A Reservation-Based Multicast Protocol for WDM Optical Star Networks

Kshirasagar Naik, *Member, IEEE*, David S. L. Wei, *Member, IEEE*, Danny Krizanc, and Sy-Yen Kuo, *Fellow, IEEE*

***Abstract*—In this paper, we present a reservation-based medium access control (MAC) protocol with multicast support for wavelength-division multiplexing networks. Our system is based on the single-hop, passive optical star architecture. Of the available wavelengths (channels), one channel is designated as a control channel, and the remaining channels are used for data transmission. Each node is equipped with a pair of fixed transceiver to access the control channel, and a fixed transmitter and a tunable receiver to access data channels. For easy implementation of th*e protocol in hardware and for precisely computing the protocol's processing overhead, we give a register-transfer model of the protocol. We simulate the protocol to study its throughput behavior, and present its analytic model. For a node to be able to send data packets in successive data slots with no time gap between them, in spite of the situation that the protocol's execution time may be longer than data transmission time, we propose the idea of multiple MAC units at each node. Unicast throughput of our protocol reaches the theoretically possible maximum throughput for MAC protocols with distributed control, and the multicast throughput is at least as good as, and even better than, those delivered by existing MAC protocols with distributed control.**

***Index Terms*—Hardware implementation, medium access control (MAC) protocol, multicast, optical star, wavelength-division multiplexing (WDM).**

I. Introduction

An OPTICAL star's broadcast-and-select mechanism has received much attention in the construction of wavelength-division multiplexing (WDM)-based local area networks [5]. Medium access control (MAC) protocols for *single-destination* traffic have been proposed for the optical star in [19] and [20]. To transmit *multidestination* traffic, single-hop MAC protocols with multicast support have been reported [1], [3], [17], [22]–[24].

The multicast protocols reported in [1], [3], and [22] are *reservation* oriented, whereas the protocols in [23] and [24] use *preallocation* strategies. All these protocols assume that their channels (data and, where applicable, control) are slotted and there is no central controller. Because an optical star coupler broadcasts a data frame to all the optical links connected to it, the protocols presented in [1], [3], [17], [23]multicast a data packet by transmitting it just once. An underlying assumption behind this *just-once* transmission is that all the desired receivers are available at the same time. The concept of *just-once* transmission has been relaxed in [10] by partitioning a multicast group into a number of smaller subgroups and transmitting the data frame once to each subgroup. This approach more effectively balances the usage of transmitter and receiver resources, but there is no guarantee that it improves system performance. The problem of minimizing the number of transmissions, as a result of partitioning, under the condition that the packet delay is minimum has been proved to be NP-complete [14]. A heuristic scheduling algorithm, called maximum-destination scheduling, has been proposed in [14]. This maximum-destination scheduling algorithm produces lower mean delay compared to a no-partition scheduling algorithm. However, in the case that a multicast packet in which many destination addresses are the same as those of unicast data packets from other sources is scheduled first, the number of unicast data packets blocked for a long time by the multicast packet is increased. To address this problem, a priority-based partition scheduling algorithm has been studied in [12]. In this protocol, transmission of multicast packets with more destination address overlap is postponed.

Simulation studies performed in [15] show that, for single-hop WDM networks, a multicast scheduling algorithm which always tries to partition a multicast transmission into smaller subgroups may not always produce lower mean packet delay than a multicast scheduling algorithm which does not partition multicast transmissions. Based on this observation, a hybrid multicast scheduling algorithm has been proposed in [15]. The idea of hybrid multicast scheduling is like this: depending on the average utilization of the data channels and the receivers, the scheduling algorithm dynamically chooses to employ a scheduling algorithm which always tries to partition multicast groups or a scheduling algorithm which does not partition multicast groups. The hybrid algorithm produces lower mean packet delay. However, no study has been performed in [10] and [15] to observe the effect of multicast group partitioning on system throughput.

The multicast protocol in [22] works in a different way as explained in what follows. Each node is equipped with one fixed transmitter and one tunable receiver. Multicast transmissions from all the nodes are represented by a traffic demand matrix and are processed as a batch. The protocol partitions the set of physical receivers into virtual receivers and transforms the original network with multicast traffic into a network with unicast traffic, and recommends to use any unicast protocol to achieve multicast. However, for best throughput, the above partitioning

Manuscript received December 31, 2002; revised November 12, 2003. This paper was presented in part at GLOBECOM 2002, Taipei, Taiwan.

K. Naik is with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON N2L 3G1, Canada (e-mail: knaik@swen.uwaterloo.ca).

D. S. L. Wei is with the Department of Computer and Information Science, Fordham University, Bronx, NY 10458 USA (e-mail: wei@dsm.fordham.edu).

D. Krizanc is with the Mathematics and Computer Science Department, Wesleyan University, Middletown CT 06457 USA (e-mail: dkrizanc@caucus.cs.wesleyan.edu).

S.-Y. Kuo is with the Electrical Engineering Department, National Taiwan University, Taipei 106, Taiwan (e-mail: sykuo@cc.ee.ntu.edu.tw).

Digital Object Identifier 10.1109/JSAC.2004.829638

0733-8716/04$20.00 © 2004 IEEE

Authorized licensed use limited to: National Taiwan University. Downloaded on January 19, 2009 at 23:35 from IEEE Xplore. Restrictions apply.

must be done in an optimal manner, which is an NP-complete problem.

Some MAC protocols use a control channel to exchange transmission related information [1], [10], [24], whereas some protocols do not use a control channel [2], [4], [23]. Some advantages of not using a control channel are that the underlying hardware may require one less pair of dedicated transceivers, and one more channel is available for data transmission [4]. Among the MAC protocols which do not use a control channel, the one presented in [23] supports multicast traffic.

A different kind of reservation-based protocol has been proposed in [17], which uses an unslotted system with a centralized scheduler. To provide bandwidth on demand, an unslotted access protocol with a centralized scheduler has been proposed in [18]. The MAC protocol reported in [18] is based on a look-ahead capability to overcome the effects of head-of-line blocking.

A common concern about the design of a MAC protocol for gigabits per second networks is the requirement of very low processing overhead. The processing times of several MAC protocols with unicast support have been rated as "low," "moderate," "high," and "very high" [9]. However, those processing times have not been precisely computed. A higher processing time means a longer gap between the transmissions of two successive data packets, which lead to lower throughput. Other factors contributing to lower throughput are propagation delay and transceiver tuning times. The centralized scheduling-based protocol in [18] mitigates the effect of propagation delay by measuring those quantities for each node and taking them into account when permitting a node to transmit a packet. The multicast protocol in [9] also considers an estimated value of propagation delay when scheduling a transmission. The multicast protocol studied in [22] recognizes that transceiver tuning times are nonnegligible.

The motivations for our work are, thus, as follows.

- There is a need for specifying a MAC protocol in detail using simple operations such as arithmetic and logical operations. There are two advantages of such a detailed specification. First, one can easily implement such a specification in hardware. Second, one can precisely compute the protocol processing overhead. If a protocol involves a computationally expensive step such as partitioning a multicast address set into a number of smaller sets [10], [14], or partitioning a set of physical receivers into virtual receivers [22], which are NP-complete problems, then the protocol will pose two problems. First, the processing overhead will be too high resulting in low throughput because of longer gaps between transmissions of successive packets. Second, it will be very difficult to precisely compute the protocol's processing overhead. We will give a high-level description of our protocol for easy comprehension followed by a register-transfer model for hardware implementation.
- It is highly unlikely that system designers will employ separate protocols for unicast service and multicast service at the MAC level. Therefore, it is prudent to design a single MAC protocol which can deliver high throughput for both kinds of traffic. We strive to achieve the maximum theoretically possible throughput for unicast traffic with distributed control [11] and higher throughput than the existing MAC protocols with distributed control such as those reported in [1], [3], and [23].

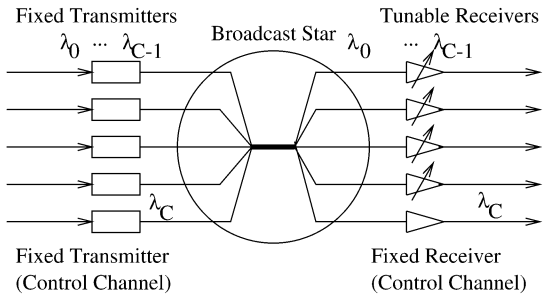

Fig. 1. Broadcast-and-select network using an optical star.

With the above mentioned motivations in our mind, in this paper, we present a new reservation-based MAC protocol with multicast support. We present analytic, simulation, and hardware implementation models of the protocol. We have identified an inherent difficulty in constructing an exact analytic model of the protocol. Therefore, we give an analytic model of the minimum guaranteed throughput of an N-node network using our protocol. To verify the analytic model and to obtain a network's accurate throughput, we simulate the protocol in Java. Simulation results show that our protocol delivers maximum possible unicast throughput, and the protocol's multicast throughput is also better than existing protocols using a control channel. Finally, we give a register-transfer model [16]of the protocol and propose a method to precisely compute the processing time overhead of the protocol for a given number of nodes and clock speed of the MAC hardware. As an example, we will show that for a 100-node network and the MAC protocol's hardware running at 200 MHz, the processing time is 20 μs. Accurate computation of the processing overhead will allow us to take suitable measures to eliminate its effect on the protocol's throughput. In this paper, we show how multiple MAC units can be used to eliminate the effect of processing overhead and receiver tuning latency.

In Section II, we present our protocol and its simulated throughput. An analytic model of the protocol's throughput is developed in Section III. In Section IV, we compare the throughput performance of the protocol with those of the existing multicast protocols. A hardware implementation model of the protocol and its computation time are analyzed in Section V. The idea of multiple MAC units to offset the effect of processing-time overhead on throughput is presented in Section VI. Some concluding remarks are given in Section VII.

II. Our MAC Protocol

In this section, we outline the system model, discuss the protocol in detail, and present its simulated throughput.

A. System Model

We consider a local network of N stations interconnected by a passive optical star, as shown in Fig. 1. We assume that all

Authorized licensed use limited to: National Taiwan University. Downloaded on January 19, 2009 at 23:35 from IEEE Xplore. Restrictions apply.

nodes are equidistant from the optical star. The main implication of this assumption is that all the packets between all pairs of nodes suffer from an identical propagation delay, denoted by R. $C+1$ wavelengths (channels) are assumed to be available. In general, $C \leq N$. Wavelengths λ_0 through λ_{C-1} are used as data channels, and wavelength λ_C is used as the control channel. We assume that each station is equipped with two fixed transmitters, one fixed receiver, and one tunable receiver. For accessing the control channel, a pair of fixed transmitter and receiver is tuned to the control channel λ_C. The data channel transmitter of node $i, 0 \leq i \leq N-1$, is fixed tuned to wavelength $\lambda_{(i \bmod C)}$.

The control channel is viewed as a sequence of *control frames* (CF), with each control frame being subdivided into N minislots—one for each node. That is, node i puts its transmission request in the ith minislot of a control frame. A minislot is a tuple: $\langle \texttt{src}, \texttt{delay}, \texttt{mc\_list} \rangle$, where `src` denotes the source node (node i), `delay` is the delay already suffered by the packet that node i wants to transmit, and `mc_list` is a multicast list of destination nodes to which node i wants to send a packet. For convenience, $delay(i)$ and $mclist(i)$ refer to the `delay` and `mc_list` components of the ith minislot, respectively. If node i does not have a transmission request in a particular control frame, its minislot takes the value $\langle i, 0, \phi \rangle$.

B. Protocol Description

After a propagation delay of R time units from the start of a control frame, all nodes receive all the N minislots of the control frame. Based on the contents of the N minislots, each node decides which nodes will transmit and which nodes will receive in the next data slot. Since all the nodes use the same decision procedure and use an identical set of N minislots, they arrive at the same conclusion. Basically, any decision procedure for this purpose must resolve two kinds of conflicts: *destination conflict* and *source conflict*. These are explained below as two requirements of our MAC protocol.

R1) Resolving *destination conflict*: A destination conflict is said to occur if multiple minislots of a control frame have the same destination node in their `mc_list`, i.e., $mclist(i) \cap mclist(j) \neq \phi, \forall i, j$ and $i \neq j$. A MAC protocol must choose a set of source nodes such that there is no destination conflict among their `mc_lists`.

R2) Resolving *source conflict*: In the system model, we have assumed that the transmitters for the data channels are fixed tuned. Since $C \leq N$ in general, the data channels of several source nodes are fixed tuned to the same wavelength. A *source conflict* is said to arise if multiple nodes with the same transmitter wavelength have nonempty `mc_lists`, i.e., they compete to use a shared data channel. Our MAC protocol must choose at most one source node from each group of nodes sharing the same transmitter wavelength.

The two conflicts can be resolved in either order. We resolve them in the R1)–R2) order. After satisfying R2), each node knows who are going to transmit next. To receive data in the next data slot from node j, the ith node tunes its receiver to wavelength $\lambda_{(j \bmod C)}$, if $i \in mclist(j)$ and node j belongs to the subset of nodes identified to transmit a packet in the following data slot. Next, we explain how to resolve the two conflicts.

To satisfy R1), we construct an undirected graph $G = (V, E)$, called a *no-overlap graph*, as follows. Corresponding to node i, we have $v_i \in V$. Thus, $|V| = N$. There is an edge $(v_i, v_j) \in E$ if and only if $mclist(i) \cap mclist(j) = \phi$. To be able to select as many nodes as possible satisfying R1), we need to select the largest clique from G. However, the problem of finding the largest clique in a graph is NP-complete. Besides, it is believed that even to get an approximate solution to this problem is still extremely difficult (see [7] and [8]for details). At the MAC protocol level, we need to keep any computation very simple. Therefore, instead of finding the maximum clique, taking a greedy approach we find a clique containing a certain node, say, node i, such that $delay(i)$ is the maximum delay among all nodes. This can be done by sequentially examining the nodes starting from the node with maximum delay and if the currently examined node is connected to each node of the clique so far constructed, the node is included and the clique grows by one node. The delay criterion introduces the notion of fairness into the protocol. If two nodes have packets with identical delays, then the node with larger identifier may be chosen. Let N_{R1} be the set of nodes selected by the greedy approach to resolve conflict R1). Next, to satisfy R2), some nodes in N_{R1} are deleted as follows.

Let $N_{R1}^i \subseteq N_{R1}$ be the set of nodes whose data channel transmitters have been fixed tuned to wavelength λ_i. To satisfy R2), a number of different criteria may be used. Two such criteria are as follows. First, select member j from N_{R1}^i such that $delay(j)$ is the maximum among nodes in N_{R1}^i. Second, select member j from N_{R1}^i such that $|mclist(j)|$ is the maximum among its group members. The multicast protocol is presented as follows.

Step 1) Collect the N minislots of a control frame.

Step 2) Find the node index, say i, with maximum $delay(i)$. Construct the no-overlap graph, and find a clique starting with node i in a greedy manner. Let N_{R1} represent the node set in the clique.

Step 3) For each data channel wavelength λ_j, let $N_{R1}^j \subseteq N_{R1}$ be the set of nodes assigned wavelength λ_j. Select one node from the group N_{R1}^j using any criterion as explained above. Let N_s be the selected node set. Nodes belonging to N_s will transmit next.

Step 4) Node i tunes its receiver to wavelength $\lambda_{(j \bmod C)}$ if $i \in mclist(j)$ and $j \in N_s$.

Step 5) All nodes in the set N_s transmit their packets in the next data slot.

Remark: The hardware implementation model does not explicitly construct no-overlap graphs. Rather, we directly identify a clique from the set of $mclist()s$.

C. Performance Simulation

We simulate the protocol to study its performance. Packets are generated for transmission at each node as independent

Authorized licensed use limited to: National Taiwan University. Downloaded on January 19, 2009 at 23:35 from IEEE Xplore. Restrictions apply.

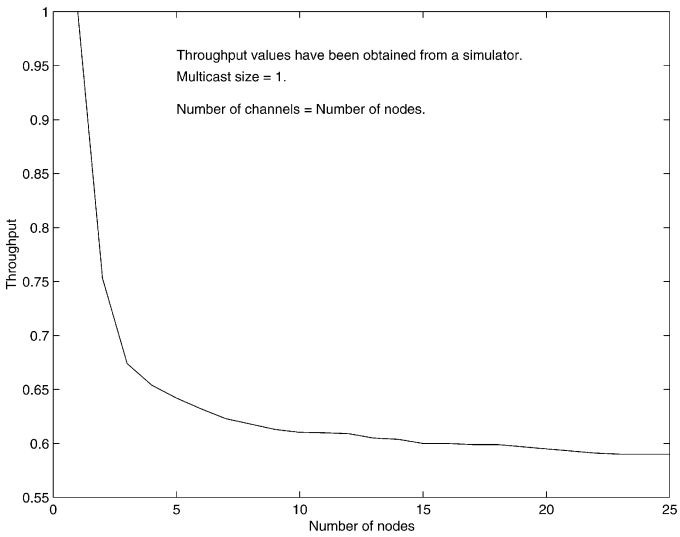

Fig. 2. Unicast throughput behavior as a function of the number of nodes.

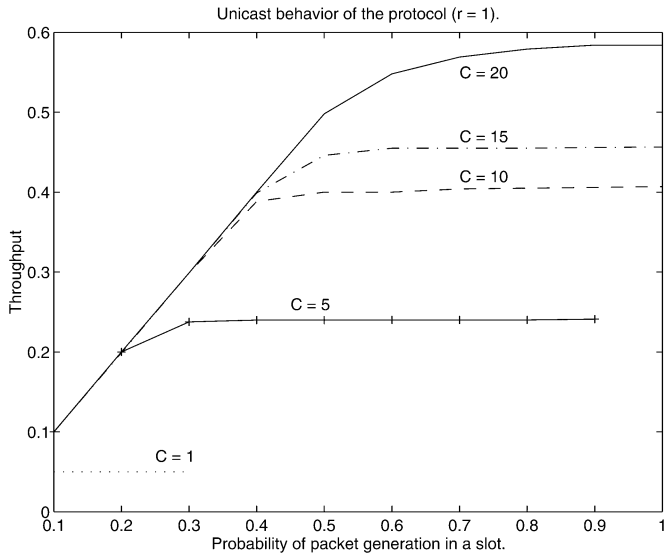

Fig. 3. Unicast throughput behavior of the protocol.

Bernoulli processes.[1] We assume infinite input queueing at each node. Each packet is destined to a multicast group of randomly chosen r nodes—the multicast group excludes the source of the packet. We have assumed that packets are queued up at their source nodes until they are scheduled for transmission. In each round of scheduling, the source node of a packet with maximum delay is made a member of N_s in Step 3) of the protocol. Thus, except in heavy traffic condition (for instance when every node makes a transmission request in every slot) which leads to an infinite queue of requests at each node, a packet is eventually transmitted.

We express the performance of the protocol in terms of *receiver throughput*. The receiver throughput, or simply throughput, is defined as the fraction of the nodes chosen by the protocol to receive packets. Referring to Step 3) of the protocol, the throughput is given by $|N_s| \times (r/N)$, where N_s is the set of nodes identified to transmit packets during the following data slot, r is the size of multicast group, and N is the number of stations (node) in the network.

Fig. 2 shows the simulated unicast throughput of the protocol. This throughput is the maximum possible throughput of a synchronous MAC protocol [11]. Though in [11], the maximum throughput achievable is given for measuring the utilization of each output trunk for an $N \times N$ switch with input queueing, it can be applied to the measurement of the utilization of the (receiving) nodes of a synchronous MAC protocol, when $N = C$, for unicast operations. That is, for large N, $N = C$, and $r = 1$, the maximum achievable throughput is 0.586 [11] and simulation results show that our protocol achieves the maximum possible unicast throughput.

Further, we simulated the protocol for 20 nodes $(N = 20)$ and for several values of multicast size r and data channel C. Packets were generated with probabilities from 0.1 to 0.9. We vary the value of r from 1 to 19. The case of $r = 1$ corresponds to unicast operation, whereas $r = 19$ corresponds to broadcast. In Fig. 3, we show the unicast throughput performance of the protocol as a function of packet generation probability and number of data channels C.

[1]In Bernoulli processes, a single packet is generated with probability p during a slot.

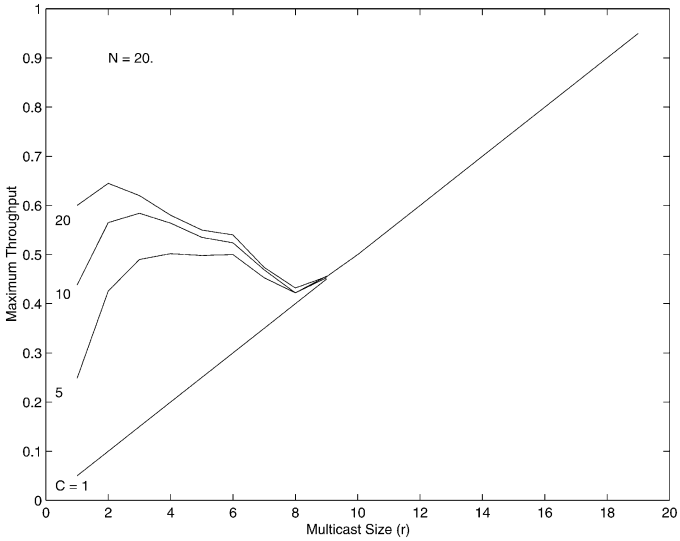

Fig. 4. Multicast throughput behavior of the protocol for varying number of channels.

In Fig. 4, we show the simulated multicast throughput performance of the protocol for $N = 20$ and different values of C. For $N = C$ and $r = 1$, the protocol delivers the maximum achievable throughput of 0.586. The humps in throughput behavior can be explained as follows. As we increase the value of r, the probability of two `mc_lists` overlapping also increases, which leads to reduced clique size, but the increase in throughput is due to the fact that throughput is directly proportional to multicast size. The initial increase in throughput is due to the fact that for $r = 2$, the clique size is more than half of the clique size for $r = 1$. When the value of r is further increased, there is drastic fall in the size of the clique, which leads to a valley in the throughput. For larger value of r, the clique size is always one and the throughput is proportional to the value of r, which explains the linear nature of throughput beyond the valley.

Authorized licensed use limited to: National Taiwan University. Downloaded on January 19, 2009 at 23:35 from IEEE Xplore. Restrictions apply.

III. Analytic Model of the Protocol's Throughput

We consider an N node network. Assuming that a node does not send a message to itself, the multicast size r is in the range of 1 to $N-1$. For simplicity, each node generates a random multicast list of constant size r. We are interested in the maximum throughput in the sense that a node makes a multicast request during each control slot.

First, we compute the probability that two multicast sets independently chosen by two nodes do not intersect. Denote this probability by p. Second, we compute the expected size of a clique selected in Step 2) of the multicast protocol. This is in fact a well known problem in random graphs and the obtained probability p is the edge-probability of the random graph, called no-overlap graph in our protocol description. Third, to take requirement R2) into account (i.e., to eliminate source conflicts), some nodes of the clique obtained above are dropped from the list of nodes to transmit next. Finally, the throughput is computed as the ratio of the product of the multicast size (r) and the number of successful transmitters computed above to N.

A. Computation of Edge-Probability p

Let $S = \{1, 2, \ldots, N\}$ be the set of all possible destinations. Given two nodes i and j in S, we define the possible destinations of messages from i and j by S_i and S_j, respectively, where $S_i = S - \{i\}$ and $S_j = S - \{j\}$. Let $rS_i \subseteq S_i$ such that $|rS_i| = r$. Similarly, let $rS_j \subseteq S_j$ such that $|rS_j| = r$. Both rS_i and rS_j are random subsets of S_i and S_j, respectively. Let I denote the event that rS_i and rS_j do not intersect. We are interested in computing the probability p of event I. For this purpose, we analyze the following two cases.

Case 1) $j \in rS_i$: In this case, rS_i does not overlap with $(n-r)!/(r!(n-2r)!)$ instances of rS_j.

Case 2) $j \notin rS_i$: In this case, rS_i does not overlap with $(n-1-r)!/(r!(n-1-2r)!)$ instances of rS_j.

However, $P_r(j \in rS_i) = r/(n-1)$ and $P_r(j \notin rS_i) = 1 - r/(n-1)$. Therefore, p can be expressed as follows:

$$p = \frac{\frac{r}{(n-1)} * \frac{(n-r)!}{r!(n-2r)!} + \left(1 - \frac{r}{(n-1)}\right) * \frac{(n-1-r)!}{r!(n-1-2r)!}}{\frac{(n-1)!}{r!(n-1-r)!}}. \quad (1)$$

Thus, the no-overlap graph is an undirected graph with N nodes and there exists an edge between nodes i and j, $\forall\, i, j$, and $i \neq j$, with probability p.

B. Computation of Expected Clique Size

Now, we analyze the average size of a clique identified in Step 2) of the multicast protocol. The details of the computation process is shown in *Block A* of Fig. 6. The clique identification process works as follows. Corresponding to the N stations in a system, we have N nodes in the no-overlap graph (random graph). The ith node of the graph represents the multicast list of the ith station. The clique construction process picks up the node, say j, with maximum delay and incrementally builds a clique by considering all other nodes one at a time. Thus, there are N steps in the construction process, and let $CL(i)$ denote a clique identified after the ith step. Initially, $CL(1) = \{j\}$. Let node k be evaluated in step $i+1$ to find if it can be added to the clique $CL(i)$. Essentially, this evaluation involves checking if node k has edges with all nodes in $CL(i)$. If node k has edges with all nodes in $CL(i)$, then $CL(i+1) = CL(i) \cup \{k\}$; otherwise, $CL(i+1) = CL(i)$. Note that once a node has been examined, it can not be reexamined for the rest of the algorithm whether it is included in the clique or not. We want to compute the expected size of the clique found by this greedy algorithm. It is believed that it is extremely difficult to get even an approximate solutions for this problem (see e.g., [7] and [8]). Fortunately, using the theorem of Geometric Distribution and Chebyshev's inequality, we still can derive a lower bound which shows that our greedy algorithm can produce an average clique whose size is bounded above the given lower bound for our protocol. Let σ denote the size of the clique produced by the greedy algorithm, and let ρ_i denote the total number of nodes that need to be examined to grow a clique from size 1 to size i. It is not hard to see that $P_r(\sigma \geq k) = P_r(\rho_k \leq N)$. Let δ_i be the random variable representing the expected number of nodes that need to be examined in order to grow the clique of size i by one. Then, $\delta_i = \rho_{i+1} - \rho_i$, $i = 0, 1, \ldots$ are independent random variables, $\delta_0 = 0$, and $P_r(\delta_i = j) = p^i(1-p^i)^{j-1}$. We now have

$$\rho_j = \sum_{i=0}^{j-1} \delta_i, \quad j \geq 1.$$

Since random variable δ_i is geometrically distributed and according to Geometric Distribution Theorem, we have

$$E(\rho_k) = \sum_{i=0}^{k-1} E(\delta_i) = \sum_{i=0}^{k-1} \frac{1}{p^i}$$

and

$$\mathrm{Var}(\rho_k) = \sum_{i=0}^{k-1} \mathrm{Var}(\delta_i) = \sum_{i=0}^{k-1} \frac{1-p^i}{(p^i)^2}.$$

Also, using Chebyshev's inequality, we have

$$\begin{aligned} P_r(\rho_k > N) &= P_r(\rho_k - E(\rho_k) > N - E(\rho_k)) \\ &\leq P_r(|\rho_k - E(\rho_k)| > \epsilon), \\ &\quad \text{where } 0 \leq \epsilon \leq N - E(\rho_k) \\ &\leq \frac{\mathrm{Var}(\rho_k)}{\epsilon^2}. \end{aligned}$$

We, thus, have $P_r(\sigma \geq k) = P_r(\rho_k \leq N) \geq 1 - (\mathrm{Var}(\rho_k)/\epsilon^2)$. Therefore, the expected clique size $E(\sigma)$ is bounded above

$$\sum_{k=1}^{\xi} k \cdot P_r(\sigma = k) \quad (2)$$

where $P_r(\sigma = k) = P_r(\sigma \geq k) - P_r(\sigma \geq k+1)$ and $\xi = \min_k\{\rho_k \geq N\}$.

C. Taking Requirement R2) Into Account

Assuming that C data channels are uniformly distributed over the N source nodes, with $C \leq N$, each channel is assigned to N/C nodes in the network. In other words, a channel serves a certain group of source nodes. If several nodes from a group appear in the clique of nodes which can possibly transmit next, the

Authorized licensed use limited to: National Taiwan University. Downloaded on January 19, 2009 at 23:35 from IEEE Xplore. Restrictions apply.

protocol chooses only one node out of them. In the following, we analyze the average size of the set of nodes which are chosen to transmit packets. This is formulated as follows.

We are given that N number of nodes share C number of channels, such that each channel is shared by a distinct group N/C nodes with no overlap among the groups. Given a subset of those nodes with size M, $M \leq N$, we need to compute the number of distinct channels to be used by those nodes after choosing at most one node from each group of nodes sharing the same channel. We randomly choose M nodes from the set of N nodes, and let X be the number of different channels used by those M nodes. We are interested in $E(X)$, the expected value of X. $E(X)$ is computed as follows.

Let $T = N/C$. For simplicity, we let T to be an integer. The number of ways to choose M nodes out of N is $\binom{N}{M}$. The number of ways to choose M nodes such that none of them use a certain channel, say i, is $\binom{N-T}{M}$. Then, the probability that channel i is not used by those M nodes, denoted by Q, is

$$Q = \frac{\binom{N-T}{M}}{\binom{N}{M}}.$$

Thus, the probability that channel i is used by some nodes in the chosen set of M nodes is $1 - Q$.

Next, we define variables $Y_1, Y_2, \ldots, Y_C$ as follows:

$$\begin{aligned} \forall\, i,\ 1 \leq i \leq C,\ Y_i &= 1 \text{ if channel } i \text{ is} \\ &\quad \text{used by those } M \text{ nodes.} \\ &= 0 \text{ if channel } i \text{ is not} \\ &\quad \text{used by those } M \text{ nodes.} \end{aligned}$$

We have $X = Y_1 + Y_2 + \cdots + Y_C$.

Also, $E(X) = E(Y_1 + Y_2 + \cdots + Y_C) = E(Y_1) + E(Y_2) + \cdots + E(Y_C)$.

However, $E(Y_i) = 1(1 - Q) + 0(Q) = 1 - Q$, for all i.

Thus, $E(X) = C \times (1 - Q)$.

D. Throughput Comparison: Analytic Versus Simulation

The protocol's average throughput in a N-node network, denoted by T_{av}, is given by $(r/N) \times |N_s|$, where N_s is the set of nodes that can transmit packets in a data slot and r is the multicast size

$$\begin{aligned} T_{\text{av}} &= \left(\frac{r}{N}\right) \times N_s \\ &= \left(\frac{r}{N}\right) \times C \times (1 - Q) \\ &= \left(\frac{r}{N}\right) \times C \times \left(1 - \frac{\binom{N - \frac{N}{C}}{E(\sigma)}}{\binom{N}{E(\sigma)}}\right). \end{aligned}$$

Here, we remind the reader that $E(\sigma)$ is the clique size obtained from (2), which in turn is computed using (1).

In Fig. 5, we compare the protocol's throughput obtained from its analytic model and simulation for $N = C = 50$ for various values of multicast size. In the figure, the solid line is obtained from simulations, and the broken line is obtained from the analytic model. It may be noted that throughput obtained from simulation of the protocol is always higher than the one obtained from the analytic model, because the analytic model gives the minimum guaranteed throughput. The mismatch between the two plots in the figure is due to the variance in the computed average clique size in a random graph [8].

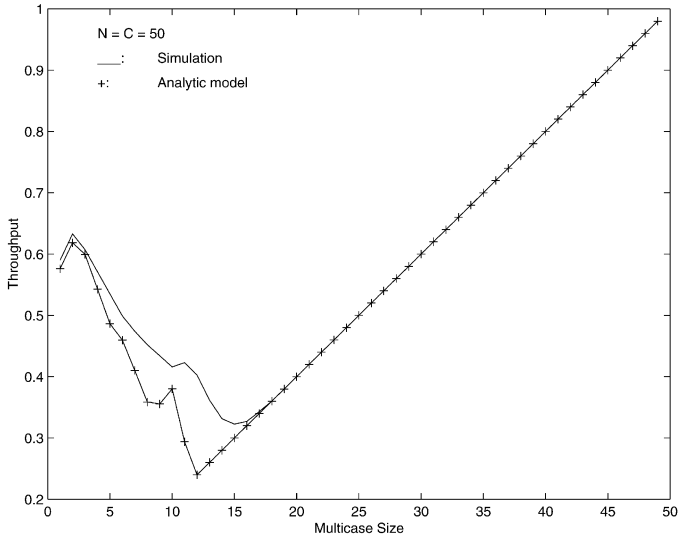

Fig. 5. Throughput comparison.

IV. Performance Comparison

In this section, we compare the performance of our protocol with other multicast protocols [1], [3], [23]. It is difficult to make a straightforward comparison of the performances of these protocols due to several reasons as explained below.

First, there are different ways of representing the protocols' throughput. Borella and Mukherjee [3] define two types of throughput, namely *transmitter throughput* and *receiver throughput*. Transmitter (receiver) throughput is defined to be the mean number of transmitters (receivers) in use at steady state. Rouskas and Ammar [23] define throughput as the expected number of packets successfully received per slot. Bandai *et al.* [1] define throughput as the average number of users which can reserve the data channel(s) per time slot. Modiano [17] expresses throughput performance in terms of *throughput efficiency*, where throughput efficiency is defined as the ratio of the *lower bound* on the number of transmissions required to the *average* number of transmissions that the protocol requires per successful multicast.

Second, each multicast message is sent to a *constant* number, say, r, of destination nodes in [1] and [17], whereas the number of destinations of a multicast message is a random number upper-bounded by r in [3]. The situation is more complex in [23], where a slot is predefined to be a unicast, broadcast, or multicast slot, and there is a certain way of simultaneous multicast and unicast.

Third, there are different queueing constraints. It is assumed in [1] and [23] that each station (node) can hold at most one data packet at a time. Thus, all packets that cannot be scheduled for

Authorized licensed use limited to: National Taiwan University. Downloaded on January 19, 2009 at 23:35 from IEEE Xplore. Restrictions apply.

TABLE I
THROUGHPUT COMPARISON

Parameters	Protocol	Throughput	Ours
N = 50, C = 10, r = 1	Borella and	0.18	0.2
N = 50, C = 10, r = 25	Mukherjee [3]	0.3	0.5
N = 50, C = 10, r = 49		0.54	0.98
N = C = 8, r = 1	Rouskas and	0.50	0.616
N = C = 8, r = 4	Ammar [23]	0.56	0.56
N = C = 8, r = 7		0.63	0.875
N = 50, C = 10, r = 5	Bandai, Shiokawa	0.4	0.46
N = 70, C = 10, r = 5	and Sasase [1]	0.35	0.41

transmission in the next data slot after their arrival (generation) are considered to be lost. The protocols in [3] and [17], and in this paper assume that all packets are queued at each station.

In Table I, we compare the throughput of our protocol with those in [1], [3], and [23]. In the absence of comparable data, we leave out the comparison with [17]. Table I shows that our protocol performs as good as, and even better than, the other protocols with distributed control. In the absence of comparable data, we have not compared our protocol's performance with the one reported in [17]. Moreover, the protocols in [17] and [18] use a centralized scheduler with access to global transmission requests in a network. The throughput performance of the protocol in [1] is not significantly different from that of our protocol. For small multicast size, the protocol in [23] and our protocol perform with a small difference, whereas our protocol performs better when the multicast size is large. For both unicast and multicast traffic, our protocol performs better than the protocol studied in [3]. It may be noted that the protocols studied in [3] and [1] use a control channel, whereas the protocols in [23] do not use any control channel.

The MAC protocol in [18] achieves better unicast throughput than the protocol presented in this paper due to their look-ahead mechanism in packet scheduling. Since their centralized scheduler collects requests from all nodes and queues them up, it can look ahead into their buffers and transmit a packet other than the one at the head of the queue. This look-ahead mechanism lowers destination conflicts and thereby increases unicast throughput. However, it is not known how the look-ahead mechanism will affect multicast throughput.

V. HARDWARE IMPLEMENTATION MODEL

Since the MAC protocol is invoked once every control frame, it is important that the computation time of the protocol is kept as small as possible. In this section, we explain how the protocol can be implemented in hardware. The hardware model of the protocol will allow us to precisely compute the protocol overhead.

Let us assume that the `delay` and `mc_list` components of a minislot are represented by an *integer* and a *bit-vector*, respectively. If the ith node wants to send a multicast message to the jth node, then the jth bit of $mclist(i)$ bit-vector is set to 1, otherwise, it is set to 0. The main idea behind representing the `mc_list` as a bit-vector is to be able to use logical operations, such as AND and OR, in the computation of a clique. This leads to linear time complexity of the protocol.

In Fig. 6, we show the flowchart representation of the multicast protocol assuming that the `mc_list` components are represented as bit-vectors. The flowchart has been presented from the viewpoint of the ith node. We have structured the flowchart into three blocks: *Block A*, *Block B*, and *Block C*. *Block A* computes the clique containing the node with maximum delay. This computation corresponds to Step 2) of the algorithm. *Block B* eliminates a few nodes from the clique to resolve source conflicts. This computation corresponds to Step 3) of the algorithm. *Block C* decides if node i is a receiver or not. If the ith node is a receiver, it tunes to the wavelength of the transmitter of the appropriate node selected in *Block B*.

Now, we explain the variables used in the flowchart. N registers, denoted by $R(0)$ through $R(N-1)$, contain the `mc_list` bit-vectors received from nodes 0 through $N-1$. N registers, denoted by $MR(0)$ through $MR(N-1)$ and called *mask* registers, are used in selecting one station out of a group of stations sharing the same data channel wavelength, i.e., in resolving source conflicts. Mask register $MR(i)$ is initialized as follows.

```
Procedure to initialize MR(i)
  The jth bit of MR(i) is referred to as
  MR(i)[j].
Step 1) Let m = i mod C.
Step 2) ∀ j, 0 ≤ j ≤ N − 1, set MR(i)[j] = 0,
        if m = (j mod C), j ≠ i.
Step 3) ∀ j, 0 ≤ j ≤ N − 1, set MR(i)[j] = 1,
        if m ≠ (j mod C), j ≠ i.
Step 4) Set MR(i)[i] = 1.
End
```

We use a third set of N registers $RB(0)$ through $RB(N-1)$ to test whether a certain bit of another register is 1 or 0. For this purpose, we use registers $RB(0)$ through $RB(N-1)$, where the ith bit of $RB(i)$ is set to 1, while all other bits of $RB(i)$ are set to 0. Register RA holds the clique at the end of *Block A*, and RX and RY are registers to hold temporary values. In register RA, we accumulate the nodes of a clique in an incremental manner-if node i belongs to the clique then the ith bit of RA is set to 1, otherwise, it is set to 0. In register RX, we accumulate the multicast sets of all the nodes which have so far been collected in RA. The AND and OR operations in Steps 2, 4, 10, 12, 17, and 21 are performed bit-wise, whereas the AND operation in step 22 is an AND on two boolean variables BX and BY. We also assume that variables k and j are stored in the processor's registers. At this point, we remind the reader that the protocol does not construct a no-overlap graph; rather, a clique is directly constructed in an incremental manner. A clique of size s is enlarged to a clique of size $s+1$ by using AND and OR operations on `mc_list` bit-vectors and the bit-vector representation of the clique of size s.

In the flowchart of Fig. 6, we do not show the computation of the node index with maximum $delay(i)$. The maximum of $delay(i)$ can be computed in a sequence of comparisons as minislots arrive at a node. Thus, when the last minislot of a control frame arrives, identification of the node index, say, j, such that $delay(j)$ is the maximum among all nodes, will be completed.

The flowchart of Fig. 6 can be implemented in hardware in a straightforward manner using the *register-transfer* model [16]

Authorized licensed use limited to: National Taiwan University. Downloaded on January 19, 2009 at 23:35 from IEEE Xplore. Restrictions apply.

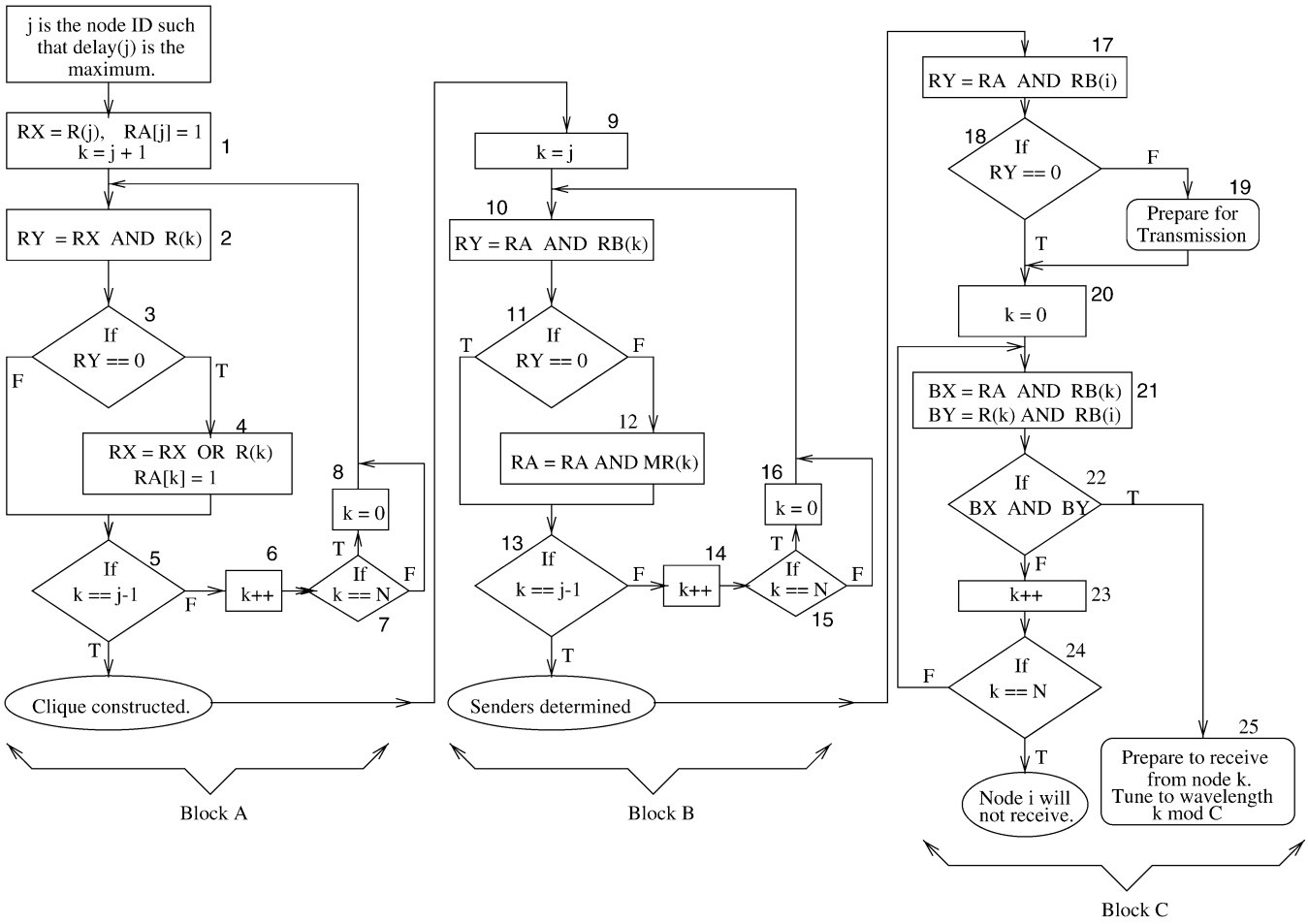

Fig. 6. Flowchart form of the multicast protocol for the ith node.

TABLE II
CYCLE COST OF THE COMPUTATION STEPS IN THE FLOWCHART

Step	Cycles	Step	Cycles	Step	Cycles	Step	Cycles	Step	Cycles
1	2+3	2	2	3	2	4	3+3	5	3
6	1	7	2	8	1	9	1	10	2
11	2	12	3	13	3	14	1	15	2
16	1	17	2	18	2	19		20	1
21	4	22	2	23	1	24	2	25	

of hardware implementation. This model has the advantage that each decision block can be executed in *two* clock cycles, rather than consuming *four* clock cycles as it is the case in general purpose microprocessors, say, *Intel i486* [6]. The computation and decision blocks of the flowchart resemble machine instructions of a microprocessor. In Table II, we give the machine-cycle cost of each block of instruction of the flowchart.

There is a loop in each of *Block A*, *Block B*, and *Block C*, which is executed at most N times. The loop costs of *Block A*, *Block B*, and *Block C* are 17, 14, and 9 cycles, respectively. Thus, the loop cost is $(17 + 14 + 9) * N = 40 * N$ cycles. Added to this are the costs of blocks 1, 9, 17, 18, and 20, which sum up to 11 cycles. Thus, the protocol's computation time is $(11 + 40 * N) \simeq 40 * N$ cycles. Assuming that the controller operates at 200 MHz (1 cycle = 5 ns), the computation time of the protocol for a 100-node network is 20 μs.

VI. MULTIPLE MAC UNITS

A protocol's throughput will be degraded by its processing time and transceiver tuning time. These overheads lead to a gap between transmission of successive data frames from the same source resulting in lower throughput. There have been efforts to mitigate their effect on throughput. To overcome the effects of propagation delays, tuning, and processing, the centralized scheduler in the MAC protocol proposed in [18] measures the delays between the nodes and the hub and takes that delay into account when scheduling transmissions. By measuring the time that it takes a node to respond to a transmit command, the scheduler obtains an estimate of the round-trip delay for that node. The round-trip delay is used to inform a node of its turn to transmit in a timely manner. By synchronizing all of the nodes to the hub, the transmissions of different nodes can be scheduled back-to-back, with minimal gaps between transmissions.

Authorized licensed use limited to: National Taiwan University. Downloaded on January 19, 2009 at 23:35 from IEEE Xplore. Restrictions apply.

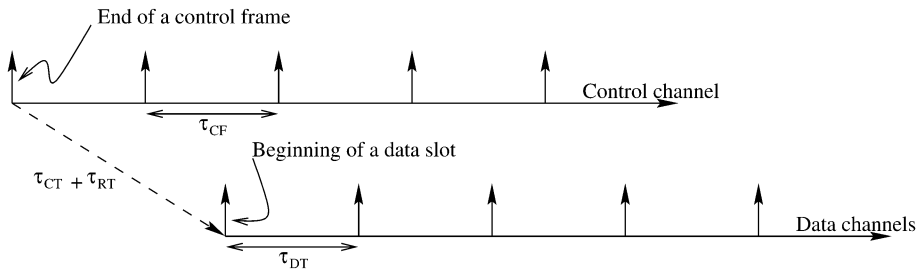

Fig. 7. Timing relation.

To mitigate the effect of propagation delay on throughput, a similar approach has been adopted in [9], where nodes schedule their transmissions to account for propagation delays. In [9], nodes offset their transmissions with the worst-case propagation delay in order to make sure that both the transmitter and receiver had sufficient time to respond to control messages.

In this paper, we propose a novel way to reduce the gap between the transmissions of successive data frames by the same node. The concept of multiple MAC units presented in this paper takes into account propagation time, protocol execution time, receiver tuning time, and data transmission time so that these individual times do not form a bottleneck. The concept of multiple MAC units does not measure those delays, rather it works on their estimations. A worst-case estimation may simply produce a small number of additional MAC units without deteriorating system performance. And, the cost of an additional MAC unit at the hardware level is insignificant.

In order to fine-tune the protocol, there is a need to identify the protocol's timing parameters. For improved performance, it is necessary to carry out several activities in parallel. In this section, we identify these actions and compute their durations. Specifically, we will analyze control frame length, computation time of the protocol, and data transmission time in addition to propagation delay and tuning times of the receivers.

Control frame length (τ_{CF}): Each control frame has N minislots of the structure $\langle \mathtt{src}, \mathtt{delay}, \mathtt{mc\_list} \rangle$. The source and the delay fields can be represented by 2 bytes each to accommodate a large number of nodes and a wide range of delay. The destination list can be represented by a N-bit vector. Thus, a control frame with N minislots consists of $(32 + N) * N$ bits. For $N = 100$ and a transmission rate of 1 Gb/s, a control frame has a duration of 13.2 μs.

Computation time of the protocol (τ_{CT}): In Section V, we showed that the computation time of the protocol is $40 * N$ cycles. Assuming that the MAC hardware has a clock running at 200 MHz, the clock duration is 5 ns. Thus, for 100 nodes, the protocol's computation time $\tau_{\mathrm{CT}} = 40 * 100 * 5$ ns $= 20$ μs. For high speed MAC hardware running at, say, 500 MHz, the computation time is 8 μs.

Data transmission time (τ_{DT}): With a data rate of 1 Gb/s and packets of length 1000 bytes, the data transmission time $\tau_{\mathrm{DT}} = 8$ μs. It is possible that the value of τ_{DT} is different from τ_{CF}, but there is no apparent advantage in making the value of τ_{DT} different from τ_{CF}. Since a source's transmitter and a destination's receiver are tuned to the same wavelength for the entire duration of a data slot, there is no gain in running the control channel at twice (or half) the rate of the data channels. This is explained as follows. On the one hand, assuming that $\tau_{\mathrm{CF}} = 0.5 * \tau_{\mathrm{DT}}$ (that is, the control channel runs at a higher rate than the data channels), even if a MAC unit readies itself to send data, a transmitter/receiver pair will have to wait until the completion of the present data slot. On the other hand, assuming that $\tau_{\mathrm{CF}} = 2 * \tau_{\mathrm{DT}}$ (that is, the control channel runs at a lower rate than the data channels), the transmitters and receivers will sit idle. Therefore, we assume that $\tau_{\mathrm{CF}} = \tau_{\mathrm{DT}}$.

Propagation delay (δ): With a local area network (LAN) diameter of 10 km, the propagation delay is about 33.3 μs.

Receiver tuning time (τ_{RT}): Receiver tuning time is much less than the other durations discussed above. Kobrinski *et al.* [13] have reported a very high-speed DFB laser amplifier with a 1-ns wavelength-switching time. Though τ_{RT} can be negligibly small, we incorporate this delay in our studies in the following.

The representative values of different timing parameters, shown above, suggest that τ_{CF} and δ can be several times greater than τ_{CT} and τ_{DT}. Thus, there may be a few control frames in flight before the first one is processed and decision made. Though control frames can be sent back-to-back, processing those frames without delay is not feasible with a single MAC unit if $\tau_{\mathrm{CT}} > \tau_{\mathrm{CF}}$. As a result, data packets cannot be sent in consecutive data slots from the same source. Hence, there is a need for multiple MAC units to process consecutive control frames in an overlapped manner so that data packets can be sent in consecutive data slots from the same source. In the following, we compute the number of such MAC units.

Referring to Fig. 7, nodes receive control frames at intervals of τ_{CF}. (The events denoting the completion of receiving control frames are shown in the figure.) After receiving a control frame, a node starts processing it—a MAC unit decides whether the node will send and/or receive data during the next data slot. If the node decides that it will receive a packet during the next data slot, it will tune its receiver to the desired wavelength. Thus, a duration of $\tau_{\mathrm{CT}} + \tau_{\mathrm{RT}}$ must elapse between the instance representing the end of receiving a control frame and the instance of starting to receive a data packet. Since $\tau_{\mathrm{CF}} = \tau_{\mathrm{DT}}$, data packets can periodically arrive at a node at an interval of τ_{DT} (The events denoting the beginning of arrival of data packets are shown in the figure.) Fig. 7 suggests that a node periodically receives control frames and readies itself to receive data packets at the same periodicity. Thus, a node needs β, where $\beta = \lceil (\tau_{\mathrm{CT}} + \tau_{\mathrm{RT}})/\tau_{\mathrm{CF}} \rceil$, numbers of MAC units. If the MAC units are numbered 0 through $\beta - 1$, then the jth control frame, $j \geq 0$, is processed by the kth MAC unit, where $k = j \bmod \beta$.

As an example, for a 100-node network with a transmission speed of 1 Gb/s, 200-MHz MAC hardware, and high-speed DFB

Authorized licensed use limited to: National Taiwan University. Downloaded on January 19, 2009 at 23:35 from IEEE Xplore. Restrictions apply.

laser amplifier with 1 ns tuning time, we have $\tau_{\mathrm{CF}} = 13.2\ \mu$s, $\tau_{\mathrm{CT}} = 20\ \mu$s, and $\tau_{\mathrm{RT}} = 1$ ns. Thus, $\beta = 2$.

VII. Concluding Remarks

We proposed a reservation-based MAC protocol with multicast support for single-hop passive optical star networks. In additional to its throughput simulation, we presented an analytical model of the protocol's throughput and a hardware implementation model of the protocol. We identified the inherent difficulty in obtaining a precise analytical model of the protocol's throughput, namely, computing the average clique size in a random graph. Therefore, we restricted ourselves to computing the lower bound on the throughput of our protocol. Throughput simulations verify our analytical model and give a more accurate picture of the protocol's throughput. The nodes collect multicast requests from all other nodes through a control channel which is accessed by each node through a pair of fixed-tuned transmitter and receiver. The available data channels are shared among the nodes. Each node is equipped with a fixed transmitter and a tunable receiver to send/receive data. The protocol operates in two phases: first, through a control channel, each node collects the multicast requests of all other nodes; second, destination and source conflicts are resolved to decide which nodes will transmit/receive next. Simulation results show that the unicast throughput of our protocol is the best possible throughput. Comparison of multicast throughput with other synchronous multicast protocols show the superiority of our protocol.

We proposed how the second phase above can be implemented in hardware. The hardware implementation model allowed us to precisely analyze the computation time of the protocol. If the protocol overhead (that is, computation time) is larger than the duration of a control frame, then using a single MAC unit, it is not possible to send data packets in consecutive data slots. Therefore, we suggested how multiple MAC units can be used to offset the effect of protocol overhead on throughput performance. Our present interest is to specify the protocol in VHDL and implement it using field programmable gate arrays (FPGA).

Acknowledgment

The authors would like to thank the anonymous reviewers for their helpful comments.

References

[1] M. Bandai, S. Shiokawa, and I. Sasase, "Performance analysis of a multicast protocol in WDM-based single-hop lightwave networks," in *Proc. IEEE GLOBECOM*, 1997, pp. 561–565.

[2] K. Bogineni, K. M. Sivalingam, and P. W. Dowd, "Low-complexity multiple access protocols for wavelength-division multiplexed photonic networks," *IEEE J. Select. Areas Commun.*, vol. 11, pp. 590–604, May 1993.

[3] M. S. Borella and B. Mukherjee, "A reservation-based multicasting protocol for WDM local lightwave networks," in *Proc. IEEE ICC*, June 1995, pp. 1277–1281.

[4] ——, "Efficient scheduling of nonuniform packet traffic in a WDM/TDM local lightwave network with arbitrary transceiver tuning latencies," *IEEE J. Select. Areas Commun.*, vol. 14, pp. 923–934, June 1996.

[5] C. A. Brackett, "Dense wavelength division multiplexing networks: Principles and applications," *IEEE J. Select. Areas Commun.*, vol. 8, pp. 948–964, Aug. 1990.

[6] *Hardware Reference Manual i486 Microprocessor*, Intel Corp., Santa Clara, CA, 1990.

[7] M. Garey and D. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. San Francisco, CA: Freeman, 1978.

[8] P. G. Gazmuri, "Independent sets in random sparse graphs," *Networks*, vol. 14, pp. 367–377, 1984.

[9] F. Jia and B. Mukherjee, "The receiver collision avoidance (RCA) protocol for a single-hop WDM lightwave network," *J. Lightwave Technol.*, vol. 11, pp. 1053–1065, May/June 1993.

[10] J. P. Jue and B. Mukherjee, "The advantages of partitioning multicast transmissions in a single-hop optical WDM network," in *Proc. IEEE ICC*, June 1997, pp. 427–431.

[11] M. J. Karol, M. G. Hluchyj, and S. P. Morgan, "Input versus output queueing in a space-division packet switch," *IEEE Trans. Commun.*, vol. COM-35, no. 12, pp. 1347–1356, Dec. 1987.

[12] T. Kitamura, M. Iizuka, and M. Sakuta, "A new partition scheduling algorithm by prioritizing the transmission of multicast packets with less destination address overlap in WDM single-hop networks," in *Proc. IEEE GLOBECOM*, 2001, pp. 1469–1473.

[13] H. Kobrinski *et al.*, "Fast wavelength-switching of laser transmitters and amplifiers," *IEEE J. Select. Areas Commun.*, vol. 8, pp. 1190–1201, Aug. 1990.

[14] H.-C. Lin and C.-H. Wang, "Minimizing the number of multicast transmissions in single-hop WDM networks," in *Proc. IEEE ICC*, 2000, pp. 1645–1649.

[15] ——, "A hybrid multicast scheduling algorithm for single-hop WDM networks," *J. Lightwave Technol.*, vol. 19, no. 11, pp. 1654–1664, Nov. 2001.

[16] M. C. McFarland, A. C. Parker, and R. Camposano, "Tutorial on high-level synthesis," in *Proc. 25th Design Automation Conf.*, 1988, pp. 330–336.

[17] E. Modiano, "Random algorithms for scheduling multicast traffic in WDM broadcast-and-select networks," *IEEE Trans. Networking*, vol. 7, no. 3, pp. 425–434, June 1999.

[18] E. Modiano and R. Barry, "A novel medium access control protocol for WDM-based LAN's and access networks using a master/slave scheduler," *J. Lightwave Technol.*, vol. 18, no. 4, pp. 461–468, Apr. 2000.

[19] B. Mukherjee, "WDM-based local lightwave networks, Part 1: Single-hop systems," *IEEE Networks Mag.*, pp. 12–27, May 1992.

[20] ——, "WDM-based local lightwave networks, Part 2: Multihop systems," *IEEE Networks Mag.*, pp. 20–32, July 1992.

[21] K. Naik, D. Wei, D. Krizanc, and S.-Y. Kuo, "A reservation based medium access control protocol with multicast support for optical star networks," in *Proc. IEEE GLOBECOM*, Nov. 2002, pp. 2798–2802.

[22] Z. Ortiz, G. N. Rouskas, and H. G. Perros, "Scheduling of multicast traffic in tunable-receiver WDM networks with nonnegligible tuning latencies," *Proc. ACM SIGCOMM*, pp. 301–310, 1997.

[23] G. N. Rouskas and M. H. Ammar, "Multi-destination communication over tunable-receiver single-hop WDM networks," *IEEE J. Select. Areas Commun.*, vol. 15, pp. 501–511, Apr. 1997.

[24] W.-Y. Tseng and S.-Y. Kuo, "A combinational media access protocol for multicast traffic in single-hop WDM LANs," in *Proc. IEEE GLOBECOM*, 1998, pp. 294–299.

Kshirasagar Naik (M'94) received the B.S. and M.Tech. degrees from Sambalpur University, Sambalpur, India, and the Indian Institute of Technology, Kharagpur, respectively, the M.Math. degree in computer science from the University of Waterloo, Waterloo, ON, Canada, and the Ph.D. degree in electrical and computer engineering from Concordia University, Montreal, QC, Canada.

He was a Faculty Member with the University of Aizu, Aizu, Japan, and Carleton University, Ottawa, ON, Canada. At present, he is an Associate Professor in the Department of Electrical and Computer Engineering, University of Waterloo. He served as a Program Co-Chair of the 5th International Conference on Information Technology held in Bhubaneswar, India, December 2002. His research interests are in the areas of testing communication protocols, wireless communication protocols, smart antenna-based MAC protocols for wireless LAN and optical networks, mobile computing, data access in wireless networks, power-conscious software systems for portable devices, and coexistence of 802.11 and Bluetooth technologies.

Authorized licensed use limited to: National Taiwan University. Downloaded on January 19, 2009 at 23:35 from IEEE Xplore. Restrictions apply.

David S. L. Wei (S'87–M'90) received the Ph.D. degree in computer and information science from the University of Pennsylvania, Philadelphia, in 1991.

He is currently an Associate Professor of Computer and Information Science at Fordham University, Bronx, NY. From May 1993 to August 1997, he was on the Faculty of the School of Computer Science and Engineering, University of Aizu, Aizu, Japan, as an Associate Professor and then a Professor. He has authored and coauthored more than 60 technical papers in the areas of distributed and parallel processing, mobile computing, quantum computing, and optical communications in various archival journals and conference proceedings. He served on the program committee and as a session chair for several reputed international conferences. He is Co-Chair of Power Aware Communication and Software, Minitrack in the Software Track at the 34th Hawaii International Conference on Systems Sciences (HICSS-34). Currently, he focuses his research effort on wireless communications and mobile computing.

Dr. Wei is a Guest Editor of the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS for the Special Issue on Mobile Computing and Networking.

Danny Krizanc received the B.Sc. degree from the University of Toronto, Toronto, ON, Canada, in 1983 and the Ph.D. degree from Harvard University, Cambridge, MA, in 1988, both in computer science.

He held positions at the Centruum voor Wiskunde en Informatica, Amsterdam, The Netherlands, the University of Rochester, Rochester, NY, and Carleton University, Ottawa, ON, Canada, before joining Wesleyan University, Middletown, CT, as an Associate Professor in 1999. His research focus is the design and analysis of algorithms, especially as applied to distributed computing and networking.

Sy-Yen Kuo (S'85–M'88–SM'98–F'01) received the B.S. degree in electrical engineering from National Taiwan University, Taipei, Taiwan, in 1979, the M.S. degree in electrical and computer engineering from the University of California at Santa Barbara, in 1982, and the Ph.D. degree in computer science from the University of Illinois at Urbana-Champagne, Urbana, IL, in 1987.

Since 1991, he has been with National Taiwan University, where he is currently a Professor and the Chairman of the Department of Electrical Engineering. He spent his sabbatical year as a Visiting Researcher at AT&T Laboratories-Research, NJ, from 1999 to 2000. He was the Chairman of the Department of Computer Science and Information Engineering, National Dong Hwa University, Hualien, Taiwan, from 1995 to 1998, a Faculty Member in the Department of Electrical and Computer Engineering, University of Arizona, Tucson, from 1988 to 1991, and an Engineer with Fairchild Semiconductor and Silvar-Lisco, both in California, from 1982 to 1984. In 1989, he also worked as a summer faculty Fellow at Jet Propulsion Laboratory, California Institute of Technology. His current research interests include software reliability engineering, mobile computing, dependable systems and networks, and optical WDM networks. He has published more than 200 papers in journals and conferences.

Dr. Kuo received the Distinguished Research Award from the National Science Council, Taiwan (1997–2005). He was also a recipient of the Best Paper Award at the 1996 International Symposium on Software Reliability Engineering, the Best Paper Award in the simulation and test category at the 1986 IEEE/ACM Design Automation Conference (DAC), the National Science Foundation's Research Initiation Award in 1989, and the IEEE/ACM Design Automation Scholarship in 1990 and 1991.

Authorized licensed use limited to: National Taiwan University. Downloaded on January 19, 2009 at 23:35 from IEEE Xplore. Restrictions apply.