JOURNAL OF PARALLEL AND DISTRIBUTED COMPUTING 17, 259–265 (1993)

Design and Analysis of a Hybrid Access Control to an Optical Star Using WDM

YORAM OFEK

IBM, T. J. Watson Research Center, P.O. Box 704, Yorktown Heights, New York 10598

AND

MOSHE SIDI*

Electrical Engineering Department, Technion, Haifa 32000, Israel

A passive optical star is an ideal shared medium, from both fault tolerant and access synchronization points of view. The communication over an optical star merges to a single point in space and then broadcasts back to all the nodes. This circular symmetry facilitates the solution for two basic distributed synchronization problems, which are presented in this work: (i) the generation of a global event clock for synchronizing the nodes' operation, and (ii) distributed scheduling for accessing the shared passive medium, which is a hybrid (deterministic and random) technique. We present, prove, and analyze this hybrid scheduling algorithm, which is equivalent to a distributed queue, and, therefore, is also algorithmically fair. Furthermore, our solution has two additional properties: destination overflow prevention and *destination fairness*. The effective solution of these problems can be used for efficiently implementing a local area network based on a passive optical star.

1. INTRODUCTION

A passive optical star is an attractive configuration of a local area network, since it has extremely fault tolerant attributes (due to the passive nature of the medium). Yet, the implementation of a passive optical star has some major synchronization problems:

1. Bit synchronization: The problem of recovering the receiving clock from the incoming serial bit streams from different sources.

2. Generating a global clock from an ensemble of asynchronous high speed local clocks: The global clock is used for synchronizing the access in order to achieve full utilization.

3. Access and flow control with minimum wasted time on scheduling: A passive optical star has two main resources: (i) time and (ii) spectrum. In this design we use a fraction of the spectrum, in order to maximize the access efficiency in a fair manner; i.e., all nodes should have an equal opportunity to transmit, and if only one node is busy it should be able to transmit continuously.

4. Destination overflow prevention and destination fairness: Prevention of packet loss due to finite buffering at the destination can result in fairness problems at the destination that can be characterized as a synchronization problem of multiple sources with respect to a single destination.

The bit synchronization problem has been formulated and solved in [4, 5]. This paper addresses the other three problems. In Section 2, a synchronization condition and an algorithm for generating a global event clock is presented and analyzed. The distributed hybrid (deterministic and random) scheduling algorithm, which is based on the global clock, is presented and proved in Section 3. The hybrid algorithm also includes signaling for destination overflow prevention; since this mechanism is integrated with the source access and global state synchronization, we obtain the desired *destination fairness* property. The analysis of this hybrid algorithm is presented in Section 4, which shows that small delay is obtained under light load conditions, and maximum throughput with bounded delay is obtained under heavy load conditions.

Our method is equivalent to a *distributed queue*, which means that a FIFO access order is preserved, and all nodes have equal opportunity to access the network. We comment that a recent attempt to realize a distributed queue, in the design of DQDB [2] (IEEE 802.6 standard), is only an approximation and it suffers from an inherent fairness problem. Most previous work on access control methods for passive optical stars was random; based on CSMA/CD (e.g., [1, 9, 10]); or deterministic, based on TDMA (time division multiple access) with reservation.

* This research was conducted in part when this author was visiting the IBM T. J. Watson Research Center.

0743-7315/93 $5.00

2. GENERATING A GLOBAL CLOCK

We define a global clock such that in every global clock period or "tick" there is a time interval, strictly greater than zero, in which all the local copies of the global clock read the same value, see [6, 7, 8] for more details. In this section we present a method for constructing such a clock and then we compute the necessary safety margin.

Consider a network composed of n nodes that are interconnected via a passive optical star. The passive optical star can be viewed as an ideal shared medium in which all communication merges to a single point in space and then broadcasts back to all the nodes. Each node has a full-duplex port, which connects the node to an array of star couplers [3]. The optical star array is constructed in a very small area in space, so the nodes have cyclic symmetry and are indistinguishable in their spatial location (i.e., the nodes can be arranged in any arbitrary order around the center of the star).

Every node has its own asynchronous high speed local clock, which is used for the serial transmission over the optical star. The global clock is generated from this ensemble of local clocks. The synchronization and scheduling algorithms are executed continuously by the star interface hardware. The local clocks, while nominally of the same value, run at slightly different rates. Let the rates of the n local clocks be denoted by $c_1, c_2, \ldots, c_n$, and assume that the manufacturer's specification guarantees that (i) the slowest possible clock is $c_{min}^{nominal}$ and (ii) the maximum drift is ρ. Therefore, $c_{min}^{nominal} \leq c_i \leq c_{min}^{nominal}(1 + \rho)$ for all $1 \leq i \leq n$.

At each node time is divided into basic units referred to as slots. Each node also maintains a local slot counter which is incremented at the start of each new time slot. The local clock is used for measuring the duration of a time slot. According to the local clock, the duration of a slot is K_s (in local clock cycles). Thus, the actual time slot duration at node i is $T_s^i = K_s/c_i$. We define the actual maximum time slot duration in a given network to be $T_{max} = K_s/c_{min}$, where c_{min} is the actual slowest clock in the system.

2.1. Synchronization Protocol

The requirement of a well-defined global state leads to the following condition:

Let SC_i represent the slot counter value of node i. Then, for any two nodes i and j, and for any slot counter value k, there exists a nonzero time interval such that $SC_i = SC_j = k$ $(1 \leq i, j \leq n)$. More specifically, we require that this time interval is greater or equal to μT_{max} (μ is an arbitrary constant, such that $\rho < \mu < 1$). The *synchronization window*, Δ_{max}, is defined as the time interval during which the slot counters increment their values, $\Delta_{max} < (1 - \mu)T_{max}$.

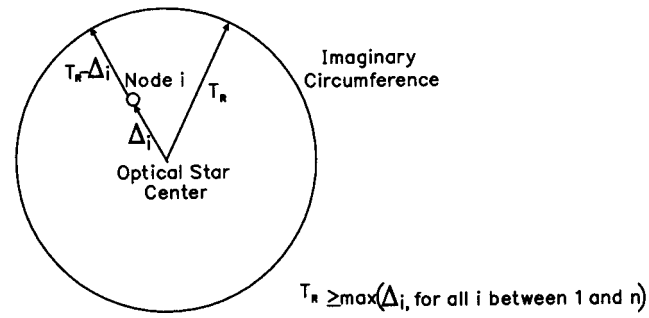


FIG. 1. The optical star timing.

The Synchronization Principle. The center of the star serves as a single point of time reference for all the n nodes of the network. Let Δ_i be the propagation delay of node i from the star's center. An imaginary circle with radius R (such that the delay $T_R \geq \Delta_i$, for all $1 \leq i \leq n$) is drawn around the n nodes of the star. Each node i knowing its propagation delay from the center can place itself on this imaginary circumference, by delaying the information it receives or transmits by $T_R - \Delta_i$, as shown in Fig. 1.

The roundtrip delay from the imaginary circumference to the center of the optical star, $2T_R$, is divided into f time slots of equal duration, such that, $fK_s/c_{min}^{nominal} = 2T_R$ (K_s is the nominal number of clock cycles in one time slot).

Synchronization Algorithm for Node i. 1. Node i will increment its slot counter after $T_R - \Delta_i$ from the time it has received the first bit of the current time slot.

2. Node i that is scheduled to transmit at time slot k will actually start to transmit after an additional delay of $T_R - \Delta_i$ from the time its slot counter was incremented to k.

2.2. Synchronization Analysis

Next we compute the necessary safety margin for ensuring that successive transmissions from different nodes will not collide with one another; i.e., we want to ensure that the tail of a previous packet will not overlap with the head of the next packet. Collisions may occur at the center of the star, where the signals merge and split. More specifically, in this section we compute what part of the time slot should be left empty as a safety margin, K_{margin}. If each slot has K_s local clock cycles, then the maximum packet size $K_s - K_{margin}$.

Timing Errors. The star can have two types of timing errors:

1. Clock drift, ρ, is the constant clock drift (given by the clock manufacturer), such that $c_{max}^{nominal} = c_{min}^{nominal}$ $(1 + \rho)$.

2. Placement error, δ_s, is the maximum error in measuring the node delay from the center of the optical star, Δ_i (in bit periods).

Safety Margin. The safety margin is determined by the maximum timing difference between two successive transmissions from two different nodes. The interval in which synchronization information is delayed on a node i is $2(T_R - \Delta_i)$. Thus, the maximum error associated with this delay is when Δ_i is zero; i.e., the node is very close to the star's center. The error associated with this delay is $K_{margin} = 2(2\delta_s/c_{min}^{nominal} + \rho f K_s)$ and the *synchronization window* is $\Delta_{max} = K_{margin}/c_{min}^{nominal}$, which is also the lower bound of the global clock period. For example, let δ_s = 10 bit period, $\rho = 10^{-5}$, $K_s = 10^4$ bits, and $f = 10$; then $K_{margin} = 2(2 \times 10 + 1) = 42$ bits, which is only 0.42% of the slot size. Note that the placement error, δ, is by far more significant than the clock drift, ρ (a similar result was obtained in [8]).

3. DISTRIBUTED SCHEDULING FOR HYBRID ACCESS CONTROL

The control channel makes an implicit use of signaling with multiple wavelengths. The idea is to use part of the wide optical spectrum in order to simplify the control of a multiple access data channel. The control channel is used for regulating the access to one or more shared data channels, each with a common wavelength. In the following discussion only a single data channel with wavelength λ_{DATA} is considered.

The scheduling algorithm is designed such that the following characteristics are obtained:

1. *Random and deterministic access modes.* When a node has a packet to transmit and it does not know the load on other nodes, it can transmit it with no delay. If collision occurs the resolution is deterministic such that no additional time slots are wasted. In the deterministic access mode the operation simulates a distributed queue. In every slot each node can add one packet to the queue. Packets that arrive at the same time slot are put in the distributed queue in a predefined order. The network will transfer one packet at a time on a first come, first served basis. As a result of this access policy, all nodes have equal opportunity to add packets into this distributed queue; i.e., a fair access is maintained among the optical star's nodes.

2. *Destination overflow prevention and fairness.* The traffic pattern to the destination can be arbitrary and bursty; the scheduling algorithm is extended such that packets will not be lost as a result of destination overflows. This mechanism avoids a collision at the destination's interface, and ensures *destination fairness,* i.e., equal opportunity by all nodes to transmit successfully to a given destination.

3.1. Principles of Operation

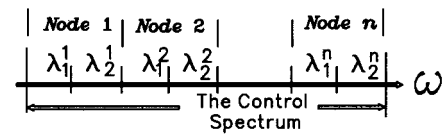
Each node has three transmitters with distinct wavelengths, one for data and two for control. It has two receivers, one fixed for data and one tunable for control. The tunable receiver is used for scanning the control spectrum in order to detect which one is active (implicit control detection). Only the transmission over the data channel is explicitly decoded; i.e., the transmitter clock is recovered with a phase-locked loop or by using the *conservative code* [4, 5].

3.2. The Control Channnel

Implicit signaling is used for control, which means that the different wavelength signals are not decoded explicitly. The receive operation over the control channel just requires the ability to determine, at every time slot, whether or not a specific control wavelength was active; i.e., the existence or absence of a specific wavelength at a certain time slot *constitutes one bit of information.* The advantage of implicit signaling is that it does not require the recovery of the receiving clock from the serial bit stream. Note that the control channel adds essentially no complexity to the digital electronic in the serial interface. It requires only the capability to store some state variables and to count the reservation requests.

For the following scheduling algorithm each node uses two control bits. Therefore, two distinct wavelengths are assigned to each node; i.e., node i will signal its control information using λ_1^i and λ_2^i. The control wavelengths used by node i have the following meaning (as shown in Fig. 2):

- λ_1^i, access request or reservation for one packet in one time slot;
- λ_2^i, overflow indication; it is inactive (no signal is transmitted) as long as the node is in an overflow condition. All other nodes will not transmit data to node i when its overflow signal is inactive. This signal is used also for



λ_1^i	λ_2^i		
0	1	None	*(I'm alive)*
1	1	Packet request	*(I'm alive)*
0	0	Overflow signal	
1	0	Packet request and overflow signal	

FIG. 2 The control channel.

fault tolerance, such that only while this signal is active will this node be considered by other nodes to be "alive."

3.3. Access and Flow Control

The algorithms for access and flow control are combined together. All nodes place themselves on the imaginary circumference (Fig. 1), and as a result, all nodes will execute the following algorithm based on the same global state information.

Data Structure and Variables.

- $OR(t), OR(t-1), \ldots, OR(t-f)$ are $f+1$ variables which record the total number of outstanding requests at the $f+1$ most recent time slots.
- TMP is a temporary variable which is used for updating $OR(t)$ at the end of the time slot.
- Request queue—each time a node receives its own packet reservation signal; it will put in the request queue the time its request should be granted. Note that the requests are granted using the first come, first served policy.
- SC_i is the slot counter value at node i. This is the node's local copy of the global clock, and at all time $t = SC_i$.
- $SLF - RSV$ is the self-reservation variable, which indicates that the node has received its own reservation signal in a random access slot. This variable will contain the time in which the packet that was sent at random should be sent again in case a collision has occurred.
- $OVF_1, OVF_2, \ldots, OVF_n$ are n 1-bit overflow flags—one for each node. OVF_i is set to 1 if no signal was detected on wavelength λ_2^i, and to 0 otherwise.

The above variables change their values according to the state information detected by the tunable receiver. The tunable receiver scans the control spectrum from λ_1^1 to λ_2^n, and it does so for every time slot. Initially the value of the above variables is set to zero.

General Description of the Algorithm. The algorithm operates in an infinite loop with two main procedures:

1. SCAN SLOT procedure—in this procedure the control spectrum is scanned in order to detect the active wavelengths.
2. END of SLOT procedure—in this procedure the state and action of the node for the next time slot are determined. This procedure is further comprised of the following procedures:

(i) My-Turn procedure—which checks if this node has its turn to transmit in the next time slot.

(ii) Request procedure—when the node's interface receives a new packet to transmit, the packet will be sent at random (if there are no outstanding requests), or it will be scheduled deterministically using the implicit signaling.

(iii) Input buffer overflow procedure—when the node i input buffer is full, it will signal it by turning off its λ_2^i transmitter, and will continue to do so as long as its input buffer is full.

(iv) Collision resolution procedure—which determines if the current slot was randomly accessed, and then checks if a collision has occurred. If there was a collision it is resolved by scheduling the colliding requests in a deterministic manner.

ALGORITHM FOR NODE i AT TIME SLOT t

```
1     SCAN SLOT Procedure
1.1     Detect λ_1^j active                          { deterministic access signaling }
1.1.1     TMP = TMP + 1
1.1.2     If j = i and OR(t − f) > 0,                { node i self-request }
                then write t + TMP into the Request Queue
1.1.3     If j = i and OR(t − f) = 0,                { self-request and random access }
                then SLF − RSV = t + TMP
1.2     If detect λ_2^j not active,
1.2.1     then OVF_j = 1 else OVF_j = 0
2     END of SLOT Procedure                          { after scanning λ_2^n }
2.1     If OR(t − f) = 0, then call COLLISION-RESOLUTION Procedure
2.2     Initialization of SLOT t + 1 -
2.2.1     If TMP > 0 ∧ OR(t − f) > 0, then TMP = TMP − 1
2.2.2     SLF − RSV = 0
2.2.3     For k = f to 1, Do OR(t − k) = OR(t − k + 1)
2.2.4     OR(t) = TMP
2.2.5     SC_i = SC_i + 1
2.2.6     t = SC_i
```

2.3 If (HEAD of Request Queue = t), then call MY-TURN Procedure
2.4 If request to send a new data packet -
2.4.1 If Request Queue not FULL and $OR(t) > 0$, then signal in the next with λ_1^i
2.4.2 If $OR(t) = 0$, { random access slot }
then send packet and signal in the next slot with λ_1^i
2.5 If Input Buffer Overflow Hazard -
2.5.1 Signal it by turning off λ_2^i, as long that the overflow hazard exists
3 **COLLISION-RESOLUTION Procedure**
3.1 If $TMP - OR(t) = 1$, then $TMP = TMP - 1$ { no collision has occurred }
3.2 If $TMP - OR(t) > 1$ and $SLV - RSV > 0$, { my collision has occurred }
then write $SLF - RSV$ into the Request Queue
4 **MY-TURN Procedure**
4.1 Destination overflow flag is not set -
4.1.1 Send packet at the next time slot
4.1.2 Read Request Queue
4.2 Destination overflow flag is set -
4.2.1 Read Request Queue
4.2.2 Send another packet and put this packet in its place

3.4. *Correctness of the Algorithm*

In this section we prove the correctness of the algorithm. In the following proofs the numbers in parentheses are references to the algorithm's statements. A time slot gets one of two attributes: random access slot when $OR(t) = 0$ and deterministic access slot when $OR(t) > 0$.

LEMMA 1 (Consistency). *At the beginning of each time slot t all nodes have the same outstanding request vector,* $\mathbf{OR}(t) = OR(t), OR(t-1), \ldots, OR(t-f)$.

Proof. By induction on t. Initially, at all nodes $\mathbf{OR}(0) = 0$. Let $\mathbf{OR}(t)$ be the value of this vector at time t. At the beginning of a time slot $TMP = OR(t)$ and then it is being incremented by the SCAN SLOT Procedure (1.1.1). Since all nodes scan the spectrum in the same way, at the end of the time slot they all have the same value for TMP. If TMP is decremented by 1 all nodes will perform this operation since this decrement depends on its current value, and the values of $OR(t)$ and $OR(t-f)$ in which all nodes have the same value. Thus, at the end of each time slot when $\mathbf{OR}(t)$ is being updated with TMP (statements 2.2.3 to 2.2.6), and therefore $\mathbf{OR}(t+1)$ will be consistent at all nodes. ■

COROLLARY 1. *All nodes will consider a random access slot consistently.*

A slot can be accessed at random only if $OR(t) = 0$ (2.4.2). By Lemma 1 all nodes have the same value for this variable.

LEMMA 2 (Uniqueness). *The request queues of any two nodes cannot contain the same scheduling requests.*

Proof. By Lemma 1 at the beginning of a time slot the value of TMP is the same on all the nodes, and it is incremented by 1 each time a reservation signal is detected (1.1.1). A reservation is stored in the request queue only if the node detects its own signal, which is unique. Therefore, for every reservation signal a single reservation is made on one of the star's nodes. When the next signal is detected TMP is incremented and, therefore, the same value of a scheduling request cannot be stored again in any of the request queues. ■

THEOREM 1 (Random Access). *In a random access slot:* (i) *If no packet was transmitted, no reservation signals will be detected and no packets will be scheduled for transmission in the request queues; or* (ii) *if one packet was transferred successfully (no collision), then this packet will not be added to the outstanding request queue; or* (iii) *if more than one packet was transmitted, they will all be scheduled deterministically for future transmission.* (iv) *In the initialization of slot* $t+1$ TMP *is not decremented by* 1.

Proof. In a random access slot a packet is transmitted together with its reservation signals (2.4.2), and therefore, the proof of (i) is trivial. (ii) If there is only one signal in this slot, it implies that this was a successful random access, and TMP is decremented by 1 (3.1) and remains equal to $OR(t)$. In this case, the $SLF - RSV$ is discarded. (iii) If there is more than one signal in this slot (collision has occurred), then the nodes involved will reschedule the transmission of their packets by placing $SLF - RSV$ in their request queue (3.2), as if this was a deterministic access slot (1.1.3). (iv) In the initialization of slot $t+1$ (2.2) after a random access slot TMP is not decremented by 1 since $OR(t-f) = 0$ (2.2.1). ■

THEOREM 2 (Deterministic Access). *The TMP is incremented by* 1 *if and only if one node has scheduled a transmission request into its request queue.*

Proof. *TMP* was incremented by one when a reservation signal was detected (1.1.1). One node on the star will detect that this is its own signal and schedule its packet for transmission by putting $TMP + t$ into its request queue (1.1.2). And in the other direction, by Lemma 2, it is clear that the request queues of all the nodes contain different values; therefore, at each time slot only one node can be scheduled for transmission. ■

COROLLARY 2. *A single node will transmit in every deterministic access slot.*

COROLLARY 3. *A single node can fully load the optical star.*

THEOREM 3 (Fairness). *The algorithm provides source and destination fairness.*

Proof. We can observe directly from the algorithm that at every time slot each node can make a single request, thus *source fairness,* and when a destination is congested the transmission to it from all other nodes is disabled and enabled in a globally synchronous manner, thus *destination fairness.* ■

4. ANALYSIS OF THE ALGORITHM

Let q_n denote the total number of outstanding requests awaiting at all nodes at the beginning of slot n. Let A_n be the number of new packets arriving at the system during slot n. We assume that A_n is independent from slot to slot identically distributed in each slot. In addition, A_n does not depend on the state of the system. The system evolves according to the following,

$$q_{n+1} = A_n + (q_n - 1)^+ - I(A_n = 1)I(q_{n-f} = 0), \quad (1)$$

where $I(V) = 1$ if the event V holds and $I(V) = 0$ otherwise, and f is the end to end propagation delay in slot units. The explanation of (1) is simple. The number of outstanding requests in the system evolves as in a regular discrete-time queue, except that if the system was empty in slot $n - f$, slot n is a random access slot, hence, if a single packet arrives, it is transmitted successfully.

In the following we present the analysis for $f = 1$, i.e., propagation delay of a single slot. To that end we define $u_n^1 = q_{n-1}$ and $u_n^2 = q_n$. Then the evolution for $f = 1$ is

$$u_{n+1}^1 = u_n^2 \text{ and } u_{n+1}^2 = A_n + (u_n^2 - 1)^+ - I(A_n = 1)I(u_n^1 = 0). \quad (2)$$

Let $G_n(x, y) = E[x^{u_n^1} y^{u_n^2}] = \sum_{i=0}^{\infty}\sum_{j=0}^{\infty} Prob[u_n^1 = i, u_n^2 = j]x^i y^j$. Then

$$\begin{aligned} G_{n+1}(x, y) &= E[x^{u_{n+1}^1} y^{u_{n+1}^2}] = E[x^{u_n^2} y^{A_n + (u_n^2 - 1)^+ - I(A_n = 1)I(u_n^1 = 0)}] \\ &= Prob[u_n^1 = 0, u_n^2 = 0] \\ &\quad \sum_{m=0}^{\infty} Prob[A_n = m] y^{m - I(m=1)} \\ &\quad + \sum_{j=1}^{\infty} Prob[u_n^1 = 0, u_n^2 = j] \\ &\quad \sum_{m=0}^{\infty} Prob[A_n = m] y^{-1}(xy)^j y^{m - I(m=1)} \\ &\quad + \sum_{i=1}^{\infty} Prob[u_n^1 = i, u_n^2 = 0] \\ &\quad \sum_{m=0}^{\infty} Prob[A_n = m] y^m \\ &\quad + \sum_{i=1}^{\infty}\sum_{j=1}^{\infty} Prob[u_n^1 = i, u_n^2 = j] \\ &\quad \sum_{m=0}^{\infty} Prob[A_n = m] y^{-1}(xy)^j y^m. \end{aligned} \quad (3)$$

In steady state ($n \to \infty$) we have from (3) (we derive the steady-state condition later)

$$\begin{aligned} G(x, y) = {} & G(0, 0)[A(y) - a_1(y - 1)] \\ & + [G(0, xy) - G(0, 0)]y^{-1}[A(y) - a_1(y - 1)] \\ & + [G(1, 0) - G(0, 0)]A(y) + [G(1, xy) \\ & - G(0, xy) - G(1, 0) + G(0, 0)]y^{-1}A(y), \end{aligned} \quad (4)$$

where $A(y)$ is the generating function of the arrivals, i.e., $A(y) = \sum_{m=0}^{\infty} Prob[A_n = m]y^m$ and $a_1 = Prob[A_n = 1]$.

Letting $x = 0$ in (4) we have

$$G(0, y) = G(0, 0)a_1(1 - y) + G(1, 0)A(y). \quad (5)$$

By substituting $y = 1$ and $y = 0$ in (5) we have

$$G(0, 1) = G(1, 0); \quad G(0, 0) = G(1, 0)a_0/(1 - a_1), \quad (6)$$

where $a_0 = Prob[A_n = 0]$.

Now let $x = 1$ in (4) to obtain

$$\begin{aligned} G(1, y) = {} & \frac{G(0, 0)a_1(y - 1)(1 - y)}{y - A(y)} + \frac{G(0, y)a_1(1 - y)}{y - A(y)} \\ & + \frac{G(1, 0)(y - 1)A(y)}{y - A(y)}. \end{aligned} \quad (7)$$

By letting $y \to 1$ in (7) and using the fact that $G(0, 1) = G(1, 0)$ (see (6)) we obtain (we use L'Hopitals' rule)

$$G(1, 0) = \frac{1 - A'(1)}{1 - a_1}, \quad (8)$$

where $A'(1)$ is $dA(y)/dy$ at $y = 1$. It is obvious from (8) that the condition for steady state is $A'(1) < 1$.

Note that since $G(1, 0)$ is known, $G(0, 0)$ is also known

(see (6)); therefore $G(0, y)$ is known (see (5)); therefore, the steady-state generating function of q_n (that is $G(1, y)$) is completely determined (see (7)).

To obtain the average number of outstanding requests in the system in steady state, L, we take the derivative of $G(1, y)$ with respect to y at $y = 1$ to obtain

$$L = A'(1) + \frac{A''(1)}{2[1 - A'(1)]} - \frac{a_0 a_1}{1 - a_1}, \tag{9}$$

where $A''(1) = d^2A(y)/dy^2$ computed at $y = 1$.

For Poisson arrivals with rate λ we have

$$L = \lambda + \frac{\lambda^2}{2(1 - \lambda)} - \frac{\lambda e^{-2\lambda}}{1 - \lambda e^{-\lambda}}. \tag{10}$$

We now turn to compute the waiting time of a packet in the system. To that end, let us tag a packet that arrives in slot n. Let $\hat{A}_n$ be the number of packets that arrive during the slot with the tagged packet. Note that the distribution of the number of arrivals in any slot (A_n) is different than the distribution of the number of arrivals in the slot that the tagged packet arrived ($\hat{A}_n$). In fact, $Prob[\hat{A}_n = m] = ma_m/A'(1)$ for $m = 1, 2, \ldots$. Let J (a random variable) denote the position of the tagged packet among the packets that arrive at the same time slot. Then the waiting time of the tagged packet is (here we assume that arrivals are recorded at the beginning of a slot)

$$W = [1 - I(\hat{A}_n = 1)I(q_{n-1} = 0)][2 + (q_n - 1)^+ + J - 1] \tag{11}$$

since if slot n is a random slot and the tagged packet is the single packet to arrive, it does not wait at all. Otherwise, by the scheduling protocol, it waits for all packets ahead of it and two additional slots. From (11) it is possible to derive the Laplace transform of the waiting time distribution. Here we only derive the expected waiting time. We use the facts that $E[J - 1] = A''(1)/(2A'(1))$, $E[I(q_{n-1} = 0)(q_n - 1)^+] = G(0, 0)(1 - a_1) + G(1, 0)[A'(1) - 1]$ (this is obtained from (5)), and that the arrivals do not depend on the state of the system. We obtain

$$E[W] = 2\frac{A'(1) - a_1}{A'(1)(1 - a_1)} + L - \frac{A'(1) - a_1}{1 - a_1} - \frac{a_1}{A'(1)}\frac{1 - A'(1)}{1 - a_1}[A'(1) - 1 + a_0] + \frac{A''(1)}{2A'(1)}, \tag{12}$$

where L is given in (9).

REFERENCES

1. Habbab, I. M. I., Kavehrad, M., and Sundberg, C-E. W. Protocol for very high speed optical fiber local area network with passive star topology. *IEEE J. Lightwave Technol.* **LT-5** (Dec. 1987), 1782–1793.
2. Hullet, J. L., and Evans, P. New proposal extends the reach of metro area nets. *Data Comm.* (Feb. 1988), 139–147.
3. Marhic, M. E. Combinatorial star couplers for single-mode optical fibers. *FOC/LAN'84*, 175–179.
4. Ofek, Y. The conservative code. *IEEE Trans. Comm.* **38,** 7 (July 1990), 1107–1113.
5. Ofek, Y. An encoder/decoder system and methodology utilizing conservative coding with block delimiters, for serial communication. U.S. Patent No. 4,864,303.
6. Ofek, Y., and Faiman, M. Distributed global event synchronization in a fiber optic hypergraph network. *The 7th International Conference on Distributed Computing Systems,* 1987, pp. 307–314.
7. Ofek, Y. Generating a fault tolerant global clock in a high speed distributed system. *The 9th International Conference on Distributed Computing Systems,* 1989, pp. 218–226.
8. Ofek, Y. Generating a fault tolerant global clock using high-speed control signals for the MetaNet architecture. *IEEE Trans. Comm.* to appear.
9. Schmidt, R. V., *et al.* Fibernet II: A fiber optic ethernet. *IEEE J. Selected Areas Comm.* **6,** 1 (Nov. 1983), 702–710.
10. Scholl, F. W., and Coden, M. H. Passive optical star systems for fiber optic local area networks. *IEEE J. Selected Areas Comm.* **6,** 6 (July 1988), 913–923.

Received March 18, 1992; revised August 25, 1992; accepted November 13, 1992

YORAM OFEK received his B.Sc. degree in electrical engineering from the Technion-Israel Institute of Technology in 1979, and his M.Sc. and Ph.D. degrees in electrical engineering from the University of Illinois–Urbana in 1985 and 1987, respectively. From 1979 to 1982 he was affiliated with RAFAEL as a research engineer. During 1983–1984 he was at Fermi National Accelerator Laboratory, Batavia, Illinois. Since 1987 he has been a research staff member at the IBM T. J. Watson Research Center, Yorktown Heights, New York. His main research interests are access-control, routing, flow-control and fairness in local and wide area networks, high speed optical networks, distributed algorithms and systems, clock synchronization, self-stabilization, and fault tolerance. He initiated and has been leading the research activities on the MetaRing and MetaNet architectures.

MOSHE SIDI (S'77–M'82–SM'87) received the B.Sc., M.Sc., and D.Sc. degrees from the Technion-Israel Institute of Technology, Haifa, Israel, in 1975, 1979, and 1982, respectively, all in electrical engineering. From 1975 to 1981 he was a teaching assistant and a teaching instructor at the Technion in communication and data networks courses. In 1982 he joined the faculty of the Electrical Engineering Department at the Technion. During the academic year 1983–1984 he was a post-doctoral associate in the Electrical Engineering and Computer Science Department at the Massachusetts Institute of Technology, Cambridge, Massachusetts. During 1986–1987 he was a visiting scientist at the IBM T. J. Watson Research Center, Yorktown Heights, New York. He received the New England Academic Award in 1989. He coauthored the book "Multiple Access Protocols: Performance and Analysis," Springer-Verlag, 1990. Currently he serves as the Editor for Communication Networks in the *IEEE Transactions on Communications* and as the Associate Editor for Communication Networks and Computer Networks in the *IEEE Transactions on Information Theory*. His current research interests are in queueing systems and in the area of computer communication networks.