common wavelengths used for optical communication fall between 0.83 and 1.55 microns. In the vicinity of 1550 nm there is a low-loss window where attenuation falls to approximately 0.15dB/km. A bandwidth of about 200 nm, extending from 1450 nm to 1650 nm is obtained. At 1550 nm, the 200 nm corresponding to over 25 THz of bandwidth. Taking advantage of even 1% (250 GHz) of this bandwidth in a single fiber is challenging. Most of today's systems utilize only a few tens of GHz of bandwidth. Figure 2.1 shows the low-loss transmission windows of optical fibers centered near 1.3 μm and 1.55 μm; the inset presents schematically the multichannel operation in the 1.55 μm window [3], [10].

There are three popular wavelengths used with optical fibers. The 850 nm is considered the first window of operation for optical fiber and corresponds to the wavelengths where GaAs devices operate. Even though loss and dispersion are both

relatively high in this region, it remains popular for cost sensitive applications. The 1300 nm region is the next window developed and many of today's installed systems operate in this region. The devices that operate in this region are commonly fabricated using the InGaAsP material. This region has the benefit of containing zero dispersion for normal silica based single mode fiber. The 1550 nm region is the third window and it is the lowest loss region. The principle drawback to operating in the 1550 nm region remains the high cost of the optoelectronic components [49].

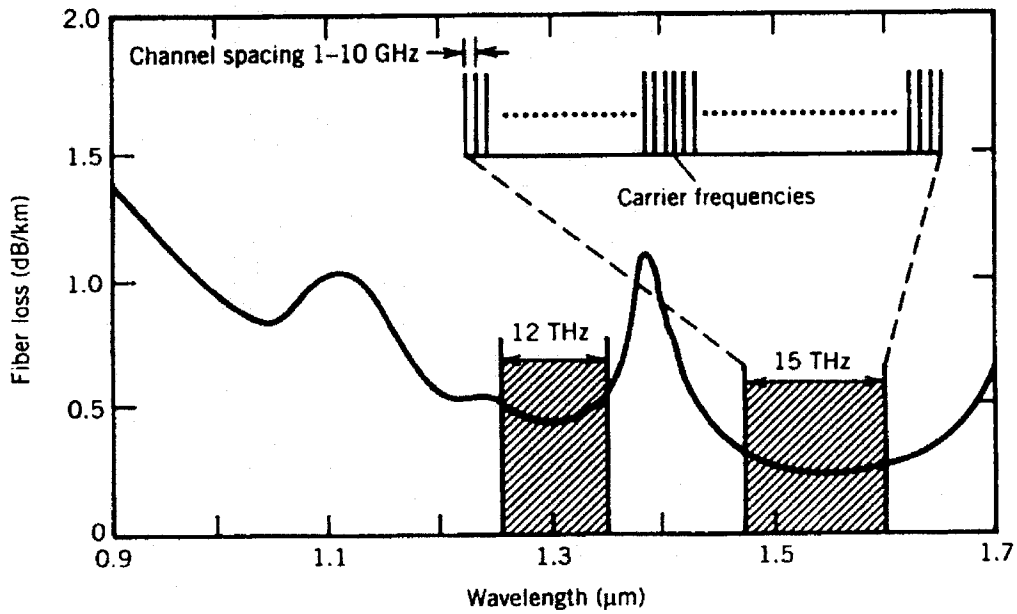


Figure 2.1: Low-loss transmission windows of silica fibers centered near 1.3 μm and 1.55 μm [10].

2.2 Optical Network Topologies

Multiaccess optical networks have three basic physical-network toplogies: ring, bus, and star. Each of these configurations is shown in Figure 2.2.

In a ring topology, the medium forms a closed loop or ring. As shown in Figure 2.2(a), all the nodes are connected to the ring. Data is transmitted from node to node in one direction. Token-passing ring is the most common ring network. The medium access is controlled by a token which is a unique type of message that is passed from node to node and gives a node the permission to transmit messages [63]. Ring topology uses the least amount of fiber. Dual counterrotating ring is another possible implementation, which doubles the fiber amount and increases throughput. In ring networks, to add a new node, the fiber must be cut and two new lines added. Dual rings can be self-healing, a failure in fiber can be detected and bypassed by creating a single ring that doesn't include the affected part [13], [23], [25].

In a bus topology, the medium consists of a single fiber to which nodes are attached. The ends of the bus are terminated by terminators that eliminate signal feedback [63]. There are two implementations for the bus. A double-bus is one of these implementations. In a double-bus network, shown in Figure 2.2(b), each node transmits to the nodes farther to the left on the lower bus and to the nodes farther to the right on the upper bus. Adding a new node, would require cutting two fibers and adding two new lines. A fault in any node or fiber would cause fragmentation. This fragmentation would allow sending in one direction, while sending in the other direction only among nodes on either side of the fault [23]. A double-bus network has the disadvantage that it requires twice as many transmitters and receivers per node as the star network [27].

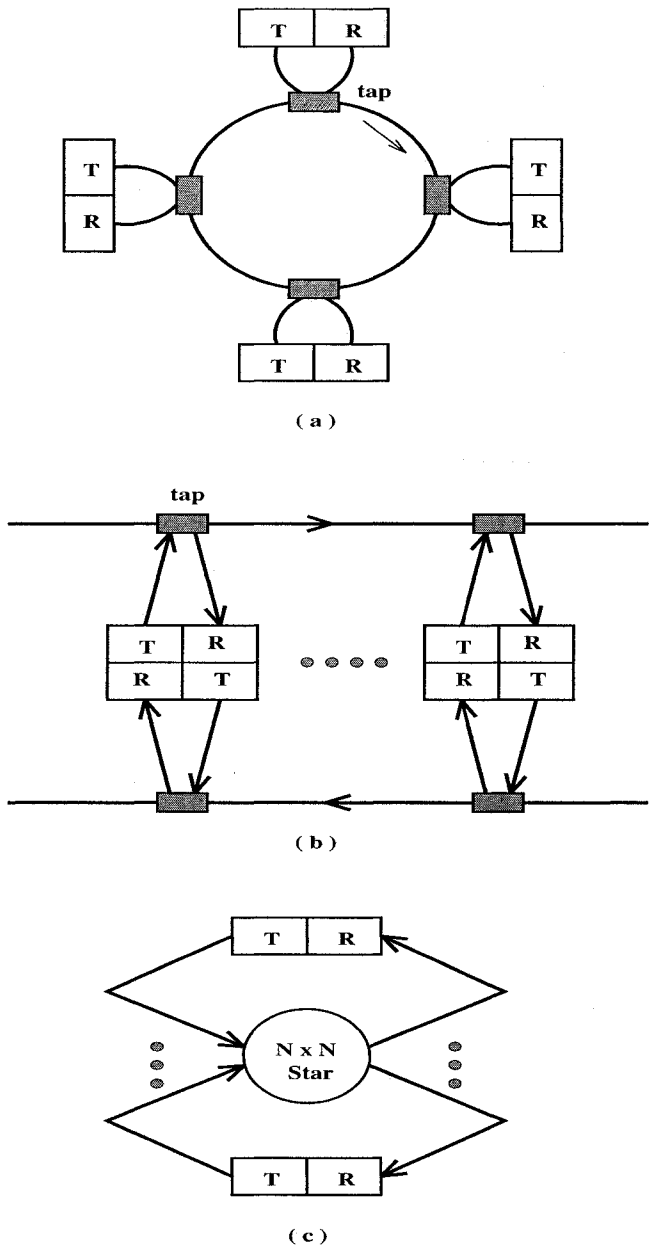


Figure 2.2: Examples of physical-network topologies of multiaccess optical networks: (a) ring , (b) bus, (c) star.

In a star topology, the nodes are directly connected to a hub. As shown in Figure 2.2(c), the outputs of all transmitters are combined by a passive star coupler and equally distributed to all receivers [20], [27]. The star topology uses the most fiber, but has the lowest average propagation delay between nodes [23]. Adding a new node would require that one line of two fibers to be run from the star coupler to the new node location. A fault in any node or fiber will only affect that node. Compared to a dual ring, the star requires 3.8 times as much as a dual ring for network consists of 100 nodes [23].

2.3 WDM networks

Various multiple access techniques have been proposed for exploiting the very large bandwidth of the optical fiber and to satisfy the increasing need for large bandwidth [37]. Among these, wavelength division multiplexing (WDM) is the most promising technique for implementing the terabit capacity optical networks. In WDM the available bandwidth of an optical fiber is divided into a large number of independent channels, each of which has a bandwidth proportionate with electronic processing speed [4]. A node may transmit on any channel by tuning its tunable transmitter(s) to one of the channels, or by selecting one of an array of lasers. A node may have a tunable receiver such that it tunes its receiver to receive from the appropriate channels, or it may have one or more fixed wavelength receivers. Multiple connections can be present at the same time, as long as they are operate on different wavelengths. The network capacity is restricted only by the number of the achievable wavelengths [6], [26]. So, parallel and concurrently operating WDM channels can be derived by having end users transmitting into and receiving from non-overlapping portions of the fiber's low-loss wavelength spectrum [16], [34]. Thus, WDM efficiently use the large optical fiber bandwidth by

simultaneous transmission of a large number of signals on different frequencies. The spacing between adjacent channels in WDM systems can be reduced if the end-user transmitters (lasers) are of high quality, which mean that they are stable and do not drift too far from their nominal operating wavelength range. The minimum channel spacing is limited by the crosstalk [16], [39], [46], [47]. Today, the term "WDM" is usually understood to mean dense WDM, that is, tens to hundreds (or more) of wavelengths, usually in the low-attenuation 1.55 μm window. To achieve a channel spacing of 1 nm or less, WDM requires narrow spectral-width lasers, typically distributed feedback (DFB) lasers or distributed bragg reflector (DBR) lasers, and optical filters to distinguish between the different wavelengths [22], [50], [57].

An alternative classification of WDM networks can be developed based on whether the nodal transceivers are tunable or not. One of the following structures is used:

Fixed Transmitter(s) and Fixed Receiver(s) (FT-FR)

Tunable Transmitter(s) and Fixed Receiver(s) (TT-FR)

Fixed Transmitter(s) and Tunable Receiver(s) (FT-TR)

Tunable Transmitter(s) and Tunable Receiver(s) (TT-TR)

FT-FR and TT-FR systems, because they use fixed receivers, may not require any coordination in control channel selection between the two communicating parties, while such coordination is usually required in the systems using FT-TR and TT-TR structure [16]. If each node is assigned a different channel under the FT-FR or FT-TR structures, then no channel collisions will occur and simple medium access control protocols can be used. The maximum number of nodes will be limited by the number of the available channels. Systems based on the TT-TR are probably the most flexible in accommodating

a scalable user population, but they also have to deal with the channel switching overhead of the transmitters and the receivers. In some systems a node may be equipped with multiple transmitters or receivers [16].

Optical WDM networks can also be classified according to the mode of transmission among network nodes. A network is said to be a single-hop or multi-hop network. In single-hop architectures, the nodes must communicate with one another in one hop [17]. The transmitted signal reaches its final destination without electronic conversion at any intermediate node [16]. This requires a significant amount of dynamic coordination between the stations since, for the packet transmission period at least, one of the transmitters of the sending stations and one of the receivers of the destination station must be tuned to the same wavelength [17]. In the single-hop environment, for the system to be efficient, the transmitters and receivers must be rapidly tunable, so that packets may be sent or received in quick succession. Given that, the tuning time for the transceivers is long in comparison to the packet transmission time, and the tunable range is small, so the challenge in this networks is the development of protocols that efficiently coordinate the data transmission [16], [17]. An example of a single-hop broadcast-and-select WDM network is shown in Figure 2.3. Each node transmits on a unique wavelength different from the others. All transmissions are broadcast to all nodes where an optical filter selects one of the wavelengths to receive [57].

Single-hop systems can be classified to pretransmission coordination systems and not-requiring any pretransmission coordination systems. Pretransmission coordination systems employ on shared control channel for the coordination of transmission times among stations [16], [30].

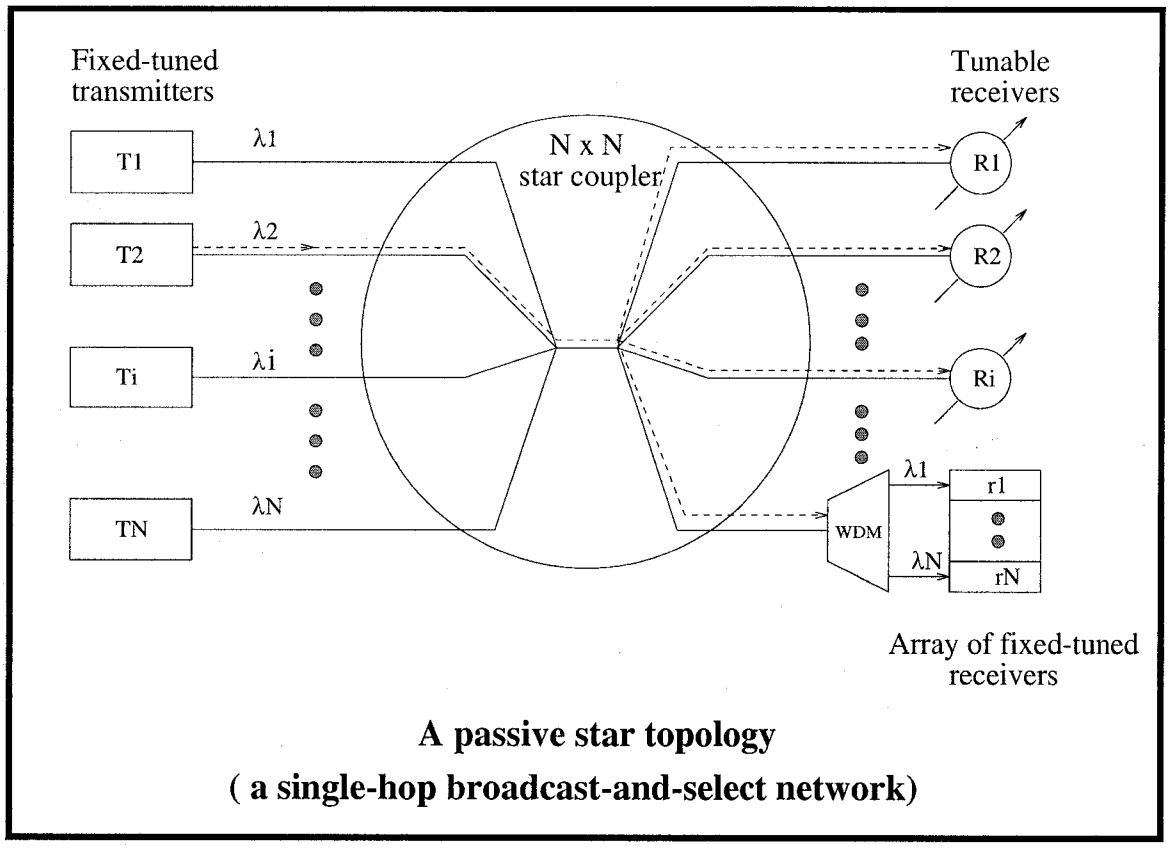
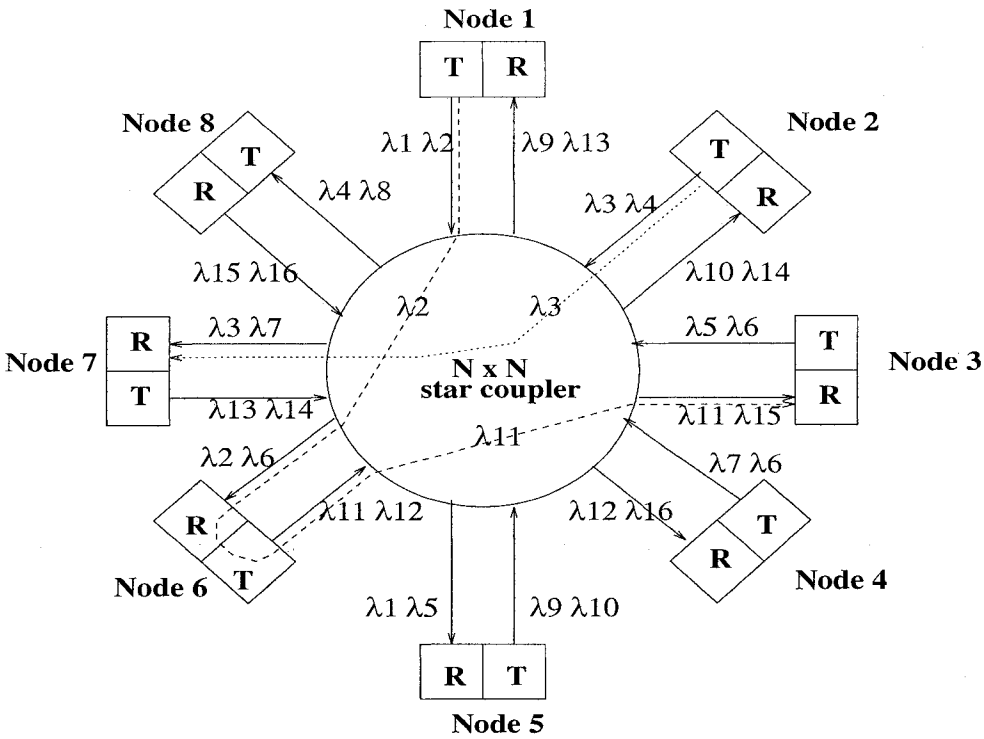


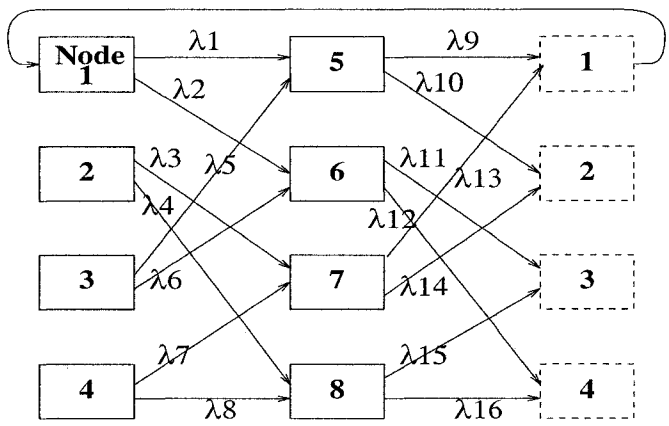
Figure 2.3: A passive star topology
(A single-hop broadcast-and-select network)

In systems which do not require pretransmission coordination, there is no control channel, and nodes are allowed to transmit or receive either, on the regular data channel, in a predetermined way or through using of contention based resolution schemes.

In multi-hop networks, in general, a signal from a source node may have to hop through zero or more intermediate nodes (conversion from optics to electronics and vice versa at intermediate nodes) before reaching its destination [17], [57]. An example of multi-hop architecture is shown in Figure 2.4. The multi-hop approach gets around the problem of needing rapidly tunable transmitters and receivers in single-hop networks by avoiding tuning altogether [57]. Each node is provided with a small number of fixed transmitters and a small number of fixed receivers. Each transmitter operates at a different wavelength. Although the network is physically a star or a bus, or any other broadcast topology, nodes can transmit signals only to those nodes which have a receiver tuned to one of their transmit wavelengths. Signals intended for other nodes will then have to be routed through intermediate nodes. At each intermediate node, the optical signal is converted to electronic form, the packet headers are decoded, and the node decides whether the received signal is intended to it or must be forwarded to another node. Thus, over the physical topology, there is a logical topology that determines the real connections among the nodes in the network. The concept is illustrated in Figure 2.4 for 8-node multihop WDM star network. Each node transmits at two fixed wavelengths and receives on two fixed wavelengths. If node 1 wishes to send a packet to node 3, the packet must go through one or more intermediate nodes. For example, node 1 can transmit on $\lambda 2$ for reception at node 6, which forwards the packet on $\lambda 11$ for reception by



(a)



(b)

Figure 2.4: (a) Example of an 8-node multihop WDM star network
(b) Directed graph for 8-nodes shufflenet multihop WDM network

the destination node 3. Thus there are two hops from node 1 to node 3. If node 1 used $\lambda 1$ instead of $\lambda 2$, there will be 4 hops from node 1 to node 3 [27].

The disadvantage of the multihop approach is that a significant portion of the network capacity is wasted because a packet goes through multiple hops before reaching its destination [57]. Also, the larger number of hops lead to longer network delays. An interesting feature of the multihop approach is that the logical topology can be changed simply by retuning some of the transmitters (or receivers) in the network [57]. The multihop architecture gives a high degree of scalability in which the number of nodes and the number of the available wavelengths are independent.

WDM networks can also be classified into two general architectural forms according to some networking techniques: the broadcast-and-select networks and the wavelength-routing networks [5], [36]. In turn, each of these can be either single-hop or multihop. A singlehop broadcast-and-select star network is illustrated in Figure 2.3 (other topologies like a bus, or a tree may be implemented as well) [27]. In broadcast-and-select architectures, each channel is transmitted by using a unique optical carrier frequency. The output of all transmitters is combined in a star coupler and then broadcast to all nodes equally. Each node can select one of the channels by using a tuning scheme [10]. Several functional possibilities exit, depending on whether the transmitters, the receivers, or both are made tunable [57]. Broadcast and select networks are suitable for LAN/MAN environments, but cannot be extended to wide area networks. Also, the network fabric is totally passive, consisting of optical couplers used as combiners and splitters, a use which enhances the network reliability. The broadcast and select supports multicast connections by having more than one receiver tuned to the same source at the same time. The

disadvantage of the broadcast and select approaches is that they require a large number of wavelengths (at least as many as there are nodes in the network), because there is no wavelength reuse in the network. As the number of channels increases, the requirements on the lasers and filters stabilities become stringent. The splitting loss is another problem with these approaches because, the power transmitted from a node is broadcast to all nodes, so each node gets only a small fraction of the transmitted power [19], [57], [59].

The second major architectural type is the wavelength routing networks. An example is shown in Figure 2.5. The wavelength routing WDM network is composed of wavelength selective (routing) elements. A connection is uniquely determined by the signal wavelength and the node from which the signal is entered into the network [27]. As shown in Figure 2.5 each node has a tunable transmitter and a tunable receiver. By tuning the transmitter to a desired wavelength, the injected signal is passively routed to the addressed receiver. So, for a complete N x N interconnection we need only N distinct wavelengths, and 2N fibers. Each receiver can be addressed by any one of the transmitters in a noninterfering way [27], [5]. This approach gives considerable wavelength reuse, reducing the number of the required wavelengths too, this approach avoid the splitting losses in the broadcast and select approach. Wavelength routing networks make it possible to dynamically change the internal routing structure according to the pattern of the traffic in the network. The principle drawback involves the fact that each node must have tunable transmitters and tunable receivers in order to achieve multiaccess. The advantages of wavelength routing networks are obtained at considerable complexity in the network routing control and hardware cost. The broadcast-and-select is suitable for LANs and MANs while the wavelength routing is suitable for WANs [57].

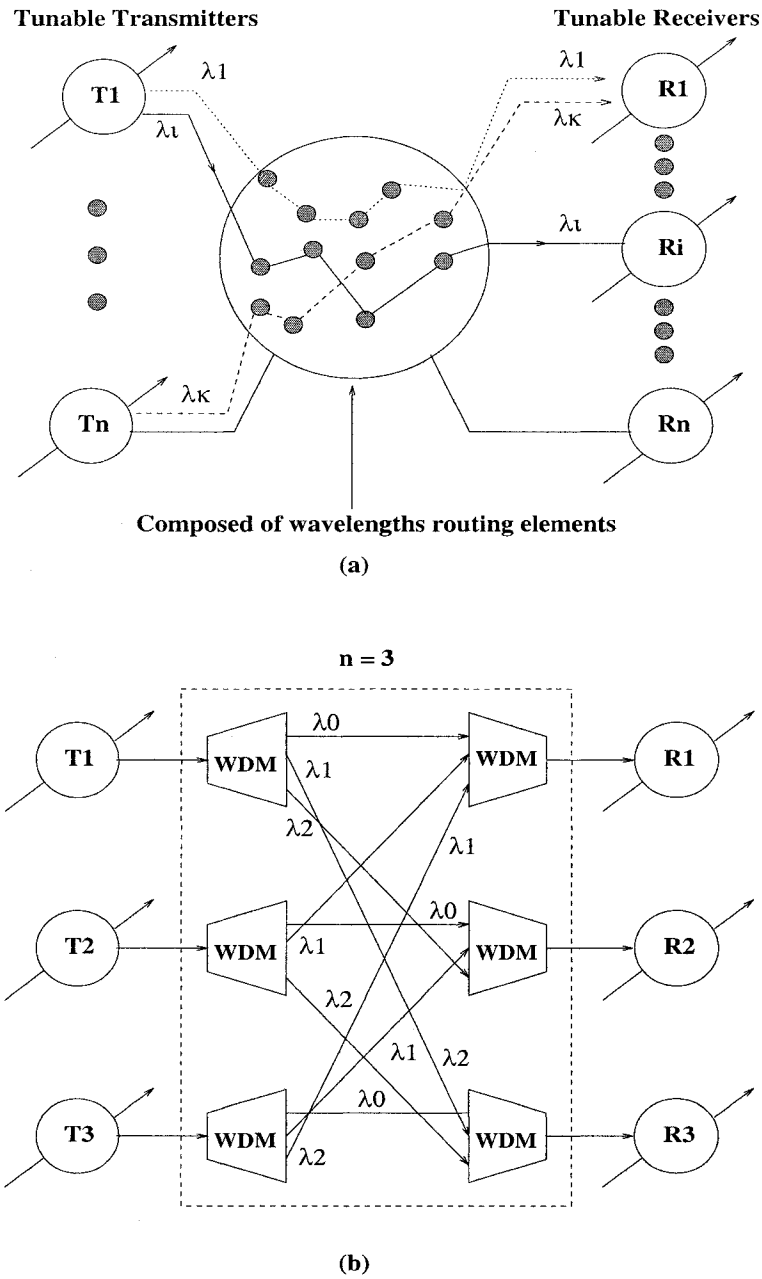


Figure 2.5: (a) Principle of wavelength routing single-hop WDMA networks (b) Example of implementation for n=3 nodes using WDM passive devices

2.4 Add / Drop Multiplexers

The availability and cost of optical components play a great role in the success of WDM networks. To make optical networking cost effective, it is important to determine a minimal set of network building blocks which would provide the functionality and flexibility need in these networks. This functionality includes multiwavelength sources and destinations, multiwavelength routing and cross-connection functions, and wavelength adding and dropping [44]. The network building blocks which implement this functionality are called network elements. A wavelength add-drop multiplexer (WADM) is one of the key network elements which is used for selectively dropping and inserting optical signals into the WDM network [56], [51]. The ability to add and drop or pass through wavelengths is very useful for controlling the data flow on and off the network and achieving successful routing of wavelengths. Also, reconfigurable ADM's offer great flexibility in the design of new network architectures.

An example for the general structure of a reconfigurable ADM is illustrated in Figure 2.6. The WADM consists of a demultiplexer followed by a group of 2 x 2 optical switches, one switch per wavelength, and a multiplexer [58], [45]. As shown in the figure, The incoming WDM signals at the input and add ports are both demultiplexed. The demultiplexed signals are then switched separately. If all of the switches are in the "bar" state, all the wavelengths go through the WADM without any disturbance. If one of them is in the "cross" state, the signal on the corresponding wavelength is "dropped" and a new signal can be "added" on the same wavelength. One or more wavelength channels can be dropped and added from and to an optical fiber [58]. Reconfigurable add-drop multiplexers are essential components in the WDM ring networks. They are required to

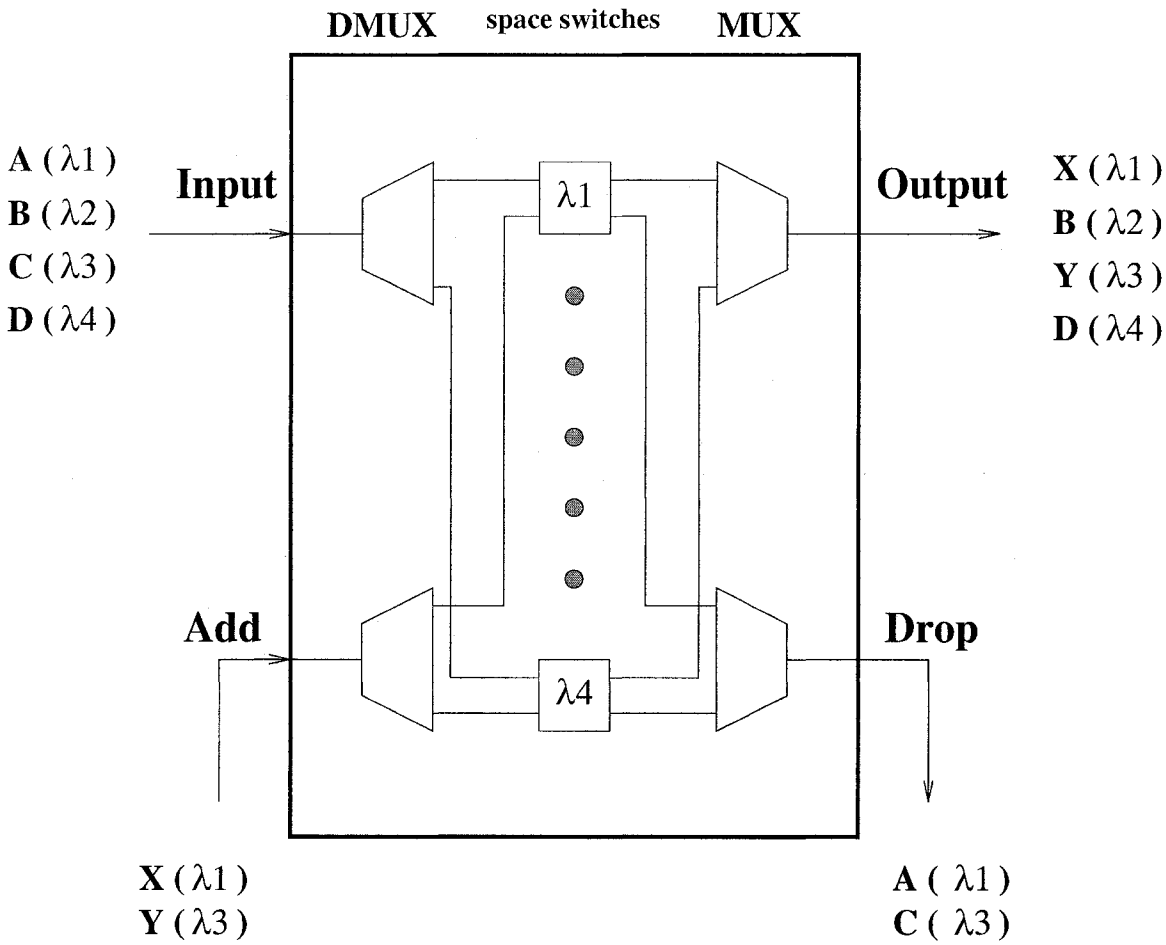


Figure 2.6: General structure of a reconfigurable ADM

flexibly configure and reconfigure optical paths through which central nodes are logically interconnected [53], [55]. In the WDM rings, each node is connected to the ring via an ADM. From figure 2.6, the INPUT and OUTPUT ports of the ADM are connected to the ring network, and the ADD and DROP ports are connected to the node.

Another reconfigurable ADM is explained in [61] that can be use for bidirectional ring networks. The architecture of this bidirectional WADM is illustrated in Figure 2.7. The signal can be added/dropped in either directions. Only a single fiber is needed for bidirectional transmission. For a unidirectional ring, a signal should go through all nodes to reach the upstream destination node. By using this WADM in a bidirectional ring, the network throughput can be increased because the signal will go through a smaller number of hops to reach its destination node. As shown in Figure 2.7(a), the WADM consists of two 8-wavelength WDM mux/demux pairs. In the WADM each channel has an Erbium-Doped Fiber Amplifier (EDFA) to boost the power before the WDM mux/demux. There are four states for the WADM. The signal can be added to the left or right hand side of the network. It can be dropped from the left or right hand side of the network. It can be passed through the WADM in each direction [61]. Figure 2.7(b) shows the structure of the switch fabric at each wavelength port. The switch consists of two optical cross switches. The first cross switch used to determine the direction of the signal, while the second switch determines whether the signal should be added/dropped or passed directly through the WADM.

2.5 WDM Rings

A severe limitation of single channel ring networks is that the bandwidth of the available single channel must be shared by all active stations. The network capacity may

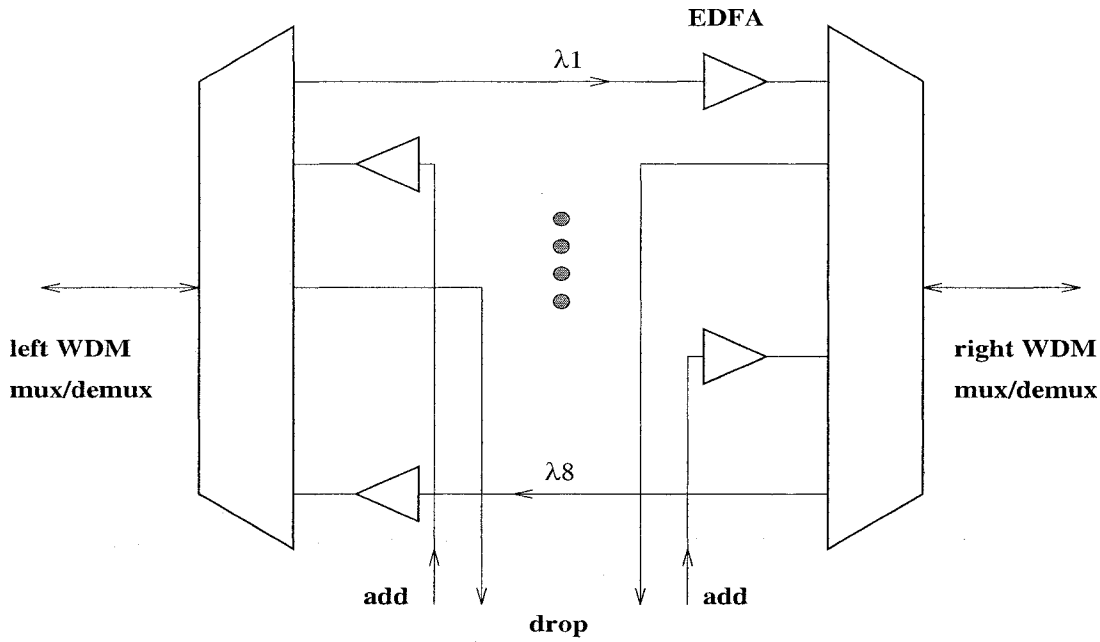


Figure 2.7: (a) Structure of bidirectional 8-wavelength WADM
(b) Structure of switch fabric at each wavelength port

be scarce for applications in which multiple nodes require access to the single channel and each one requiring very high bandwidth [52], [38]. Large bandwidth can be obtained by using multi-channel ring networks. The total available bandwidth will be a multiple of the bandwidth of each channel. Recent progress in optical networks, particularly in the area of wavelength division multiplexing serves as a strong basis for providing growth capability in the existing ring networks.

Through the concept of WDM, the lightwave technology can provide a good approach for multiple channels realization over ring networks. Ring network architectures, utilizing WDM transmission, present an efficient means for implementing optical, high speed local and metropolitan area communication. Also, its implementation appears to be feasible with state of the art components. Ring topologies have been proven to be efficient and survivable structures. They provide fault tolerance, by using an additional counter rotating ring for use under faulty conditions to reconfigure the logical ring [48], [60]. Also, they allow simple acknowledgment mechanism, by circulating the signal in the ring back to the requesting node.

In the literature, various WDM ring architectures have been considered. In [15] they have studied a WDM ring networks based on a single or multifiber ring using a key component called polarization-independent acoustically tunable optical filter (PIATOF). This device is working as an add/drop multiplexer for inserting the power of each node to the ring network. They used multifiber ring configuration to increase the number of users and to decrease the number of operating wavelengths in the network.

A WDM ring based on ADM employing a fiber Fabry-Perot filters and optical circulators was proposed by Oda [24]. The configuration of the proposed ring network is

shown in Figure 2.8. The proposed ring is unidirectional and constructed using a central node and N remote nodes and optical amplifiers. As shown in Figure 2.9 the central node consists of optical transmitters (OT's), optical receivers (OR's), a multiplexer (MUX), a demultiplexer (DMUX) and an electrical switch. In this ring, each remote node is connected to the central node using a different optical frequency. The central node transmits the multiplexed signals. At each remote node, one of the multiplexed frequencies is dropped through the ADM and another optical signal with the same frequency is added to the other unfiltered signals to transmit the signal back to the central node. The remote nodes are connected to each other by sending their signals which conveys the address information to the central node. At the central node, according to the address information and by using the electrical switch, the receiver and the sender of the considered nodes are connected. Then, the central node sends the signal on the destination frequency to be received by the destination node.

A WDM ring architecture based on the multihop principle is presented in [41]. Onto a single fiber ring a multiloop ring is embedded as an optical connectivity. A ring uses three wavelengths is shown in Figure 2.10. In each node an optical interface allows two channels to flow into the node and bypass the third signal. By exploiting the multihop principle, each node needs only two transceivers instead of three.

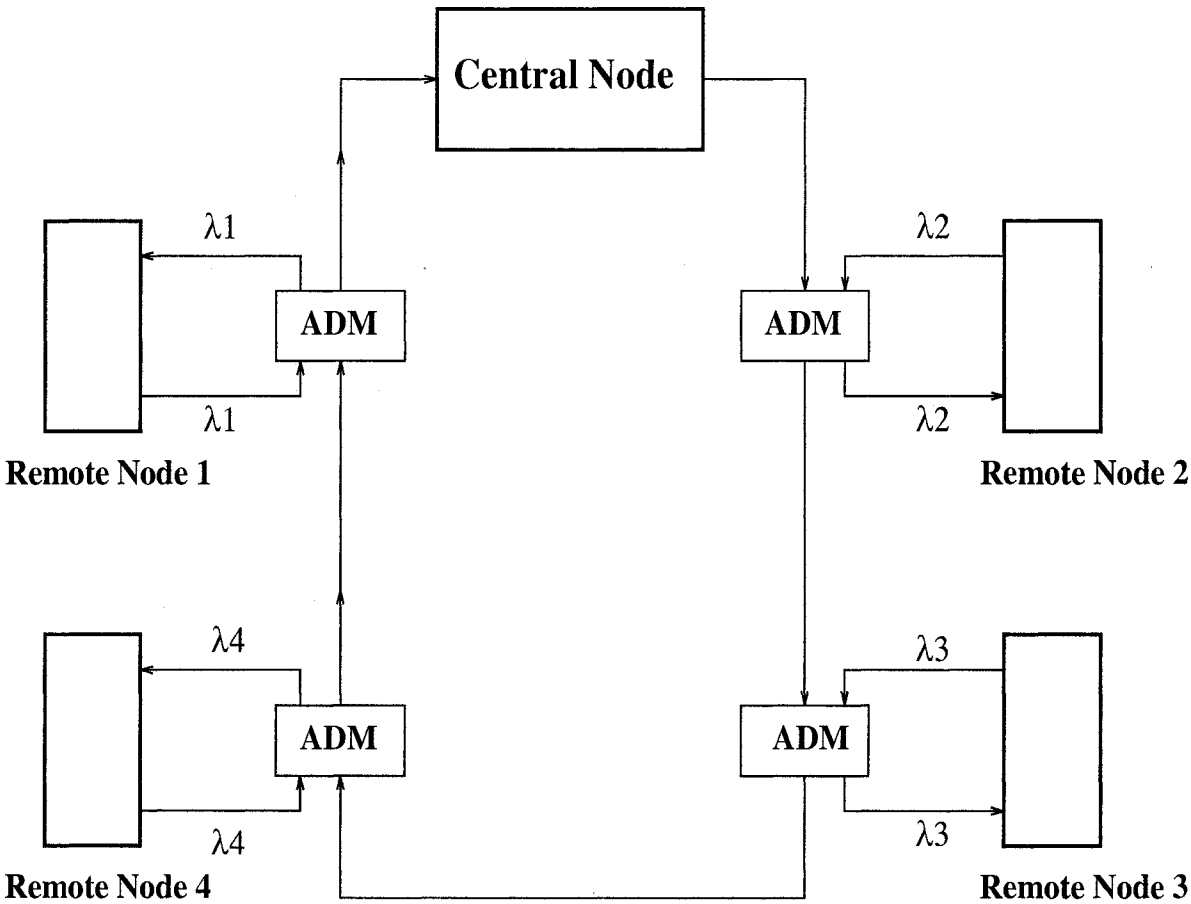


Figure 2.8: WDM Ring Network using central node

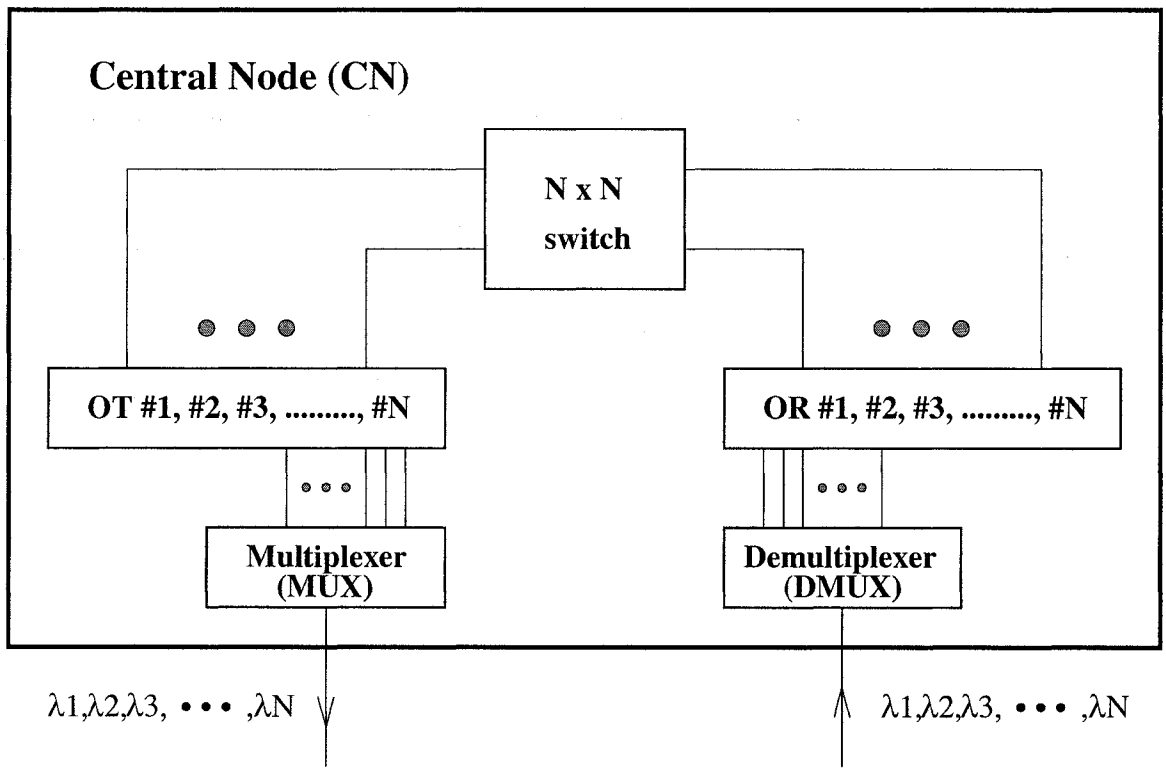


Figure 2.9: Structure of Central Node

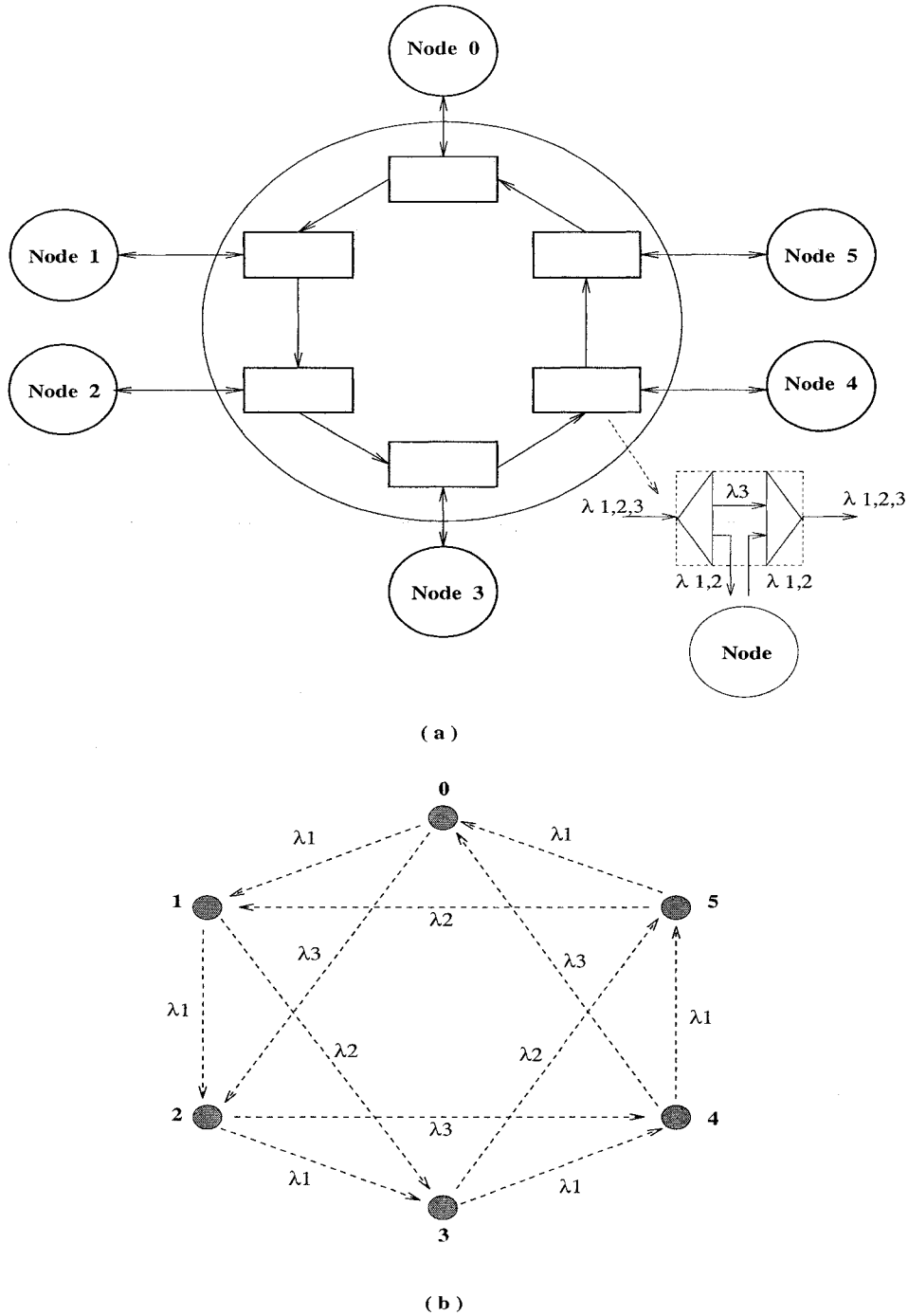


Figure 2.10: Multihop ring network (a) Fiber ring network (b) Multi-loop optical ring

CHAPTER 3

WDM RING I : A WDM RING NETWORK BASED ON INTEGRATED GROUP-SEARCH WAVELENGTH ADD/DROP MULTIPLEXER (IGSWADM)

3.1 Introduction

The success of wavelength division multiplexing optical networks depends on the availability and the cost of the optical components. Reconfigurable wavelength add/drop multiplexers (WADM) that can arbitrarily insert or delete any wavelength channel or combination of wavelength channels play a great role in the design of the new large capacity WDM ring networks. They offer added flexibility because they can respond to the network traffic changes. Ring network architectures utilizing WDM technique and WADMs are promising candidates for implementing the next generation optical networks. They present an attractive solution for optical high speed metropolitan and local area networking.

An optical multi-channel ring network architecture based on the wavelength division multiplexing technique and the add/drop multiplexers is introduced and discussed in this chapter. The key element in the proposed ring architecture is the Integrated Group-Search Wavelength Add/Drop Multiplexer (IGSWADM) which used mainly for inserting the transmission of each node after sensing the transmission

wavelengths of the other nodes. A detailed description for the IGSWADM is provided. Also, a protocol for fast circuit switched applications is proposed for use with the proposed architecture and its performance evaluated.

3.2 Network Architecture

The proposed WDM ring network architecture is shown in Figure 3.1. It consists of N nodes interconnected by two fibers forming a ring. The transmission on one fiber is in a direction opposite to that on the other fiber. Each node is connected to the ring via an integrated group-search wavelength add/drop multiplexer (IGSWADM). Each node is equipped with one tunable transmitter, one fixed transmitter, for transmission to other nodes, one fixed receiver for receiving messages, and one tunable receiver for receiving acknowledgments. The number of wavelengths (W) is equal to the number of nodes (N). Each node can transmit using any free wavelength and receive only one fixed wavelength assigned to it in addition to another wavelength determined according to the desired destination wavelength λ_d, where λ_d may be $\lambda_1, \lambda_2, \ldots, \lambda_N$. Each node is allowed to drop its own wavelength from one fiber but has to add the same wavelength when desired on the other fiber.

3.3 Integrated Group Search Wavelength Add / Drop Multiplexer (IGSWADM)

The key element in the proposed architecture is the Integrated Group-Search Wavelength Add/Drop Multiplexer (IGSWADM). It is implemented using non-linear polymer technology. Figure 3.2 shows its four wavelengths schematic diagram. For N wavelengths, it consists of two (1 x N) wavelength routers, two (N x 1) combiners, two 99:1 splitters, and a reconfigurable Add/Drop unit. Function of the splitters is to extract 1% of signal power, allowing nodes to monitor the presence of a particular wavelength.

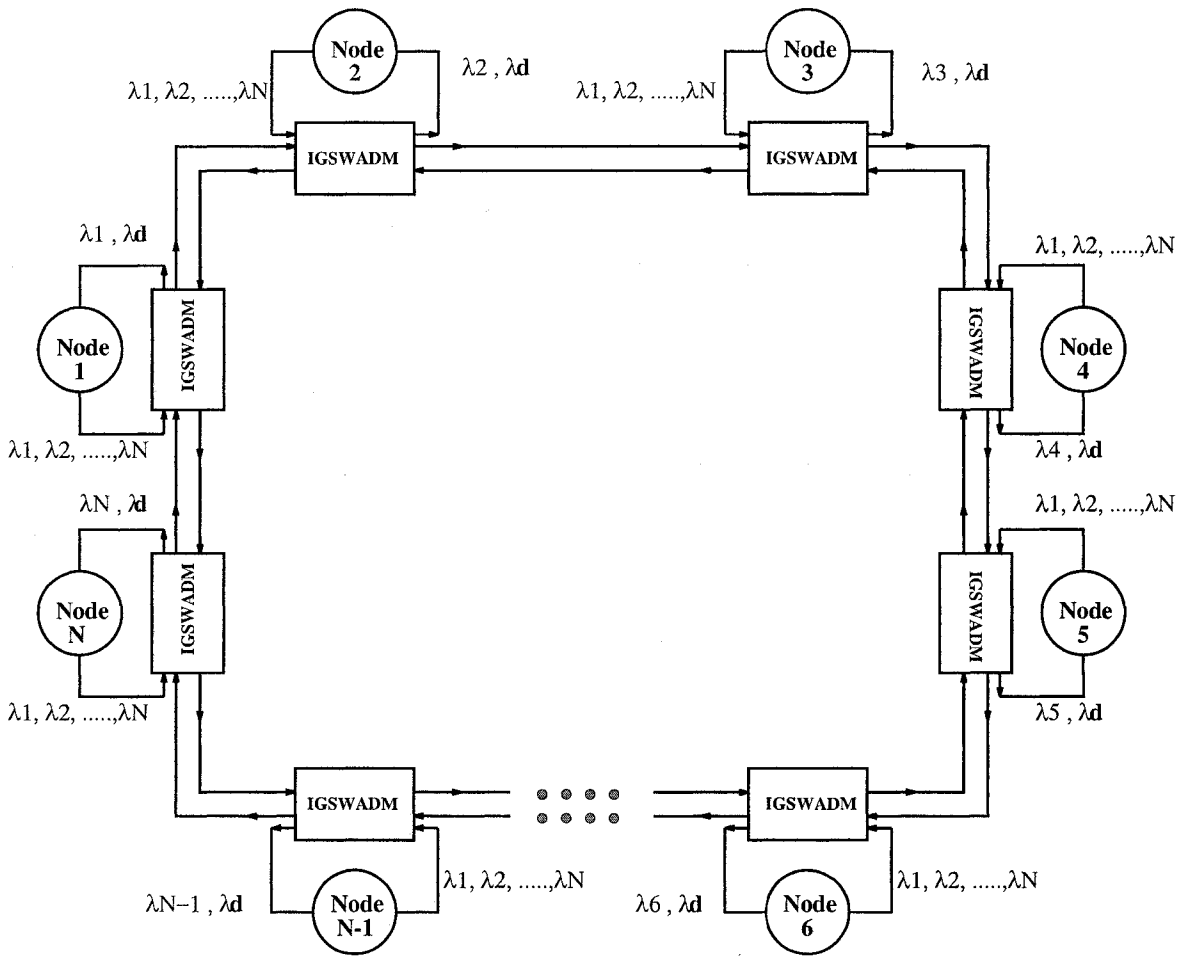


Figure 3.1: WDM Ring I network based on IGSWADM

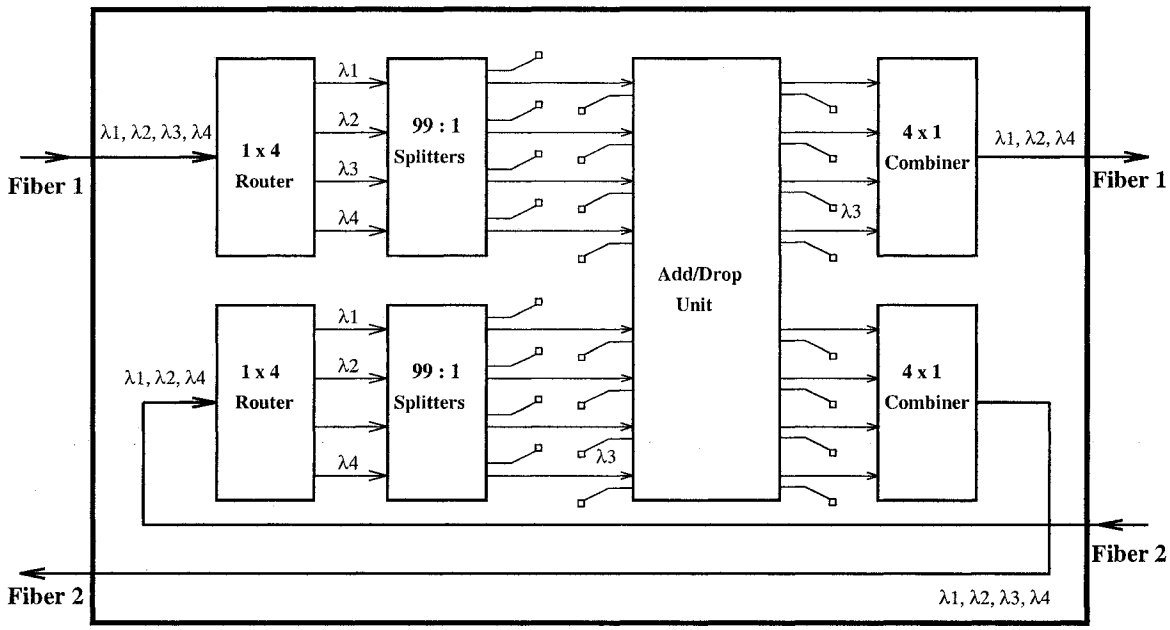


Figure 3.2: Schematic diagram for four wavelengths Integrated Group-Search Wavelength Add/Drop Multiplexer device (IGSWADM)

When a certain node wants to communicate with another node, it first senses the existence of the wavelength corresponding to the destination node on the ring. If this wavelength is free for use (i.e., not used by another node), the node will add a connection request signal on this free wavelength and will configure its add/drop unit to be ready for dropping the incoming acknowledgment signal on the same wavelength.

3.4 Medium Access Protocol

The medium access control protocol (MAC) to establish a connection between any two nodes, for example node i and node j, is as follows: Each node is assigned its own unique wavelength on which its fixed receiver is fixed tuned. Upon the arrival of a message at node i destined to node j, node i first senses the wavelength of node j (λ_j). If λ_j is free, node i sends a connection request to j on λ_j and tunes its tunable receiver to channel λ_j on the opposite fiber so that it will be able to receive the acknowledgment from node j. This connection request (CR) signal is a periodically repeated special message which contains the addresses of the sender node and the intended destination node. In the ideal case, this CR signal will propagate on the ring and go through the intervening nodes to the desired destination node. Node j, upon receiving the connection request of node i, will send out an acknowledgment on the same wavelength λ_j but on the other fiber. The tunable receiver of node i, which is tuned to channel λ_j, will receive the acknowledgment signal and will learn that node j is ready to receive its message. Node i will begin transmitting its message on wavelength λ_j. When the transmission is completed, both nodes will return to waiting mode for new arrivals.

During the propagation time of the connection request from the sender passing through intermediate nodes, it may happen that an intermediate node A may have a

connection request destined to the same destination node j and it sensed that λ_j is free, or it was already transmitting to node j on λ_j . In this cases, node A drops the connection request of node i and prevents it from reaching node j. When node A finishes its transmission it will reconfigure its add-drop multiplexer and will allow the connection request of node i to propagate to the next node in its way to the desired destination.

With this protocol, all nodes perform their transmission and reception independently and asynchronously. Each node can transmit to any other node and at the same time it can receive from any other node. For example, node i can transmit to node j on λ_j while it receives another node's transmission on its fixed assigned wavelength λ_i. Also, any two nodes can establish a full duplex connection between them, where a node transmits its connection request and messages on one fiber, while the other node transmits the acknowledgment signal and its messages to the sender on the other fiber.

In this protocol there is no possibility of deadlock. If two nodes are being sending connection requests to each other nearly at the same time, they will both receive the acknowledgment signals from each other on different wavelength channels and on different paths on the ring.

There is no timeout mechanism considered in this version of the protocol. Each node will keep sensing the desired channel till it be free and use it. A station will keep sending its connection request to the desired destination and wait for an acknowledgment.

3.5 Connection Setup Mechanism

The signaling mechanism for connection set up between two nodes, shown in Figure 3.3, is as follows: If node i want to communicate with node j, it will sense the channels to see if the wavelength of node j (λ_j) is free. After time T_{sens}, if λ_j is free, node i

will send a connection request (CR) signal using λ_j on one fiber. When the CR of node i reaches node j, it will be dropped by node j, processed in time T_{proc} and an acknowledgment (ACK) signal on λ_j will be added to the other fiber to propagate in the reverse direction through the intermediate nodes till it reaches node i. By this way, node i will learn that it can transmit its messages on λ_j to node j on the reserved connection. With this connection setup, there is a possibility that the CR is delayed at the intermediate nodes for certain waiting times W_k depending on the transmission length of these nodes to the same destination desired by node i as shown in Figure 3.3 and as follows:

$$\tau_{CR} = \begin{cases} (j-i)\tau + W_A \Rightarrow i < j \\ (N + j - i)\tau + W_B \Rightarrow i > j \end{cases}$$

$$W_A = \sum_{k=i+1}^{j-1} W_k$$

$$W_B = \sum_{k=i+1}^{N} W_k + \sum_{k=1}^{j-1} W_k$$

W_k is the waiting time of the node i connection request at node k, and k=1,2,3,..,N. The propagation delay between neighboring nodes is τ.

The duration of the acknowledgment signal τ_{ACK} is

$$\tau_{ACK} = \begin{cases} (j-i)\tau \Rightarrow i < j \\ (N + j - i)\tau \Rightarrow i > j \end{cases}$$

The total setup time W_T is

$W_T = T_{sens} + T_{proc} + \tau_{CR} + \tau_{ACK}$

For the best case W_k =0 and $\tau_{CR} = \tau_{ACK}$

So, the total setup time is : $W_T = T_{sens} + T_{proc} + 2\ \tau_{CR}$

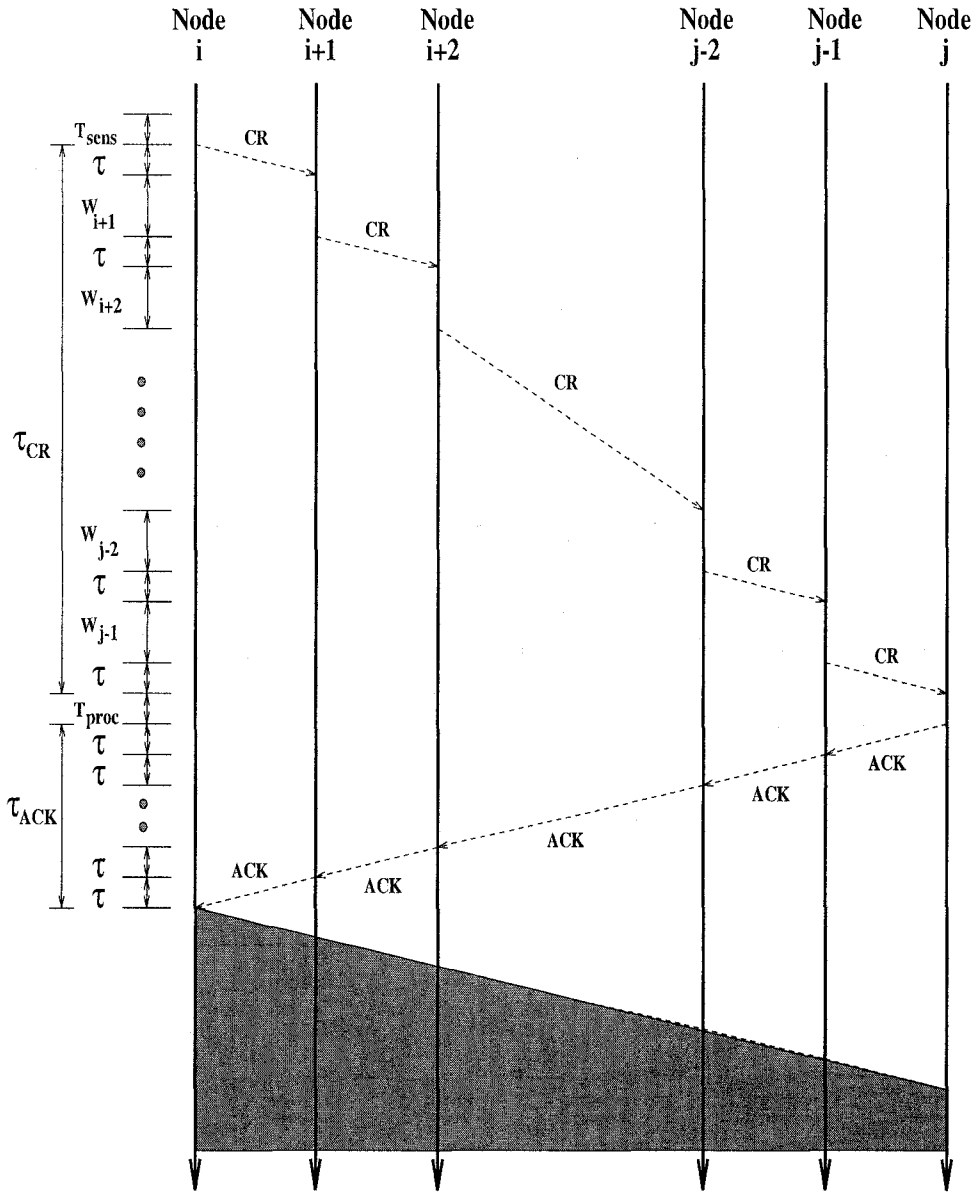


Figure 3.3: Connection setup timing

3.6 Network Performance

To evaluate the performance for the proposed network architecture and to quantify the effects of different network parameters on the overall performance, the following model and assumptions are considered.

- The network consists of N nodes.
- Number of wavelengths (W) is equal to the number of nodes (N).
- Each node has a single buffer for message storage. New arrivals are permitted only when the existing message transmitted and departs from the buffer.
- Messages arrive at each node according to Poisson processes.
- Message lengths are geometrically distributed with an average message length = ml seconds.
- The propagation delay between neighboring nodes is constant (i.e. nodes are equally separated) and is equal τ microseconds.
- Uniform traffic is considered. Transmission of each message is equally likely to any one of the other nodes.
- The transfer time of a message is a function of the source and destination respective positions on the network.

In our simulation, the following parameters were considered. The network consists of 4 nodes and there are 4 transmission channels. The mean message length, unless otherwise stated, is 10 msec. The mean arrival rate per node is varying between 10 and 400 arrivals/second. The propagation delay τ is 1 μsec. corresponding to 260 meters and network span of 1040 meters. The sensing time and the processing time at each node are neglected. We investigated the effect of these parameters on network performance.

Figure 3.4, illustrates the behavior of the normalized throughput versus message arrival rates for different message lengths and according to Table 3.1.

Table 3.1: Throughput vs. arrival rates for different message lengths.

Arrival Rate (arrivals/second)	Throughput		
	Message Length = 10 ms	Message Length = 50 ms	Message Length = 100 ms
5	0.047	0.194	0.312
10	0.090	0.311	0.440
40	0.259	0.542	0.611
60	0.347	0.587	0.636
80	0.401	0.611	0.648
150	0.507	0.646	0.666
200	0.547	0.656	0.671
240	0.580	0.661	0.674
300	0.591	0.667	0.676

As the arrival rate increases, more messages become available for transmission, throughput increases till it reaches certain saturation level where the network reaches its full capacity. Increase of message length will increase throughput. For larger message lengths throughput reaches its saturation level faster than in case of small messages.

Normalized throughput vs. propagation delay τ is plotted in Figure 3.5 at different arrival rates. Throughput values are plotted vs. propagation delay values as in Table 3.2.

Table 3.2: Throughput vs. propagation delay at different arrival rates

Propagation delay (μsec.)	Throughput		
	100 arrivals/second	200 arrivals/second	400 arrivals/second
10	0.089	0.161	0.267
50	0.088	0.156	0.253
100	0.086	0.150	0.236
200	0.083	0.140	0.200
400	0.077	0.120	0.165

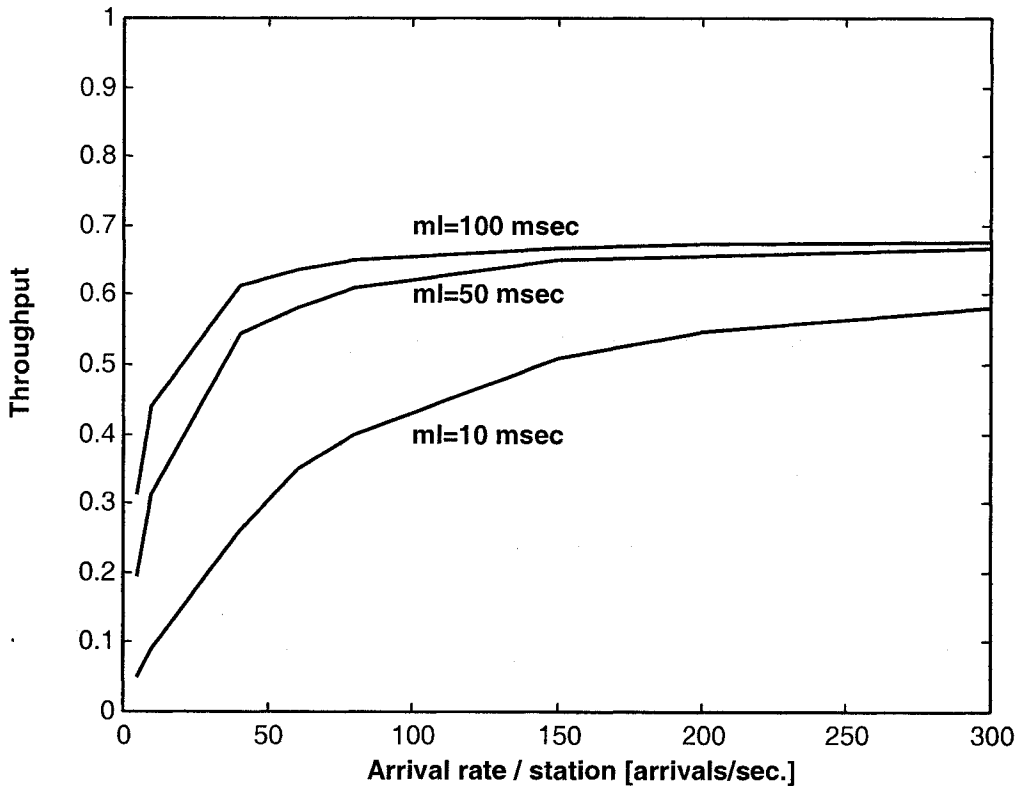


Figure 3.4: Throughput vs. arrival rate for different message lengths

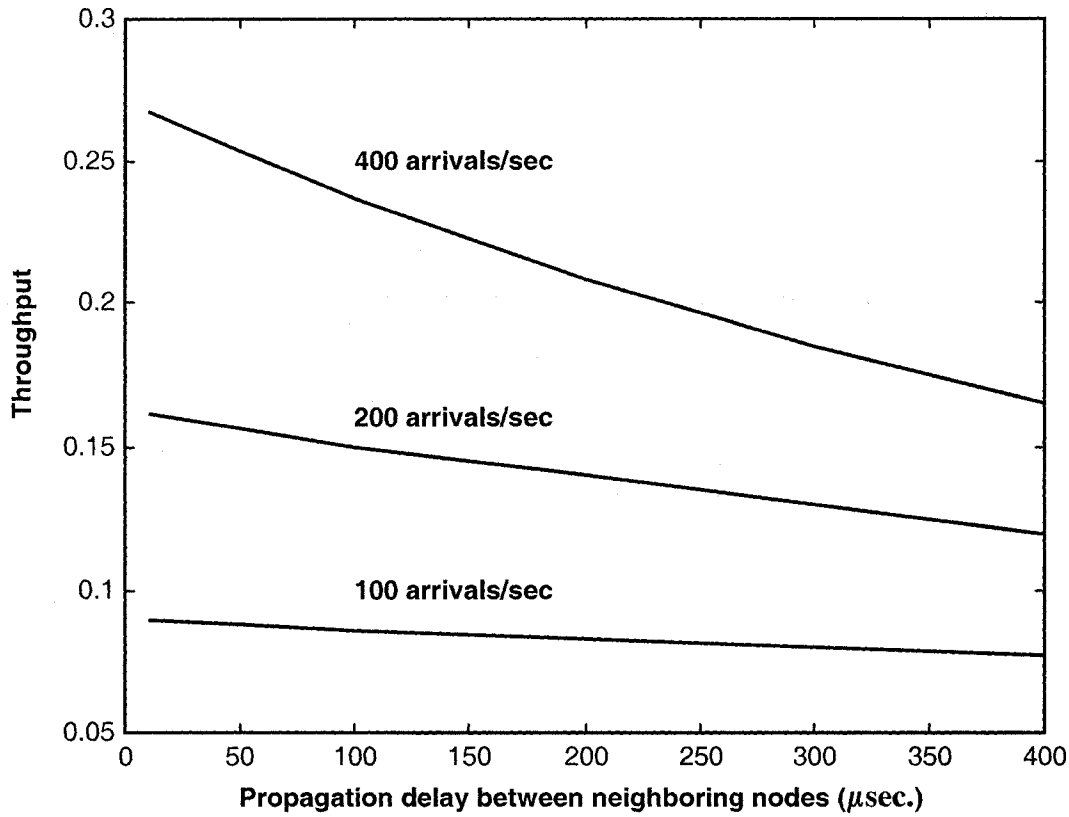


Figure 3.5: Throughput vs. propagation delay between neighboring nodes for different arrival rates

As the propagation delay increases the network throughput decreases because the messages will spend long times on the network till reach their destinations. Also, as the arrival rate increases, the effect of the propagation delay on throughput is remarkable.

In Figure 3.6, the mean access delay is plotted versus throughput for increasing arrival rates. Maximum throughput is limited to about 0.55. The delay increases proportionately with throughput and reaches larger values as throughput reaches its maximum limit.

3.7 Summary

A ring network architecture based on integrated group-search wavelength add/drop multiplexer (IGSWADM) device is introduced. This device used for adding the transmission of each node after sensing the transmission of other nodes. A detailed description for the IGSWADM is provided. A medium access protocol for controlling the operation of the network is proposed. In this protocol a node can transmit and receive at the same time. The connection setup between any two nodes is explained. The proposed protocol is suitable for the fast circuit applications. The performance of the network under this protocol is investigated. The effect of network parameters such as the message arrival rates, the message lengths, the propagation delay, is studied. The results shows that the proposed network has high performance at higher arrival rates and long message lengths.

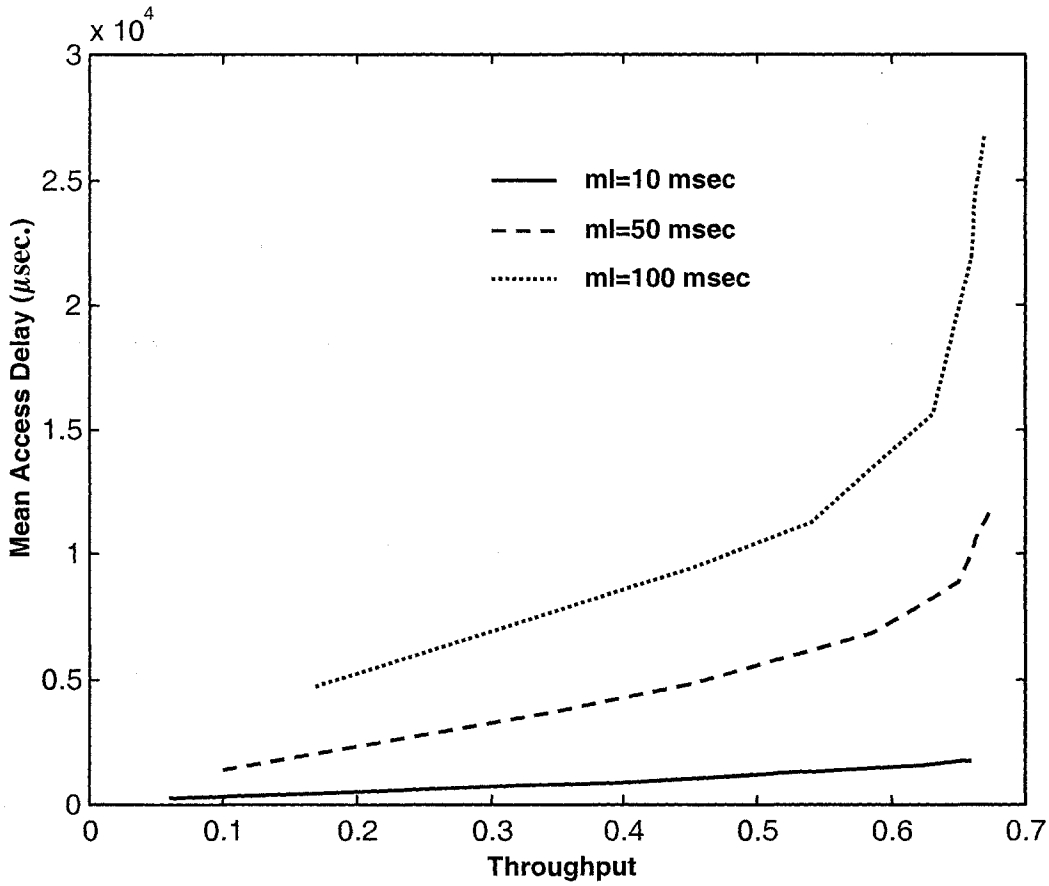


Figure 3.6: Mean access delay vs. Throughput for different arrival rates

CHAPTER 4

WDM RING II : A WDM RING NETWORK USING SIMPLE NODE ARCHITECTURE

4.1 Introduction

As the cost of a network depends to a large extend on the cost of nodes, it is essential to keep the network nodes architectures as simple as possible. This is achieved by simplifying the transmitter and the receiver hardware. Each node in the network is equipped with at least one transmitter and one receiver. In addition, to achieve connectivity between any pair of nodes, it is required that at least one of the two devices (transmitter or receiver) be tunable on a connection by connection basis.

Each node of the ring network described in chapter 3 was constructed using two transmitters. One transmitter is fixed tuned to the node wavelength for transmitting acknowledgment signals to the requesting nodes, when the node is requested by another node. The other transmitter is tunable to the desired destination wavelength for transmitting connection requests and messages to the desired destination node. Also, the architecture depended on two receivers. One receiver with fixed tuning for receiving the connection requests and messages, on the node's fixed wavelength, when the node is requested by another node. The second receiver is tunable and is meant for receiving the incoming acknowledgment signals form the desired destination node.

In this chapter, a second WDM ring network architecture (WDM Ring II) is presented and discussed in detail. This architecture is based on a simpler node construction compared to the previous one. Each node is equipped with only one tunable transmitter and only one fixed tuned receiver. As in the previous ring architecture, the key element in this architecture is the Integrated Group-Search Wavelength Add/Drop Multiplexer (IGSWADM) which used for sensing the desired destination wavelength and inserting/dropping the signals. A simple modification is made in the construction of the IGSWADM, its architecture is simpler. A detailed description of the network architecture and the IGSWADM is provided. A medium access protocol for fast circuit switched applications is proposed for controlling the operation of the network. Performance of the network is evaluated, discussed, and compared with that of the previous ring network.

4.2 Network Architecture

As shown in Figure 4.1, the proposed WDM ring network architecture consists of N nodes interconnected by two fibers forming a ring. Network nodes are connected to the ring through IGSWADM devices. Transmission direction on one fiber is opposite to that on the other fiber. Each node is equipped with only one tunable transmitter for transmitting connection requests, messages, and acknowledgment signals to other nodes, and only one fixed tuned receiver for receiving connection requests, messages, and the incoming acknowledgment signals from other nodes. Each node is assigned a reserved wavelength for reception. Number of wavelengths (W) is equal to number of nodes (N). Each node can transmit using any free wavelength according to the desired destination. A transmitting node tunes its tunable transmitter to the wavelength of the desired destination node receiver and transmits its message.

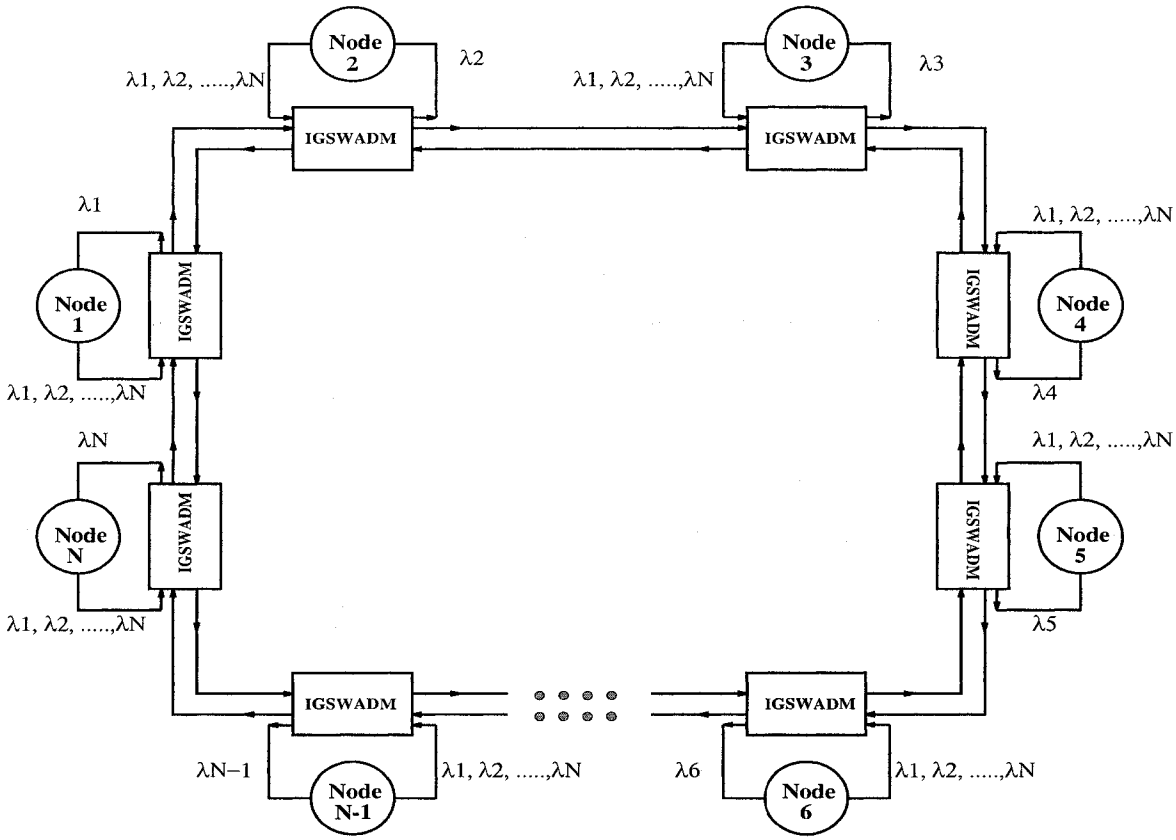


Figure 4.1: The configuration of the proposed WDM Ring II

The message propagates in the optical medium until it is extracted from the ring by the destination node's receiver. Each node can be involved in only one connection at a time. It can transmit only to one other node or receive only from one other node at a time. This means that any node can communicate with only one node at a time. A node cannot transmit and receive at the same time. For the description and evaluation, assume that each node has a buffer that can hold only one message. Each node is allowed to drop its own wavelength from fiber 1 when receiving connection requests and messages and from fiber 2 when receiving the incoming acknowledgments. So, the node receiver is connected to fiber 1 when the node is in the waiting mode, while it is connected to fiber 2 when it is in the requesting mode and waiting for an acknowledgment.

4.3 Integrated Group-Search Wavelength Add/Drop Multiplexer (IGSWADM)

The proposed ring network architecture in this chapter is based on a modified version of the IGSWADM used by the previous network architecture. Figure 4.2 shows the architecture of the modified IGSWADM implementing four wavelength channels. For N wavelengths, it consists of two (1 x N) wavelength routers, only one 99:1 splitter, two (N x 1) combiners, and a reconfigurable add/drop unit. Here, only one 99:1 splitter unit is needed and connected to fiber 1. This is because any node can be in one mode at a time, either transmission or reception. Also, this will keep the node construction more simple. In addition, the add/drop unit has a fixed configuration to drop the incoming acknowledgment signals on the node wavelength from the destination node on fiber 2. Each node needs to sense only the traffic on fiber 1. The tunable transmitter will be connected to fiber 1 when transmitting connection requests and messages, and to fiber 2 when transmitting acknowledgments.

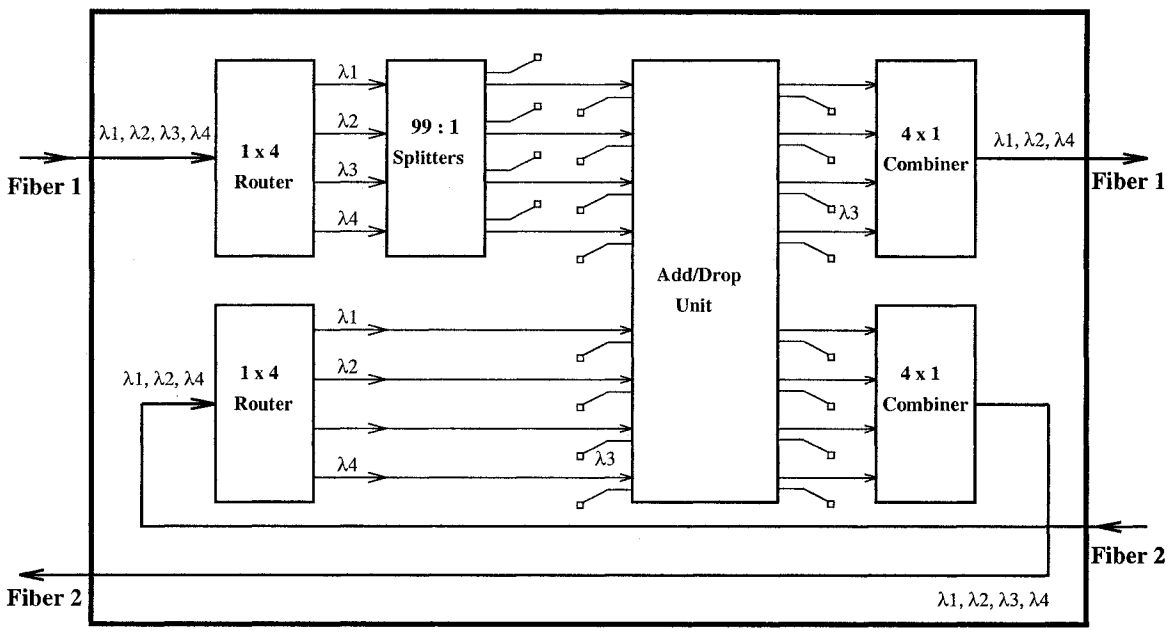


Figure 4.2: Schematic diagram of the modified four wavelengths integrated group-search wavelength add/drop multiplexer device (IGSWADM).

By sensing the traffic on fiber 1, each node will be able to monitor the presence of a particular wavelength. When a certain node wants to communicate with another node, it first senses the existence of the desired destination wavelength. If this wavelength is free (not used by any other node), the node will configure its IGSWADM add/drop unit to add a connection request signal on this free wavelength and to drop any signal will come later from the preceding nodes on this wavelength. At the same time, the requesting node is configured to drop the incoming acknowledgment on its wavelength. For example, in a network consisting of 6 nodes, when node 2 wants to setup connection with node 5, it first senses the output of the splitter for the desired destination wavelength ($\lambda 5$). If $\lambda 5$ exist, this mean that another node is already transmitting a message or a connection request to node 5. In this case, node 2 will wait till that connection is finished and $\lambda 5$ becomes free. When node 2 senses that $\lambda 5$ is free, it configures its add/drop unit to drop any signal that will arrive later on $\lambda 5$ from any preceding node, and insert its connection request on $\lambda 5$ into the traffic to node 5. Also, it waits to drop the incoming acknowledgment from node 5 on $\lambda 2$.

4.4 Medium Access Protocol

Figure 4.3 shows the flow chart of the medium access control protocol (MAC) for setting up a connection between any two nodes, for example i and j, as follows: Upon the arrival of a message at node i destined to node j, node i first checks its receiver status. If its receiver is busy, it waits till it finishes reception then reserves it for receiving the incoming acknowledgment. Next, it ignores any incoming connection request directed to it from other nodes. Then, node i senses the wavelength (λ_j) of the desired destination (node j). If λ_j is free (not used by any preceding node in transmission to node j), node i

send a connection request (CR) signal to node j on wavelength λ_j. This CR signal is a periodically repeated special message that contains the addresses of the sender and the destination. In ideal case, this CR signal will propagate on the ring and go through the intervening nodes to the destination. Upon receiving the connection request of node i, node j will send out an acknowledgment on the wavelength of the requesting node i (λ_i) on fiber 2 to propagate back to node i. Node i's receiver which is fixed tuned to λ_i will receive the acknowledgment signal and will learn that node j is ready to receive its message. Node i will begin transmitting its message on wavelength λ_j. When transmission is completed, node i will release its receiver for receiving any waiting connection request directed to it from other nodes.

With this protocol, there is a possibility that when a message arrives at node i, the node may sense the destination wavelength busy (in use) for a long time. Also, it may happen that the CR signal does not get through an intermediate node for a long time. In addition, when a CR signal reaches its destination, it may find the destination busy. A deadlock may also happen, if two nodes begin sending connection requests to each other nearly at the same time. They will both reserve their receivers for the incoming acknowledgments, but since they will not receive the connection request of each other, the acknowledgments will never be sent. To overcome all these problems, the protocol includes a timeout mechanism. If an acknowledgment is not received within certain timeout duration measured from message arrival instant, the connection setup is aborted. The requesting node releases its receiver to receive any connection request directed to it, and waits for a random time after which it tries to set up the same connection again.

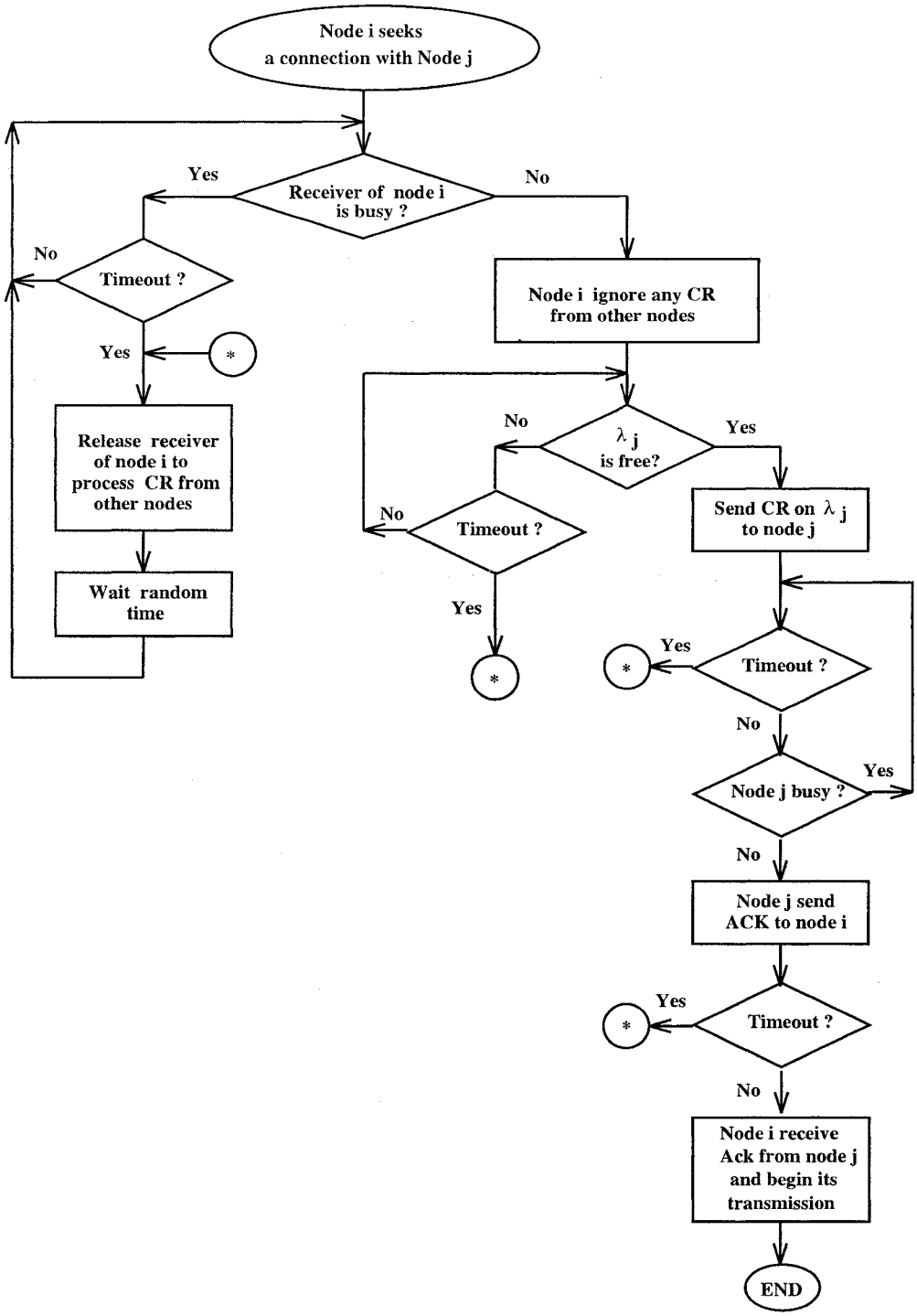


Figure 4.3: Flow chart for the medium access protocol used to setup a connection between network nodes

4.5 Network Performance

To evaluate the performance of the proposed ring architecture and to investigate the effects of the network parameters, the following assumption are used:

- The network consists of N nodes
- Each node is assigned a reserved wavelength for reception. The number of wavelengths (W) is equal to the number of nodes
- There is no queuing. Each node has a single buffer for a message storage. New arrivals are permitted only when the existing message transmission is completed and the buffer becomes empty.
- Messages arrive at each node according to Poisson processes.
- Message lengths are geometrically distributed with an average message length = *ml* seconds.
- The propagation delay between neighboring nodes is constant (i.e, nodes are equally separated) and is equal τ microseconds.
- Uniform traffic distribution among nodes is considered. Transmission of each message is equally likely to any one of the other nodes.
- The transfer time of a message is a function of the source and destination positions on the network.
- Signal sensing times and processing times at each node are neglected
- Various timeout durations are considered.

In the simulation program, the following default values are considered:

- The ring network consists of N=6 nodes
- There are W=6 transmission channels

- The mean message lengths considered are 10 ms, 50 ms, and 100 ms.
- The mean arrival rate per node is varying between 10 and 1000 arrivals/second
- The propagation delay τ is 10 μs corresponding to 2600 meters and network span of 15600 meters. In some cases propagation delay τ is varied between 10 μs and 400 μs.
- Timeout periods are varying between 0.1 ms and 2000 ms.

The effect of some of these parameters on the network performance is studied by varying them around the default values.

In Figure 4.4, the behavior of the normalized throughput versus message arrival rates is illustrated. In this study, the timeout duration is equal 1 ms and the propagation delay τ is 10 μs. The throughput values are plotted versus arrival rate values for different message lengths (ml) according to Table 4.1.

Table 4.1: Throughput vs. arrival rate for different message lengths

Arrival Rate (arrivals/second)	Throughput		
	Message Length = 10 millisecond	Message Length = 50 millisecond	Message Length = 100 millisecond
10	0.058	0.120	0.150
30	0.126	0.225	0.270
50	0.175	0.270	0.300
70	0.200	0.300	0.320
100	0.243	0.320	0.330
150	0.274	0.330	0.337
200	0.295	0.340	0.345
300	0.316	0.346	0.351
400	0.325	0.348	0.352
600	0.336	0.350	0.355
800	0.342	0.353	0.355
900	0.344	0.353	0.355
1000	0.347	0.353	0.355

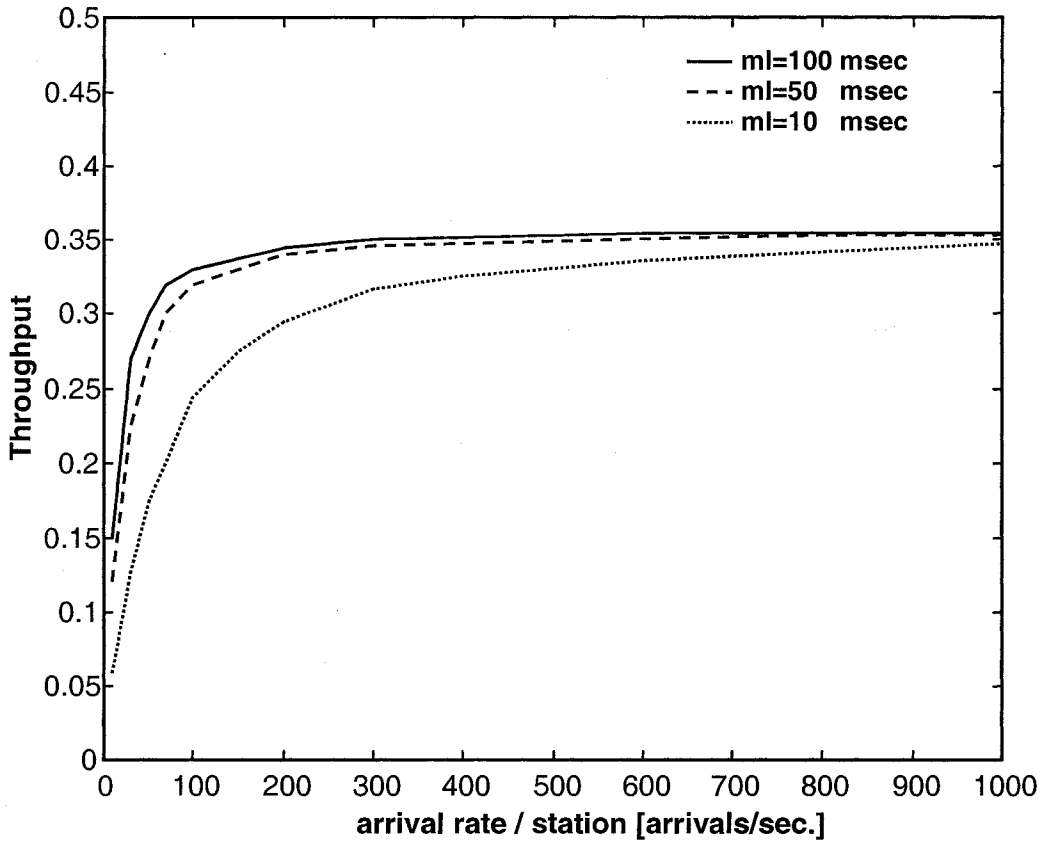


Figure 4.4: Throughput vs. arrival rate for different message lengths

As the arrival rate increases, more messages become available for transmission, throughput increases till it reaches certain saturation level where the network reaches its full capacity. Figure 4.4 does not show the effect of the timeout duration on the throughput values. This is because the selected timeout duration (1 ms) was reasonable to let the system achieves its maximum throughput and stay at the same saturation level for long range of arrival rates. The effect of timeout on the network performance is shown in the next studies.

Figure 4.5 illustrates the behavior of the throughput versus timeout durations for different arrival rates, considering mean message length equal 100 ms and propagation delay τ equal 10 μs. Throughput values are plotted versus timeout durations for various arrival rates according to Table 4.2.

Table 4.2: Throughput vs. timeout at different arrival rates

Timeout Duration (ms)	Throughput		
	10 arrivals/second	50 arrivals/second	200 arrivals/second
0.1	0.111	0.191	0.221
0.5	0.147	0.292	0.327
1	0.165	0.302	0.341
5	0.215	0.321	0.351
10	0.241	0.331	0.345
50	0.262	0.322	0.328
100	0.268	0.301	0.303
500	0.232	0.234	0.232
1000	0.191	0.186	0.179
2000	0.159	0.139	0.128

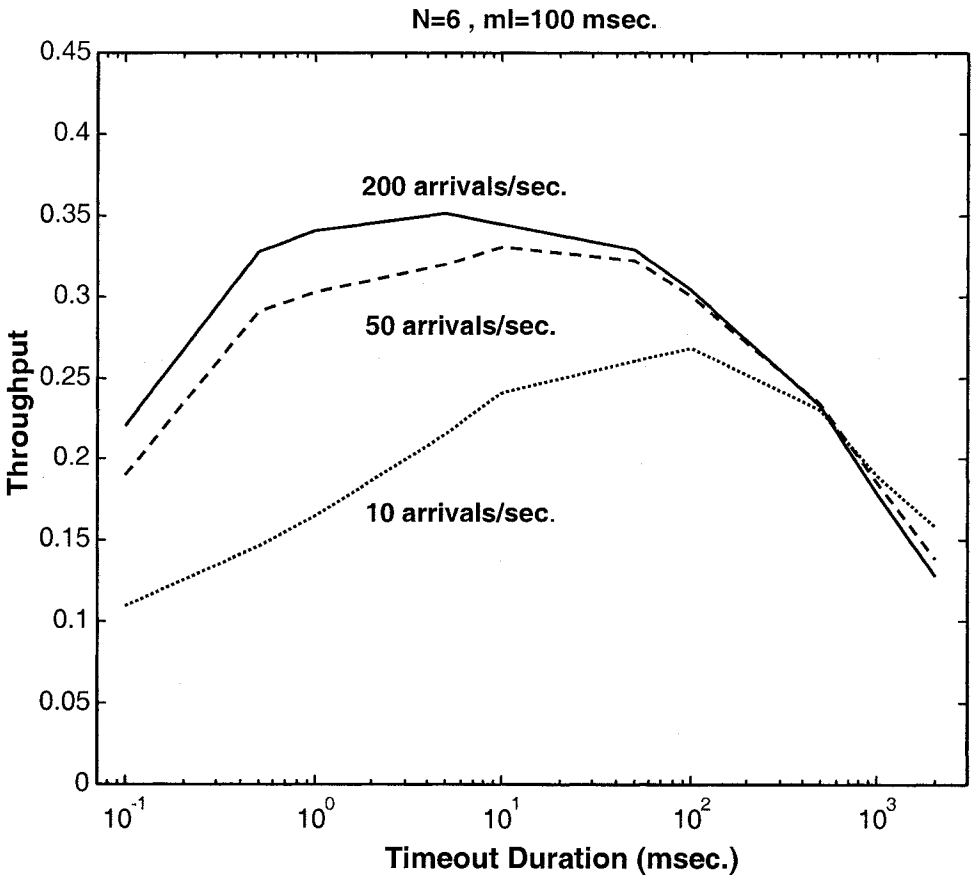


Figure 4.5: Throughput vs. timeout durations for different arrival rates

For all the considered arrival rates, as timeout duration increases, throughput first increases and then decreases. As shown in Figure 4.5, at low timeout durations, throughput values are small. This is because connection requests are timing out quickly, before they get the chance to be acknowledged. By increasing timeout duration, throughput values increase because more connection requests will be acknowledged. Increasing timeout duration even further, network nodes will spend longer amounts of time in the requesting mode and more nodes waiting for each other. As a result, fewer nodes will be available for acknowledging connection requests and throughput values decrease. An important result noticed, is that for a given arrival rate, there is an optimum timeout value at which the network reaches its maximum throughput. For 10 arrivals per second, the network achieves its maximum throughput of 0.268 at a timeout duration equal 100 ms. For 50 arrivals per second, the network achieves a maximum throughput of 0.33 at timeout duration equal 10 ms. For 200 arrivals per second, the maximum throughput equal 0.351 at a timeout duration equal 5 ms. This leads to an important conclusion that the function of the medium access control protocol can be improved by applying dynamic timeout duration. By changing the values of the timeout based on the arrival rate values at each node, it may be possible to get maximum throughput for certain arrival rates. Also, it is shown in the figure that the throughput reaches higher values for higher arrival rates, and reaches its maximum point faster than that in case of lower arrival rates.

Figure 4.6 shows the behavior of the throughput versus timeout duration for various message lengths. In this study the arrival rate of 200 arrivals per second is

considered and the propagation delay τ is equal 10 μs. Throughput values are given in Table 4.3. versus timeout durations for different message lengths.

Table 4.3: Throughput vs. timeout at different message lengths

Timeout Duration (ms)	Throughput		
	Message length = 10 ms	Message length = 50 ms	Message length = 100 ms
0.1	0.174	0.192	0.221
0.5	0.277	0.323	0.329
1	0.295	0.336	0.342
5	0.311	0.347	0.352
10	0.301	0.341	0.345
50	0.245	0.311	0.328
100	0.204	0.287	0.303
500	0.102	0.191	0.232
1000	0.065	0.142	0.179
2000	0.038	0.093	0.128

For all the considered message lengths, as timeout duration increases, throughput first increases and then decreases. As shown in Figure 4.6, at low timeout durations, connection requests are timing out quickly before being acknowledged, resulting in a low throughput. As timeout duration increases, more connection requests will be acknowledged, resulting in a higher throughput. As timeout increases even further, number of requesting nodes begins to be more than number of nodes that are available to acknowledge connection requests. This decreases throughput. An important result noticed is that the network achieves its maximum throughput at the same optimal timeout duration (5ms) for the different message lengths. This leads to conclusion that, this optimal timeout is not significantly affected by the message length. Throughput reaches higher values for long message lengths. Maximum throughput is 0.352 for message length =100 ms, 0.347 for message length =50 ms, and 0.311 for message length =10 ms.

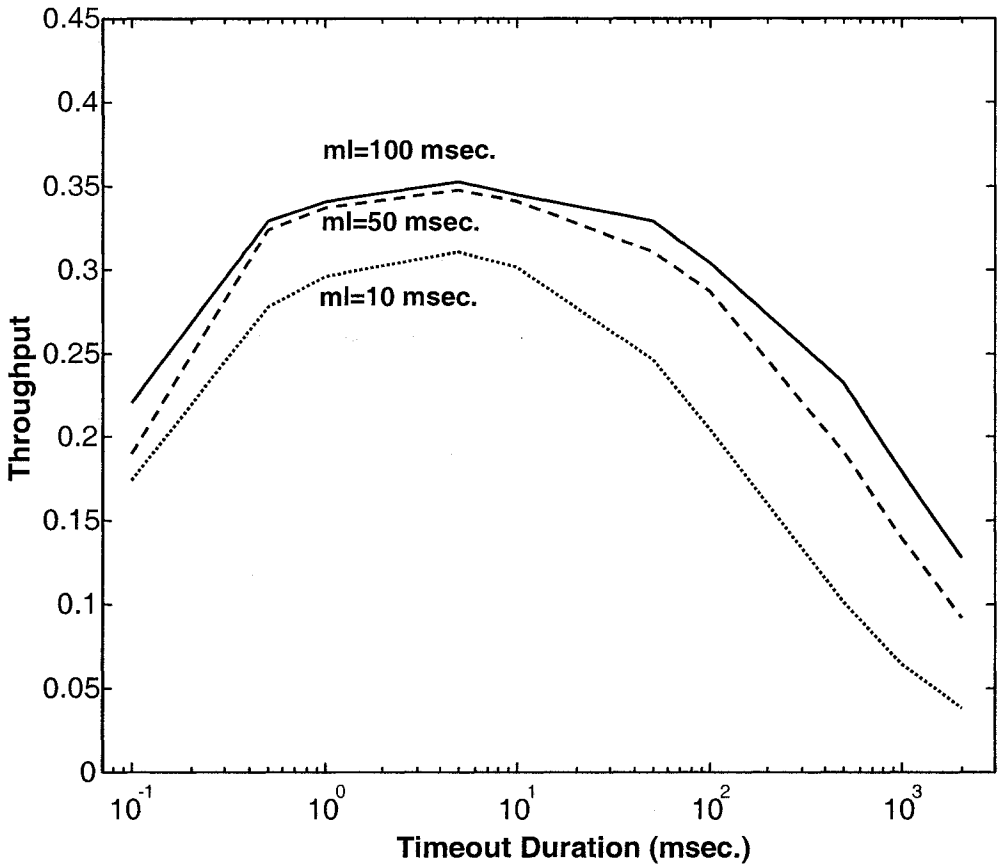


Figure 4.6: Throughput vs. timeout durations for different message lengths

Figure 4.7 shows the behavior of the network throughput versus the propagation delay between neighboring nodes, τ, for different arrival rates. In this study, the timeout duration is kept fixed at 10 ms, and a message length of 100 ms is considered. Throughput values are plotted versus propagation delay values according to Table 4.4.

Table 4.4: Throughput vs. propagation delay at different arrival rates

Propagation delay (μs)	Throughput		
	100 arrivals/second	200 arrivals/second	400 arrivals/second
10	0.273	0.301	0.315
50	0.265	0.289	0.298
100	0.253	0.271	0.275
200	0.221	0.219	0.212
400	0.111	0.106	0.080

As shown in Figure 4.7, at all the arrival rates, increasing the propagation delay between nodes, by increasing the fixed distance between neighboring nodes, will cause a decrease in network throughput. This is because, the messages will spend long times on the network till reach their destinations. Also, as the arrival rate increases, increasing the propagation delay will cause the decrease of network throughput quickly.

The following study shows the effect of simplifying the architecture of the network nodes, by having only one transmitter and only one receiver, and the need for a timeout mechanism. Figure 4.8 shows the behavior of network throughput versus arrival rates for two cases. The first case is when the network designed based on simple node structure by equipping each node with one transmitter and one receiver. In this case the medium access protocol is based on a timeout mechanism (i.e, with timeout). The second case is the design discussed in chapter 3, in which each node is equipped with two transmitters and two receivers. In this case, there is no need for a timeout mechanism as explained in chapter 3.

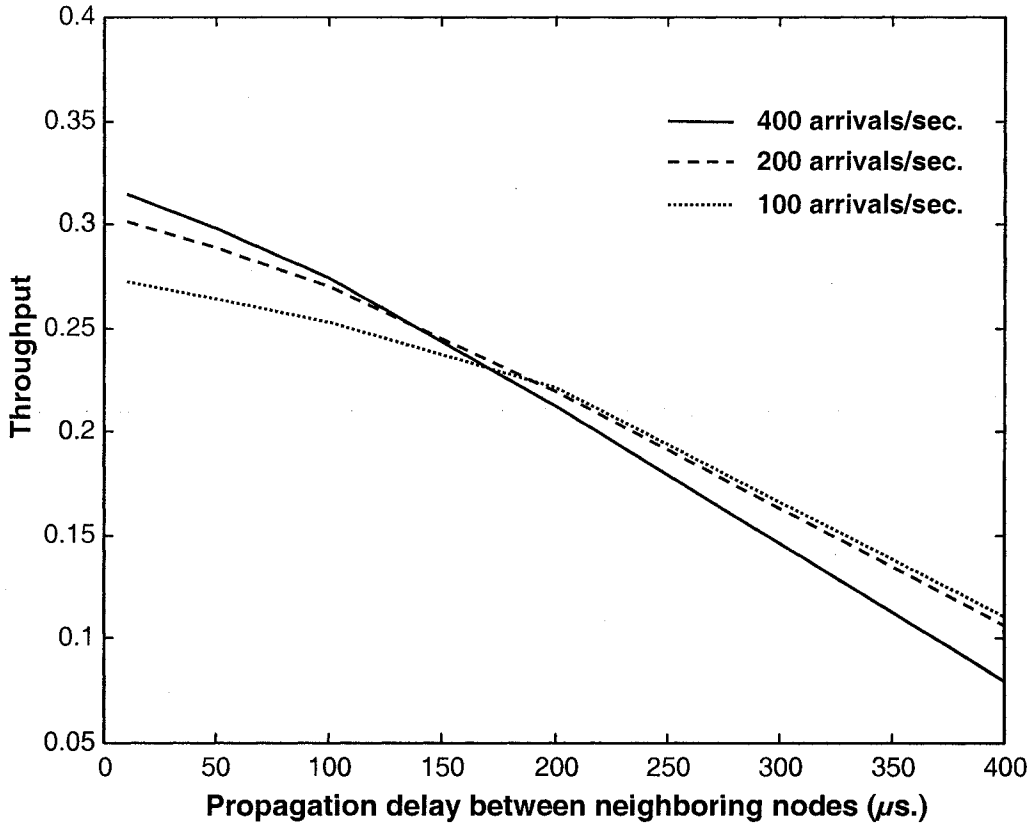


Figure 4.7: Throughput vs. propagation delay between neighboring nodes for different arrival rates

Throughput values versus arrival rates are given in Table 4.5 for case 1 and case 2. Both cases consider a network consists of 6 nodes, message length with of 100 ms, and a propagation delay equal 10 μs. In case 1, a timeout duration of 1 ms is considered. In Figure 4.8, as the arrival rate increases, the throughput increases in both cases. For case 1 (considering timeout), the network achieves a maximum throughput of about 0.355. For case 2 (without timeout), the network achieves maximum throughput of about 0.66. This means that, the network based on simple node architecture has a poor performance compared with the network where each node is equipped with two transmitters and two receivers. To simplify the network architecture, there is a need for a complicated medium access protocol to control the access to the network, and there are many restrictions on the ability of transmission and reception as explained in section 4.4 of this chapter.

Table 4.5: Throughput vs. arrival rate for case 1 and case 2

Arrival Rate (arrivals/second)	Throughput	
	Case 1 (with timeout)	Case 2 (without timeout)
10	0.151	0.409
30	0.272	0.540
50	0.300	0.571
100	0.333	0.600
200	0.345	0.615
400	0.351	0.621
600	0.355	0.643
1000	0.355	0.660

4.6 Summary

A ring network based on a simple node architecture is presented and discussed. In this network, each network node is equipped with only one transmitter and only one receiver. This node architecture affects the performance of the network, since there is a need for a timeout mechanism to overcome deadlocks during network operation.

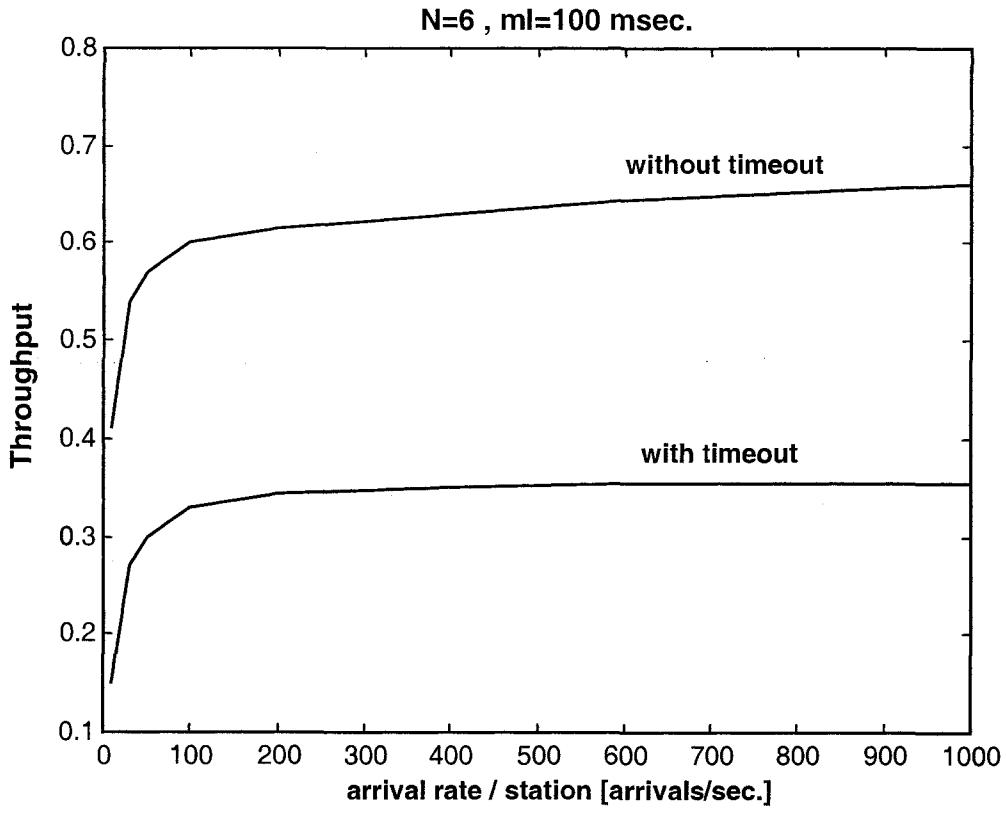


Figure 4.8: Throughput versus arrival rates for case 1 (considering timeout) and case 2 (without timeout)

A medium access protocol considering timeout mechanism is presented. The limitations on the ability of each node to access the network are considered. Each node is involved in only one transmission or one reception at a time.

In this network architecture, the design of the integrated group search add/drop multiplexer device is simplified by using only one 99:1 splitter unit. Each node will sense only the traffic on fiber 1 in case of requesting a connection with another node.

The effects of the different network parameters such as message arrival rate and timeout duration, on the network performance are investigated. The results show that, there is a great effect of selecting the suitable timeout duration for each arrival rate in each node. There is an optimal timeout duration for each arrival rate which lets the network achieve its maximum throughput. This leads to a conclusion that the function of the medium access protocol can be improved by the dynamic selection of the timeout duration. It is further shown that, the optimal timeout duration is not affected by the message lengths. The network throughput increases with increasing arrival rates and then decreases depending on the considered timeout duration. The network achieves higher performance for long message lengths.

CHAPTER 5

WDM RING III: A WDM RING NETWORK USING LOW NUMBER OF CHANNELS

5.1 Introduction

An important issue to consider in the design of a WDM network is the number of wavelengths available for use. The bandwidth of the optical fiber is limited to the low attenuation regions around 1300 nm and 1550 nm. Each region has bandwidth of approximately 200 nm. The number of wavelengths is dictated by the optical devices and not by the bandwidth available in an optical fiber. Today, the number of wavelengths available in WDM systems is limited to about 100 channels or less. Although the number of channels that is available is high, for reasons of economy, it is often necessary to restrict the number of channels in a network to a small number [21], [34], [48].

Wavelength reuse techniques allow a particular wavelength used in one place in the network also to be used in another place at the same time, thus allowing the design of optical networks with reduced number of wavelengths [14], [33].

In the last two ring network architectures, WDM Ring I and WDM Ring II, the number of wavelengths used is equal to the number of network nodes. In these networks, each node is equipped with at least one fixed receiver tuned to a reserved wavelength for reception.

In this chapter, a WDM ring network architecture using N/2 of wavelength channels, where N is the number of nodes, is presented and discussed. This architecture has the same node structure used for the WDM Ring I. Each node is equipped with two transmitters and two receivers. The number of channels is less than the number of nodes. For a ring network consisting of 6 nodes, the number of used wavelengths is equal 3. Integrated Group-Search Wavelength Add/Drop Multiplexer (IGSWADM) is used here too for sensing the desired destination wavelength and inserting/deleting the optical signals. A detailed description of the network architecture and the IGSWADM is provided. A medium access protocol for fast circuit switched applications is proposed for controlling the operation of the ring network.

5.2 Network Architecture

Figure 5.1 shows the proposed WDM Ring III architecture for a network consisting of 6 nodes interconnected by two fibers (fiber 1 and fiber 2) forming a ring. The network nodes are connected to the ring via IGSWADM devices. The direction of transmission on one fiber is opposite to that on the other fiber. Each node is assigned a reserved wavelength (λ) for receiving connection requests and messages directed to it from other nodes, and for transmitting the acknowledgement signals. In this network architecture, the number of used wavelengths (W) is half the number of nodes (N) (i.e., W=N/2). For a ring network consisting of 6 nodes, the wavelengths are assigned to the nodes as follows:

- Node 1 and node 4 use wavelength λ_1
- Node 2 and node 5 use wavelength λ_2
- Node 3 and node 6 use wavelength λ_3

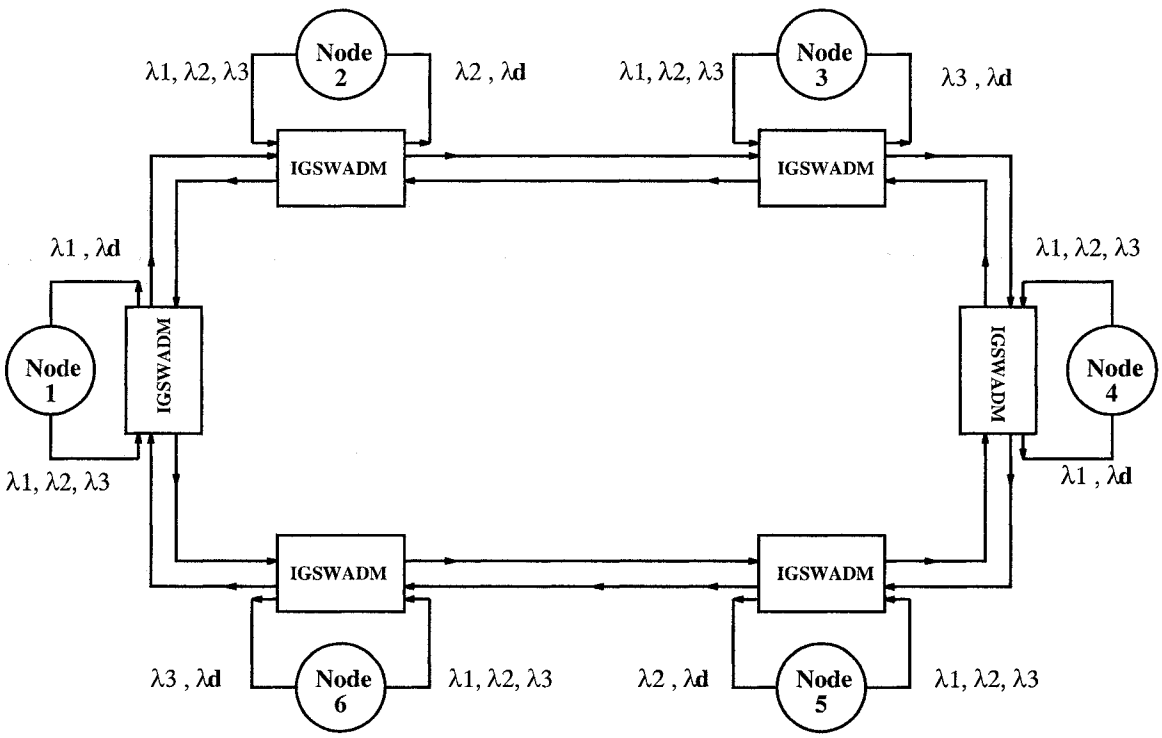


Figure 5.1: The configuration of the proposed WDM Ring III.
(for N=6 nodes)

The nodes that use the same wavelength are called "wavelength similar nodes". In this architecture, each network node is equipped with one tunable transmitter for sending the connection requests and messages on the wavelength of the destination node, one fixed transmitter for sending the acknowledgements on the node reserved wavelength, one fixed receiver for receiving the connection requests and messages directed to the node from other nodes, and one tunable receiver for receiving the incoming acknowledgments on the destination wavelength.

The following rules are used to control the operation of the proposed ring network:

- Each node has to send its connection requests and messages to the N/2 succeeding nodes using one fiber (for example fiber 1), and send its connection requests and messages to the N/2 preceding nodes using the other fiber (fiber 2). For example, for N=6, node 1 uses fiber 1 to send its connection requests to node 2, node 3, and node 4, while using fiber 2 to send to node 6, node 5, and node 4. The wavelength similar nodes can use fiber 1 or fiber 2 to send signals to each other. The whole fiber connection pattern, for N=6, is shown in Figure 5.2.
- Each node can transmit using any free wavelength according to the desired destination. A transmitting node tunes its tunable transmitter and its tunable receiver to the wavelength of the desired destination node's receiver and transmits its connection request. In the same time; it is ready to receive the incoming acknowledgement on the destination wavelength. Connection requests and messages propagate in the optical medium until it is extracted from the ring by the destination node's receiver.

- A node can be involved in two connections at a time. It can transmit to another node and receive from another node at the same time.
- Each node is allowed to drop its own wavelength from one fiber when receiving connection requests and messages and from the other fiber when receiving the incoming acknowledgments.
- A requesting node has to sense the traffic on both fibers before sending its connection request. If the desired destination wavelength is sensed busy (in use) on any fiber, it is considered busy on the other fiber and cannot be used. This rule is not applied to the wavelength similar nodes.
- For wavelength similar nodes, when one node wants to send connection request to its wavelength similar node, it senses the presence of its wavelength on both fibers. If its wavelength was busy on fiber 1, it has to use the same fiber 1 to send its connection request, and if it is busy on fiber 2, it has to use fiber 2. This will be done after dropping the incoming signal on its wavelength. This way, the same wavelength will be reused but in different part of the ring.
- A transmitting node has to drop the destination wavelength from both fibers. This will prevent the connection request, which come from preceding nodes and directed to the same destination from reaching their destination. In addition, this will allow the node to receive the incoming acknowledgement from the destination node. A requested node sends its acknowledgment reply on its reserved wavelength to go back to the requesting node on the reserved path.
- Each node can transmit to only one destination at a time and can handle only one reception at a time.

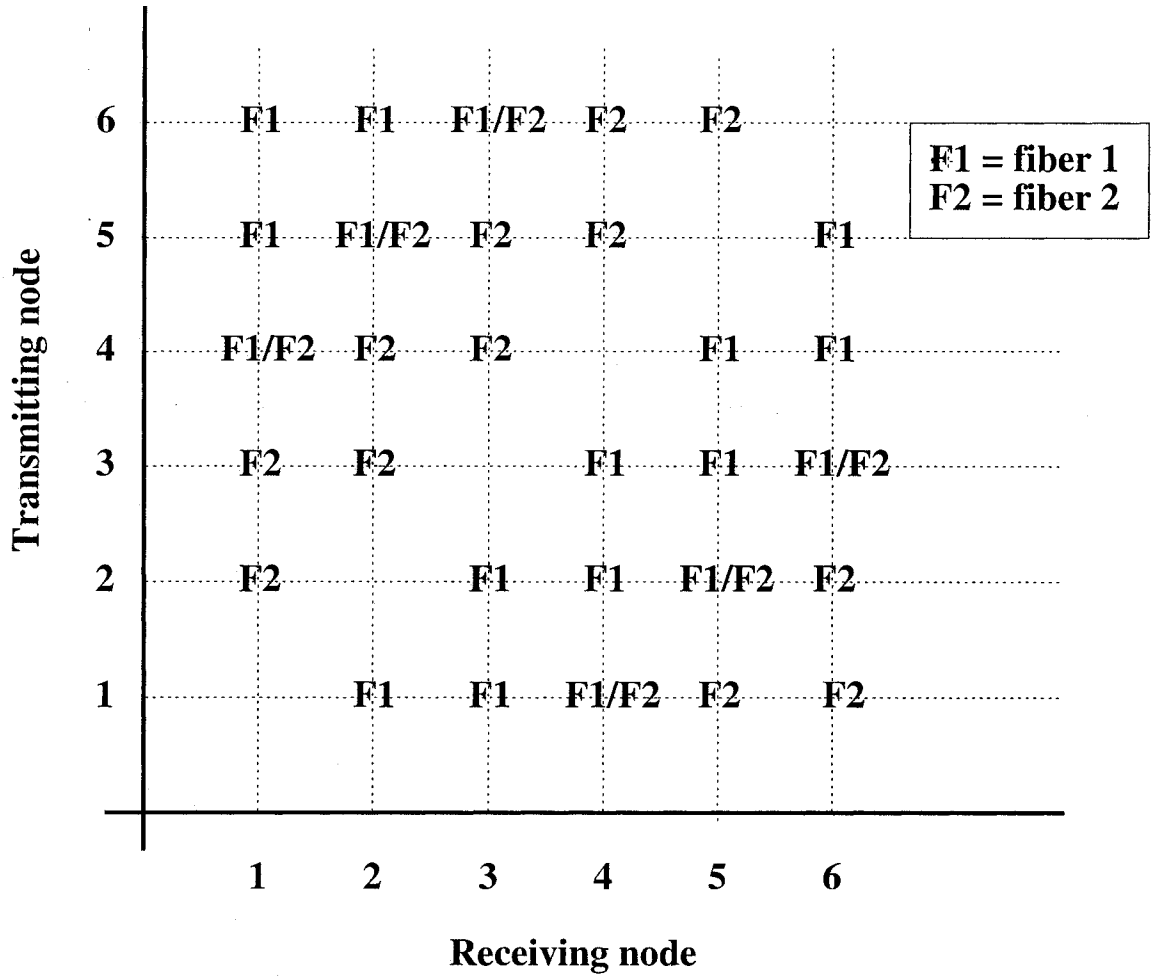
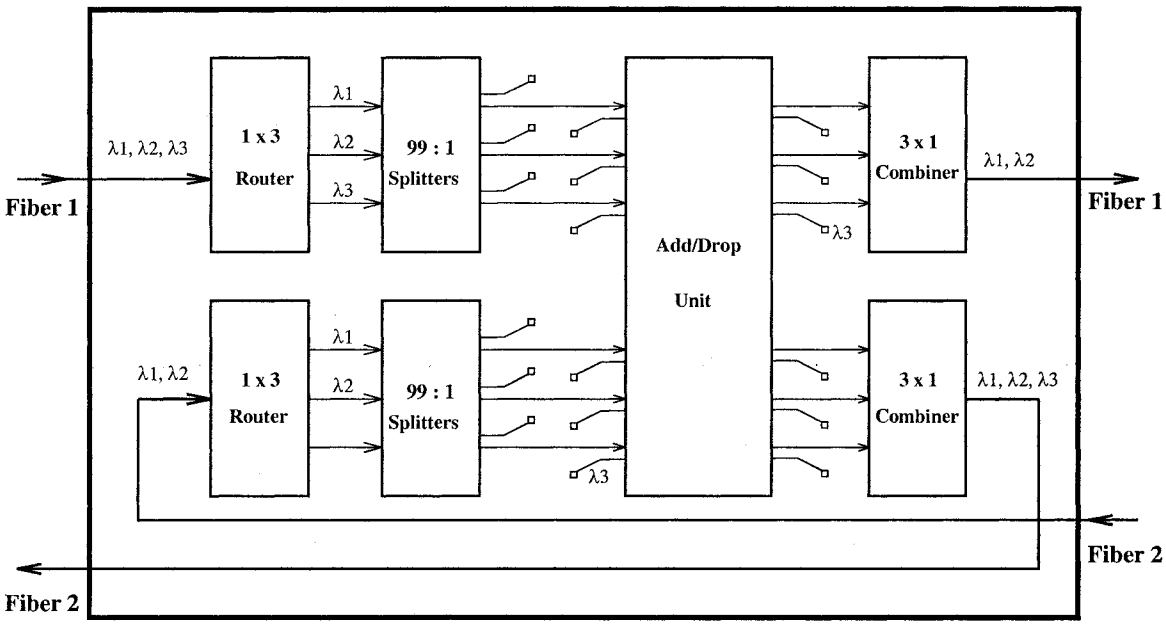


Figure 5.2: Fiber connection pattern for a ring consisting of 6 nodes

5.3 IGSWADM used in WDM Ring III Architecture

The proposed WDM Ring III architecture is based on a modified version of the IGSWADM used by the previous ring networks. The architecture of the modified IGSWADM, used by a network with 6 nodes, is illustrated in Figure 5.3. It consists of two (1 x 3) wavelength routers, two 99:1 splitters, two (3 x 1) combiners, and a reconfigurable add/drop unit. The structure and function of this IGSWADM is simpler because it handles number of wavelengths less than that in the previous IGSWADM architectures. For a network consisting of 6 nodes, the IGSWADM handles only 3 wavelengths. In this case, signals can be sensed, added to, or dropped form both fibers.

A node will add its connection request on fiber 1 when setting up connection with the N/2 succeeding nodes, and on fiber 2 when setting up connection with the N/2 preceding nodes. When a node wants to communicate with another node, it first senses the presence of the desired destination wavelength on both fibers. If this wavelength is busy (in use) on any fiber, it will be considered busy on the other fiber, and cannot be used. This is because, when a node send a connection request on a certain wavelength on one fiber, it reserve the other fiber for receiving the incoming acknowledgment on the same wavelength. For example, if node 1 was sending a connection request to node 3 on wavelength λ_3 on fiber 1, it will reserve fiber 2 for receiving the incoming acknowledgment on λ_3 . In this case, if node 2 wants to communicate with node 6, it senses that λ_3 (which assigned to node 3 and node 6) is busy on fiber 1. As a result, node 2 cannot send connection request on λ_3 to node 6 on fiber 2, because this signal interferes with the acknowledgment signal on λ_3 from node 3 to node 1 on fiber 2.



Integrated Group-Search Wavelength Add/Drop Multiplexer Device (IGSWADM)

Figure 5.3: Schematic diagram of the three wavelengths Integrated Group-Search Wavelength Add/Drop Multiplexer

This rule is not applied to the wavelength similar nodes, for example, if node 2 wants to communicate with node 5, it will sense the presence of λ_2 on both fibers, if λ_2 is busy on fiber 1, node 1 has to send its request on the same fiber 1 after dropping the incoming signal on λ_2 form fiber 1. This way λ_2 will exist on two different parts of the ring at the same time.

The next step after sensing the desired destination wavelength is the add/drop process. For example, when node 2 wants to setup a connection with node 4, and it senses the splitter output for the wavelength λ_1 to be free, it will configure its add/drop to drop any signal that will arrive later on λ_1 from any preceding node, and add its connection request on λ_1 into the traffic on fiber 1 to node 4. At the same time, it configures the add/drop unit to be ready to drop the incoming acknowledgment signal on λ_1 from node 4 on fiber 2.

5.4 Medium Access Protocol

The MAC protocol to setup a connection between two nodes, for example node 1 and node 3 is as follows: Upon the arrival of a message at node 1 destined to node 3, node 1 first senses the wavelength of node 3 (λ_3). If λ_3 is free on both fibers, node 1 sends a connection request to node 3 on λ_3 on fiber 1 and tunes its tunable receiver to channel λ_3 on fiber 2 so that it will be able to receive the incoming acknowledgment signal from node 3. The connection request (CR) signal is a periodically repeated special message contains the addresses of the sender and destination. In ideal case, this CR signal will propagate on the ring and go through the intermediate nodes to node 3. At node 3, upon receiving the connection request of node 1, it will send out an acknowledgment

signal on wavelength λ_3 on fiber 2 to propagate back to node 1. Node 1's receiver, which is tuned to λ_3, will receive the acknowledgment signal and will learn that node 3 is ready to receive its message. Node 1 will begin transmitting its message on wavelength λ_3. The same steps will be executed if node 1 wants to communicate with node 5, with a difference that node 1 will send its connection request to node 5 on fiber 2, and will wait the acknowledgment signal on fiber 1. So, depending on the rules mentioned in section 5.2 and the fiber connection pattern, the nodes send their connection requests on either fiber 1 or fiber 2.

In this protocol, all nodes perform their transmission and reception independently and asynchronously. Each node can transmit to any other node and can receive from any other node simultaneously. For example, node 2 can transmit to node 3 on λ_3, while it receives another node's transmission on its reserved wavelength λ_2 .

With this protocol, there is a possibility of deadlock. For example, if node 6 wants to setup connection with node 4 and sensed that λ_1 is free on both fibers, it will send its connection request to node 4 on λ_1 on fiber 2. In the same time if node 5 wants to setup connection with node 1 and sensed that λ_1 is free on both fibers; it will send a connection request to node 1 on λ_1 on fiber 1. In this situation, node 6 will configure its add/drop unit to drop the incoming acknowledgment on λ_1 from node 4 on fiber 1. At the same time, node 5 will configure its add/drop unit to drop the incoming acknowledgment on λ_1 from node 1 on fiber 2. As a result, node 6 will drop the connection request from node 5 to node 1, and node 5 will drop the connection request from node 6 to node 4. So, the connection requests will not reach their destinations forever, and acknowledgments will

never be sent. To overcome all these problems, the protocol has to include a timeout mechanism. If an acknowledgment is not received within certain timeout duration measured from the message arrival instant, the connection setup is aborted. The requesting node waits for a random time after which it tries to set up the same connection again.

5.5 Summary

WDM Ring III is presented and discussed. This ring architecture based on the wavelength reuse principle, the number of wavelengths used is less than the number of the network node (W=N/2). Each node is equipped with two transmitters and two receivers. In this architecture, a special fiber connection pattern is used to explain how a node connected to the succeeding and preceding nodes. The physical connection between nodes is depending on the node position on the ring. A description for an IGSWADM used with a network consists of 6 nodes and using 3 wavelengths is presented. A medium access protocol for fast circuit switching applications is proposed. In this protocol, a timeout mechanism is required to overcome the deadlock problem.

CHAPTER 6

WDM RING NETWORKS SIMULATION MODELS USING ULTRASAN

6.1 Introduction

Performance characterization of WDM networks is useful for developing strategies and architectural extensions that may improve the capability of these networks. Communication networks must be evaluated to their communication capacity, component limitations, protocol efficiency, and bottleneck analysis [11], [18], [42]. Performance of a network can be evaluated by direct measurements or by modeling. Measurements require a real network to be built, and tested. This is very difficult and costly. Instead of experimenting with a real network, a model is built and used for estimating the performance characteristics [2], [7], [8]. One of the common approaches for performance evaluation is the simulation. UltraSAN is one of the several tools for solving the simulation models. It is a graphical software tool that aids in the construction and solution of systems represented as stochastic activity networks [9], [31], [32], [35].

6.2 Stochastic Activity Networks

Stochastic Activity Network (SAN) is a tool for modeling systems graphically and mathematically. A SAN composed model for WDM Ring I is shown in Figure 6.1, its

function is explained later. SAN consists of four primitive objects: *places, activities, input gates,* and *output gates.*

Places are used to represent the "state" of the modeled system. They are represented graphically as circles (e.g., places *A*, *B*, C, *SID*, *init*, and *ACK1* shown in Figure 6.2) and may contain *tokens*. The number of tokens in a place is called the marking of that place. Token may be given different interpretations, for example, the number of tokens in place *C* represents the number of stations in the network, while that in place *A* indicates the arrival of a message.

Activities represent actions that take some specified time duration to complete. Activities are of two types, timed and instantaneous. Timed activities are represented graphically as hollow ovals. Timed activities (e.g., *arr*, *dest*, and *LTX1* shown in Figure 6.2) represent activities of the modeled system whose durations affect the system performance. Each timed activity has an activity time distribution function associated with its duration, for example, activity *arr* has an exponential distribution, while activity *dest* has a deterministic time. Instantaneous activities represent actions that complete in a very small amount of time (negligible time). They are represented as vertical lines; however, there are no instantaneous activities in Figure 6.2. When an activity completed, one token is added to each place directly connected to the output of that activity, and one token is deleted from each of the places directly connected to the input of the activity. For example, a token will be added to place *A* when activity *arr* completed, while a token is removed from place *C* when activity *fast* completed.

Case probabilities, associated with activities, represented as small circles on the right side of an activity, allow the realization of uncertainty associated with what happens

after an activity completed. For example, activity *dest*, has three cases. The choice of a particular case determine which output gate will work in the next step. If case 1 is selected this means that the function of the output gate *SOG3* will be executed. Activities are enabled if there is at least one token in each place directly connected to it, and if the predicate of each input gate connected to it holds (i.e., true).

Input gates control the enabling of activities and define the changes that will happen after an activity completion. Input gates are represented as triangles with their head point connected to the activity controlled by them. Gates *SIG1*, *SIG2*, shown in Figure 6.2, are examples for input gates. Input gates have enabling predicates, which are Boolean functions that control whether the connected activity will be enabled or not. Also, they have function that describes an action that will be taken when the connected activity completed.

Output gates are represented as triangles with their flat side connected to an activity. They are defined with a function. The function defines the changes that will happen when the connected activity completed. Gates *SOG3*, *SOG502,* shown in Figure 6.2, are examples for output gates.

It is difficult to specify large systems in terms of a single SAN model. Composed models are a hierarchical specification of SAN models. A complete model is built from one or more SAN submodels combined with "replicate" and "join" operations. The "replicate" operation replicates a submodel certain number of times. Some of the submodel places are made common among all the replicated submodels to provide a mean of communication between these submodels. For a network consisting of 6 nodes, a submodel for only one station is constructed and then replicated 6 times after selecting

some places in the submodel to be common places for the communication among these replications. The "join" operation combines different types of submodels. A set of common places is selected to provide communication among these joined submodels. For example, after a submodel for a network's medium is constructed and a submodel for the network's station is constructed and repeated 6 times, they combined together using the "join" operation. The resulting model is called the composed model.

6.3 SAN Model of WDM Ring I

The WDM Ring I network has been modeled using composed stochastic activity network models. The composed model, shown in Figure 6.1, consists of two submodels, a station submodel and a medium submodel. Form chapter 3, section 3.6, the WDM Ring I consisting of 4 nodes (stations), so, the station submodel is replicated 4 times (this is defined inside the replication operation "Rep1"). The replicated submodel and the medium submodel are joined together (this is defined inside the join operation "Join1"). The operation of the composed model is based on the medium access protocol described in chapter 3.

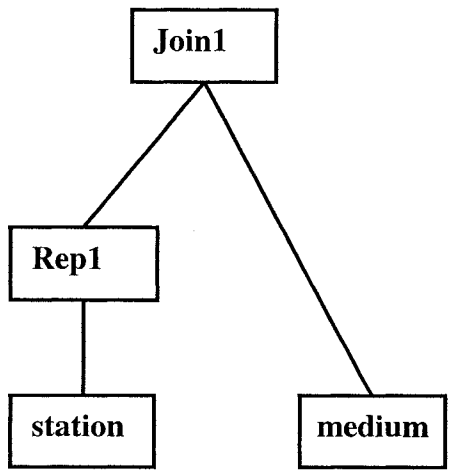


Figure 6.1: Composed Model: WDM Ring I

Figure 6.2 shows the SAN representation for the "station" submodel. In it, activity *arr* models the arrivals of new message, while place *A* represents the single buffer that holds the arrived message. The operation of the SAN composed model is described as follows: To initiate the operation of the model, the place *A* is filled with zero token (i.e., empty). This will enables the input gate *SIG1* to work, in turn, the connected activity *arr* will be enabled. After an exponentially distributed time (consider Poisson arrivals), the activity *arr* completes and a token is added to place *A* to indicate a new arrival. In turn, the input gate *SIG2* is enabled due to the existence of a token in place *A* (its predicate holds). Place *B* is originally initiated holding one token. This situation will result in activation of the activity *dest*, which has 3 cases, for selecting the message destination station in random. If case 1 is selected, the output gate *SOG3* function will be executed and a destination station is selected. IF, for example, the considered station is station number 1 (defined by place *G1*), and its message destination is station number 3 (defined by place *G3*), the output gate *SOG3* will add 3 tokens in place *G1*. Place *G1* is common place between the station submodel and the medium submodel. This means *G1* in the medium submodel will contain 3 tokens. As shown in Figure 6.3, in the medium submodel, when 3 tokens exist in place *G1*, the input gate *MIG1* is enabled and in turn the deterministic activity *s1* is initiated. After the completion of *s1*, the function of the *MOG1* will be executed and the place *L13* (which represents the status of wavelength λ_3 in station 1) is checked. If place *L13* is busy, which means that the wavelength of the destination station 3 is busy, *MOG1* will change the token in place *G1* to a number indicates that the destination is busy. The sending station 1 will wait till the place *L13* be free and *MOG1* repeats its function again. If place *L13* is free (has no token), the output

gate *MOG1* will put one token in it to indicate the existence of the wavelength of station 1 in this place. This enables the input gate *MIG13*, and in turn, initiates activity *b13* (which represents the propagation delay between station 1 and station 2). When *b13* completes, the function of out gate *MOG13* will be executed. As a result, a token will be added in place *L23* (which represents the status of wavelength λ_3 in station 2). This means that, the signal on λ_3 is propagated from station 1 to station 2 in its way to its destination. When place *L23* has a token, the input gate *MIG23* is enabled, and in turn, the activity *b23* (which represent the propagation delay between station 2 and station 3) is initiated. When *b23* completes, the function of the output gate *MOG23* is executed. As a result, a token will be added to place *L33* (which represents the status of wavelength λ_3 in station 3). When place *L33* has a token, the input gate *MIG33* is enabled, and in turn the activity *b33* (which represents the total back propagation delay between station 3 and station 1). When *b33* completed the output gate *MOG33* will but one token in the place *ACK3* that indicates that station 3 acknowledged the request of station 1. This acknowledgment will propagate back to station 1 and reaches it after time represented by the duration of the deterministic activity *b33*. The place *ACK3* is common place between the two submodels. At station submodel, shown in Figure 6.2, the existence of a token in place *ACK3* will enables the input gate *SIG3*, and in turn will initiate the activity *pretx*. When *pretx* completed, the output gate *SOG2* will generate a random number represent a message length, this number will be added to place *R1*. The tokens in place *R1*will enables the input gate *SIG4*, in turn will initiate the activity *LTX1* (its duration represents the message length). When *LTX1* completed (i.e., the message transmission completed), the output gate *SOG501* will clear the places *A*, *B*, and *TX* to allow the arrival of new

messages. In the same time, the token in place *G1* is changed to a value indicates that the transmission is finished. At the medium submodel, the new value in place *G1* will enables the input gate *MIG1*, and in turn the output gate *MOG1* will execute its function to clear the reserved connection between station 1 and station 3. Other connections setup and transmissions between different stations are established in the same described steps.

The described model was solved by the terminating simulator solver provided in UlraSAN. All the commands in the description of the model were sequences of C language statements. The network performance is estimated by using reward variables. These reward variables are defined in the model description. For example, a reward variable named *lt1* is used to count the number of completion of activity *LTX1*. This count is accumulated during the simulation run time. The accumulated count represents the number of transmissions performed by station 1 during the simulation time. Another reward variable is *t1* that counts the number of completions of activity *TX1*. The accumulated value of *t1* represents the summation of the lengths of the messages transmitted by station 1. All the performance variables specified in the simulations were estimated to a 95% level of confidence.

By using these reward variables, various measures of network performance can be calculated. For example, the network Throughput is calculated by dividing the summation of message lengths transmitted by all stations during the simulation time, by the simulation duration. The mean access delay is calculated by dividing the summation of the connection setup durations of the transmitted messages of all stations, by the summation of the number of messages transmitted by all stations during the simulation time. A detailed description for the WDM Ring I SAN model is included in appendix I.

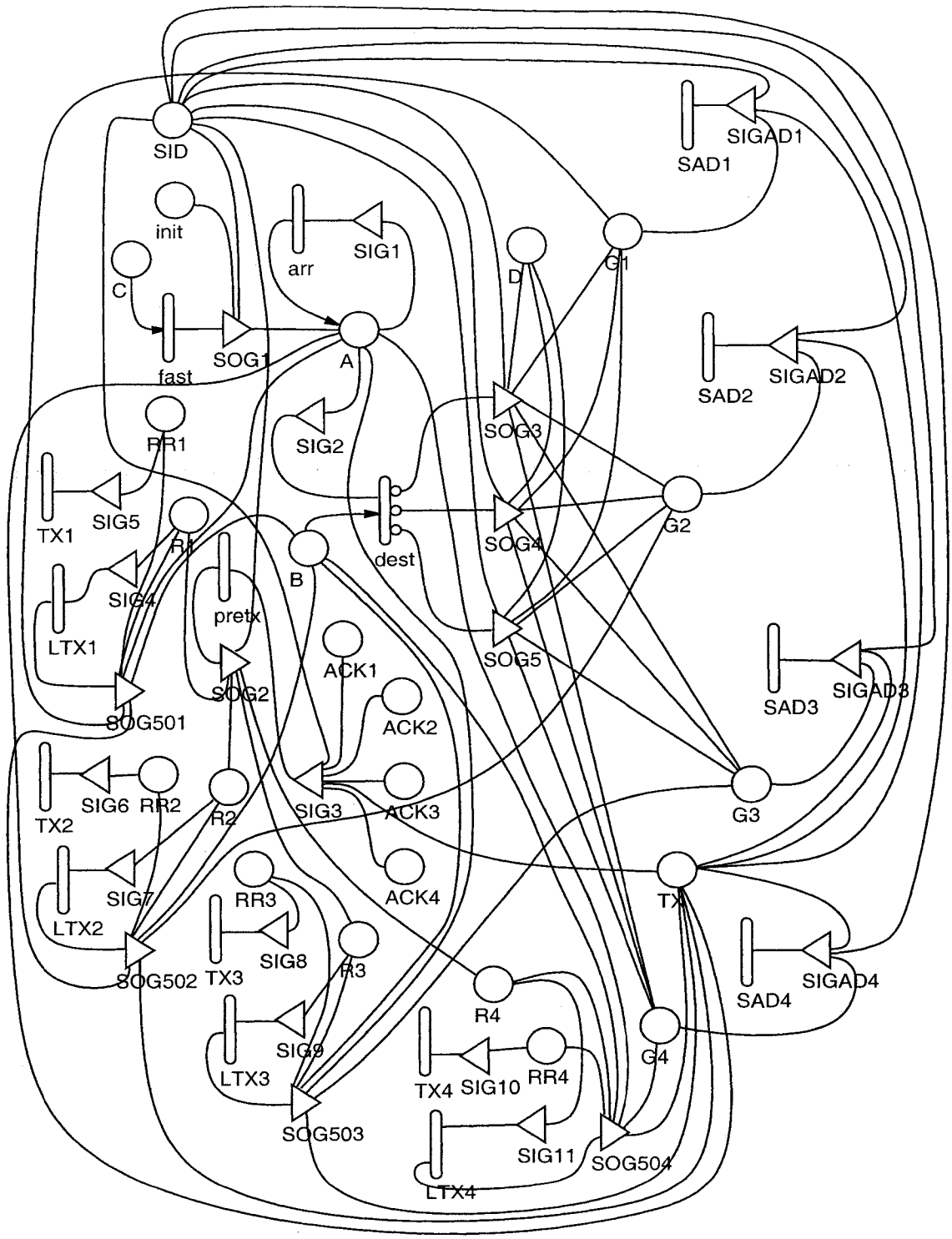


Figure 6.2: WDM Ring I's SAN submodel: station

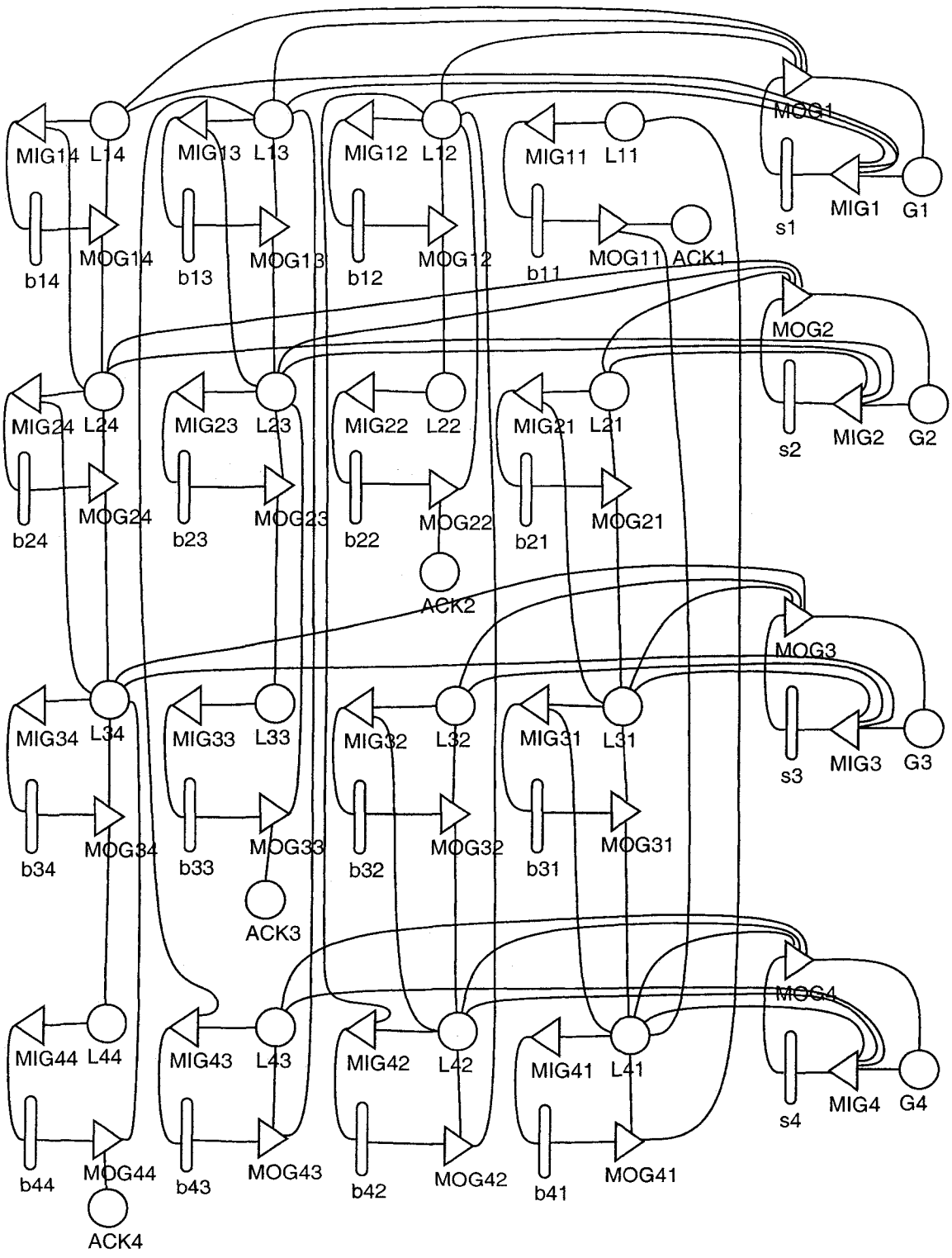


Figure 6.3: WDM Ring I's SAN submodel: medium

6. 4 SAN Model of WDM Ring II

The WDM Ring II composed SAN model consists of two submodels, a station submodel and a medium submodel. For network consisting of 6 stations, the station submodel is replicated 6 times. The replicated submodel and the medium submodel are joined together as in Figure 6.1. The operation of the composed model is based on the medium access protocol described in chapter 4.

Figure 6.4 shows the SAN representation for the "station" submodel in a network consisting of 6 stations. In this submodel, activity *arr* models the arrival of new messages, place *A* models the station buffer, activity *Tout* models the timeout duration, activity *wait* models the waiting for a random amount of time after timeout, and activity *dest* models the random selection of messages destinations. The other activities, places, and gates have a similar description as in the previous section.

The operation of the composed model is described as follows: Let us consider that station 2 wants to transmit a message to station 5. In station 2, a zero token in place *A* will enable the input gate *SIG1*, in turn, the activity *arr* will be initiated. After an exponentially distributed time the activity *arr* completes and add a token in place *A* to indicate a message arrival. As a result, the input gates *SIG2* and *SIGTO* will be enabled. The input gate *SIGTO* models the initiation of the timeout duration. In this model, the input gate *SIG2*, will be completely enabled when the receiver of station 2 (the sending station) is free. The enabling of input gate *SIG2*, will initiate the activity *dest* which has 5 cases to model the random selection of the destination. When activity *dest* completes, a case is selected and accordingly its output gate is selected. Let us consider that the destination station 5 is the selected destination. According to this selection the output gate

SOG5 will add 5 tokens in the place *G2*. This place is common between the two submodels. In the medium submodel, shown in Figure 6.5, the input gate *MIG2* will be enabled and activity *s2* will be initiated. When activity *s2* completes, the output gate *MOG2* will check the place *L25* (which represents the status of the wavelength λ_5 in station 2). If place *L25* is empty (i.e., λ_5 is free), the output gate *MOG2* will add 2 tokens in place *L25*. As a result, the input gate *MIG25* will be enabled, and activity *b25* will be initiated. When *b25* completes, the output gate *MOG25* will add 2 tokens in the place *L35* (which represents the status of wavelength λ_5 in station 3). The input gate *MIG35* will be enabled and activity *b35* will be initiated. When *b35* completes, the output gate *MOG35* will add 2 tokens in the place *L45* (which represents the status of λ_5 in station 4). The input gate *MIG45* will be enabled, and activity *b45* will be initiated. When *b45* completes, the output gate *MOG45* will add 3 tokens in the place *L55*. The input gate *MIG55* will be enabled, and activity *b55* will be initiated. When *b55* completes after a time determined according the number of token in place *L55*, the output gate *MOG55* will add 2 tokens in place *ACK5* which indicates that station 5 acknowledged the request of station 2. The place *ACK5* is a common place between the two submodels. At the station submodel, shown in Figure 6.4, the input gate *SIG3* will be enabled, and the activity *pretx* will be initiated. When *pretx* completes, the output gate *SOG2* will generate a random number representing a message length. This number will be added to the place *R2*. The input gate *SIG7* will be enabled, and activity *LTX2* (its duration represents the message length) will be initiated. When *LTX2* completes, the output gate *SOG502* will clear the places *A*, *B*, *TX*, and *L22* (which represent the status of the receiver in station 2) to allow the arrival of new messages and to handle the waiting connection requests

directed to it from other nodes . In the same time, the token in place *G2* will be changed to a value indicates that the transmission is completed. At the medium submodel, the input gate *MIG2* will be enabled, and activity *s2* will be initiated. When *s2* completes, the output gate *MOG2* will clear the reserved connection between station 2 and station 5.

Note that, during the execution of the above steps, the timeout activity *Tout* is working in parallel, such that if it is completed during the connection setup procedure (i.e., before the acknowledgment arrival to the requesting node), the connection setup is aborted. In the above scenario, if activity *Tout* in station 2 is completed before the arrival of the acknowledgment signal, the output gate *SOGTO* will clear the place *L22* (the receiver of station 2) and will change the token value in the place *G2* to indicate connection abortion. At the same time it will cause the initiation of activity *wait* to let station 2 wait a random amount of time before attempting to transmit again. When *wait* completes, station 2 will repeat all the above steps to setup the same connection again.

The terminating simulator solver provided in UltraSAN solved the composed model. The network performance measures are calculated by the same way described in the pervious section. Note that the performance of the network is affected very much by the timeout duration. This effect is explained in detail in chapter 4. A detailed description for the WDM Ring II SAN model is presented in appendix II.

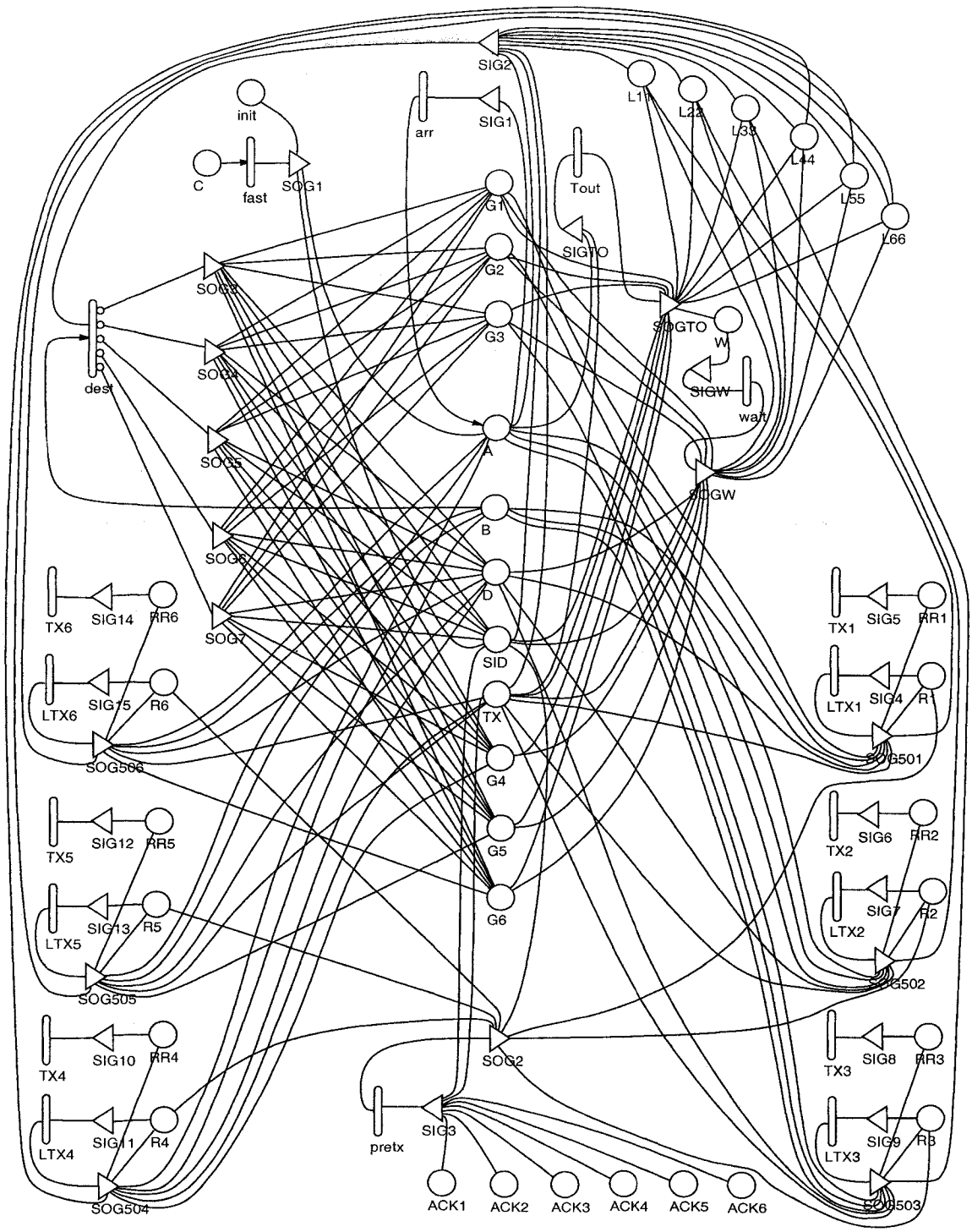


Figure 6.4: WDM Ring II's SAN submodel: station

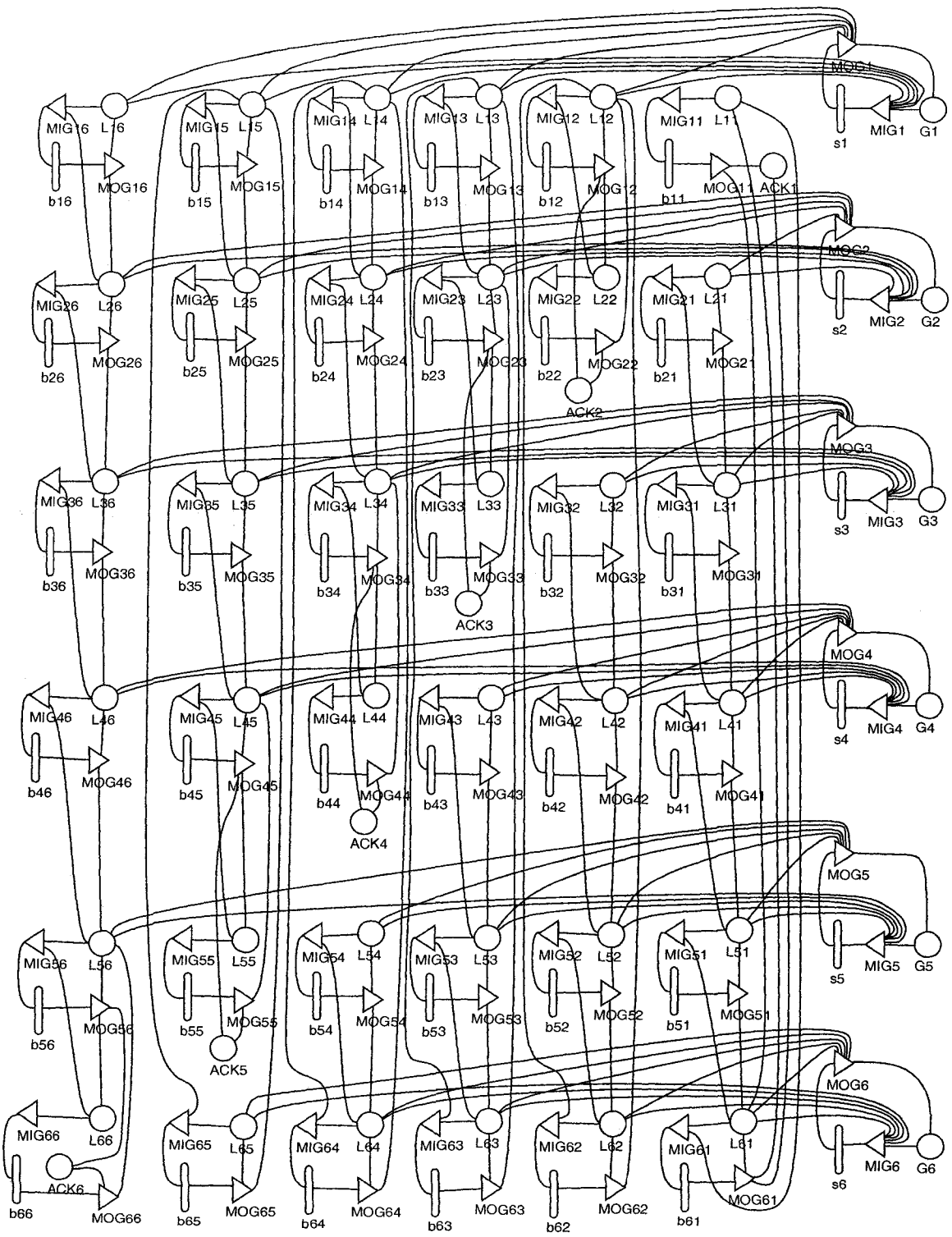


Figure 6.5: WDM Ring II's SAN submodel: medium

CHAPTER 7

CONCLUSION

This dissertation proposes three new architectures for WDM ring networks, which are suitable for a wide range of applications. The key element in the three proposed architectures named WDM Ring I, WDM Ring II, and WDM Ring III, is the Integrated Group-Search Wavelength Add/Drop Multiplexer (IGSWADM) device. Recent developments with polymer technology suggest that these devices can be fabricated economically. Since the success of the WDM networks depends heavily on the availability and the cost of the optical components which provide the functionality and flexibility needed in these networks, the IGSWADM device will become a candidate component for the optical high speed metropolitan and local area networks.

The IGSWADM device is capable of performing several functions simultaneously. These functions provide a great flexibility in the design of the WDM ring networks. The ability of this device to sense the transmission traffic for the presence of a certain wavelength allows the design of the proposed ring networks to avoid the need for a central control node and the multihop principle. In the proposed ring networks, each node has the ability to monitor the traffic before deciding to transmit. The reconfigurable add/drop unit in this device offered high flexibility in controlling the data flow on and off the designed networks and in achieving successful routing of the wavelengths.

The physical topology of the proposed ring networks is based on a double-fiber ring. The transmission on one fiber propagates in a direction opposite to that on the other fiber. The use of two fibers allows the proposed ring networks to use an easy acknowledgment mechanism. The requesting node transmits its connection setup request on one fiber and waits too the incoming acknowledgment signal on the reserved path on the other fiber. The proposed ring architectures are presented, discussed, and evaluated.

In chapter 2, an introduction for the WDM technique and its classifications are presented, the add/drop multiplexers (ADMs) are introduced. A survey of the ring networks based on WDM technique and the ADMs is provided.

In chapter 3, the WDM Ring I is introduced [62]. The operation of this ring network depends heavily on the node architectures. Each node in this ring network is equipped with two transmitters and two receivers. This node architecture allows the nodes to transmit and receive at the same time. This leads to high performance. The nodes are connected to the ring via IGSWADM devices. A medium access control protocol suitable for fast circuit switched applications is proposed for controlling the operation of the proposed network. Under this protocol, the performance of a ring consisting of 4 nodes is evaluated for different network parameters. The performance results indicate that the network has high performance at high data rates and long messages.

In chapter 4, the WDM Ring II architecture is presented and discussed [62]. This architecture is based on a simpler node construction compared to that used in WDM Ring I. Each node is equipped with only one tunable transmitter and only one fixed tuned receiver. With this node architecture a timeout mechanism is provided to overcome the

deadlock problems during the network operation. The nodes are connected to the ring through IGSWADM devices. A modified simple version of the IGSWADM, using only one splitter, is used in this architecture. A medium access protocol using a timeout mechanism is proposed for controlling the operation of the network. The limitations on the ability of each node to access the network are considered. The effects of different network parameters on the network performance are investigated and discussed. The performance is highly affected by the timeout duration. The function of the medium access protocol can be improved by the dynamic selection of the timeout duration. The optimal timeout duration is not affected by the message lengths. The network achieves higher performance for long message lengths.

In chapter 5, the WDM Ring III architecture is introduced. This architecture is based on the wavelength reuse principle. The number of wavelengths used is less than the number of nodes. Each node is equipped with two transmitters and two receivers. The communication among network nodes is based on a special fiber connection pattern to organize the physical connection between any node and its succeeding and preceding nodes. The nodes are connected to ring via IGSWADM devices. The IGSWADM used in this network handles a small number of wavelengths. A medium access protocol for fast circuit switched applications is proposed to control the operation of the network. In this protocol a timeout mechanism is provided to avoid the deadlock problems.

Table 7.1 shows a comparison among the three ring networks considering node architecture, number of used wavelengths, timeout mechanism, transmission and reception, acknowledgment mechanism, IGSWADM construction, and performance.

Table 7.1: Comparison of the three ring networks

Comparison points	WDM Ring I	WDM Ring II	WDM Ring III
Node architecture	• One tunable transmitter • One fixed transmitter • One tunable receiver • One fixed receiver	• One tunable transmitter • One fixed receiver	• One tunable transmitter • One fixed transmitter • One tunable receiver • One fixed receiver
Number of wavelengths (W) versus number of node (N)	W = N	W = N	W = N/2
Timeout mechanism	Not required	Required	Required
Transmission and reception	Each node can transmit and receive simultaneously	Each node can transmit only or receive only at a time	Each node can transmit and receive simultaneously
Acknowledgment mechanism	Destination sends ACK. signal on its wavelength	Destination sends ACK. signal on the sender wavelength	Destination sends ACK. signal on its wavelength
Transmission direction	Each node transmits to its succeeding nodes	Each node transmits to its succeeding nodes	Each node transmits to its N/2 succeeding nodes on one fiber and to its N/2 preceding nodes on the other fiber

Table 7.1 (continued): Comparison of the three ring networks

Comparison points	WDM Ring I	WDM Ring II	WDM Ring III
IGSWADM	Uses two splitters Handles N wavelengths	Uses one splitter Handles N wavelengths	Uses two splitters Handles N/2 Wavelengths
Performance	Higher performance at high arrival rates and long message lengths	Highly affected by the selected timeout duration. Higher for long message lengths	Expected to be affected by the selected timeout duration. Higher at high arrival rates and long messages

The significance of this dissertation is the presentation of three new WDM ring network architectures, and the proposal of a family of MAC control protocols that are specially designed for the IGSWADM based ring networks. The proposed network architectures overcome the severe limitation of the single channel ring networks by introducing high bandwidth multichannel ring networks. Through the concept of wavelength division multiplexing, the proposed WDM ring networks present an efficient means for implementing optical, high speed local and metropolitan area communication. The proposed ring networks are simple in construction and their implementation appears to be feasible with state of the art components. Using ring topology with these architectures allows them to be scalable and use simple control software and interfaces.

The proposed ring architectures have the advantage that they don't need for a central control node, as with the WDM ring networks presented by other researchers. The

design of the proposed networks, especially WDM Ring III, avoids the multihop technique that is used by other researchers to reduce the required number of wavelengths.

The proposal of a family of novel MAC protocols that control the operation of the proposed ring networks adds another reason for the significance of this dissertation. The proposed protocols are suitable for fast circuit switched applications.

There are several directions to extend the introduced proposals for future research. The following aspects may be considered:

- Studying the proposed architectures in more detail, by constructing and evaluating analytical models for each of the proposed ring networks.
- Compare the analytical model results with the existing simulation results for each network.
- Investigate the effects of more realistic network parameters on the network performance. For example by studying networks consisting of a larger number of nodes.
- Study the possibility of reducing the required number of wavelengths.
- Modify the proposed MAC protocols for packet switched applications.
- Study the possibility of LANs internetworking based on the proposed WDM ring networks.
- Investigate the scalability and robustness of the proposed WDM ring networks.

REFERENCES

[1] D. B. Keck, "Fundamentals of optical waveguide fibers", IEEE Communications Magazine, Vol. 23, No. 5, pp. 17 – 22, May 1985.

[2] J. L. Hammond and P. J. P. O'Reilly, "Performance Analysis of Local Computer Networks", Addison-Wesley, 1986.

[3] T. Nakagami, T. Sakurai, "Optical and Optoelectronics Devices for Optical Fiber Transmission Systems", IEEE Communications Magazine, Vol. 26, No. 1, pp. 28 – 33, Jan. 1988.

[4] P. S. Henry, "High-Capacity Lightwave Local Area Networks", IEEE Communication magazine, pp. 20 – 26, Oct. 1989.

[5] C. A. Brackett, "Dense Wavelength Division Multiplexing Networks : Principles and Applications", IEEE Journal on Selected Areas in Communications, Vol. 8, No. 6, pp. 948 – 964, Aug. 1990.

[6] N. R. Dono, P. E. Green Jr., K. Liu, R. Ramaswami, and F. Tong, "A Wavelength Division Multiple Access Network for Computer Communication", IEEE J. on Selected Areas in Communications, Vol. 8, No. 6, pp. 983 – 993, Aug. 1990.

[7] T. G. Robertazzi, "Computer Networks and Systems : Queueing Theory and Performance Evaluation", Springer-Verlag, 1990.

[8] A. M. Law and W. D. Kelton, "Simulation Modeling and Analysis", McGraw-Hill Inc., 1991.

[9] J.A. Couvillion, R. Freire, R. Johnson, W. D. Oball, M. A. Qureshi, M. Rai, W. H. Sanders, and J. E. Tvedt, "Performance Modeling with UltraSAN", IEEE Software, Vol. 8, No. 5, pp. 69 – 80, Sept. 01 1991.

[10] G. P. Agrawal, "Fiber-Optic Communication Systems", John Wiley & Sons, 1992.

[11] J. M. Senior, "Optical Fiber Communications, Principles and Practice", Prentice Hall, 1992

[12] P. E. Green, "An All-Optical Computer Network: Lessons Learned", IEEE Network, pp. 56 – 60, March 1992.

[13] P. J. Fortier, "Handbook of LAN Technology", McGrow-Hill, 1992.

[14] S. S. Wagner and T. E. Chapuran, "Multiwavelength Ring Networks for Switch Consolidation and Interconnection", pp. 1173-1179, ICC., 1992.

[15] M. I. Irshid and M. Kavehrad, "A Fully Transparent Fiber-Optic Ring Architecture For WDM Networks", Journal of Lightwave Tech., Vol. 10, No. 1, pp. 101-108, Jan. 1992.

[16] B. Mukherjee, "WDM-Based Loacl Lightwave Networks Part I : Single-Hop Systems", IEEE Network, pp. 12 – 26, May 1992.

[17] B. Mukherjee, "WDM-Based Loacl Lightwave Networks Part II : Multihop Systems", IEEE Network, pp. 20 – 31, July 1992.

[18] P. E. Green Jr., "Fiber Optic Networks", Prentice Hall, 1993

[19] W. Stallings, "Local and Metropolitan Area Networks, Macmillan", 1993.

[20] A. Ganz, and Z. Koren, "Performance and Design Evaluation of WDM Stars", IEEE J. of Lightwave Tech., Vol. 11, No. 2, pp. 358 – 366, Feb. 1993.

[21] A. F. Elrefaie, "Multiwavelength survivable ring network architectures", in ICC'93, Geneva, Switzerland, paper 48.7,May 1993.

[22] F. J. Janniello, R. Ramaswami, and G. Steinberg, "A Prototype Circuit-Switched Multi-Wavelength Optical Metropolitan-Area Networks", IEEE J. Lightwave Tech., Vol. 11, No. 5/6, pp. 777 – 782, May/June 1993.

[23] O. K. Tonguz and K. A. Falcone, "Fiber-Optic Interconnection of Local Area Networks: Physical Limitations of Topologies", Journal of Lightwave Tech., Vol. 11, No. 5/6, pp. 1040-1052, May/June 1993.

[24] K. Oda and H. Toba,"An Optical FDM-Add/Drop Multiplexing Ring Network Utilizing Fiber Fabry-Perot Filters and Optical Circulators", IEEE Photonics Tech. Letters, Vol. 5, No. 7, pp. 825-828, July 1993.

[25] B. Albert and A. P. Jayasumana, "FDDI and FDDI-II Architecture, Protocols and Performance", Artech House, 1994.

[26] J. Sauer, A. P. Jayasumana, C. Radehaus, and H. Temkin, "A Robust WDM Communication System", Proc. of IEEE Lasers and Electro-Optic Society 7th Annual Meeting (LEOS'94), Oct. 1994.

[27] D. J. G. Mestdagh, "Fundamentals of Multiaccess Optical Fiber Networks" Artech House , 1995.

[28] E. Gay, M. J. Chawki, D. H. B. Hoa, and V. Tholey, "Theoretical Simulation and Experimental Investigation on a WDM Survivable Unidirectional Open Ring Network Using Tunable Channel Selecting Receivers", J. of Lightwave Tech., Vol. 13, No. 8, pp. 1636-1647, August 1995.

[29] I. Chlamtac, A. Fumagalli, L. G. Kazovsky and P. T. Poggiolini, "A Contention/Collision Free WDM Ring Network for Multi Gigabit Packet Switched Communication", Journal of High Speed Networks, Vol. 4, pp. 201-219, 1995.

[30] R. A. Barry, and P. A. Humblet, "Models of Blocking Probability in All-Optical Networks with and without Wavelength Changers", IEEE INFOCOM, Vol. 2, pp. 402 – 412, 1995.

[31] W. H. Sanders, W. D. Obal II, M. A. Qureshi, and F. K. Widjanarko, "UltraSAN VERSION 3: ARCHITECTURE, FEATURES, AND IMPLEMENTATION", Proceedings of the AIAA Computing in Aerospace 10 Conference, San Antonio, TX, March 28-30, pp. 327-338, 1995.

[32] Center for Reliable and High Performance Computing, Coordinated Science Laboratory, University of Illinois, UltraSAN Reference Manual, 1995.

[33] Z. Zhang, and A. S. Acampora, "A Heuristic Wavelength Assignment Algorithm for Multihop WDM Networks with Wavelength Routing and Wavelength Re-Use", IEEE/ACM Transactions on Networking, Vol. 3, No. 3, pp. 281 – 288, June 1995.

[34] M. J. Minardi, and M. A. Ingram, "Adaptive Crosstalk Cancellation and Laser Frequency Drift Compensation in Dense WDM Networks", IEEE J. of Lightwave Tech., Vol. 13, No. 8, pp. 1624 – 1635, Aug. 1995.

[35] D. D. Deavours, W. D. Obal II, M. A. Qureshi, W. H. Sanders, and A. P. A. van Moorsel, " UltraSAN VERSION 3 OVERVIEW", Proceedings of the Sixth International Workshop on Petri Nets and Performance Models, Durham, NC, pp. 216-217, October 3-6, 1995.

[36] R. Ramaswami, and K. N. Sivarajan, "Routing and Wavelength Assignment in All-Optical Networks", IEEE/ACM Transactions on Networking, Vol. 3, No. 5, pp. 489 – 500, Oct. 1995.

[37] M. J. O'Mahony, "Optical Multiplexing in Fiber Networks : Progress in WDM and OTDM", IEEE Communications Magazine, pp. 82 – 88, Dec. 1995.

[38] M. Takatori, Y. Nakano, Y. Ashi, H. Fujita, M. Mizukami, and K. Itoh, "A High Performance Switch for OC-12 SONET Self-Healing Ring Networks", IEEE Journal on Selected Areas in Communications, Vol. 14, No. 2, pp. 353-361, Feb. 1996.

[39] H. Willebrand, J. Sauer, C. Radehaus, M. Sprenger, P. Kalra, A. Jayasumana, and H. Temkin, "A Wavelength Tolerant Robust Optical WDM Network based on an Intelligent Multichannel Wavelength Tracking Receiver", Proc. 2nd International Workshop on High-Speed Network Computing (HiNet 96), April 15 –19, 1996.

[40] C. A. Brackett, "Is There an Emerging Consensus on WDM Networking ?", IEEE J. of Lightwave Tech., Vol. 14, No. 6, pp. 963 – 941, June 1996.

[41] D. Guo, A. S. Acampora, "Scalable Multihop WDM Passive Ring with Optimal Wavelength Assignment and Adaptive Wavelength Routing", Journal of Lightwave Tech., Vol. 14, No. 6, pp. 1264-1277, June 1996.

[42] J. P. Jue, M. S. Borella, and B. Mukherjee, "Perfrormance Analysis of the Rainbow WDM Optical Networks Prototype", IEEE J. on Selected Areas in Communications, Vol. 14, No. 5, pp. 945 – 951, June 1996.

[43] P. E. Green, Jr., "Optical Networking Update", IEEE J. on Selected Areas in Communications, Vol. 14, No. 5, pp. 764 – 779, June 1996.

[44] R. E. Wagner, R. C. Alferness, A. A. M. Saleh, and M. S. Goodman, "MONET: Multiwavelength Optical Networking", IEEE J. of Lightwave Tech., Vol. 14, No. 6, pp. 1349-1355, June 1996.

[45] W. D. Zhong, S. Dods, J. P. R. Lacey, and R. S. Tucker, "Reconfigurable multichannel add-drop multiplexer with improved performance", Electronic Letters, Vol. 32, No. 16, pp. 1477-1478, August 1996.

[46] C. Vaishnav, M. Nieberger, A. P. Jayasumana, and J. Sauer, " Design and Performance of a Robust WDM Network", Current Developments in Optical Design and Engineering VI, SPIE 1996 International Symposium in Optical Science, Engineering and Instrumentation, pp. 370-378, Aug. 1996.

[47] S. A. Abd-Elmalak, C. Vaishnav, and A. P. Jayasumana, "Performance of a Symmetric Robust WDM Circuit-Switched Network with Token-Passing in Control Channel", Proc. of 5th International Conference on Computer Communications and Networks (IC3N), pp. 125 – 130, Oct. 1996.

[48] B. Glance, C. R. Doerr, I. P. Kaminow and R. Montagne, "Optically Restorable WDM Ring Networks Using Simple Add/Drop Circuitry", Journal of Lightwave Tech., Vol. 14, No. 11, pp. 2453-2456, Nov. 1996.

[49] S. B. Alexander, "Optical Communication Receiver Design", SPIE Optical Engineering Press, Washington USA, 1997.

[50] J. Hubner, P. Varming and M. Kristensen, "Five wavelength DFB fiber laser source for WDM systems", Electronics Letters, Vol. 33, No. 2, pp. 139 – 140, January 1997.

[51] G. Raybon, U. Koren, B. I. Miller, M. Chien, M. G. Young, R. J. Capik, K. Dreyer, and R. M. Derosier, "A Wavelength-Tunable Semiconductor Amplifier/Filter for Add/Drop Multiplexing in WDM Networks", IEEE Photonics Technology Letters, Vol. 9, No. 1, pp. 40-42, Jan. 1997.

[52] M. A. Marsan, A. Fumagalli, E. Leonardi and F. Neri, "Modelling Slotted Mutli-Channel Ring All-Optical Networks", Proceedings of the Fifth International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, pp. 146-153 , Jan. 1997.

[53] C. G. M. Vreeburg, T. Uitterdijk, Y. S. Oei, M. K. Smit, F. H. Groen, E. G. Metall, P. Demeester, and H. J. Frankena, "First InP-Based Reconfigurable Integrated Add-Drop Multiplexre" , IEEE Photonics Technology Letters, Vol. 9, No. 2, pp. 188-190, Feb. 1997.

[54] M. Sharma, H. Ibe, and T. Ozeki, "WDM Ring Network Using a Centralized Multiwavelength Light Source and Add-Drop Multiplexing Filters", J. of Lightwave Tech., Vol. 15, No. 6, pp. 917-929, June 1997.

[55] S. Y. Kim, S. B. Lee, J. Chung, S. Y. Kim, I. J. Park, J. Jeong, and S. S. Choi, " Highly Stable Optical Add/Drop Multiplexer Using Polarization Beam Splitters and Fiber Bragg Gratings", IEEE Photonics Technology Letters, Vol. 9, No. 8, pp. 1119-1121, Aug. 1997.

[56] N. Antoniodes, I. Roudas, R. E. Wagner, and S. F. Habiby, "Simulation of ASE Noise Accumulation in a Wavelength Add-Drop Multiplexer Cascade", IEEE Photonics Tech. Letters, Vol. 9, No. 9, pp. 1274-1276, Sept. 1997.

[57] R. Ramaswami, G. H. Sasaki, "Multiwavelength Optical Networks with Limited Wavelength Conversion", IEEE INFOCOM, Vol. 2, pp. 489 – 498, 1997.

[58] S. D. Dods, J. P. R. Lacey, and R. S. Tucker, "Homodyne Crosstalk in WDM Ring and Bus Networks", IEEE Photonics Tech. Letters, Vol. 9, No. 9, pp. 1285-1287, Sept. 1997.

[59] T. S. El-Bawab, C. H. Vaishnav, A. P. Jayasumana, H. Temkin, J. Sauer, and H. Willebrand, "Robust Wavelength Division Multiplexed Local Area Networks", Fiber and Integrated Optics, Vol. 16, No. 3, pp. 237 - 260, 1997.

[60] R. D. T. Lauder, J. M. Badcock, W. T. Holloway, and D. D. Sampson, "WDM Ring Network Employing a Shared Multiwavelength Incoherent Source", IEEE Photonics Technology Letters, Vol. 10, No. 2, Feb. 1998.

[61] K.-P. Ho and S.-K. Liaw, "Eight-channel bidirectional WDM add/drop multiplexer", Electronics Letters, Vol. 34, No. 10, pp. 947-948, May 1998.

[62] Abdel-Rahman M. Abdalla and Anura P. Jayasumana, "A WDM Ring Architecture Based on Integrated Group-Search Wavelength Add/Drop Multiplexers", Proceedings of the International Conference on Telecommunications ICT'98, Vol. III, Porto Carras, Greece, pp. 494-498, June 1998.

[63] D. A. Stamper, "Local Area Networks", Addison Wesley Longman, Inc., 1998.

APPENDIX I

DOCUMENTATION OF SAN MODEL OF WDM RING I

Documentation for Composed Model: *r4nodes.*

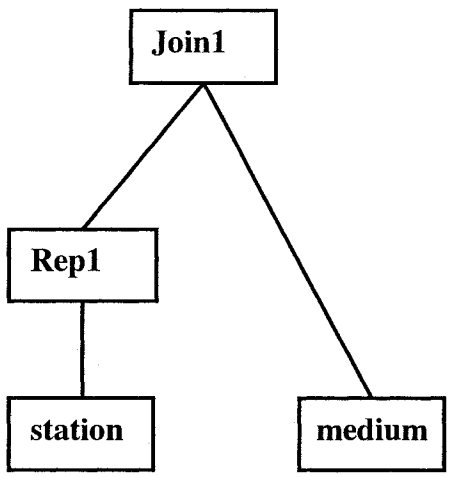


Figure I.1: Composed Model: *r4nodes*

Table I.1: Composed Model Rep Node Definitions for Project *r4nodes*

Node	**Reps**	**Common Places**
Rep1	4	ACK1
		ACK2
		ACK3
		ACK4
		C
		G1
		G2
		G3
		G4
		init

Table I.2: Composed Model Join Node Definitions for Project *r4nodes*

Node	**Common Places**		
Join1	Subtree	1	2
	ACK1	√	√
	ACK2	√	√
	ACK3	√	√
	ACK4	√	√
	G1	√	√
	G2	√	√
	G3	√	√
	G4	√	√

Table I.3: Reward Variable Definitions for Project *r4nodes*

Variable	***Definition***
lt1	Rate rewards none
	Impulse rewards Subnet = station activity = *LTX1*,value = *1*
	Simulator statistics Estimate mean Confidence Level = *0.95* Relative Confidence Interval = *0.10* Initial Transient = *1000* Batch Size =*1000* Variable type = *Interval of Time* Start of Interval = *0* Length of Interval = *1000*
lt2	Rate rewards none
	Impulse rewards Subnet = station activity = *LTX2*,value = *1*
	Simulator statistics Estimate mean Confidence Level = *0.95* Relative Confidence Interval = *0.10* Initial Transient = *1000* Batch Size =*1000* Variable type = *Interval of Time* Start of Interval = *0* Length of Interval = *1000*

Table I.3 (continue): Reward Variable Definitions for Project *r4nodes*

Variable	***Definition***
lt3	Rate rewards none
	Impulse rewards Subnet = station activity = *LTX3*,value = *1*
	Simulator statistics Estimate mean Confidence Level = *0.95* Relative Confidence Interval = *0.10* Initial Transient = *1000* Batch Size = *1000* Variable type = *Interval of Time* Start of Interval = *0* Length of Interval = *1000*
lt4	Rate rewards none
	Impulse rewards Subnet = station activity = *LTX4*,value = *1*
	Simulator statistics Estimate mean Confidence Level = *0.95* Relative Confidence Interval = *0.10* Initial Transient = *1000* Batch Size = *1000* Variable type = *Interval of Time* Start of Interval = *0* Length of Interval = *1000*

Table I.3 (continue): Reward Variable Definitions for Project *r4nodes*

Variable	***Definition***
t1	Rate rewards none
	Impulse rewards Subnet = station activity = *TX1*,value = *30000*
	Simulator statistics Estimate mean Confidence Level = *0.95* Relative Confidence Interval = *0.10* Initial Transient = *1000* Batch Size = *1000* Variable type = *Interval of Time* Start of Interval = *0* Length of Interval = *1000*
t2	Rate rewards none
	Impulse rewards Subnet = station activity = *TX2*,value = *30000*
	Simulator statistics Estimate mean Confidence Level = *0.95* Relative Confidence Interval = *0.10* Initial Transient = *1000* Batch Size = *1000* Variable type = *Interval of Time* Start of Interval = *0* Length of Interval = *1000*

Table I.3 (continue): Reward Variable Definitions for Project *r4nodes*

Variable	***Definition***
t3	Rate rewards none
	Impulse rewards Subnet = station activity = *TX3*,value = *30000*
	Simulator statistics Estimate mean Confidence Level = *0.95* Relative Confidence Interval = *0.10* Initial Transient = *1000* Batch Size = *1000* Variable type = *Interval of Time* Start of Interval = *0* Length of Interval = *1000*
t4	Rate rewards none
	Impulse rewards Subnet = station activity = *TX4*,value = *30000*
	Simulator statistics Estimate mean Confidence Level = *0.95* Relative Confidence Interval = *0.10* Initial Transient = *1000* Batch Size = *1000* Variable type = *Interval of Time* Start of Interval = *0* Length of Interval = *1000*

Table I.3 (continue): Reward Variable Definitions for Project *r4nodes*

Variable	***Definition***
rrr1	Rate rewards Subnet = station Predicate: *1* Function: *MARK(RR1)*
	Impulse rewards none
	Simulator statistics Estimate mean Confidence Level = *0.95* Relative Confidence Interval = *0.10* Initial Transient = *1000* Batch Size = *1000* Variable type = *Instant of Time* Start of Interval = *1000* Length of Interval = *1000*
rrr2	Rate rewards Subnet = station Predicate: *1* Function: *MARK(RR2)*
	Impulse rewards none
	Simulator statistics Estimate mean Confidence Level = *0.95* Relative Confidence Interval = *0.10* Initial Transient = *1000* Batch Size = *1000* Variable type = *Instant of Time* Start of Interval = *1000* Length of Interval = *1000*

Table I.3 (continue): Reward Variable Definitions for Project *r4nodes*

Variable	***Definition***
rrr3	Rate rewards Subnet = station Predicate: *1* Function: *MARK(RR3)*
	Impulse rewards none
	Simulator statistics Estimate mean Confidence Level = *0.95* Relative Confidence Interval = *0.10* Initial Transient = *1000* Batch Size = *1000* Variable type = *Instant of Time* Start of Interval = *1000* Length of Interval = *1000*
rrr4	Rate rewards Subnet = station Predicate: *1* Function: *MARK(RR4)*
	Impulse rewards none
	Simulator statistics Estimate mean Confidence Level = *0.95* Relative Confidence Interval = *0.10* Initial Transient = *1000* Batch Size = *1000* Variable type = *Instant of Time* Start of Interval = *1000* Length of Interval = *1000*

Table I.3 (continue): Reward Variable Definitions for Project *r4nodes*

Variable	***Definition***
ad1	Rate rewards none
	Impulse rewards Subnet = station activity = *SAD1*,value = *1*
	Simulator statistics Estimate mean Confidence Level = *0.95* Relative Confidence Interval = *0.10* Initial Transient = *1000* Batch Size = *1000* Variable type = *Interval of Time* Start of Interval = *0* Length of Interval = *1000*
ad2	Rate rewards None
	Impulse rewards Subnet = station activity = *SAD2*,value = *1*
	Simulator statistics Estimate mean Confidence Level = *0.95* Relative Confidence Interval = *0.10* Initial Transient = *1000* Batch Size = *1000* Variable type = *Interval of Time* Start of Interval = *0* Length of Interval = *1000*

Table I.3 (continue): Reward Variable Definitions for Project *r4nodes*

Variable	***Definition***
ad3	Rate rewards none
	Impulse rewards Subnet = station activity = *SAD3*,value = *1*
	Simulator statistics Estimate mean Confidence Level = *0.95* Relative Confidence Interval = *0.10* Initial Transient = *1000* Batch Size = *1000* Variable type = *Interval of Time* Start of Interval = *0* Length of Interval = *1000*
ad4	Rate rewards None
	Impulse rewards Subnet = station activity = *SAD4*,value = *1*
	Simulator statistics Estimate mean Confidence Level = *0.95* Relative Confidence Interval = *0.10* Initial Transient = *1000* Batch Size = *1000* Variable type = *Interval of Time* Start of Interval = *0* Length of Interval = *1000*

Table I.4: Study Editor Set Definitions for Project *r4nodes*

Study	Experiment	Variable	Type	Value
st3	exp1	arrivrate	double	10
	exp2	arrivrate	double	50
	exp3	arrivrate	double	100
	exp4	arrivrate	double	200
	exp5	arrivrate	double	300
	exp6	arrivrate	double	500
	exp7	arrivrate	double	700
	exp8	arrivrate	double	1000
	exp9	arrivrate	double	1500
	exp10	arrivrate	double	2000
	exp11	arrivrate	double	3000
	exp12	arrivrate	double	4000
	exp13	arrivrate	double	5000
	exp14	arrivrate	double	6000

Table I.4 (continue): Study Editor Set Definitions for Project *r4nodes*

Study	Experiment	Variable	Type	Value
st2	exp1	arrivrate	double	10
	exp2	arrivrate	double	50
	exp3	arrivrate	double	100
	exp4	arrivrate	double	200
	exp5	arrivrate	double	300
	exp6	arrivrate	double	500
	exp7	arrivrate	double	700
	exp8	arrivrate	double	1000
	exp9	arrivrate	double	1500
	exp10	arrivrate	double	2000
	exp11	arrivrate	double	3000
	exp12	arrivrate	double	4000
	exp13	arrivrate	double	5000

Table I.5: Non-zero or Variable Initial Markings for SAN Model *station*

Place	*Marking*
B	*1*
C	*4*
init	*1*

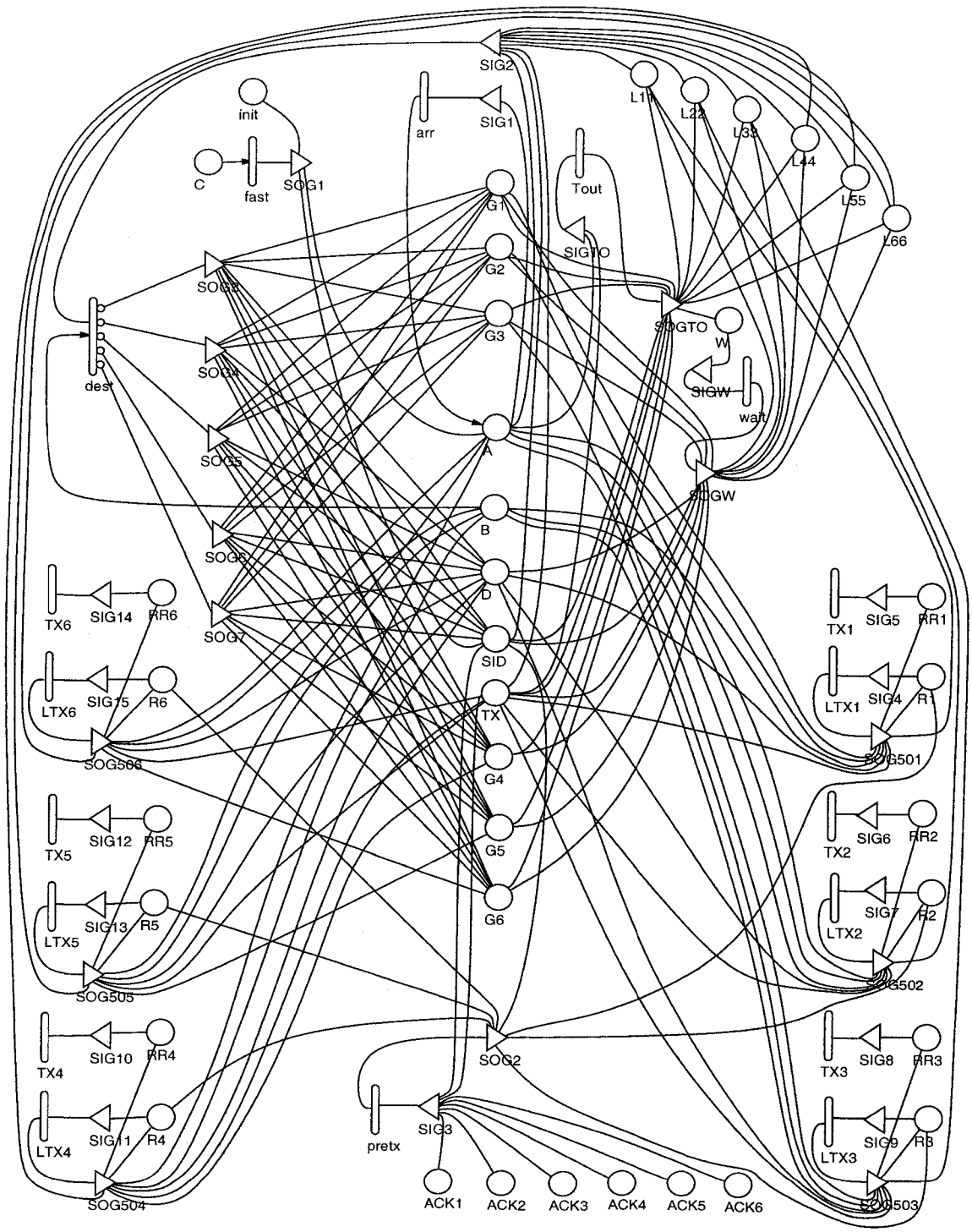


Figure I.2: SAN Model: *station*

Table I.6: Activity Time Distributions for SAN Model *station*

Activity	***Distribution***	***Parameter values***
LTX1	deterministic	
	value	*MARK(R1) * 0.0005*
LTX2	deterministic	
	value	*MARK(R2) * 0.0005*
LTX3	deterministic	
	value	*MARK(R3) * 0.0005*
LTX4	deterministic	
	value	*MARK(R4) * 0.0005*
SAD1	deterministic	
	value	*0.0002*
SAD2	deterministic	
	value	*0.0002*
SAD3	deterministic	
	value	*0.0002*
SAD4	deterministic	
	value	*0.0002*
TX1	deterministic	
	value	*0.000000000000001*
TX2	deterministic	
	value	*0.000000000000001*
TX3	deterministic	
	value	*0.000000000000001*
TX4	deterministic	
	value	*0.000000000000001*
arr	exponential	
	rate	*GLOBAL_D(arrivrate)*
dest	deterministic	
	value	*0.000000000000001*
fast	deterministic	
	value	*0.000000000000001*
pretx	deterministic	
	value	*0.00000000000001*

Table I.7: Activity Case Probabilities for SAN Model *station*

Activity	*Case*	*Probability*
dest	1	*0.33*
	2	*0.33*
	3	*0.34*

Table I.8: Input Gate Definitions for SAN Model *station*

Gate	*Definition*
SIG1	Predicate *MARK(A)< 1*
	Function *identity*
SIG10	Predicate *MARK(RR4)>= 30000*
	Function *MARK(RR4)=0 ;*
SIG11	Predicate *MARK(R4)>0*
	Function *identity*
SIG2	Predicate *MARK(A) == 1*
	Function *identity*
SIG3	Predicate *(MARK(SID)==MARK(ACK1) || MARK(SID)==MARK(ACK2) || MARK(SID)==MARK(ACK3) || MARK(SID)==MARK(ACK4) ||*
	Function *MARK(TX)=1;* *if (MARK(SID)==MARK(ACK1)) { MARK(ACK1)=0 ; }* *else if (MARK(SID)==MARK(ACK2)) { MARK(ACK2)=0; }* *else if (MARK(SID)==MARK(ACK3)) { MARK(ACK3)=0; }* *else if (MARK(SID)==MARK(ACK4)) { MARK(ACK4)=0; }*
SIG4	Predicate *MARK(R1) >0*
	Function *identity*

Table I.8 (continue): Input Gate Definitions for SAN Model *station*

Gate	***Definition***
SIG5	Predicate *MARK(RR1)>=30000*
	Function *MARK(RR1) = 0 ;*
SIG6	Predicate *MARK(RR2)>=30000*
	Function *MARK(RR2) = 0 ;*
SIG7	Predicate *MARK(R2) >0*
	Function *identity*
SIG8	Predicate *MARK(RR3)>=30000*
	Function *MARK(RR3) = 0 ;*
SIG9	Predicate *MARK(R3) >0*
	Function *identity*
SIGAD1	Predicate *MARK(G1) >0 && MARK(G1) !=500 && MARK(TX) == 0 && MARK(SID)==1*
	Function *identity*
SIGAD2	Predicate *MARK(G2) >0 && MARK(G2) !=500 && MARK(TX) == 0 && MARK(SID)==2*
	Function *identity*
SIGAD3	Predicate *MARK(G3) >0 && MARK(G3) !=500 && MARK(TX) == 0 && MARK(SID)==3*
	Function *identity*
SIGAD4	Predicate *MARK(G4) >0 && MARK(G4) !=500 && MARK(TX) == 0 && MARK(SID)==4*
	Function *identity*

Table I.9: Output Gate Definitions for SAN Model *station*

Gate	*Definition*
SOG1	*if (MARK(SID)==0) {MARK(SID)=MARK(init); MARK(init)++;* *MARK(A)=0;}*
SOG2	*Double TTT;* */* time_t ttt; */* *int ttt;* *int dum;* *srand((unsigned)time(&ttt));* *TTT=rand() / 32768.0;* *if (TTT < 0.000000001) { TTT=0.000000001;}* *dum=1+(int)((log(TTT))/(log(0.99)));* *if (MARK(SID)==1) {MARK(R1)=dum;}* *else if (MARK(SID)==2) {MARK(R2)=dum;}* *else if (MARK(SID)==3) {MARK(R3)=dum;}* *else if (MARK(SID)==4) {MARK(R4)=dum;}*
SOG3	*if (MARK(SID)==1) {MARK(D)=MARK(SID)+1;* *MARK(G1)=MARK(D);}* *else if (MARK(SID)==2) {MARK(D)=MARK(SID)+1;* *MARK(G2)=MARK(D);}* *else if (MARK(SID)==3) {MARK(D)=MARK(SID)+1;* *MARK(G3)=MARK(D);}* *else if (MARK(SID)==4) {MARK(D)=MARK(SID)-3 ;* *MARK(G4)=MARK(D);}*
SOG4	*if (MARK(SID)==1) {MARK(D)=MARK(SID)+2;* *MARK(G1)=MARK(D);}* *else if (MARK(SID)==2) {MARK(D)=MARK(SID)+2;* *MARK(G2)=MARK(D);}* *else if (MARK(SID)==3) {MARK(D)=MARK(SID)+2;* *MARK(G3)=MARK(D);}* *else if (MARK(SID)==4) {MARK(D)=MARK(SID)-2;* *MARK(G4)=MARK(D);}*
SOG5	*if (MARK(SID)==1) {MARK(D)=MARK(SID)+3;* *MARK(G1)=MARK(D);}* *else if (MARK(SID)==2) {MARK(D)=MARK(SID)-1;* *MARK(G2)=MARK(D);}* *else if (MARK(SID)==3) {MARK(D)=MARK(SID -1;* *MARK(G3)=MARK(D);}* *else if (MARK(SID)==4) {MARK(D)=MARK(SID)-1;* *MARK(G4)=MARK(D);}*

Table I.9 (continue): Output Gate Definitions for SAN Model *station*

Gate	***Definition***
SOG501	*MARK(G1)=500; MARK(A)=0; MARK(B)=1; MARK(TX)=0; MARK(RR1)=MARK(RR1)+MARK(R1); MARK(R1)=0;*
SOG502	*MARK(G2)=500; MARK(A)=0; MARK(B)=1; MARK(TX)=0; MARK(RR2)=MARK(RR2)+MARK(R2); MARK(R2)=0;*
SOG503	*MARK(G3)=500; MARK(A)=0; MARK(B)=1; MARK(TX)=0; MARK(RR3)=MARK(RR3)+MARK(R3); MARK(R3)=0;*
SOG504	*MARK(G4)=500; MARK(A)=0; MARK(B)=1; MARK(TX)=0; MARK(RR4)=MARK(RR4)+MARK(R4); MARK(R4)=0;*

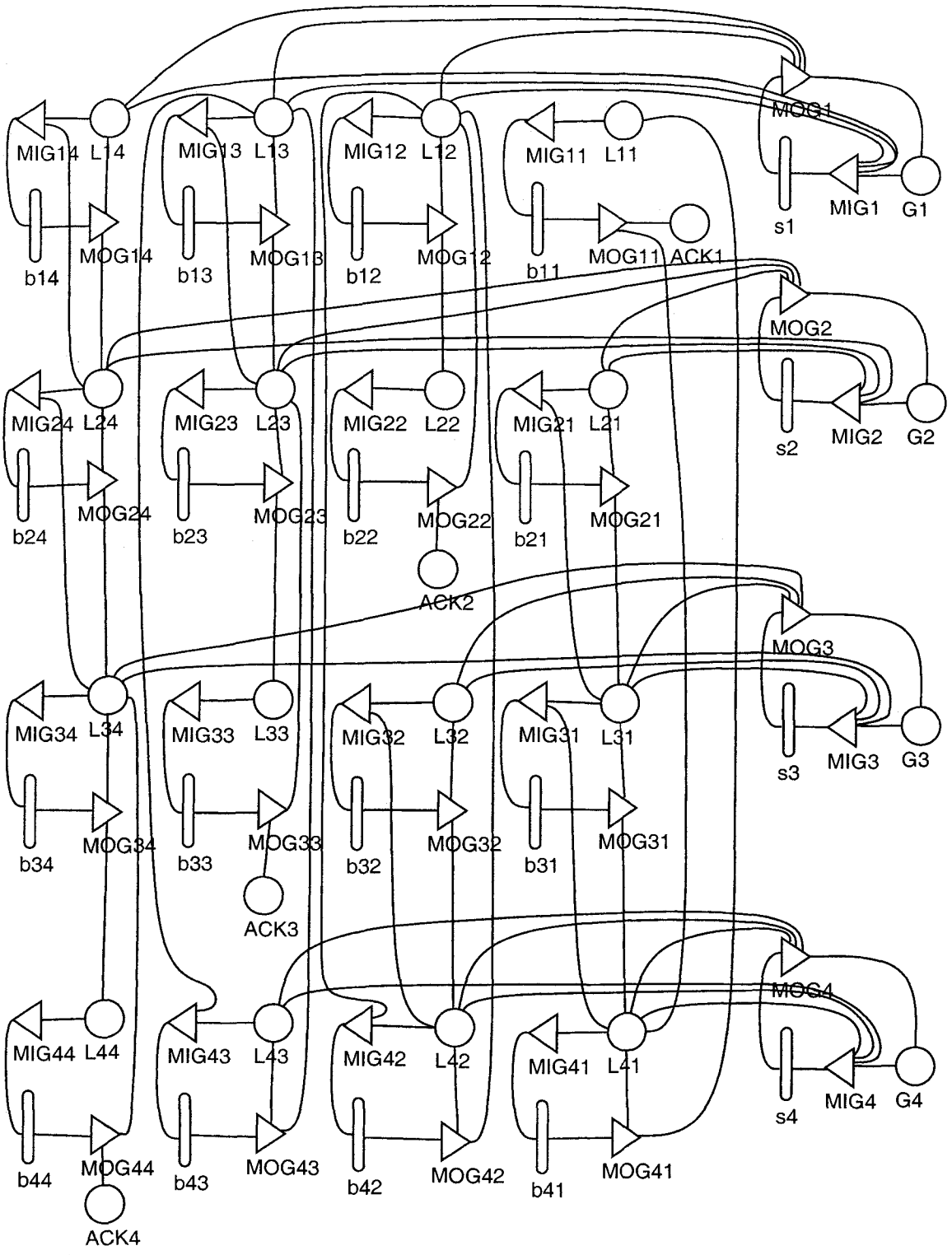


Figure I.3: SAN Model: *medium*

Table I.10: Activity Time Distributions for SAN Model *medium*

Activity	*Distribution*	*Parameter values*
b11	deterministic	
	value	*MARK(L11)*0.00001*
b12	deterministic	
	value	*0.00001*
b13	deterministic	
	value	*0.00001*
b14	deterministic	
	value	*0.00001*
b21	Deterministic	
	value	*0.00001*
b22	Deterministic	
	value	*MARK(L22)*0.00001*
b23	Deterministic	
	value	*0.00001*
b24	Deterministic	
	value	*0.00001*
b31	Deterministic	
	value	*0.00001*
b32	Deterministic	
	value	*0.00001*
b33	Deterministic	
	value	*MARK(L33)*0.00001*
b34	Deterministic	
	value	*0.00001*
b41	Deterministic	
	value	*0.00001*
b42	Deterministic	
	value	*0.00001*
b43	Deterministic	
	value	*0.00001*
b44	Deterministic	
	value	*MARK(L44)*0.00001*
s1	deterministic	
	value	*0.000000000000001*
s2	deterministic	
	value	*0.000000000000001*
s3	deterministic	
	value	*0.000000000000001*
s4	deterministic	
	value	*0.000000000000001*

Table I.11: Input Gate Definitions for SAN Model medium

Gate	*Definition*
MIG1	Predicate *MARK(G1)==2 || MARK(G1)==3 || MARK(G1)==4 || || MARK(G1)==500* *|| (MARK(G1)==12 && MARK(L12)==0)* *|| (MARK(G1)==13 && MARK(L13)==0)* *|| (MARK(G1)==14 && MARK(L14)==0)*
	Function *identity*
MIG11	Predicate *MARK(L11)==1 || MARK(L11)==2 || MARK(L11)==3*
	Function *MARK(L11)=123 ;*
MIG12	Predicate *MARK(L12)==1 || MARK(L12)==3 || MARK(L12)==4 || || MARK(L12)==500*
	Function *identity*
MIG13	Predicate *MARK(L13)==1 || MARK(L13)==4 || MARK(L13)==500* *|| (MARK(L13)==101 && MARK(L23)==0)* *|| (MARK(L13)==104 && MARK(L23)==0)*
	Function *identity*
MIG14	Predicate *MARK(L14)==1 || MARK(L14)==500 || (MARK(L14)==101 && MARK(L24)==0)*
	Function *identity*
MIG2	Predicate *MARK(G2)==1 || MARK(G2)==3 || MARK(G2)==4 || MARK(G2)==500* *|| (MARK(G2)==21 && MARK(L21)==0)* *|| (MARK(G2)==23 && MARK(L23)==0)* *|| (MARK(G2)==24 && MARK(L24)==0)*
	Function *identity*

Table I.11 (continue): Input Gate Definitions for SAN Model medium

Gate	***Definition***
MIG21	Predicate *MARK(L21)==2 || MARK(L21)==500* *|| (MARK(L21)==102 && MARK(L31)==0)*
	Function *identity*
MIG22	Predicate *MARK(L22)==1 || MARK(L22)==2 || MARK(L22)==3*
	Function *MARK(L22)=123;*
MIG23	Predicate *MARK(L23)==1 || MARK(L23)==2 || MARK(L23)==4 || MARK(L23)==500*
	Function *identity*
MIG24	Predicate *MARK(L24)==1 || MARK(L24)==2 || MARK(L24)==500* *|| (MARK(L24)==101 && MARK(L34)==0)* *|| (MARK(L24)==102 && MARK(L34)==0)*
	Function *identity*
MIG3	Predicate *MARK (G3)= =1 || MARK(G3)= =2 || MARK(G3)= =4 || MARK(G3)= =500* *||(MARK(G3)= =31 && MARK(L31)= =0)* *||(MARK(G3)= =32 && MARK(L32)= =0)* *||(MARK(G3)= =34 && MARK(L34)= =0)*
	Function *identity*
MIG31	Predicate *MARK(L31)==2 || MARK(L31)==3|| MARK(L31)==500* *||(MARK(L31)= =102 && MARK(L41)= =0)* *||(MARK(L31)= =103 && MARK(L41)= =0)*
	Function *identity*
MIG32	Predicate *MARK(L32)= =3|| MARK(L32)= =500* *|| (MARK(L32)= =103&& MARK(L42)= =0)*
	Function *identity*

Table I.11 (continue): Input Gate Definitions for SAN Model medium

Gate	***Definition***
MIG33	Predicate *MARK(L33)= =1||MARK(L33)= =2|| MARK(L33)= =3||*
	Function *MARK(L33)=123;*
MIG34	Predicate *MARK(L34)= =1 || MARK(L34)= =2|| MARK(L34)= =3|| MARK(L34)= =500*
	Function *identity*
MIG4	Predicate *MARK(G4)==1|| MARK(G4)==2|| MARK(G4)==3|| || MARK(G4)==500 ||(MARK(G4)==41&& MARK(L41)==0) ||(MARK(G4)==42&& MARK(L42)==0) ||(MARK(G4)==43&& MARK(L43)==0)*
	Function *identity*
MIG41	Predicate *MARK(L41)==2||MARK(L41)==3|| MARK(L41)==4|| MARK(L41)==500*
	Function *identity*
MIG42	Predicate *MARK(L42)==3|| MARK(L42)==4|| MARK(L42)==500|| ||(MARK(L42)==103&& MARK(L12)==0) ||(MARK(L42)==104&& MARK(L12)==0)*
	Function *identity*
MIG43	Predicate *MARK(L43)==4|| MARK(L43)==500 ||(MARK(L43)==104&& MARK(L13)==0)*
	Function *identity*
MIG44	Predicate *MARK(L44)==1|| MARK(L44)==2|| MARK(L44)==3||*
	Function *MARK(L44)=123;*

Table I.12: Output Gate Definitions for SAN Model *medium*

Gate	*Definition*
MOG1	*if(MARK(G1)==2|| MARK(G1)==12){ if (MARK(L12)==0) {MARK(L12)=1; MARK(G1)=200;} else{MARK(G1)=12;} } else if (MARK(G1)==3 ||MARK(G1)==13){ if (MARK(L13)==0) {MARK(L13)=1; MARK(G1)=200;} else{MARK(G1)=13;} } if (MARK(G1)==4|| MARK(G1)==14){ if (MARK(L14)==0) {MARK(L14)=1; MARK(G1)=200;} else{MARK(G1)=14;} } else if (MARK(G1)==500) { if (MARK(L12)==12) {MARK(L12)=500; MARK(G1)=0;} if (MARK(L13)==13) {MARK(L13)=500; MARK(G1)=0;} if (MARK(L14)==14) {MARK(L14)=500; MARK(G1)=0;}*
MOG11	*MARK(ACK1)=MARK(L41)-110; if (MARK(L41)==14) { MARK(L41)=41;}*
MOG12	*if (MARK(L12)==1) { MARK(L22)=1; MARK(L12)=MARK(L12)+10;} else (MARK(L12)==4) {MARK(L22)=2; MARK(L12)=MARK(L12)+10;} else (MARK(L12)==3) {MARK(L22)=3; MARK(L12)=MARK(L12)+10;} if (MARK(L22)==123) {MARK(L22)=0; MARK(L12)=0;}*
MOG13	*if (MARK(L23)==0) { if (MARK(L13)==1 || MARK(L13)==101) {MARK(L23)=1; MARK(L13)=13;} else if (MARK(L13)==4 || MARK(L13)==104) { MARK(L23)=4; MARK(L13)=43;} } else if (MARK(L23)==11 ||MARK(L23)==14) {MARK(L23)=500; MARK(L13)=0;} else {MARK(L13)=MARK(L13)+100;} }*
MOG14	*if (MARK(L24)==0) {MARK(L24)=1; MARK(L14)=14;} else if (MARK(L24)==14) {MARK(L24)=500; MARK(L14)=0;} else {MARK(L14)=MARK(L14)+100;}*
MOG2	*if (MARK(G2)==1 || MARK(G2)==21){ if (MARK(L21)==0) {MARK(L21)=2; MARK(G2)=200; } else {MARK(G2)=21;} } else if (MARK(G2)==3 || MARK(G2)==23){ if (MARK(L23)==0) {MARK(L23)=2; MARK(G2)=200;} else {MARK(G2)=23;} } else if (MARK(G2)==4 || MARK(G2)==24) { if (MARK(L24)==0) {MARK(L24)=2; MARK(G2)=200;} else {MARK(G2)=24;} } else if (MARK(G2)==500) { if (MARK(L21)==21) {MARK(L21)=500; MARK(G2)=0;} if (MARK(L23)==23) {MARK(L23)=500; MARK(G2)=0;} if (MARK(L24)==24) {MARK(L24)=500; MARK(G2)=0;} }*
MOG21	*if (MARK(L31)==0) { MARK(L31)=2; MARK(L21)=21;} else if (MARK(L31)==21) {MARK(L31)=500; MARK(L21)=0;} else{MARK(L21)=MARK(L21)+100;}}*

Table I.12 (continue): Output Gate Definitions for SAN Model *medium*

Gate	*Definition*
MOG22	*MARK(ACK2)=MARK(L12)-10;* *if (MARK(L12)==11) {MARK(L12)=12;}*
MOG23	*if (MARK(L23)==1) {MARK(L33)=2; MARK(L23)=MARK(L23)+10; }* *else if(MARK(L23)==2){MARK(L33)=1; MARK(L23)=MARK(L23)+10; }* *else if (MARK(L23)==4){MARK(L33)=3; MARK(L23)=MARK(L23)+10; }* *if (MARK(L33)==123) {MARK(L33)=0; MARK(L23)=0;}*
MOG24	*if (MARK(L34)==0) { if (MARK(L24)==1 || MARK(L24)==101)* *{MARK(L34)=1; MARK(L24)=14;}* *else if (MARK(L24)==2 || MARK(L24)==102) {MARK(L34)=2;* *MARK(L24)=24;}* *else if (MARK(L34)==11 || MARK(L34)==12){MARK(L34)=500;* *MARK(L24)=0;}* *else {MARK(L24)=MARK(L24)+100;}*
MOG3	*if (MARK(G3)==1|| MARK(G3)==31) { if (MARK(L31)==0)* *{MARK(L31)=3; MARK(G3)=200;}* *else {MARK(G3)=31;} }* *else if (MARK(G3)==2 || MARK(G3)==32) { if (MARK(L32)==0)* *{MARK(L32)=3; MARK(G3)=200;}* *else {MARK(G3)=32;}}* *else if (MARK(G3)==4 || MARK(G3)==34) { if (MARK(L34)==0)* *{MARK(L34)=3; MARK(G3)=200;}* *else {MARK(G3)=34;} }* *else if (MARK(G3)==500) { if (MARK(L31)==31){ MARK(L31)=500;* *MARK(G3)=0;}* *if (MARK(L32)==32) {MARK(L32)=500; MARK(G3)=0;}* *if (MARK(L34)==34) {MARK(L34)=500; MARK(G3)=0;} }*
MOG31	*if (MARK(L41)==0) { if (MARK(L31)==2 || MARK(L31)==102)* *{MARK(L41)=3; MARK(L31)=21;}* *else if (MARK(L31)==3 || MARK(L31)==103)* *{MARK(L41)=3; MARK(L31)=31;} }* *else if (MARK(L41)==21 || MARK(L41)==13){MARK(L41)=500;* *MARK(L31)=0;}* *else {MARK(L31)=MARK(L31)+100;}*
MOG32	*if (MARK(L42)==0) {MARK(L42)=3; MARK(L32)=32;}* *else if (MARK(L42)==32) {MARK(L42)=500; MARK(L32)=0;}* *else {MARK(L32)=MARK(L32)+100;}*
MOG33	*MARK(ACK3)=MARK(L23)-10;* *if (MARK(L23)==12) {MARK(L23)=23;}*

Table 12 (continue): Output Gate Definitions for SAN Model *medium*

Gate	*Definition*
MOG34	*if (MARK(L34)==1) {MARK(L44)=3; MARK(L34)=MARK(L34)+10; } else if (MARK(L34)==2) {MARK(L44)=2; MARK(L34)=MARK(L34)+10; } else if (MARK(L34)==3) {MARK(L44)=1; MARK(L34)=MARK(L34)+10;} if (MARK(L44)==123) {MARK(L44)=0; MARK(L34)=0;}*
MOG4	*if (MARK(G4)==1 || MARK(G4)==41) { if (MARK(L41)==0) {MARK(L41)=4; MARK(G4)=200;} else {MARK(G4)=41;} } else if (MARK(G4)==2 || MARK(G4)==42) { if (MARK(L42)==0) {MARK(L42)=4; MARK(G4)=200;} else {MARK(G4)=42;} } else if (MARK(G4)==3 || MARK(G4)==43) { if (MARK(L43)==0){MARK(L43)=4; MARK(G4)=200;} else {MARK(G4)=43;} } else if (MARK(G4)==500) { if (MARK(L41)==41) {MARK(L41)=500; MARK(G4)=0;} if (MARK(L42)==42) {MARK(L42)=500; MARK(G4)=0;} if (MARK(L43)==43) {MARK(L43)=500; MARK(G4)=0;} }*
MOG41	*if (MARK(L41)==2) {MARK(L11)=3; MARK(L41)=MARK(L41)+10;} else if (MARK(L41)==3) {MARK(L11)=2; MARK(L41)=MARK(L41)+10;} else if (MARK(L41)==4) {MARK(L11)=1; MARK(L41)= MARK(L41)+10;} if (MARK(L11)==123) {MARK(L11)=0; MARK(L41)=0;}*
MOG42	*if (MARK(L12)==0) { if (MARK(L42)==3 || MARK(L42)==103){ MARK(L12)=3; MARK(L42)=32;} else if (MARK(L42)==4 || MARK(L42)==104) { MARK(L12)=4; MARK(L42)=42;} } else if (MARK(L12)==13 || MARK(L12)==14) {MARK(L12)=500; MARK(L42)=0;} else {MARK(L42)=MARK(L42)+100;}*
MOG43	*if (MARK(L13)==0) {MARK(L13)=4; MARK(L43)=43;} else if (MARK(L13)==43) {MARK(L13)=500; MARK(L43)=0;} else {MARK(L43)=MARK(L43)+100;} }*
MOG44	*MARK(ACK4)=MARK(L34)-10; if (MARK(L34)==13){MARK(L34)=34;}*

APPENDIX II

DOCUMENTATION OF SAN MODEL OF WDM RING II

Documentation for Composed Model: *r6nodes.*

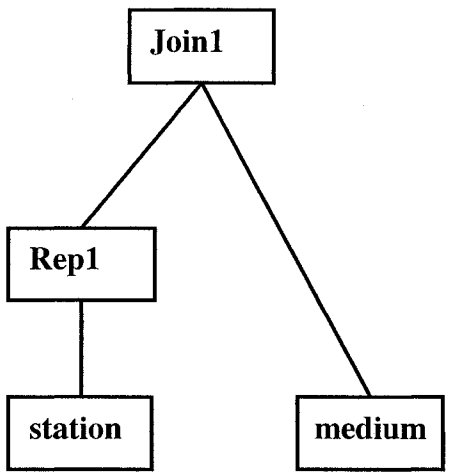


Figure II.1: Composed Model: *r6nodes*

Table II.1: Composed Model Rep Node Definitions for Project *r6nodes*

Node	Reps	Common Places
Rep1	6	ACK1
		ACK2
		ACK3
		ACK4
		ACK5
		ACK6
		C
		G1
		G2
		G3
		G4
		G5
		G6
		L11
		L22
		L33
		L44
		L55
		L66
		init

Table II.2: Composed Model Join Node Definitions for Project *r6nodes*

Node	Common Places		
Join1	Subtree	1	2
	ACK1	√	√
	ACK2	√	√
	ACK3	√	√
	ACK4	√	√
	ACK5	√	√
	ACK6	√	√
	G1	√	√
	G2	√	√
	G3	√	√
	G4	√	√
	G5	√	√
	G6	√	√
	L11	√	√
	L22	√	√
	L33	√	√
	L44	√	√
	L55	√	√
	L66	√	√

Table II.3: Reward Variable Definitions for Project *r6nodes*

Variable	***Definition***
lt1	Rate rewards none
	Impulse rewards Subnet = station activity = *LTX1*,value = *1*
	Simulator statistics Estimate mean Confidence Level = *0.95* Relative Confidence Interval = *0.10* Initial Transient = *1000* Batch Size =*1000* Variable type = *Interval of Time* Start of Interval = *0* Length of Interval = *1000*
lt2	Rate rewards none
	Impulse rewards Subnet = station activity = *LTX2*,value = *1*
	Simulator statistics Estimate mean Confidence Level = *0.95* Relative Confidence Interval = *0.10* Initial Transient = *1000* Batch Size =*1000* Variable type = *Interval of Time* Start of Interval = *0* Length of Interval = *1000*

Table II.3 (continue): Reward Variable Definitions for Project *r6nodes*

Variable	***Definition***
lt3	Rate rewards none
	Impulse rewards Subnet = station activity = *LTX3*,value = *1*
	Simulator statistics Estimate mean Confidence Level = *0.95* Relative Confidence Interval = *0.10* Initial Transient = *1000* Batch Size = *1000* Variable type = *Interval of Time* Start of Interval = *0* Length of Interval = *1000*
lt4	Rate rewards none
	Impulse rewards Subnet = station activity = *LTX4*,value = *1*
	Simulator statistics Estimate mean Confidence Level = *0.95* Relative Confidence Interval = *0.10* Initial Transient = *1000* Batch Size = *1000* Variable type = *Interval of Time* Start of Interval = *0* Length of Interval = *1000*

Table II.3 (continue): Reward Variable Definitions for Project *r6nodes*

Variable	***Definition***
lt5	Rate rewards none
	Impulse rewards Subnet = station activity = *LTX5*,value = *1*
	Simulator statistics Estimate mean Confidence Level = *0.95* Relative Confidence Interval = *0.10* Initial Transient = *1000* Batch Size = *1000* Variable type = *Interval of Time* Start of Interval = *0* Length of Interval = *1000*
lt6	Rate rewards none
	Impulse rewards Subnet = station activity = *LTX6*,value = *1*
	Simulator statistics Estimate mean Confidence Level = *0.95* Relative Confidence Interval = *0.10* Initial Transient = *1000* Batch Size =*1000* Variable type = *Interval of Time* Start of Interval = *0* Length of Interval = *1000*

Table II.3 (continue): Reward Variable Definitions for Project *r6nodes*

Variable	***Definition***
t1	Rate rewards none
	Impulse rewards Subnet = station activity = *TX1*,value = *30000*
	Simulator statistics Estimate mean Confidence Level = *0.95* Relative Confidence Interval = *0.10* Initial Transient = *1000* Batch Size = *1000* Variable type = *Interval of Time* Start of Interval = *0* Length of Interval = *1000*
t2	Rate rewards none
	Impulse rewards Subnet = station activity = *TX2*,value = *30000*
	Simulator statistics Estimate mean Confidence Level = *0.95* Relative Confidence Interval = *0.10* Initial Transient = *1000* Batch Size = *1000* Variable type = *Interval of Time* Start of Interval = *0* Length of Interval = *1000*

Table II.3 (continue): Reward Variable Definitions for Project *r6nodes*

Variable	***Definition***
t3	Rate rewards none
	Impulse rewards Subnet = station activity = *TX3*,value = *30000*
	Simulator statistics Estimate mean Confidence Level = *0.95* Relative Confidence Interval = *0.10* Initial Transient = *1000* Batch Size = *1000* Variable type = *Interval of Time* Start of Interval = *0* Length of Interval = *1000*
t4	Rate rewards none
	Impulse rewards Subnet = station activity = *TX4*,value = *30000*
	Simulator statistics Estimate mean Confidence Level = *0.95* Relative Confidence Interval = *0.10* Initial Transient = *1000* Batch Size = *1000* Variable type = *Interval of Time* Start of Interval = *0* Length of Interval = *1000*

Table II.3 (continue): Reward Variable Definitions for Project *r6nodes*

Variable	***Definition***
t5	Rate rewards none
	Impulse rewards Subnet = station activity = *TX5*,value = *30000*
	Simulator statistics Estimate mean Confidence Level = *0.95* Relative Confidence Interval = *0.10* Initial Transient = *1000* Batch Size = *1000* Variable type = *Interval of Time* Start of Interval = *0* Length of Interval = *1000*
t6	Rate rewards none
	Impulse rewards Subnet = station activity = *TX6*,value = *30000*
	Simulator statistics Estimate mean Confidence Level = *0.95* Relative Confidence Interval = *0.10* Initial Transient = *1000* Batch Size = *1000* Variable type = *Interval of Time* Start of Interval = *0* Length of Interval = *1000*

Table II.3 (continue): Reward Variable Definitions for Project *r6nodes*

Variable	*Definition*
rrr1	Rate rewards Subnet = station Predicate: *1* Function: *MARK(RR1)*
	Impulse rewards none
	Simulator statistics Estimate mean Confidence Level = *0.95* Relative Confidence Interval = *0.10* Initial Transient = *1000* Batch Size = *1000* Variable type = *Instant of Time* Start of Interval = *1000* Length of Interval = *1000*
rrr2	Rate rewards Subnet = station Predicate: *1* Function: *MARK(RR2)*
	Impulse rewards none
	Simulator statistics Estimate mean Confidence Level = *0.95* Relative Confidence Interval = *0.10* Initial Transient = *1000* Batch Size = *1000* Variable type = *Instant of Time* Start of Interval = *1000* Length of Interval = *1000*

Table II.3 (continue): Reward Variable Definitions for Project *r6nodes*

Variable	***Definition***
rrr3	Rate rewards Subnet = station Predicate: *1* Function: *MARK(RR3)*
	Impulse rewards none
	Simulator statistics Estimate mean Confidence Level = *0.95* Relative Confidence Interval = *0.10* Initial Transient = *1000* Batch Size = *1000* Variable type = *Instant of Time* Start of Interval = *1000* Length of Interval = *1000*
rrr4	Rate rewards Subnet = station Predicate: *1* Function: *MARK(RR4)*
	Impulse rewards none
	Simulator statistics Estimate mean Confidence Level = *0.95* Relative Confidence Interval = *0.10* Initial Transient = *1000* Batch Size = *1000* Variable type = *Instant of Time* Start of Interval = *1000* Length of Interval = *1000*

Table II.3 (continue): Reward Variable Definitions for Project *r6nodes*

Variable	*Definition*
rrr5	Rate rewards Subnet = station Predicate: *1* Function: *MARK(RR5)*
	Impulse rewards none
	Simulator statistics Estimate mean Confidence Level = *0.95* Relative Confidence Interval = *0.10* Initial Transient = *1000* Batch Size = *1000* Variable type = *Instant of Time* Start of Interval = *1000* Length of Interval = *1000*
rrr6	Rate rewards Subnet = station Predicate: *1* Function: *MARK(RR6)*
	Impulse rewards none
	Simulator statistics Estimate mean Confidence Level = *0.95* Relative Confidence Interval = *0.10* Initial Transient = *1000* Batch Size = *1000* Variable type = *Instant of Time* Start of Interval = *1000* Length of Interval = *1000*

Table II.4: Study Editor Set Definitions for Project *r6nodes*

Study	Experiment	Variable	Type	Value
st1	exp1	arrivrate timeout	double double	10 0
	exp2	arrivrate timeout	double double	30 0
	exp3	arrivrate timeout	double double	50 0
	exp4	arrivrate timeout	double double	70 0
	exp5	arrivrate timeout	double double	100 0
	exp6	arrivrate timeout	double double	150 0
	exp7	arrivrate timeout	double double	200 0
	exp8	arrivrate timeout	double double	250 0
	exp9	arrivrate timeout	double double	300 0
	exp10	arrivrate timeout	double double	350 0
	exp11	arrivrate timeout	double double	400 0
	exp12	arrivrate timeout	double double	500 0
	exp13	arrivrate timeout	double double	600 0
	exp14	arrivrate timeout	double double	700 0
	exp15	arrivrate timeout	double double	800 0
	exp16	arrivrate timeout	double double	900 0
	exp17	arrivrate timeout	double double	1000 0

Table II.4 (continue): Study Editor Set Definitions for Project *r6nodes*

Study	Experiment	Variable	Type	Value
st2	exp1	arrivrate timeout	double double	10 0
	exp2	arrivrate timeout	double double	30 0
	exp3	arrivrate timeout	double double	50 0
	exp4	arrivrate timeout	double double	70 0
	exp5	arrivrate timeout	double double	100 0
	exp6	arrivrate timeout	double double	150 0
	exp7	arrivrate timeout	double double	200 0
	exp8	arrivrate timeout	double double	300 0
	exp9	arrivrate timeout	double double	400 0
	exp10	arrivrate timeout	double double	600 0
	exp11	arrivrate timeout	double double	800 0
	exp12	arrivrate timeout	double double	1000 0
	exp13	arrivrate timeout	double double	2000 0
	exp14	arrivrate timeout	double double	3000 0
	exp15	arrivrate timeout	double double	3500 0

Table II.4 (continue): Study Editor Set Definitions for Project *r6nodes*

Study	Experiment	Variable	Type	Value
st3	exp1	arrivrate timeout	double double	10 0
	exp2	arrivrate timeout	double double	30 0
	exp3	arrivrate timeout	double double	50 0
	exp4	arrivrate timeout	double double	70 0
	exp5	arrivrate timeout	double double	100 0
	exp6	arrivrate timeout	double double	150 0
	exp7	arrivrate timeout	double double	200 0
	exp8	arrivrate timeout	double double	300 0
	exp9	arrivrate timeout	double double	400 0
	exp10	arrivrate timeout	double double	600 0
	exp11	arrivrate timeout	double double	800 0
	exp12	arrivrate timeout	double double	1000 0
	exp13	arrivrate timeout	double double	2000 0
	exp14	arrivrate timeout	double double	3000 0

Table II.4 (continue): Study Editor Set Definitions for Project *r6nodes*

Study	Experiment	Variable	Type	Value
st4	exp1	arrivrate timeout	double double	4000 0
	exp2	arrivrate timeout	double double	5000 0
	exp3	arrivrate timeout	double double	6000 0
	exp4	arrivrate timeout	double double	7000 0
	exp5	arrivrate timeout	double double	8000 0
	exp6	arrivrate timeout	double double	9000 0
	exp7	arrivrate timeout	double double	10000 0
	exp8	arrivrate timeout	double double	12000 0
	exp9	arrivrate timeout	double double	15000 0
	exp10	arrivrate timeout	double double	17000 0
	exp11	arrivrate timeout	double double	19000 0
	exp12	arrivrate timeout	double double	20000 0

Table II.4 (continue): Study Editor Set Definitions for Project *r6nodes*

Study	Experiment	Variable	Type	Value
st5	exp1	arrivrate timeout	double double	1000 0
	exp2	arrivrate timeout	double double	2000 0
	exp3	arrivrate timeout	double double	3000 0
	exp4	arrivrate timeout	double double	4000 0
	exp5	arrivrate timeout	double double	6000 0
	exp6	arrivrate timeout	double double	8000 0
	exp7	arrivrate timeout	double double	10000 0
	exp8	arrivrate timeout	double double	12000 0
	exp9	arrivrate timeout	double double	15000 0
	exp10	arrivrate timeout	double double	17000 0
	exp11	arrivrate timeout	double double	19000 0
	exp12	arrivrate timeout	double double	20000 0

Table II.4 (continue): Study Editor Set Definitions for Project *r6nodes*

Study	Experiment	Variable	Type	Value
st6	exp1	arrivrate timeout	double double	10 0
	exp2	arrivrate timeout	double double	30 0
	exp3	arrivrate timeout	double double	50 0
	exp4	arrivrate timeout	double double	70 0
	exp5	arrivrate timeout	double double	100 0
	exp6	arrivrate timeout	double double	200 0
	exp7	arrivrate timeout	double double	300 0
	exp8	arrivrate timeout	double double	400 0
	exp9	arrivrate timeout	double double	500 0
	exp10	arrivrate timeout	double double	600 0
	exp11	arrivrate timeout	double double	700 0
	exp12	arrivrate timeout	double double	800 0

Table II.4 (continue): Study Editor Set Definitions for Project *r6nodes*

Study	Experiment	Variable	Type	Value
st7	exp1	arrivrate timeout	double double	10 0
	exp2	arrivrate timeout	double double	20 0
	exp3	arrivrate timeout	double double	30 0
	exp4	arrivrate timeout	double double	40 0
	exp5	arrivrate timeout	double double	50 0
	exp6	arrivrate timeout	double double	100 0
	exp7	arrivrate timeout	double double	200 0
	exp8	arrivrate timeout	double double	400 0
	exp9	arrivrate timeout	double double	600 0
	exp10	arrivrate timeout	double double	800 0
	exp11	arrivrate timeout	double double	1000 0
	exp12	arrivrate timeout	double double	2000 0
	exp13	arrivrate timeout	double double	3000 0
	exp14	arrivrate timeout	double double	5000 0
	exp15	arrivrate timeout	double double	7000 0
	exp16	arrivrate timeout	double double	10000 0
	exp17	arrivrate timeout	double double	20000 0

Table II.4 (continue): Study Editor Set Definitions for Project *r6nodes*

Study	Experiment	Variable	Type	Value
st8	exp1	arrivrate timeout	double double	200 0.0001
	exp2	arrivrate timeout	double double	200 0.001
	exp3	arrivrate timeout	double double	200 0.005
	exp4	arrivrate timeout	double double	200 0.01
	exp5	arrivrate timeout	double double	200 0.05
	exp6	arrivrate timeout	double double	200 0.1
	exp7	arrivrate timeout	double double	200 0.5
	exp8	arrivrate timeout	double double	200 0.7
	exp9	arrivrate timeout	double double	200 1
	exp10	arrivrate timeout	double double	200 1.5
	exp11	arrivrate timeout	double double	200 1.7
	exp12	arrivrate timeout	double double	200 1.9
	exp13	arrivrate timeout	double double	200 2

Table II.4 (continue): Study Editor Set Definitions for Project *r6nodes*

Study	Experiment	Variable	Type	Value
st9	exp1	arrivrate timeout	double double	10 0.0001
	exp2	arrivrate timeout	double double	10 0.0005
	exp3	arrivrate timeout	double double	10 0.001
	exp4	arrivrate timeout	double double	10 0.005
	exp5	arrivrate timeout	double double	10 0.01
	exp6	arrivrate timeout	double double	10 0.05
	exp7	arrivrate timeout	double double	10 0.1
	exp8	arrivrate timeout	double double	10 0.3
	exp9	arrivrate timeout	double double	10 0.5
	exp10	arrivrate timeout	double double	10 1.7
	exp11	arrivrate timeout	double double	10 1.9
	exp12	arrivrate timeout	double double	10 2
	exp13	arrivrate timeout	double double	0 0

Table II.4 (continue): Study Editor Set Definitions for Project *r6nodes*

Study	Experiment	Variable	Type	Value
st10	exp1	arrivrate timeout	double double	50 0.0001
	exp2	arrivrate timeout	double double	50 0.0005
	exp3	arrivrate timeout	double double	50 0.001
	exp4	arrivrate timeout	double double	50 0.005
	exp5	arrivrate timeout	double double	50 0.01
	exp6	arrivrate timeout	double double	50 0.05
	exp7	arrivrate timeout	double double	50 0.1
	exp8	arrivrate timeout	double double	50 0.5
	exp9	arrivrate timeout	double double	50 0.7
	exp10	arrivrate timeout	double double	50 1
	exp11	arrivrate timeout	double double	50 1.5
	exp12	arrivrate timeout	double double	50 1.7
	exp13	arrivrate timeout	double double	50 2

Table II.4 (continue): Study Editor Set Definitions for Project *r6nodes*

Study	Experiment	Variable	Type	Value
st11	exp1	arrivrate timeout	double double	200 0.0001
	exp2	arrivrate timeout	double double	200 0.0005
	exp3	arrivrate timeout	double double	200 0.001
	exp4	arrivrate timeout	double double	200 0.005
	exp5	arrivrate timeout	double double	200 0.01
	exp6	arrivrate timeout	double double	200 0.05
	exp7	arrivrate timeout	double double	200 0.1
	exp8	arrivrate timeout	double double	200 0.5
	exp9	arrivrate timeout	double double	200 0.7
	exp10	arrivrate timeout	double double	200 1
	exp11	arrivrate timeout	double double	200 1.5
	exp12	arrivrate timeout	double double	200 1.7
	exp13	arrivrate timeout	double double	200 2

Table II.4 (continue): Study Editor Set Definitions for Project *r6nodes*

Study	Experiment	Variable	Type	Value
st12	exp1	arrivrate timeout	double double	200 0.0001
	exp2	arrivrate timeout	double double	200 0.0005
	exp3	arrivrate timeout	double double	200 0.001
	exp4	arrivrate timeout	double double	200 0.005
	exp5	arrivrate timeout	double double	200 0.01
	exp6	arrivrate timeout	double double	200 0.05
	exp7	arrivrate timeout	double double	200 0.1
	exp8	arrivrate timeout	double double	200 0.5
	exp9	arrivrate timeout	double double	200 1
	exp10	arrivrate timeout	double double	200 2

Table II.5: Non-zero or Variable Initial Markings for SAN Model *station*

Place	*Marking*
B	*1*
C	*6*
init	*1*

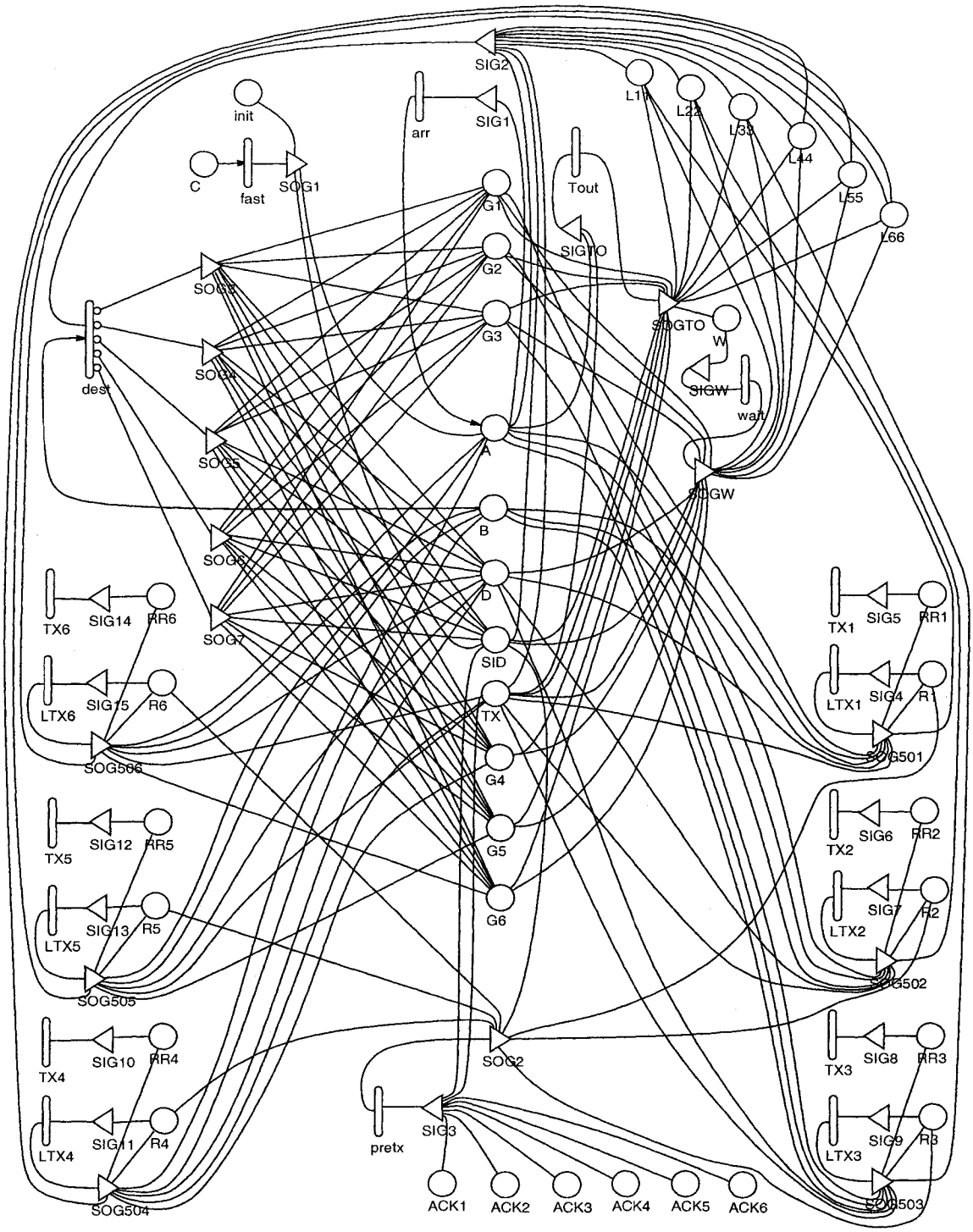


Figure II.2: SAN Model: *station*

Table II.6: Activity Time Distributions for SAN Model *station*

Activity	*Distribution*	*Parameter values*
LTX1	deterministic	
	value	*MARK(R1)*0.001*
LTX2	deterministic	
	value	*MARK(R2)*0.001*
LTX3	deterministic	
	value	*MARK(R3)*0.001*
LTX4	deterministic	
	value	*MARK(R4)*0.001*
LTX5	deterministic	
	value	*MARK(R5)* 0.001*
LTX6	deterministic	
	value	*MARK(R6)* 0.001*
TX1	deterministic	
	value	*0.000000000000001*
TX2	deterministic	
	value	*0.000000000000001*
TX3	deterministic	
	value	*0.000000000000001*
TX4	deterministic	
	value	*0.000000000000001*
TX5	deterministic	
	value	*0.000000000000001*
TX6	deterministic	
	value	*0.000000000000001*
Tout	deterministic	
	value	*GLOBAL_D(timeout)*
arr	exponential	
	rate	*GLOBAL_D(arrivrate)*
dest	deterministic	
	value	*0.000012*
fast	deterministic	
	value	*0.000000000000001*
pretx	deterministic	
	value	*0.00000000000001*
wait	deterministic	
	value	*MARK(W)* 0.00001*

Table II.7: Activity Case Probabilities for SAN Model *station*

Activity	*Case*	*Probability*
dest	1	*0.2*
	2	*0.2*
	3	*0.2*
	4	*0.2*
	5	*0.2*

Table II.8: Input Gate Definitions for SAN Model *station*

Gate	*Definition*
SIG1	Predicate *MARK(A)< 1*
	Function *identity*
SIG10	Predicate *MARK(RR4)>= 30000*
	Function *MARK(RR4)=0 ;*
SIG11	Predicate *MARK(R4)>0*
	Function *identity*
SIG12	Predicate *MARK(RR5)>= 30000*
	Function *MARK(RR5)=0 ;*
SIG13	Predicate *MARK(R5) > 0*
	Function *identity*
SIG14	Predicate *MARK(RR6) >= 30000*
	Function *MARK(RR6)=0 ;*
SIG15	Predicate *MARK(R6)> 0*
	Function *identity*

Table II.8 (continue): Input Gate Definitions for SAN Model *station*

Gate	***Definition***
SIG2	Predicate *(MARK(A) > 0 && MARK(SID)==1 && MARK(L11)==0 && MARK(TX)==0) ||* *(MARK(A) > 0 && MARK(SID)==2 && MARK(L22)==0 && MARK(TX)==0) ||* *(MARK(A) > 0 && MARK(SID)==3 && MARK(L33)==0 && MARK(TX)==0) ||* *(MARK(A) > 0 && MARK(SID)==4 && MARK(L44)==0 && MARK(TX)==0) ||* *(MARK(A) > 0 && MARK(SID)==5 && MARK(L55)==0 && MARK(TX)==0) ||* *(MARK(A) > 0 && MARK(SID)==6 && MARK(L66)==0 && MARK(TX)==0)*
	Function *if (MARK(SID)==1) {MARK(A)=2; MARK(L11)=999; }* *else if (MARK(SID)==2) {MARK(A)=2; MARK(L22)=999;}* *else if (MARK(SID)==3) {MARK(A)=2; MARK(L33)=999;}* *else if (MARK(SID)==4) {MARK(A)=2; MARK(L44)=999;}* *else if (MARK(SID)==5) {MARK(A)=2; MARK(L55)=999;}* *else if (MARK(SID)==6) {MARK(A)=2; MARK(L66)=999;}*
SIG3	Predicate *(MARK(TX)==0) && (MARK(SID)==MARK(ACK1) || MARK(SID)==MARK(ACK2) || MARK(SID)==MARK(ACK3) || MARK(SID)==MARK(ACK4) || MARK(SID)==MARK(ACK5) || MARK(SID)==MARK(ACK6))*
	Function *MARK(TX)=1;* *if (MARK(SID)==MARK(ACK1)) { MARK(ACK1)=0 ; }* *else if (MARK(SID)==MARK(ACK2)) { MARK(ACK2)=0; }* *else if (MARK(SID)==MARK(ACK3)) { MARK(ACK3)=0; }* *else if (MARK(SID)==MARK(ACK4)) { MARK(ACK4)=0; }* *else if (MARK(SID)==MARK(ACK5)) { MARK(ACK5)=0; }* *else if (MARK(SID)==MARK(ACK6)) { MARK(ACK6)=0; }*
SIG4	Predicate *MARK(R1) >0*
	Function *identity*

Table II.8 (continue): Input Gate Definitions for SAN Model *station*

Gate	*Definition*
SIG5	Predicate *MARK(RR1)>= 30000*
	Function *MARK(RR1)=0 ;*
SIG6	Predicate *MARK(RR2)>= 30000*
	Function *MARK(RR2)=0 ;*
SIG7	Predicate *MARK(R2)>0*
	Function *identity*
SIG8	Predicate *MARK(RR3)>= 30000*
	Function *MARK(RR3)=0 ;*
SIG9	Predicate *MARK(R3) > 0*
	Function *identity*
SIGTO	Predicate *MARK(A)> 0 && MARK(TX)==0*
	Function *identity*
SIGW	Predicate *MARK(W)> 0*
	Function *MARK(W)=0;*

Table II.9: Output Gate Definitions for SAN Model *station*

Gate	***Definition***
SOG1	*if (MARK(SID)==0) {MARK(SID)=MARK(init); MARK(init)++; MARK(A)=0;}*
SOG2	*#include <time.h>* *double TTT;* *time_t t ;* */* int ttt; */* *int dum;* *srand((unsigned)time(&t));* *TTT=rand() / 32768.0;* *if (TTT < 0.000000001) { TTT=0.000000001;}* *dum=1+(int)((log(TTT))/(log(0.99)));* *if (MARK(SID)==1) {MARK(R1)=dum;}* *else if (MARK(SID)==2) {MARK(R2)=dum;}* *else if (MARK(SID)==3) {MARK(R3)=dum;}* *else if (MARK(SID)==4) {MARK(R4)=dum;}* *else if (MARK(SID)==5) {MARK(R5)=dum;}* *else if (MARK(SID)==6) {MARK(R6)=dum;}*
SOG3	*if (MARK(SID)==1) { if (MARK(D) > 0) {MARK(G1)=MARK(D);}* *else {MARK(D)=MARK(SID)+1; MARK(G1)=MARK(D);} }* *else if (MARK(SID)==2) { if (MARK(D) > 0) {MARK(G2)=MARK(D);}* *else {MARK(D)=MARK(SID)+1; MARK(G2)=MARK(D);} }* *else if (MARK(SID)==3) { if (MARK(D) > 0) {MARK(G3)=MARK(D);}* *else {MARK(D)=MARK(SID)+1; MARK(G3)=MARK(D);} }* *else if (MARK(SID)==4) { if (MARK(D) > 0) {MARK(G4)=MARK(D);}* *else {MARK(D)=MARK(SID)+1; MARK(G4)=MARK(D);} }* *else if (MARK(SID)==5) { if (MARK(D) > 0) {MARK(G5)=MARK(D);}* *else {MARK(D)=MARK(SID)+1; MARK(G5)=MARK(D);} }* *else if (MARK(SID)==6) { if (MARK(D) > 0) {MARK(G6)=MARK(D);}* *else {MARK(D)=MARK(SID)-5; MARK(G6)=MARK(D);} }*

Table II.9 (continue): Output Gate Definitions for SAN Model *station*

Gate	*Definition*
SOG4	*if (MARK(SID)==1) { if (MARK(D) > 0) {MARK(G1)=MARK(D);}* *else {MARK(D)=MARK(SID)+2; MARK(G1)=MARK(D);} }* *else if (MARK(SID)==2) { if (MARK(D) > 0) {MARK(G2)=MARK(D);}* *else {MARK(D)=MARK(SID)+2; MARK(G2)=MARK(D);} }* *else if (MARK(SID)==3) { if (MARK(D) > 0) {MARK(G3)=MARK(D);}* *else {MARK(D)=MARK(SID)+2; MARK(G3)=MARK(D);} }* *else if (MARK(SID)==4) { if (MARK(D) > 0) {MARK(G4)=MARK(D);}* *else {MARK(D)=MARK(SID)+2; MARK(G4)=MARK(D);} }* *else if (MARK(SID)==5) { if (MARK(D) > 0) {MARK(G5)=MARK(D);}* *else {MARK(D)=MARK(SID)-4; MARK(G5)=MARK(D);} }* *else if (MARK(SID)==6) { if (MARK(D) > 0) {MARK(G6)=MARK(D);}* *else {MARK(D)=MARK(SID)-4; MARK(G6)=MARK(D);} }*
SOG5	*if (MARK(SID)==1) { if (MARK(D) > 0) {MARK(G1)=MARK(D);}* *else {MARK(D)=MARK(SID)+3; MARK(G1)=MARK(D);} }* *else if (MARK(SID)==2) { if (MARK(D) > 0) {MARK(G2)=MARK(D);}* *else {MARK(D)=MARK(SID)+3; MARK(G2)=MARK(D);} }* *else if (MARK(SID)==3) { if (MARK(D) > 0) {MARK(G3)=MARK(D);}* *else {MARK(D)=MARK(SID)+3; MARK(G3)=MARK(D);} }* *else if (MARK(SID)==4) { if (MARK(D) > 0) {MARK(G4)=MARK(D);}* *else {MARK(D)=MARK(SID)-3; MARK(G4)=MARK(D);} }* *else if (MARK(SID)==5) { if (MARK(D) > 0) {MARK(G5)=MARK(D);}* *else {MARK(D)=MARK(SID)-3; MARK(G5)=MARK(D);} }* *else if (MARK(SID)==6) { if (MARK(D) > 0) {MARK(G6)=MARK(D);}}*
SOG501	*MARK(G1)=500; MARK(A)=0; MARK(B)=1; MARK(TX)=0;* *MARK(D)=0;* *MARK(RR1)=MARK(RR1)+MARK(R1); MARK(R1)=0; MARK(L11)=0;*
SOG502	*MARK(G2)=500; MARK(A)=0; MARK(B)=1; MARK(TX)=0;* *MARK(D)=0;* *MARK(RR2)=MARK(RR2)+MARK(R2); MARK(R2)=0; MARK(L22)=0;*
SOG503	*MARK(G3)=500; MARK(A)=0; MARK(B)=1; MARK(TX)=0;* *MARK(D)=0;* *MARK(RR3)=MARK(RR3)+MARK(R3); MARK(R3)=0; MARK(L33)=0;*
SOG504	*MARK(G4)=500; MARK(A)=0; MARK(B)=1; MARK(TX)=0;* *MARK(D)=0;* *MARK(RR4)=MARK(RR4)+MARK(R4); MARK(R4)=0; MARK(L44)=0;*
SOG505	*MARK(G5)=500; MARK(A)=0; MARK(B)=1; MARK(TX)=0;* *MARK(D)=0;* *MARK(RR5)=MARK(RR5)+MARK(R5); MARK(R5)=0; MARK(L55)=0;*
SOG506	*MARK(G6)=500; MARK(A)=0; MARK(B)=1; MARK(TX)=0;* *MARK(D)=0;* *MARK(RR6)=MARK(RR6)+MARK(R6); MARK(R6)=0; MARK(L66)=0;*

Table II.9 (continue): Output Gate Definitions for SAN Model *station*

Gate	*Definition*
SOG6	*if (MARK(SID)==1) { if (MARK(D) > 0) {MARK(G1)=MARK(D);}* *else {MARK(D)=MARK(SID)+4; MARK(G1)=MARK(D);} }* *else if (MARK(SID)==2)* *{ if (MARK(D) > 0) {MARK(G2)=MARK(D);}* *else {MARK(D)=MARK(SID)+4; MARK(G2)=MARK(D);} }* *else if (MARK(SID)==3)* *{ if (MARK(D) > 0) {MARK(G3)=MARK(D);}* *else {MARK(D)=MARK(SID)-2; MARK(G3)=MARK(D);} }* *else if (MARK(SID)==4)* *{ if (MARK(D) > 0) {MARK(G4)=MARK(D);}* *else {MARK(D)=MARK(SID)-2; MARK(G4)=MARK(D);} }* *else if (MARK(SID)==5)* *{ if (MARK(D) > 0)* *{MARK(G5)=MARK(D);}* *else {MARK(D)=MARK(SID)-2; MARK(G5)=MARK(D);}* *else {MARK(D)=MARK(SID)-2; MARK(G6)=MARK(D);} }
SOG7	*if (MARK(SID)==1) { if (MARK(D) > 0) {MARK(G1)=MARK(D);}* *else {MARK(D)=MARK(SID)+5; MARK(G1)=MARK(D);} }* *else if (MARK(SID)==2)* *{ if (MARK(D) > 0) {MARK(G2)=MARK(D);}* *else {MARK(D)=MARK(SID)-1; MARK(G2)=MARK(D);} }* *else if (MARK(SID)==3)* *{ if (MARK(D) > 0) {MARK(G3)=MARK(D);}* *else {MARK(D)=MARK(SID)-1; MARK(G3)=MARK(D);} }* *else if (MARK(SID)==4)* *{ if (MARK(D) > 0) {MARK(G4)=MARK(D);}* *else {MARK(D)=MARK(SID)-1; MARK(G4)=MARK(D);} }* *else if (MARK(SID)==5)* *{ if (MARK(D) > 0)* *{MARK(G5)=MARK(D);}* *else {MARK(D)=MARK(SID)-1; MARK(G5)=MARK(D);}* *else if (MARK(SID)==6) { if (MARK(D) > 0) {MARK(G6)=MARK(D);}* *else {MARK(D)=MARK(SID)-1; MARK(G6)=MARK(D);} }

Table II.9 (continue): Output Gate Definitions for SAN Model *station*

Gate	***Definition***
SOGTO	*#include <time.h>* *double TT;* *time_t t;* *int WR;* *srand((unsigned)time(&t));* *TT=rand() / 32768.0 ;* *if (TT < 0.000000001) {TT=0.000000001;}* *WR=8+(int)(9 * TT);* *MARK(TX)=2; MARK(W)=WR;* *if (MARK(SID)==1) {if (MARK(G1) > 0 && MARK(G1) !=600)* *{MARK(G1)=600;}* *if (MARK(L11)==999) {MARK(L11)=0;} }* *else if (MARK(SID)==2) { if (MARK(G2) > 0 && MARK(G2) !=600)* *{MARK(G2)=600;}* *if (MARK(L22)==999) {MARK(L22)=0;} }* *else if (MARK(SID)==3) { if (MARK(G3) > 0 && MARK(G3) !=600)* *{MARK(G3)=600;}* *if (MARK(L33)==999) {MARK(L33)=0;} }* *else if (MARK(SID)==4) { if (MARK(G4) > 0 && MARK(G4) !=600)* *{MARK(G4)=600;}* *if (MARK(L44)==999) {MARK(L44)=0;} }* *else if (MARK(SID)==5) { if (MARK(G5) > 0 && MARK(G5) !=600)* *{MARK(G5)=600;}* *if (MARK(L55)==999) {MARK(L55)=0;} }* *else if (MARK(SID)==6) { if (MARK(G6) > 0 && MARK(G6) !=600)* *{MARK(G6)=600;} }*
SOGW	*if (MARK(SID)==1) { if (MARK(L11)==0 && MARK(D) > 0)* *{MARK(L11)=999; MARK(G1)=MARK(D);} }* *else if (MARK(SID)==2) { if (MARK(L22)==0 && MARK(D) > 0)* *{MARK(L22)=999; MARK(G2)=MARK(D);} }* *else if (MARK(SID)==3) { if (MARK(L33)==0 && MARK(D) > 0)* *{MARK(L33)=999; MARK(G3)=MARK(D);} }* *else if (MARK(SID)==4) { if (MARK(L44)==0 && MARK(D) > 0)* *{MARK(L44)=999; MARK(G4)=MARK(D);} }* *else if (MARK(SID)==5) { if (MARK(L55)==0 && MARK(D) > 0)* *{MARK(L55)=999; MARK(G5)=MARK(D);} }*

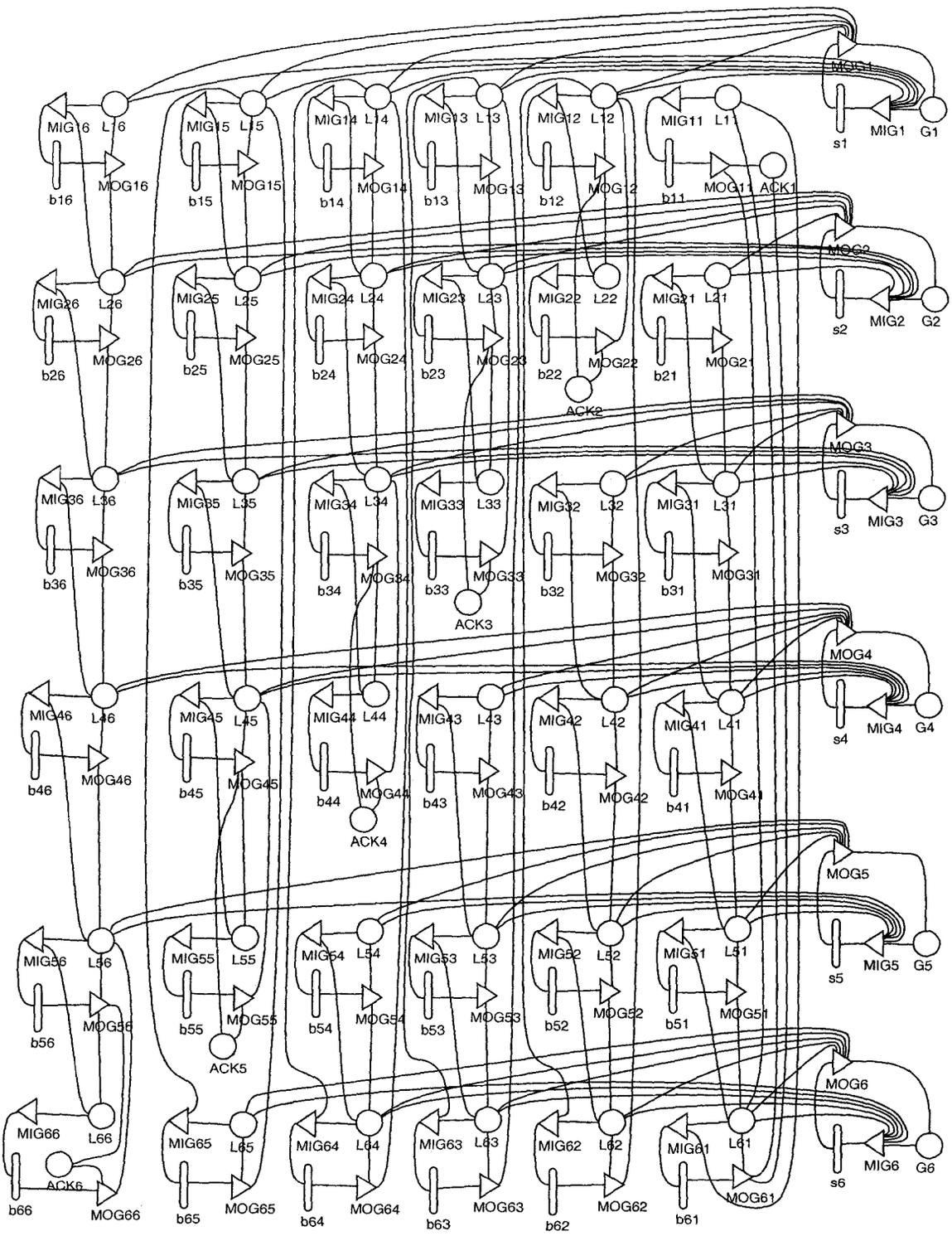


Figure 3: SAN Model: medium

Table 10: Activity Time Distributions for SAN Model *medium*

Activity	***Distribution***	***Parameter values***
b11	deterministic	
	value	*MARK(L11)*0.00001*
b12	deterministic	
	value	*0.00001*
b13	deterministic	
	value	*0.00001*
b14	deterministic	
	value	*0.00001*
b15	deterministic	
	value	*0.00001*
b16	deterministic	
	value	*0.00001*
b21	deterministic	
	value	*0.00001*
b22	deterministic	
	value	*MARK(L22)*0.00001*
b23	deterministic	
	value	*0.00001*
b24	deterministic	
	value	*0.00001*
b25	deterministic	
	value	*0.00001*
b26	deterministic	
	value	*0.00001*
b31	deterministic	
	value	*0.00001*
b32	deterministic	
	value	*0.00001*
b33	deterministic	
	value	*MARK(L33)*0.00001*
b34	deterministic	
	value	*0.00001*
b35	deterministic	
	value	*0.00001*
b36	deterministic	
	value	*0.00001*
b41	deterministic	
	value	*0.00001*
b42	deterministic	
	value	*0.00001*
b43	deterministic	
	value	*0.00001*

Table II.10 (continue): Activity Time Distributions for SAN Model *medium*

Activity	*Distribution*	*Parameter values*
b44	deterministic	
	value	*MARK(L44)*0.00001*
b45	deterministic	
	value	*0.00001*
b46	deterministic	
	value	*0.00001*
b51	deterministic	
	value	*0.00001*
b52	deterministic	
	value	*0.00001*
b53	deterministic	
	value	*0.00001*
b54	deterministic	
	value	*0.00001*
b55	deterministic	
	value	*MARK(L55)*0.00001*
b61	deterministic	
	value	*0.00001*
b62	deterministic	
	value	*0.00001*
b63	deterministic	
	value	*0.00001*
b64	deterministic	
	value	*0.00001*
b65	deterministic	
	value	*0.00001*
b66	deterministic	
	value	*MARK(L66)*0.00001*
s1	deterministic	
	value	*0.000000000000001*
s2	deterministic	
	value	*0.000000000000001*
s3	deterministic	
	value	*0.000000000000001*
s4	deterministic	
	value	*0.000000000000001*
s5	deterministic	
	value	*0.000000000000001*
s6	deterministic	
	value	*0.000000000000001*

Table II.11: Input Gate Definitions for SAN Model medium

Gate	***Definition***
MIG1	Predicate *MARK(G1)==2 || MARK(G1)==3 || MARK(G1)==4 || MARK(G1)==5 || MARK(G1)==6 || MARK(G1)==500 || MARK(G1)==600 || (MARK(G1)==12 && MARK(L12)==0) || (MARK(G1)==13 && MARK(L13)==0) || (MARK(G1)==14 && MARK(L14)==0) || (MARK(G1)==15 && MARK(L15)==0) || (MARK(G1)==16 && MARK(L16)==0)*
	Function *identity*
MIG11	Predicate *MARK(L11)==1 || MARK(L11)==2 || MARK(L11)==3 || MARK(L11)==4 || MARK(L11)==5*
	Function *MARK(L11)=123 ;*
MIG12	Predicate *MARK(L12)==1 || MARK(L12)==3 || MARK(L12)==4 || MARK(L12)==5 || MARK(L12)==6 || MARK(L12)==500 || MARK(L12)==600 || (MARK(L12)==101 && MARK(L22)==0) || (MARK(L12)==106 && MARK(L22)==0) || (MARK(L12)==105 && MARK(L22)==0) || (MARK(L12)==104 && MARK(L22)==0) || (MARK(L12)==103 && MARK(L22)==0)*
	Function *identity*
MIG13	Predicate *MARK(L13)==1 || MARK(L13)==6 || MARK(L13)==5 || MARK(L13)==4 || MARK(L13)==500 || MARK(L13)==600 || (MARK(L13)==101 && MARK(L23)==0) || (MARK(L13)==106 && MARK(L23)==0) || (MARK(L13)==105 && MARK(L23)==0) || (MARK(L13)==104 && MARK(L23)==0)*
	Function *identity*

Table II.11 (continue): Input Gate Definitions for SAN Model medium

Gate	***Definition***
MIG14	Predicate *MARK(L14)==1 || MARK(L14)==6 || MARK(L14)==5 || MARK(L14)==500 || MARK(L14)==600 || (MARK(L14)==101 && MARK(L24)==0) || (MARK(L14)==106 && MARK(L24)==0) || (MARK(L14)==105 && MARK(L24)==0)*
	Function *identity*
MIG15	Predicate *MARK(L15)==1 || MARK(L15)==6 || MARK(L15)==500 || MARK(L15)==600 || (MARK(L15)==101 && MARK(L25)==0) || (MARK(L15)==106 && MARK(L25)==0)*
	Function *identity*
MIG16	Predicate *MARK(L16)==1 || MARK(L16)==500 || MARK(L16)==600 || (MARK(L16)==101 && MARK(L26)==0)*
	Function *identity*
MIG2	Predicate *MARK(G2)==1 || MARK(G2)==3 || MARK(G2)==4 || MARK(G2)==5 || MARK(G2)==6 || MARK(G2)==500 ||MARK(G2)==600 || (MARK(G2)==21 && MARK(L21)==0) || (MARK(G2)==23 && MARK(L23)==0) || (MARK(G2)==24 && MARK(L24)==0) || (MARK(G2)==25 && MARK(L25)==0) || (MARK(G2)==26 && MARK(L26)==0)*
	Function *identity*
MIG21	Predicate *MARK(L21)==2 || MARK(L21)==500 || MARK(L21)==600 || (MARK(L21)==102 && MARK(L31)==0)*
	Function *identity*
MIG22	Predicate *MARK(L22)==1 || MARK(L22)==2 || MARK(L22)==3 || MARK(L22)==4 || MARK(L22)==5*
	Function *MARK(L22)=123;*

Table II.11 (continue): Input Gate Definitions for SAN Model medium

Gate	***Definition***
MIG23	Predicate *MARK(L23)==2 || MARK(L23)==1 || MARK(L23)==6 || MARK(L23)==5 || MARK(L23)==4 || MARK(L23)==500 || MARK(L23)==600 || (MARK(L23)==102 && MARK(L33)==0) || (MARK(L23)==101 && MARK(L33)==0) || (MARK(L23)==106 && MARK(L33)==0) || (MARK(L23)==105 && MARK(L33)==0)*
	Function *identity*
MIG24	Predicate *MARK(L24)==2 || MARK(L24)==1 || MARK(L24)==6 || MARK(L24)==5 || MARK(L24)==500 || MARK(L24)==600 || (MARK(L24)==102 && MARK(L34)==0) || (MARK(L24)==101 && MARK(L34)==0) || (MARK(L24)==106 && MARK(L34)==0) || (MARK(L24)==105 && MARK(L34)==0)*
	Function *identity*
MIG25	Predicate *MARK(L25)==2 || MARK(L25)==1 || MARK(L25)==6 || MARK(L25)==500 || MARK(L25)==600 || (MARK(L25)==102 && MARK(L35)==0) || (MARK(L25)==101 && MARK(L35)==0) || (MARK(L25)==106 && MARK(L35)==0)*
	Function *identity*
MIG26	Predicate *MARK(L26)= =2 || MARK(L26)= =1 || MARK(L26)= =500 || MARK(L26)= = 600 ||(MARK(L26)= =102 && MARK(L36)= =0) ||(MARK(L26)= =101 && MARK(L36)= =0)*
	Function *identity*

Table II.11 (continue): Input Gate Definitions for SAN Model medium

Gate	***Definition***
MIG3	Predicate *MARK (G3)= =1 || MARK(G3)= =2 || MARK(G3)= =4 || MARK(G3)= =5 || MARK(G3)= =6 || MARK(G3)= =500 || MARK(G3)= =600 ||(MARK(G3)= =31 && MARK(L31)= =0) ||(MARK(G3)= =32 && MARK(L32)= =0) ||(MARK(G3)= =34 && MARK(L34)= =0) ||(MARK(G3)= =35 && MARK(L35)= =0) ||(MARK(G3)= =36 && MARK(L36)= =0)*
	Function *identity*
MIG31	Predicate *MARK(L31)= =3 || MARK(L31)= =2|| MARK(L31)= =500 || MARK(L31)= =600 ||(MARK(L31)= =103 && MARK(L41)= =0) ||(MARK(L31)= =102 && MARK(L41)= =0)*
	Function *identity*
MIG32	Predicate MARK(L32)= =3|| MARK(L32)= =500||MARK(L32)= =600 || (MARK(L32)= =103&& MARK(L42)= =0)
	Function *identity*
MIG33	Predicate MARK(L33)= =1||MARK(L33)= =2|| MARK(L33)= =3|| MARK(L33)= =4||MARK(L33)= =5
	Function MARK(L33)=123;
MIG34	Predicate MARK(L34)= =3 || MARK(L34)= =2|| MARK(L34)= =1|| MARK(L34)= =6|| MARK(L34)= =5|| MARK(L34)= =500|| MARK(L34)= =600 ||(MARK(L34)= =103&&MARK(L44)= =0) ||(MARK(L34)= =102&& MARK(L44)= =0) ||(MARK(L34)= =101&& MARK(L44)= =0) ||(MARK(L34)= =106&& MARK(L44)= =0) ||(MARK(L34)= =105&& MARK(L44)= =0)
	Function *identity*

Table II.11 (continue): Input Gate Definitions for SAN Model medium

Gate	***Definition***
MIG35	Predicate *MARK(L35)==3|| MARK(L35)==2|| MARK(L35)==1||* *MARK(L35)==6|| MARK(L35)==500||MARK(L35)= =600* *||(MARK(L35)==103&& MARK(L45)==0)* *||(MARK(L35)==102&& MARK(L45)==0)* *||(MARK(L35)==101&& MARK(L45)==0)* *||(MARK(L35)==106&& MARK(L45)==0)*
	Function *identity*
MIG36	Predicate *MARK(L36)==3||MARK(L36)==2|| MARK(L36)==1||* *MARK(L36)==500|| MARK(L36)==600* *||(MARK(L36)==103&& MARK(L46)==0)* *||(MARK(L36)==102&& MARK(L46)==0)* *||(MARK(L36)==101&& MARK(L46)==0)*
	Function *identity*
MIG4	Predicate *MARK(G4)==1|| MARK(G4)==2|| MARK(G4)==3||* *MARK(G4)==5|| MARK(G4)==6* *|| MARK(G4)==500|| MARK(G4)==600* *||(MARK(G4)==41&& MARK(L41)==0)* *||(MARK(G4)==42&& MARK(L42)==0)* *||(MARK(G4)==43&& MARK(L43)==0)* *||(MARK(G4)==45&& MARK(L45)==0)* *||(MARK(G4)==46&& MARK(L46)==0)*
	Function *identity*
MIG41	Predicate *MARK(L41)==4||MARK(L41)==3|| MARK(L41)==2||* *MARK(L41)==500|| MARK(L41)==600* *||(MARK(L41)==104&& MARK(L51)==0)* *||(MARK(L41)==103&& MARK(L51)==0)* *||(MARK(L41)==102&& MARK(L51)==0)*
	Function *identity*

Table II.11 (continue): Input Gate Definitions for SAN Model medium

Gate	***Definition***
MIG42	Predicate *MARK(L42)==4|| MARK(L42)==3|| MARK(L42)==500|| MARK(L42)==600* *||(MARK(L42)==104&& MARK(L52)==0)* *||(MARK(L42)==103&& MARK(L52)==0)*
	Function *identity*
MIG43	Predicate *MARK(L43)==4|| MARK(L43)==500|| MARK(L43)==600* *||(MARK(L43)==104&& MARK(L53)==0)*
	Function *identity*
MIG44	Predicate *MARK(L44)==1|| MARK(L44)==2|| MARK(L44)==3|| MARK(L44)==4|| MARK(L44)==5*
	Function *MARK(L44)=123;*
MIG45	Predicate *MARK(L45)==4|| MARK(L45)==3|| MARK(L45)==2|| MARK(L45)==1|| MARK(L45)==6||* *MARK(L45)==500|| MARK(L45)==600* *||(MARK(L45)==104&& MARK(L55)==0)* *||(MARK(L45)==103&& MARK(L55)==0)* *||(MARK(L45)==102&& MARK(L55)==0)* *||(MARK(L45)==101&& MARK(L55)==0)* *||(MARK(L45)==106&& MARK(L55)==0)*
	Function *identity*
MIG46	Predicate *MARK(L46)==4|| MARK(L46)==3|| MARK(L46)==2|| MARK(L46)==1||* *MARK(L46)==500|| MARK(L46)==600* *||(MARK(L46)==104&& MARK(L56)==0)* *||(MARK(L46)==103&& MARK(L56)==0)* *||(MARK(L46)==102&& MARK(L56)==0)*
	Function *identity*

Table II.11 (continue): Input Gate Definitions for SAN Model medium

Gate	***Definition***
MIG5	Predicate *MARK(G5)==1|| MARK(G5)==2|| MARK(G5)==3|| MARK(G5)==4|| MARK(G5)==6* *|| MARK(G5)==500|| MARK(G5)==600* *||(MARK(G5)==51&& MARK(L51)==0)* *||(MARK(G5)==52&& MARK(L52)==0)* *||(MARK(G5)==53&& MARK(L53)==0)* *||(MARK(G5)==54&& MARK(L54)==0)* *||(MARK(G5)==56&& MARK(L56)==0)*
	Function *identity*
MIG51	Predicate *MARK(L51)==5|| MARK(L51)==4|| MARK(L51)==3|| MARK(L51)==2||* *MARK(L51)==500|| MARK(L51)==600* *||(MARK(L51)==105&& MARK(L61)==0)* *||(MARK(L51)==104&& MARK(L61)==0)* *||(MARK(L51)==103&& MARK(L61)==0)* *||(MARK(L51)==102&& MARK(L61)==0)*
	Function *identity*
MIG52	Predicate *MARK(L52)==5|| MARK(L52)==4|| MARK(L52)==3|| MARK(L52)==500|| MARK(L52)==600* *||(MARK(L52)==105&& MARK(L62)==0)* *||(MARK(L52)==104&& MARK(L62)==0)* *||(MARK(L52)==103&& MARK(L62)==0)*
	Function *identity*
MIG53	Predicate *MARK(L53)==5|| MARK(L53)==4|| MARK(L53)==500|| MARK(L53)==600* *||(MARK(L53)==105&& MARK(L63)==0)* *||(MARK(L53)==104&& MARK(L63)==0)*
	Function *identity*
MIG54	Predicate *MARK(L54)==5|| MARK(L54)==500|| MARK(L54)==600* *||(MARK(L54)==105&& MARK(L64)==0)*
	Function *identity*

Table II.11 (continue): Input Gate Definitions for SAN Model medium

Gate	***Definition***
MIG55	Predicate *MARK(L55)==1|| MARK(L55)==2|| MARK(L55)==3|| MARK(L55)==4|| MARK(L55)==5*
	Function *MARK(L55)=123;*
MIG56	Predicate *MARK(L56)==5|| MARK(L56)==4|| MARK(L56)==3|| MARK(L56)==2|| MARK(L56)==1* *|| MARK(L56)==500|| MARK(L56)==600* *||(MARK(L56)==105&& MARK(L66)==0)* *||(MARK(L56)==104&& MARK(L66)==0)* *||(MARK(L56)==103&& MARK(L66)==0)* *||(MARK(L56)==102&& MARK(L66)==0)* *||(MARK(L56)==101&& MARK(L66)==0)*
	Function *identity*
MIG6	Predicate *MARK(G6)==1|| MARK(G6)==2|| MARK(G6)==3|| MARK(G6)==4|| MARK(G6)==5* *|| MARK(G6)==500|| MARK(G6)==600* *||(MARK(G6)==61&& MARK(L61)==0)* *||(MARK(G6)==62&& MARK(L62)==0)* *||(MARK(G6)==63&& MARK(L63)==0)* *||(MARK(G6)==64&& MARK(L64)==0)* *||(MARK(G6)==65&& MARK(L65)==0)*
	Function *identity*
MIG61	Predicate *MARK(L61)==6|| MARK(L61)==5|| MARK(L61)==4|| MARK(L61)==3|| MARK(L61)==2|| MARK(L61)==500|| MARK(L61)==600* *||(MARK(L61)==106&& MARK(L11)==0)* *||(MARK(L61)==105&& MARK(L11)==0)* *||(MARK(L61)==104&& MARK(L11)==0)* *||(MARK(L61)==103&& MARK(L11)==0)* *||(MARK(L61)==102&& MARK(L11)==0)*
	Function *identity*

Table II.11 (continue): Input Gate Definitions for SAN Model medium

Gate	***Definition***
MIG62	Predicate *MARK(L62)==6|| MARK(L62)==5|| MARK(L62)==4|| MARK(L62)==3|| MARK(L62)==500|| MARK(L62)==600 ||(MARK(L62)==106&& MARK(L12)==0) ||(MARK(L62)==105&& MARK(L12)==0) ||(MARK(L62)==104&& MARK(L12)==0) ||(MARK(L62)==103&& MARK(L12)==0)*
	Function *identity*
MIG63	Predicate *MARK(L63)==6|| MARK(L63)==5|| MARK(L63)==4|| MARK(L63)==500|| MARK(L63)==600 ||(MARK(L63)==106&& MARK(L13)==0) ||(MARK(L63)==105&& MARK(L13)==0) ||(MARK(L63)==104&& MARK(L13)==0)*
	Function *identity*
MIG64	Predicate *MARK(L64)==6|| MARK(L64)==5|| MARK(L64)==500|| MARK(L64)==600 ||(MARK(L64)==106&& MARK(L14)==0) ||(MARK(L64)==105&& MARK(L14)==0)*
	Function *identity*
MIG65	Predicate *MARK(L65)==6|| MARK(L65)==500|| MARK(L65)==600 ||(MARK(L65)==106&& MARK(L15)==0)*
	Function *identity*
MIG66	Predicate *MARK(L66)==1|| MARK(L66)==2|| MARK(L66)==3|| MARK(L66)==4|| MARK(L66)==5*
	Function *MARK(L66)=123;*

Table II.12: Output Gate Definitions for SAN Model *medium*

Gate	*Definition*
MOG1	*if(MARK(G1)==2|| MARK(G1)==12){ if (MARK(L12)==0) {MARK(L12)=1; MARK(G1)=200;} else{MARK(G1)=12;} } else if (MARK(G1)==3 ||MARK(G1)==13){ if (MARK(L13)==0) {MARK(L13)=1; MARK(G1)=200;} else{MARK(G1)=13;} } else if (MARK(G1)==4|| MARK(G1)==14){ if (MARK(L14)==0) {MARK(L14)=1; MARK(G1)=200;} else{MARK(G1)=14;} } else if (MARK(G1)==5 || MARK(G1)==15){ if (MARK(L15)==0) {MARK(L15)=1; MARK(G1)=200;} else{MARK(G1)=15;} } else if (MARK(G1)==6|| MARK(G1)==16){ if (MARK(L16)==0){MARK(L16)=1; MARK(G1)=200;} else{MARK(G1)=16;}} else if (MARK(G1)==500) { if (MARK(L12)==12) {MARK(L12)=500; MARK(G1)=0;} if (MARK(L13)==13) {MARK(L13)=500; MARK(G1)=0;} if (MARK(L14)==14) {MARK(L14)=500; MARK(G1)=0;} if (MARK(L15)==15) {MARK(L15)=500; MARK(G1)=0;} if (MARK(L16)==16) {MARK(L16)=500; MARK(G1)=0;}} else if (MARK(G1)==600){ { if (MARK(L12)==1 || MARK(L12)==101) {MARK(L12)=0; MARK(G1)=0;} else if (MARK(L12)==11 || MARK(L12)==111 || MARK(L12)==12){MARK(L12)=600; MARK(G1)=0;} else {MARK(G1)=0;} } { if (MARK(L13)==1|| MARK(L13)==101) {MARK(L13)=0; MARK(G1)=0;} else if (MARK(L13)==13) {MARK(L13)=600; MARK(G1)=0;} else {MARK(G1)=0;} } { if (MARK(L14)==1|| MARK(L14)==101) {MARK(L14)=0; MARK(G1)=0;} else if (MARK(L14)==14) {MARK(L14)=600; MARK(G1)=0;} else {MARK(G1)=0;}} { if (MARK(L15)==1|| MARK(L15)==101) {MARK(L15)=0; MARK(G1)=0;} else if (MARK(L15)==15) {MARK(L15)=600; MARK(G1)=0;} else {MARK(G1)=0;} } { if (MARK(L16)==1|| MARK(L16)==101) {MARK(L16)=0; MARK(G1)=0;} }*
MOG11	*if (MARK(L61) != 600) { if (MARK(L61) > 100) {MARK(ACK1)=MARK(L61)-110;} else {MARK(ACK1)=MARK(L61)-10;} if (MARK(L61)==16 || MARK(L61)==116) { MARK(L61)=61;} }*

Table II.12 (continue): Output Gate Definitions for SAN Model *medium*

Gate	*Definition*
MOG12	*if (MARK(L12)==600) { if (MARK(L22) > 0 && MARK(L22) < 999) {MARK(L12)=0; MARK(L22)=0; MARK(ACK2)=0;} else {MARK(L12)=0;} } else { if (MARK(L22)==0) { if (MARK(L12)==1 || MARK(L12)==101) { MARK(L22)=1; MARK(L12)=MARK(L12)+10;} else if (MARK(L12)==6 || MARK(L12)==106) {MARK(L22)=2; MARK(L12)=MARK(L12)+10;} else if (MARK(L12)==5 || MARK(L12)==105) {MARK(L22)=3; MARK(L12)=MARK(L12)+10;} else if (MARK(L12)==4 || MARK(L12)==104) {MARK(L22)=4; MARK(L12)=MARK(L12)+10;} else if (MARK(L12)==3 || MARK(L12)==103) {MARK(L22)=5; MARK(L12)=MARK(L12)+10;} } else if (MARK(L22)==123) {MARK(L22)=0; MARK(L12)=0;} else if (MARK(L22)==999) {MARK(L12)=MARK(L12)+100;} }*
MOG13	*if (MARK(L13)==600) { if ((MARK(L23)==1 || MARK(L23)==101) || (MARK(L23)==6 || MARK(L23)==106) || (MARK(L23)==5 || MARK(L23)==105) || (MARK(L23)==4 || MARK(L23)==104)) {MARK(L13)=0; MARK(L23)=0;} else {MARK(L13)=0; MARK(L23)=600;} } else { if (MARK(L23)==0) { if (MARK(L13)==1 || MARK(L13)==101) {MARK(L23)=1; MARK(L13)=13;} else if (MARK(L13)==6 || MARK(L13)==106) { MARK(L23)=6; MARK(L13)=63;} else if (MARK(L13)==5 || MARK(L13)==105) { MARK(L23)=5; MARK(L13)=53;} else if (MARK(L13)==4 || MARK(L13)==104) { MARK(L23)=4; MARK(L13)=43;} } else if (MARK(L23)==11 || MARK(L23)==111 || MARK(L23)==16 || MARK(L23)==116 || MARK(L23)==15 || MARK(L23)==115 || MARK(L23)==14 || MARK(L23)==114) {MARK(L23)=500; MARK(L13)=0;} }*

Table II.12 (continue): Output Gate Definitions for SAN Model *medium*

Gate	*Definition*
MOG14	*if (MARK(L14)==600)* *{ if ((MARK(L24)==1 || MARK(L24)==101) ||* *(MARK(L24)==6 || MARK(L24)==106)* *|| (MARK(L24)==5 || MARK(L24)==105))* *{MARK(L14)=0; MARK(L24)=0;}* *else {MARK(L14)=0; MARK(L24)=600;} }* *else {if (MARK(L24)==0) { if (MARK(L14)==1 ||* *MARK(L14)==101) {MARK(L24)=1; MARK(L14)=14;}* *else if (MARK(L14)==6 || MARK(L14)==106)* *{MARK(L24)=6; MARK(L14)=64;}* *else if (MARK(L14)==5 || MARK(L14)==105)* *{MARK(L24)=5; MARK(L14)=54;}}* *else if (MARK(L24)==14 || MARK(L24)==64 ||* *MARK(L24)==54) {MARK(L24)=500; MARK(L14)=0;}* *else {MARK(L14)=MARK(L14)+100;} }*
MOG15	*if (MARK(L15)==600)* *{ if ((MARK(L25)==1 || MARK(L25)==101) ||* *(MARK(L25)==6 || MARK(L25)==106))* *{MARK(L15)=0; MARK(L25)=0;}* *else {MARK(L15)=0; MARK(L25)=600;} }* *else { if (MARK(L25)==0){ if (MARK(L15)==1 ||* *MARK(L15)==101) {MARK(L25)=1; MARK(L15)=15;}* *else if (MARK(L15)==6 || MARK(L15)==106)* *{MARK(L25)=6; MARK(L15)=65;}}* *else if (MARK(L25)==15 || MARK(L25)==65)* *{MARK(L25)=500; MARK(L15)=0;}*
MOG16	*if (MARK(L16)==600)* *{ if (MARK(L26)==1 || MARK(L26)==101) {MARK(L16)=0;* *MARK(L26)=0;}* *else {MARK(L16)=0; MARK(L26)=600;} }* *else { if (MARK(L26)==0) {MARK(L26)=1; MARK(L16)=16;}* *else if (MARK(L26)==16) {MARK(L26)=500; MARK(L16)=0;}* *else {MARK(L16)=MARK(L16)+100;} }*

Table II.12 (continue): Output Gate Definitions for SAN Model *medium*

Gate	*Definition*
MOG2	*if (MARK(G2)==1 || MARK(G2)==21){ if (MARK(L21)==0) {MARK(L21)=2; MARK(G2)=200; } else {MARK(G2)=21;} } else if (MARK(G2)==3 || MARK(G2)==23){ if (MARK(L23)==0) {MARK(L23)=2; MARK(G2)=200;} else {MARK(G2)=23;} } else if (MARK(G2)==4 || MARK(G2)==24) { if (MARK(L24)==0) {MARK(L24)=2; MARK(G2)=200;} else {MARK(G2)=24;} } else if (MARK(G2)==5 || MARK(G2)==25){ if (MARK(L25)==0) {MARK(L25)=2; MARK(G2)=200;} else {MARK(G2)=25;} } else if (MARK(G2)==6 || MARK(G2)==26) { if (MARK(L26)==0) {MARK(L26)=2; MARK(G2)=200;} else {MARK(G2)=26;} } else if (MARK(G2)==500) { if (MARK(L21)==21) {MARK(L21)=500; MARK(G2)=0;} if (MARK(L23)==23) {MARK(L23)=500; MARK(G2)=0;} if (MARK(L24)==24) {MARK(L24)=500; MARK(G2)=0;} if (MARK(L25)==25) {MARK(L25)=500; MARK(G2)=0;} if (MARK(L26)==26) {MARK(L26)=500; MARK(G2)=0;} } else if (MARK(G2)==600) {{ if (MARK(L21)==2 || MARK(L21)==102) {MARK(L21)=0; MARK(G2)=0;} else if (MARK(L21)==21) {MARK(L21)=600; MARK(G2)=0;} else {MARK(G2)=0;}} { if (MARK(L23)==2 || MARK(L23)==102) {MARK(L23)=0; MARK(G2)=0;} else if (MARK(L23)==12 || MARK(L23)==112 || MARK(L23)==23) {MARK(L23)=600; MARK(G2)=0;} else {MARK(G2)=0;} } { if (MARK(L24)==2 || MARK(L24)==102) {MARK(L24)=0; MARK(G2)=0;} else if (MARK(L24)==24) {MARK(L24)=600; MARK(G2)=0;} else {MARK(G2)=0;} } { if (MARK(L25)==2 || MARK(L25)==102) {MARK(L25)=0; MARK(G2)=0;} else if (MARK(L25)==25) {MARK(L25)=600; MARK(G2)=0;} else {MARK(G2)=0;} } { if (MARK(L26)==2 || MARK(L26)==102) {MARK(L26)=0; MARK(G2)=0;} }*
MOG21	*if (MARK(L21)==600) { if (MARK(L31)==2 || MARK(L31)==102) {MARK(L21)=0; MARK(L31)=0;} else {MARK(L21)=0; MARK(L31)=600;} } else { if (MARK(L31)==0) { MARK(L31)=2; MARK(L21)=21;} else if (MARK(L31)==21) {MARK(L31)=500; MARK(L21)=0;} else{MARK(L21)=MARK(L21)+100;}}*

Table II.12 (continue): Output Gate Definitions for SAN Model *medium*

Gate	*Definition*
MOG22	*if (MARK(L12) != 600)* *{ if (MARK(L12) > 100) { MARK(ACK2)=MARK(L12)-110;}* *else {MARK(ACK2)=MARK(L12)-10;}* *if (MARK(L12)==11 || MARK(L12)==111) {MARK(L12)=12;} }*
MOG23	*if (MARK(L23)==600){ if (MARK(L33) > 0 && MARK(L33) < 999)* *{MARK(L23)=0; MARK(L33)=0; MARK(ACK3)=0;}* *else {MARK(L23)=0;} }* *else { if (MARK(L33)==0) { if (MARK(L23)==2 || MARK(L23)==102)* *{MARK(L33)=1; MARK(L23)=MARK(L23)+10;}* *else if (MARK(L23)==1 || MARK(L23)==101) {MARK(L33)=2;* *MARK(L23)=MARK(L23)+10; }* *else if (MARK(L23)==6 || MARK(L23)==106) {MARK(L33)=3;* *MARK(L23)=MARK(L23)+10; }* *else if (MARK(L23)==5 || MARK(L23)==105) {MARK(L33)=4;* *MARK(L23)=MARK(L23)+10; }* *else if (MARK(L23)==4 || MARK(L23)==104) {MARK(L33)=5;* *MARK(L23)=MARK(L23)+10; } }* *else if (MARK(L33)==123) {MARK(L33)=0; MARK(L23)=0;}}*
MOG24	*if (MARK(L24)==600)* *{ if ((MARK(L34)==2 || MARK(L34)==102) ||* *(MARK(L34)==1 || MARK(L34)==101)* *|| (MARK(L34)==6 || MARK(L34)==106) ||* *(MARK(L34)==5 || MARK(L34)==105))* *{MARK(L24)=0; MARK(L34)=0;}* *else {MARK(L24)=0;MARK(L34)=600;} }* *else { if (MARK(L34)==0) { if (MARK(L24)==2 ||* *MARK(L24)==102) {MARK(L34)=2; MARK(L24)=24;}* *else if (MARK(L24)==1 || MARK(L24)==101)* *{MARK(L34)=1; MARK(L24)=14;}* *else if (MARK(L24)==6 || MARK(L24)==106)* *{MARK(L34)=6; MARK(L24)=64;}* *else if (MARK(L24)==5 || MARK(L24)==105)* *{MARK(L34)=5; MARK(L24)=54;} }* *else if (MARK(L34)==12 || MARK(L34)==112 ||* *MARK(L34)==11 || MARK(L34)==111 || MARK(L34)==16 ||* *MARK(L34)==116 || MARK(L34)==15 ||* *MARK(L34)==115)* *{MARK(L34)=500; MARK(L24)=0;}}*

Table II.12 (continue): Output Gate Definitions for SAN Model *medium*

Gate	*Definition*
MOG25	*if (MARK(L25)==600)* *{ if ((MARK(L35)==2 || MARK(L35)==102) ||* *(MARK(L35)==1 || MARK(L35)==101)* *|| (MARK(L35)==6 || MARK(L35)==106))* *{MARK(L25)=0; MARK(L35)=0;}* *else {MARK(L25)=0; MARK(L35)=600;} }* *else { if (MARK(L35)==0) {if (MARK(L25)==2 ||* *MARK(L25)==102){MARK(L35)=2; MARK(L25)=25;}* *else if (MARK(L25)==1 || MARK(L25)==101)* *{MARK(L35)=1; MARK(L25)=15;}* *else if (MARK(L25)==6 || MARK(L25)==106)* *{MARK(L35)=6; MARK(L25)=65;} }* *else if (MARK(L35)==25 || MARK(L35)==15 ||* *MARK(L35)==65)* *{MARK(L35)=500; MARK(L25)=0;}* *else {MARK(L25)=MARK(L25)+100;} }*
MOG26	*if (MARK(L26)==600)* *{ if ((MARK(L36)==2 || MARK(L36)==102) ||* *(MARK(L36)==1 || MARK(L36)==101))* *{MARK(L26)=0; MARK(L36)=0;}* *else {MARK(L26)=0; MARK(L36)=600;} }* *else { if (MARK(L36)==0) { if (MARK(L26)==2 ||* *MARK(L26)==102) {MARK(L36)=2; MARK(L26)=26;}* *else if (MARK(L26)==1 || MARK(L26)==101)* *{MARK(L36)=1; MARK(L26)=16;}}* *else if (MARK(L36)==26 || MARK(L36)==16) {MARK(L36)=500;* *MARK(L26)=0;}* *else {MARK(L26)=MARK(L26)+100;} }*

Table II.12 (continue): Output Gate Definitions for SAN Model *medium*

Gate	*Definition*
MOG3	*if (MARK(G3)==1|| MARK(G3)==31) { if (MARK(L31)==0) {MARK(L31)=3; MARK(G3)=200;} else {MARK(G3)=31;} } else if (MARK(G3)==2 || MARK(G3)==32) { if (MARK(L32)==0) {MARK(L32)=3; MARK(G3)=200;} else {MARK(G3)=32;}} else if (MARK(G3)==4 || MARK(G3)==34) { if (MARK(L34)==0) {MARK(L34)=3; MARK(G3)=200;} else {MARK(G3)=34;} } else if (MARK(G3)==5 || MARK(G3)==35) { if (MARK(L35)==0) {MARK(L35)=3; MARK(G3)=200;} else {MARK(G3)=35;} } else if (MARK(G3)==6 || MARK(G3)==36) { if (MARK(L36)==0) {MARK(L36)=3; MARK(G3)=200;} else {MARK(G3)=36;} } else if (MARK(G3)==500) { if (MARK(L31)==31){ MARK(L31)=500; MARK(G3)=0;} if (MARK(L32)==32) {MARK(L32)=500; MARK(G3)=0;} if (MARK(L34)==34) {MARK(L34)=500; MARK(G3)=0;} if (MARK(L35)==35) {MARK(L35)=500; MARK(G3)=0;} if (MARK(L36)==36){MARK(L36)=500; MARK(G3)=0;}} else if (MARK(G3)==600) { { if (MARK(L31)==3 || MARK(L31)==103) {MARK(L31)=0; MARK(G3)=0;} else if (MARK(L31)==31) {MARK(L31)=600; MARK(G3)=0;} else {MARK(G3)=0;} } { if (MARK(L32)==3 || MARK(L32)==103) {MARK(L32)=0; MARK(G3)=0;} else if (MARK(L32)==32) {MARK(L32)=600; MARK(G3)=0;} else {MARK(G3)=0;} } { if (MARK(L34)==3 || MARK(L34)==103) {MARK(L34)=0; MARK(G3)=0;} else if (MARK(L34)==13 ||MARK(L34)==113 || MARK(L34)==34) {MARK(L34)=600; MARK(G3)=0;} else {MARK(G3)=0;} } { if (MARK(L35)==3 || MARK(L35)==103) {MARK(L35)=0; MARK(G3)=0;} else if (MARK(L35)==35) {MARK(L35)=600; MARK(G3)=0;} else {MARK(G3)=0;} } {if (MARK(L36)==3 || MARK(L36)==103) {MARK(L36)=0; MARK(G3)=0;} }*

Table II.12 (continue): Output Gate Definitions for SAN Model *medium*

Gate	*Definition*
MOG31	*if (MARK(L31)==600)* *{ if ((MARK(L41)==3 || MARK(L41)==103) || (MARK(L41)==2 || MARK(L41)==102)) {MARK(L31)=0; MARK(L41)=0;}* *else {MARK(L31)=0; MARK(L41)=600;}}* *else { if (MARK(L41)==0) { if (MARK(L31)==3 || MARK(L31)==103) {MARK(L41)=3; MARK(L31)=31;}* *else if (MARK(L31)==2 || MARK(L31)==102) {MARK(L41)=2; MARK(L31)=21;} }* *else if (MARK(L41)==31 || MARK$(L41)==21){MARK(L41)=500; MARK(L31)=0;} else {MARK(L31)=MARK(L31)+100;} }*
MOG32	*if (MARK(L32)==600)* *{ if (MARK(L42)==3 || MARK(L42)==103) {MARK(L32)=0; MARK(L42)=0;} else {MARK(L32)=0; MARK(L42)=600;} }* *else { if (MARK(L42)==0) {MARK(L42)=3; MARK(L32)=32;}* *else if (MARK(L42)==32) {MARK(L42)=500; MARK(L32)=0;}* *else {MARK(L32)=MARK(L32)+100;} }*
MOG33	*if (MARK(L23) != 600)* *{ if (MARK(L23) > 100) {MARK(ACK3)=MARK(L23)-110;}* *else {MARK(ACK3)=MARK(L23)-10;}* *if (MARK(L23)==12 || MARK(L23)==112) {MARK(L23)=23;} }*
MOG34	*if (MARK(L34)==600) { if (MARK(L44) > 0 && MARK(L44) < 999) {MARK(L34)=0; MARK(L44)=0; MARK(ACK4)=0;}* *else {MARK(L34)=0;}} else { if (MARK(L44)==0) { if (MARK(L34)==3 || MARK(L34)==103) {MARK(L44)=1; MARK(L34)=MARK(L34)+10; }* *else if (MARK(L34)==2 || MARK(L34)==102) {MARK(L44)=2; MARK(L34)=MARK(L34)+10; }* *else if (MARK(L34)==1 || MARK(L34)==101) {MARK(L44)=3; MARK(L34)=MARK(L34)+10; }* *else if (MARK(L34)==6 || MARK(L34)==106) {MARK(L44)=4; MARK(L34)=MARK(L34)+10;}* *else if (MARK(L34)==5 || MARK(L34)==105) {MARK(L44)=5; MARK(L34)=MARK(L34)+10; } }* *else if (MARK(L44)==123) {MARK(L44)=0; MARK(L34)=0;}* *else if (MARK(L44)==999) {MARK(L34)=MARK(L34)+100;} }*

Table II.12 (continue): Output Gate Definitions for SAN Model *medium*

Gate	*Definition*
MOG35	*if (MARK(L35)==600) { if ((MARK(L45)==3 || MARK(L45)==103) || (MARK(L45)==2 || MARK(L45)==102) || (MARK(L45)==1 || MARK(L45)==101) || (MARK(L45)==6 || MARK(L45)==106)) {MARK(L35)=0; MARK(L45)=0;} else {MARK(L35)=0; MARK(L45)=600;} } else { if (MARK(L45)==0) { if (MARK(L35)==3 || MARK(L35)==103) {MARK(L45)=3; MARK(L35)=35;} else if (MARK(L35)==2 || MARK(L35)==102) {MARK(L45)=2; MARK(L35)=25;} else if (MARK(L35)==1 || MARK(L35)==101) {MARK(L45)=1; MARK(L35)=15; else if (MARK(L35)==6 || MARK(L35)==106) {MARK(L45)=6; MARK(L35)=65;}} else if (MARK(L45)==13 || MARK(L45)==113 || MARK(L45)==12 || MARK(L45)==112 || MARK(L45)==11 || MARK(L45)==111 || MARK(L45)==16 || MARK(L45)==116) {MARK(L45)=500; MARK(L35)=0;} else {MARK(L35)=MARK(L35)+100;} }*
MOG36	*if (MARK(L36)==600) { if ((MARK(L46)==3 || MARK(L46)==103) || (MARK(L46)==2 || MARK(L46)==102) || (MARK(L46)==1 || MARK(L46)==101)) { MARK(L36)=0; MARK(L46)=0;} else {MARK(L36)=0; MARK(L46)=600;} } else { if (MARK(L46)==0) { if (MARK(L36)==3 || MARK(L36)==103) {MARK(L46)=3; MARK(L36)=36;} else if (MARK(L36)==2 || MARK(L36)==102) {MARK(L46)=2; MARK(L36)=26;} else if (MARK(L36)==1 || MARK(L36)==101) {MARK(L46)=1; MARK(L36)=16;} } else if (MARK(L46)==36 || MARK(L46)==26 || MARK(L46)==16) {MARK(L46)=500; MARK(L36)=0;} else {MARK(L36)=MARK(L36)+100;} }*

Table II.12 (continue): Output Gate Definitions for SAN Model *medium*

Gate	*Definition*
MOG4	*if (MARK(G4)==1 || MARK(G4)==41) { if (MARK(L41)==0) {MARK(L41)=4; MARK(G4)=200;} else {MARK(G4)=41;} } else if (MARK(G4)==2 || MARK(G4)==42) { if (MARK(L42)==0) {MARK(L42)=4; MARK(G4)=200;} else {MARK(G4)=42;} } else if (MARK(G4)==3 || MARK(G4)==43) { if (MARK(L43)==0){MARK(L43)=4; MARK(G4)=200;} else {MARK(G4)=43;} } else if (MARK(G4)==5 || MARK(G4)==45) { if (MARK(L45)==0) {MARK(L45)=4; MARK(G4)=200;} else {MARK(G4)=45;} } else if (MARK(G4)==6 || MARK(G4)==46) { if (MARK(L46)==0){MARK(L46)=4; MARK(G4)=200;} else {MARK(G4)=46;} } else if (MARK(G4)==500) { if (MARK(L41)==41) {MARK(L41)=500; MARK(G4)=0;} if (MARK(L42)==42) {MARK(L42)=500; MARK(G4)=0;} if (MARK(L43)==43) {MARK(L43)=500; MARK(G4)=0;} if (MARK(L45)==45) {MARK(L45)=500; MARK(G4)=0;} if (MARK(L46)==46) {MARK(L46)=500; MARK(G4)=0;}} else if (MARK(G4)==600) { { if (MARK(L41)==4 || MARK(L41)==104) {MARK(L41)=0; MARK(G4)=0;} else if (MARK(L41)==41) {MARK(L41)=600; MARK(G4)=0;} else {MARK(G4)=0;} } { if (MARK(L42)==4 || MARK(L42)==104) {MARK(L42)=0; MARK(G4)=0;} else if (MARK(L42)==42) {MARK(L42)=600; MARK(G4)=0;} else {MARK(G4)=0;} } { if (MARK(L43)==4 || MARK(L43)==104) {MARK(L43)=0; MARK(G4)=0;} else if (MARK(L43)==43) {MARK(L43)=600; MARK(G4)=0;} else {MARK(G4)=0;} } { if (MARK(L45)==4 || MARK(L45)==104) {MARK(L45)=0; MARK(G4)=0;} else if (MARK(L45)==14 || MARK(L45)==114 || MARK(L45)==45) {MARK(L45)=600; MARK(G4)=0;} else {MARK(G4)=0; } } { if (MARK(L46)==4 || MARK(L46)==104) {MARK(L46)=0; MARK(G4)=0;} }*

Table II.12 (continue): Output Gate Definitions for SAN Model *medium*

Gate	*Definition*
MOG41	*if (MARK(L41)==600)* *{if ((MARK(L51)==4 || MARK(L51)==104) || (MARK(L51)==3 || MARK(L51)==103) || (MARK(L51)==2 || MARK(L51)==102))* *{MARK(L41)=0; MARK(L51)=0;}* *else {MARK(L41)=0; MARK(L51)=600;} }* *else { if (MARK(L51)==0){ if (MARK(L41)==4 || MARK(L41)==104) {MARK(L51)=4; MARK(L41)=41;}* *else if (MARK(L41)==3 || MARK(L41)==103) {MARK(L51)=3; MARK(L41)=31;}* *else if (MARK(L41)==2 || MARK(L41)==102) {MARK(L51)=2; MARK(L41)=21;} }* *else if (MARK(L51)==41 || MARK(L51)==31 || MARK(L51)==21)* *{MARK(L51)=500; MARK(L41)=0;}* *else {MARK(L41)=MARK(L41)+100;}}
MOG42	*if (MARK(L42)==600)* *{ if ((MARK(L52)==4 || MARK(L52)==104) || (MARK(L52)==3 || MARK(L52)==103))* *{MARK(L42)=0; MARK(L52)=0;}* *else {MARK(L42)=0; MARK(L52)=600;} }* *else { if (MARK(L52)==0) { if (MARK(L42)==4 || MARK(L42)==104){ MARK(L52)=4; MARK(L42)=42;}* *else if (MARK(L42)==3 || MARK(L42)==103) { MARK(L52)=3; MARK(L42)=32;} }* *else if (MARK(L52)==42 || MARK(L52)==32) {MARK(L52)=500; MARK(L42)=0;}* *else {MARK(L42)=MARK(L42)+100;} }
MOG43	*if (MARK(L43)==600)* *{ if (MARK(L53)==4 || MARK(L53)==104) {MARK(L43)=0; MARK(L53)=0;}* *else {MARK(L43)=0; MARK(L53)=600;} }* *else { if (MARK(L53)==0) {MARK(L53)=4; MARK(L43)=43;}* *else if (MARK(L53)==43) {MARK(L53)=500; MARK(L43)=0;}* *else {MARK(L43)=MARK(L43)+100;} }
MOG44	*if (MARK(L34) != 600)* *{ if (MARK(L34) > 100) {MARK(ACK4)=MARK(L34)-110;}* *else {MARK(ACK4)=MARK(L34)-10;}* *if (MARK(L34)==13 || MARK(L34)==113){MARK(L34)=34;} }

Table II.12 (continue): Output Gate Definitions for SAN Model *medium*

Gate	*Definition*
MOG45	*if (MARK(L45)==600) { if (MARK(L55) > 0 && MARK(L55) < 999) {MARK(L45)=0; MARK(L55)=0; MARK(ACK5)=0;} else {MARK(L45)=0;} } else { if (MARK(L55)==0) { if (MARK(L45)==4 || MARK(L45)==104) {MARK(L55)=1; MARK(L45)=MARK(L45)+10;} else if (MARK(L45)==3 || MARK(L45)==103) {MARK(L55)=2; MARK(L45)=MARK(L45)+10;} else if (MARK(L45)==2 || MARK(L45)==102) {MARK(L55)=3; MARK(L45)=MARK(L45)+10;} else if (MARK(L45)==1 || MARK(L45)==101) {MARK(L55)=4; MARK(L45)=MARK(L45)+10;} else if (MARK(L45)==6 || MARK(L45)==106) {MARK(L55)=5; MARK(L45)=MARK(L45)+10;} } else if (MARK(L55)==123) {MARK(L55)=0; MARK(L45)=0;} else if (MARK(L55)==999) {MARK(L45)=MARK(L45)+100;} } }*
MOG46	*if (MARK(L46)==600) { if ((MARK(L56)==4 || MARK(L56)==104) || (MARK(L56)==3 || MARK(L56)==103) || (MARK(L56)==2 || MARK(L56)==102) || (MARK(L56)==1 || MARK(L56)==101)) {MARK(L46)=0; MARK(L56)=0;} else {MARK(L46)=0; MARK(L56)=600;} } else { if (MARK(L56)==0) { if (MARK(L46)==4 || MARK(L46)==104) {MARK(L56)=4; MARK(L46)=46;} else if (MARK(L46)==3 || MARK(L46)==103) {MARK(L56)=3; MARK(L46)=36;} else if (MARK(L46)==2 || MARK(L46)==102) {MARK(L56)=2; MARK(L46)=26;} else if (MARK(L46)==1 || MARK(L46)==101) {MARK(L56)=1; MARK(L46)=16;} } else if (MARK(L56)==14 || MARK(L56)==114 || MARK(L56)==13 || MARK(L56)==113 || MARK(L56)==12 || MARK(L56)==112 || MARK(L56)==11 || MARK(L56)==111) {MARK(L56)=500; MARK(L46)=0;} }*

Table II.12 (continue): Output Gate Definitions for SAN Model *medium*

Gate	*Definition*
MOG5	*if (MARK(G5)==1 || MARK(G5)==51) { if (MARK(L51)==0) {MARK(L51)=5; MARK(G5)=200;} else {MARK(G5)=51;}} else if (MARK(G5)==2 || MARK(G5)==52) { if (MARK(L52)==0) {MARK(L52)=5; MARK(G5)=200;} else {MARK(G5)=52;}} else if (MARK(G5)==3 || MARK(G5)==53){ if (MARK(L53)==0){MARK(L53)=5; MARK(G5)=200;} else {MARK(G5)=53;}} else if (MARK(G5)==4 || MARK(G5)==54) {if (MARK(L54)==0) {MARK(L54)=5; MARK(G5)=200;} else {MARK(G5)=54;}} else if (MARK(G5)==6 || MARK(G5)==56) { if (MARK(L56)==0) {MARK(L56)=5; MARK(G5)=200;} else {MARK(G5)=56;}} else if (MARK(G5)==500) {if (MARK(L51)==51) {MARK(L51)=500; MARK(G5)=0;} if (MARK(L52)==52) {MARK(L52)=500; MARK(G5)=0;} if (MARK(L53)==53) {MARK(L53)=500; MARK(G5)=0;} if (MARK(L54)==54) {MARK(L54)=500; MARK(G5)=0;} if (MARK(L56)==56) {MARK(L56)=500; MARK(G5)=0;} } else if (MARK(G5)==600) { { if (MARK(L51)==5 || MARK(L51)==105) {MARK(L51)=0; MARK(G5)=0;} else if (MARK(L51)==51) {MARK(L51)=600; MARK(G5)=0;} else {MARK(G5)=0;} } { if (MARK(L52)==5 || MARK(L52)==105) {MARK(L52)=0; MARK(G5)=0;} else if (MARK(L52)==52) {MARK(L52)=600; MARK(G5)=0;} else {MARK(G5)=0;} } { if (MARK(L53)==5 || MARK(L53)==105) {MARK(L53)=0; MARK(G5)=0;} else if (MARK(L53)==53) {MARK(L53)=600; MARK(G5)=0;} else {MARK(G5)=0;} } { if (MARK(L54)==5 || MARK(L54)==105) {MARK(L54)=0; MARK(G5)=0;} else if (MARK(L54)==54) {MARK(L54)=600; MARK(G5)=0;} else {MARK(G5)=0;} } { if (MARK(L56)==5 || MARK(L56)==105) {MARK(L56)=0; MARK(G5)=0;} }*

Table II.12 (continue): Output Gate Definitions for SAN Model *medium*

Gate	*Definition*
MOG51	*if (MARK(L51)==600)* *{ if ((MARK(L61)==5 || MARK(L61)==105) ||* *(MARK(L61)==4 || MARK(L61)==104)* *|| (MARK(L61)==3 || MARK(L61)==103) ||* *(MARK(L61)==2 || MARK(L61)==102))* *{MARK(L51)=0; MARK(L61)=0;}* *else {MARK(L51)=0; MARK(L61)=600;} }* *else {if (MARK(L61)==0) { if (MARK(L51)==5 ||* *MARK(L51)==105) {MARK(L61)=5; MARK(L51)=51;}* *else if (MARK(L51)==4 || MARK(L51)==104)* *{MARK(L61)=4; MARK(L51)=41;}* *else if (MARK(L51)==3 || MARK(L51)==103)* *{MARK(L61)=3; MARK(L51)=31;}* *else if (MARK(L51)==2 || MARK(L51)==102)* *{MARK(L61)=2; MARK(L51)=21;}}* *else if (MARK(L61)==15 || MARK(L61)==115 ||* *MARK(L61)==14 || MARK(L61)==114 ||* *MARK(L61)==13 || MARK(L61)==113 || MARK(L61)==12 ||* *MARK(L61)==112)* *{MARK(L61)=500; MARK(L51)=0;}* *else {MARK(L51)=MARK(L51)+100;} }*
MOG52	*if (MARK(L52)==600)* *{ if ((MARK(L62)==5 || MARK(L62)==105)||* *(MARK(L62)==4 || MARK(L62)==104)* *|| (MARK(L62)==3 || MARK(L62)==103))* *{MARK(L52)=0; MARK(L62)=0;}* *else {MARK(L52)=0; MARK(L62)=600;} }* *else { if (MARK(L62)==0) { if (MARK(L52)==5 ||* *MARK(L52)==105) {MARK(L62)=5; MARK(L52)=52;}* *else if (MARK(L52)==4 || MARK(L52)==104)* *{MARK(L62)=4; MARK(L52)=42;}* *else if (MARK(L52)==3 || MARK(L52)==103)* *{MARK(L62)=3; MARK(L52)=32;} }* *else if (MARK(L62)==52 || MARK(L62)==42 ||* *MARK(L62)==32)* *{MARK(L62)=500; MARK(L52)=0;} }*

Table II.12 (continue): Output Gate Definitions for SAN Model *medium*

Gate	*Definition*
MOG53	*if (MARK(L53)==600)* *{ if ((MARK(L63)==5 || MARK(L63)==105) || (MARK(L63)==4 || MARK(L63)==104)) {MARK(L53)=0; MARK(L63)=0;}* *else {MARK(L53)=0; MARK(L63)=600;} }* *else { if (MARK(L63)==0) { if (MARK(L53)==5 || MARK(L53)==105) {MARK(L63)=5; MARK(L53)=53;}* *else if (MARK(L53)==4 || MARK(L53)==104) {MARK(L63)=4; MARK(L53)=43;} }* *else if (MARK(L63)==53 || MARK(L63)==43) {MARK(L63)=500; MARK(L53)=0;}* *else {MARK(L53)=MARK(L53)+100;} }*
MOG54	*if (MARK(L54)==600)* *{ if (MARK(L64)==5 || MARK(L64)==105) {MARK(L54)=0; MARK(L64)=0;}* *else {MARK(L54)=0; MARK(L64)=600;} }* *else { if (MARK(L64)==0) {MARK(L64)=5; MARK(L54)=54;}* *else if (MARK(L64)==54) {MARK(L64)=500; MARK(L54)=0;}* *else {MARK(L54)=MARK(L54)+100;} }*
MOG55	*if (MARK(L45) != 600)* *{ if (MARK(L45) > 100) {MARK(ACK5)=MARK(L45)-110;}* *else {MARK(ACK5)=MARK(L45)-10;}* *if (MARK(L45)==14 || MARK(L45)==114) {MARK(L45)=45;}}*
MOG56	*if (MARK(L56)==600) { if (MARK(L66) > 0 && MARK(L66) < 999){MARK(L56)=0; MARK(L66)=0;MARK(ACK6)=0;}* *else{MARK(L56)=0;} }* *else { if (MARK(L66)==0) { if (MARK(L56)==5 || MARK(L56)==105)* *{MARK(L66)=1; MARK(L56)=MARK(L56)+10;}* *else if (MARK(L56)==4 || MARK(L56)==104) {MARK(L66)=2; MARK(L56)=MARK(L56)+10;}* *else if (MARK(L56)==3 || MARK(L56)==103) {MARK(L66)=3; MARK(L56)=MARK(L56)+10;}* *else if (MARK(L56)==2 || MARK(L56)==102) {MARK(L66)=4; MARK(L56)=MARK(L56)+10;}* *else if (MARK(L56)==1 || MARK(L56)==101) {MARK(L66)=5; MARK(L56)=MARK(L56)+10;} }* *else if (MARK(L66)==123) {MARK(L66)=0; MARK(L56)=0;}* *else if (MARK(L66)==999) {MARK(L56)=MARK(L56)+100;}}*

Table II.12 (continue): Output Gate Definitions for SAN Model *medium*

Gate	*Definition*
MOG6	*if (MARK(G6)==1 || MARK(G6)==61) { if (MARK(L61)==0) {MARK(L61)=6; MARK(G6)=200;} else {MARK(G6)=61;} } else if (MARK(G6)==2 || MARK(G6)==62) { if (MARK(L62)==0) {MARK(L62)=6; MARK(G6)=200;} else {MARK(G6)=62;} } if (MARK(G6)==3 || MARK(G6)==63) { if (MARK(L63)==0) {MARK(L63)=6; MARK(G6)=200;} else {MARK(G6)=63;} } else if (MARK(G6)==4 || MARK(G6)==64){ if (MARK(L64)==0){MARK(L64)=6; MARK(G6)=200;} else {MARK(G6)=64;} } else if (MARK(G6)==5 || MARK(G6)==65) { if (MARK(L65)==0) {MARK(L65)=6; MARK(G6)=200;} else {MARK(G6)=65;} } else if (MARK(G6)==500) { if (MARK(L61)==61) {MARK(L61)=500; MARK(G6)=0;} if (MARK(L62)==62) {MARK(L62)=500; MARK(G6)=0;} if (MARK(L63)==63) {MARK(L63)=500; MARK(G6)=0;} if (MARK(L64)==64) {MARK(L64)=500; MARK(G6)=0;} if (MARK(L65)==65) {MARK(L65)=500; MARK(G6)=0;} } else if (MARK(G6)==600) { { if (MARK(L61)==6 || MARK(L61)==106) {MARK(L61)=0; MARK(G6)=0;} else if (MARK(L61)==16 || MARK(L61)==116 || MARK(L61)==61) { MARK(L61)=600; MARK(G6)=0;} else {MARK(G6)=0;} } { if (MARK(L62)==6 || MARK(L62)==106) {MARK(L62)=0; MARK(G6)=0;} else if (MARK(L62)==62) {MARK(L62)=600; MARK(G6)=0;} else{MARK(G6)=0;} } { if (MARK(L63)==6 || MARK(L63)==106) {MARK(L63)=0; MARK(G6)=0;} else if (MARK(L63)==63){MARK(L63)=600; MARK(G6)=0;} else {MARK(G6)=0;} } { if (MARK(L64)==6 || MARK(L64)==106) {MARK(L64)=0; MARK(G6)=0;} else if (MARK(L64)==64) {MARK(L64)=600; MARK(G6)=0;} else {MARK(G6)=0;} } { if (MARK(L65)==6 || MARK(L65)==106) {MARK(L65)=0; MARK(G6)=0;}*

Table II.12 (continue): Output Gate Definitions for SAN Model *medium*

Gate	*Definition*
MOG61	*if (MARK(L61)==600) {if (MARK(L11) > 0 && MARK(L11) < 999) {MARK(L61)=0; MARK(L11)=0; MARK(ACK1)=0;} else {MARK(L61)=0;} } else { if (MARK(L11)==0) { if (MARK(L61)==6 || MARK(L61)==106) {MARK(L11)=1; MARK(L61)=MARK(L61)+10;} else if (MARK(L61)==5 || MARK(L61)==105) {MARK(L11)=2; MARK(L61)=MARK(L61)+10;} else if (MARK(L61)==4 || MARK(L61)==104) {MARK(L11)=3; MARK(L61)=MARK(L61)+10;} else if (MARK(L61)==3 || MARK(L61)==103) {MARK(L11)=4; MARK(L61)=MARK(L61)+10;} else if (MARK(L61)==2 || MARK(L61)==102){MARK(L11)=5; MARK(L61)=MARK(L61)+10;}} else if (MARK(L11)==123){MARK(L11)=0; MARK(L61)=0;} else if (MARK(L11)==999) {MARK(L61)=MARK(L61)+100;} }*
MOG62	*if (MARK(L62)==600) { if ((MARK(L12)==6 || MARK(L12)==106) || (MARK(L12)==5 || MARK(L12)==105) || (MARK(L12)==4 || MARK(L12)==104) || (MARK(L12)==3 || MARK(L12)==103)) {MARK(L62)=0; MARK(L12)=0;} else {MARK(L62)=0; MARK(L12)=600;} } else { if (MARK(L12)==0){if (MARK(L62)==6 || MARK(L62)==106) {MARK(L12)=6; MARK(L62)=62;} else if (MARK(L62)==5 || MARK(L62)==105) {MARK(L12)=5; MARK(L62)=52;} else if (MARK(L62)==4 || MARK(L62)==104) {MARK(L12)=4; MARK(L62)=42;} else if (MARK(L62)==3 || MARK(L62)==103) {MARK(L12)=3; MARK(L62)=32;} } else if (MARK(L12)==16 || MARK(L12)==116 || MARK(L12)==15 || MARK(L12)==115 || MARK(L12)==14 || MARK(L12)==114 || MARK(L12)==13 || MARK(L12)==113) {MARK(L12)=500; MARK(L62)=0;} else {MARK(L62)=MARK(L62)+100;} }*

Table II.12 (continue): Output Gate Definitions for SAN Model *medium*

Gate	*Definition*
MOG63	*if (MARK(L63)==600)* *{ if ((MARK(L13)==6 || MARK(L13)==106) ||* *(MARK(L13)==5 || MARK(L13)==105)* *|| (MARK(L13)==4 || MARK(L13)==104))* *{MARK(L63)=0; MARK(L13)=0;}* *else {MARK(L63)=0; MARK(L13)=600;} }* *else { if (MARK(L13)==0) { if (MARK(L63)==6 ||* *MARK(L63)==106) {MARK(L13)=6; MARK(L63)=63;}* *else if (MARK(L63)==5 || MARK(L63)==105)* *{MARK(L13)=5; MARK(L63)=53;}* *else if (MARK(L63)==4 || MARK(L63)==104)* *{MARK(L13)=4; MARK(L63)=43;} }* *else if (MARK(L13)==63 || MARK(L13)==53 ||* *MARK(L13)==43)* *{MARK(L13)=500; MARK(L63)=0;} }*
MOG64	*if (MARK(L64)==600)* *{ if ((MARK(L14)==6 || MARK(L14)==106) ||* *(MARK(L14)==5 || MARK(L14)==105))* *{MARK(L64)=0; MARK(L14)=0;}* *else {MARK(L64)=0; MARK(L14)=600;} }* *else { if (MARK(L14)==0) { if (MARK(L64)==6 ||* *MARK(L64)==106){MARK(L14)=6; MARK(L64)=64;}* *else if (MARK(L64)==5 || MARK(L64)==105)* *{MARK(L14)=5; MARK(L64)=54;} }* *else if (MARK(L14)==64 || MARK(L14)==54) {MARK(L14)=500;* *MARK(L64)=0; else {MARK(L64)=MARK(L64)+100;} }*
MOG65	*if (MARK(L65)==600)* *{if (MARK(L15)==6 || MARK(L15)==106) {MARK(L65)=0;* *MARK(L15)=0;}* *else {MARK(L65)=0; MARK(L15)=600;} }* *else {if (MARK(L15)==0) {MARK(L15)=6; MARK(L65)=65;}* *else if (MARK(L15)==65) {MARK(L15)=500; MARK(L65)=0;} }*
MOG66	*if (MARK(L56) != 600)* *{ if (MARK(L56) > 100) {MARK(ACK6)=MARK(L56)-110;}* *else {MARK(ACK6)=MARK(L56)-10;}* *if (MARK(L56)==15 || MARK(L56)==115) {MARK(L56)=56;}}*