Golden Ratio Scheduling for Flow Control with Low Buffer Requirements

Shivendra S. Panwar, *Member, IEEE,* Thomas K. Philips, and Mon-Song Chen

Abstract—**In this paper, we describe a method of flow control that requires very few buffers to be allocated at each node to virtual circuits (or sessions) that have to traverse many links. Transmissions are scheduled using the Golden Ratio Policy of Itai and Rosberg. We show that the buffer requirements of a session grow at most logarithmically with the number of slots allotted to it. As an immediate consequence, intra-network delays are bounded.**

I. Introduction

CONSIDER a network of computers (or nodes) interconnected by transmission links (or edges) communicating with each other via a store and forward mechanism. The virtual circuit that is set up between two communicating processes in distinct computers is called a *session*. Each link in the network may carry traffic from hundreds of sessions, and each session is allotted a share of the network's capacity. The task of allocating capacity to the sessions and controlling the entry and forwarding of their messages is referred to as *flow control*.

In this paper we describe and analyze a flow control policy based on time division multiple access (TDMA) that can allocate capacity to sessions in any desired proportion. We assume that the capacity of each link is the same. We further assume that the time axis is *slotted*, and that messages can be broken up into an integral number of constant length packets, each of which can be transmitted in a slot. At each node a *transmission schedule* is defined for each link over which packets are to be transmitted. The transmission schedule is a mapping from the set of all sessions that use a link onto the set of all slots available on that link. The transmission schedule on every link is *periodic*, i.e., if slot i is allotted to a given session, then slot $i + N$ is allotted to the same session. N is called the *period* of the schedule, and is the same for every link. The transmission schedule is constructed using the Golden Ratio Policy of Itai and Rosberg [8]. As the time axis is slotted, and as not all the slots need to be preallocated to sessions, we believe that the Golden Ratio Policy will find use as a flow and congestion control mechanism in Asynchronous Transfer Mode (ATM) networks [1], [6], [11].

Slots are preassigned to sessions, and a slot is used if and only if a packet from its preassigned session awaits transmission. Unassigned slots are not ever used. Consequently, the service provided by the policy is circuit switching, even though it may be implemented using packet switching technology. While this causes slightly longer queueing delays at the source, it has the important benefit of greatly reducing buffer requirements and obviating the need for window flow control. At each node, buffers are preallocated to sessions. As the number of buffers required at a node is upper bounded by the sum of the buffer requirements for each session, this gives us an upper bound on the number of buffers required at every node.

As the schedule is periodic, we restrict our attention to a single *transmission cycle*, which is defined to be a block of N contiguous slots. A session is allotted slots according to its capacity requirements, and is allotted the same number of slots on every link that it passes through. Consequently, some slots may remain unused. The resulting flow control policy has the following properties.

1) Sessions require very few buffers at intermediate nodes.
2) Intra-network delays are bounded (and low).
3) It is stable in heavy and/or bursty traffic.
4) The policy is distributed; i.e., after all the nodes on a session's path have been informed of its capacity requirements and the period of the schedule, no further communication between nodes is required.

In related work, Hahne [5] showed that if, at every node, sessions were allowed to transmit in a round robin fashion and sufficiently large windows are used, then sessions would automatically be allotted their max-min fair rates (for the definition of max-min fairness see [3]. Note that this policy is completely distributed, and requires no communication between processors. Unfortunately, the window size required is very large. When the window size was restricted, max–min fairness was not achieved in general, even though the throughput of a session was lower bounded. Mukherji [13] has analyzed a TDMA policy in which each session is allotted at least one slot during each transmission cycle, thus lower bounding its throughput, and accommodating isochronous traffic such as voice. Uncommitted slots are allocated to sessions according to their instantaneous requirements. In addition, Mukherji provides upper bounds on the delays experienced

Paper approved by the Editor for Communication Networks of the IEEE Communication Society. Manuscript received January 15, 1989; revised November 15, 1990. This work was supported in part by the New York State Center for Advanced Technology in Telecommunications, Polytechnic University, Brooklyn, NY 11201, and by the National Science Foundation, under Grant NCR-8909719. This paper was presented in part at IEEE GLOBECOM '88, Hollywood, FL, November 28–December 1, 1988 and at the Allerton Conference on Communication, Control, and Computing, University of Illinois, Urbana-Champaign, IL, October 3–5, 1990.

S. S. Panwar is with the Department of Electrical Engineering and Computer Science, Polytechnic University, Brooklyn, NY 11201.

T. K. Phillips and M.-S Chen are with IBM T. J. Watson Research Center, Yorktown Heights, NY 10598.

IEEE Log Number 9107309.

by packets, and describes efficient approximation algorithms for the construction of transmission schedules.

There is a vast body of literature on other forms of flow control including, but not limited to, mechanisms of the store and forward kind. The interested reader is directed to the excellent surveys in [4], [12], and [18].

The remainder of the paper is organized as follows. In Section II, we describe the Golden Ratio Policy. Section III is devoted to an exploration of its properties. In Section IV, we examine the performance of the policy, and discuss design considerations in Section V. Finally, in Section VI, we draw our conclusions and identify some open problems.

II. The Golden Ratio Policy

The policy to be described is an extension of the Golden Ratio Policy of Itai and Rosberg [8] to a network. Suppose we have S sessions on a given link labeled $1, 2, \cdots, S$, and session i is to be allotted X_i slots, where for clarity, we have suppressed the subscript identifying the link. The session is assigned the same number of slots on every link it passes through. The period of the schedule N is given, and is the same throughout the network. We further require that $\sum_{i=1}^{s} X_i = N$ on every link. Consequently, it may be necessary to introduce dummy sessions on some links so as to ensure that all of the above constraints are met. Let $\phi^{-1} \triangleq \frac{\sqrt{5}-1}{2} = .6180339\cdots$. ϕ is known as the Golden Ratio, and is related to the Fibonacci numbers via

$$F_k = \frac{\phi^k - (1-\phi)^k}{\sqrt{5}}. \tag{1}$$

The Fibonacci numbers can also be generated by the linear recurrence $F_{k+1} = F_k + F_{k-1}$, with $F_0 \triangleq 0$ and $F_1 \triangleq 1$. The next ten numbers in the sequence are 1, 2, 3, 5, 8, 13, 21, 34, 55, and 89. Following [*&itairos*], mark off the points $\phi^{-1} \bmod 1$, $2\phi^{-1} \bmod 1, \cdots, N\phi^{-1} \bmod 1$ on a circle of circumference 1, dividing it into N intervals. Allot to session 1 the slots corresponding to $\phi^{-1} \bmod 1$, $2\phi^{-1} \bmod 1, \cdots, X_1\phi^{-1} \bmod 1$, to session 2 to those corresponding to $(X_1+1)\phi^{-1} \bmod 1, \cdots, (X_1+X_2)\phi^{-1} \bmod 1$, and so on until all the slots have been allotted.

Consider the following example: $S = 3$, $N = 8$, $X_1 = X_2 = 3$, $X_3 = 2$. The network is assumed to have just two nodes, say 1 and 2, and a single edge joining them. All the sessions are assumed to start at node 1 and end at node 2. The transmission schedule on the edge is $2\,1\,3\,2\,1\,2\,1\,3\cdots$ as can be verified by direct computation.

We define a *frame* to be a transmission cycle which starts at the slot corresponding to the first mark on the circle following 0 in the clockwise direction. In the example above, $2\,1\,3\,2\,1\,2\,1\,3$ is a frame.

Note that successive transmission permits to a session are evenly spaced. This property of the Golden Ratio policy has been explored in great detail [16], [17] and has found use in multiplicative hashing [10] and the design of TDMA protocols [7], [8], [15]. Note that any irrational number could have been used to generate the sequence of slots- the Golden Ratio is optimal in that the transmission schedule generated by it is as evenly spaced as possible [10], [16], [17].

III. Properties of the Golden Ratio Policy

Our primary objective is to minimize the number of buffers that must be allotted to a session so that a transmitted packet is never lost. As sessions are preassigned slots, they do not interact with each other, and so it suffices to examine the buffer requirements of a single session. In all that follows, unless stated explicitly to the contrary, we shall assume the following six conditions to hold.

1) N, the period of the schedule, is the same on every link.
2) All links have the same capacity.
3) X, the number of slots allotted to the chosen session, is the same on every link it traverses, and slots that are not allocated to a session are not ever used.
4) At every node, there is one transmission buffer per link to hold the packet being transmitted.
5) Nodes can transmit packets on all outgoing links and receive packets on all incoming links simultaneously.
6) Frames are not necessarily synchronized, i.e. frames at different nodes do not necessarily start at the same time. Any offset between the frames, however, do not vary with the time (if not, and if a fast frame was upstream of a slow one, it would eventually cause the downstream node's buffers to overflow with probability 1).

With these preliminaries behind us, we state and prove our main theorems. The proof of the first theorem, being lengthy, is deferred to the appendix.

Theorem 1: Under assumptions 1–6, if $N = F_k$, $k \geq 4$ and $X = F_{k1}$. $3 \leq k_1 \leq k$, a session requires at most two buffers at every node (other than its source and destination) that it passes through. Furthermore, the bound is tight. If $k_1 \leq 2$ or $k_1 = k$ (i.e., if $X = 1$ or $X = N$), exactly one buffer is required.

This theorem, while elegant, is restrictive for two reasons.

1) It is not always possible to force N to be a Fibonacci number.
2) As $\sum_{i=1}^{s} X_i = N$, some of the X_i may not be Fibonacci numbers.

In both cases, the above result does not always hold, because the interpermit intervals are most regular when N and X_i are Fibonacci numbers [8]. In the following important special case, however, we can prove a variant of Theorem 1.

Theorem 2: In addition to assumptions 1–5, assume that

1) N is not a Fibonacci number.
2) There is a network wide frame synchronization—i.e, the slots corresponding to ϕ^{-1}, $2\phi^{-1}, \cdots$ start at exactly the same time at every node.

Then if $X \geq 2$ is a Fibonacci number, a session requires at most two buffers at every node (other than its source and destination) that it passes through. Furthermore, the bound is tight. If $X = 1$ or $X = N$ exactly one buffer is required.

Proof: If $X = 1$ or N, each packet is forwarded before the next one arrives, so that exactly one buffer is required. If $X \geq 2$ we proceed as follows. Let F_k be the

smallest Fibonacci number that is larger than N. Construct a schedule with F_k slots, and allocate the first N slots (those corresponding to $\phi^{-1}, \cdots, N\phi^{-1}$) as before. Do not allocate the remaining slots. From Theorem 1, the session requires at most two buffers. Next, at each node, delete the slots corresponding to $(N+1)\phi^{-1}, \cdots, F_k\phi^{-1}$. Network wide frame synchronization ensures that the deleted slots are positioned identically at every node, and consequently their deletion cannot change the relative positions of the slots allotted to the session. The buffer requirement cannot therefore change. □

Note that the proof of the last Theorem does *not* assume that session 1 is allotted the same slots at every node. It may be allotted the slots corresponding to $\phi^{-1}, 2\phi^{-1}, \cdots$ at its start, those corresponding to $m\phi^{-1}$, $(m+1)\phi^{-1}$, $\cdots$ at the next node and so on. In fact, Theorem 2 holds if each node is synchronized to its neighbors within one slot time. When the number of slots allotted to a session is not a Fibonacci number, the number of buffers required cannot be computed exactly. The following upper bound, however, holds.

Theorem 3: In addition to assumptions 1–5, assume that

1) N is a Fibonacci number, or if it is not, that there is network wide frame synchronization.
2) X is not a Fibonacci number.

Then the number of buffers required by the session at every node other than the source and destination is upper bounded by $2m$ where m is the smallest number of distinct Fibonacci numbers that X can be represented as the sum of; and $2m$ in turn is upper bounded by $2\left\lceil \frac{\lceil \log_\phi \sqrt{5} X_1 \rceil - 1}{2} \right\rceil$.

Proof: Let $F_{k_1} < X < F_{k_1+1}$. From [9] we know that we can always effect the decomposition $X = F_{i_1} + \cdots + F_{i_m}$, $F_{i_j} \neq F_{i_l}$, $j \neq l$. Furthermore, $k_1 = i_1 \geq i_2 + 2 \geq i_3 + 4 \geq \cdots \geq i_m + 2m - 2 \geq 2$. Decompose the session into m "minisessions" labeled $1, \cdots m$ by allocating the session's slots to these minisessions in the following manner. Assign to minisession 1 the slots corresponding to $\phi^{-1}, \cdots, F_{i_1}\phi^{-1}$, to minisession 2 those corresponding to $(F_{i_1}+1)\phi^{-1}, \cdots, (F_{i_1}+F_{i_2})\phi^{-1}$, and so on until all X slots have been alotted. Each minisession requires at most two buffers, and the buffer requirements are at worst additive. It immediately follows that the number of buffers required is upper bounded by $2m$. In the worst case, we have $X = F_{k_1} + F_{k_1-2} + \cdots + F_1$ if k_1 is odd, and $X = F_{k_1} + F_{k_1-2} + \cdots + F_2$ if k_1 is even. There are at most $\left\lceil \frac{k_1}{2} \right\rceil$ distinct numbers in the decomposition. In addition, $\frac{\phi^{k_1}}{\sqrt{5}} < F_{k_1} + 1 \leq X \leq F_{k_1+1} - 1 \leq \frac{\phi^{k_1+1}}{\sqrt{5}}$, so that $k_1 = \lceil \log_\phi (\sqrt{5} X) \rceil - 1$, implying that

$$\text{Number of buffers required} \leq 2\left\lceil \frac{k_1}{2} \right\rceil = 2\left\lceil \frac{\lceil \log_\phi \sqrt{5} X \rceil - 1}{2} \right\rceil. \tag{2}$$

□

When k_1 is odd, this bound can be tightened. Two cases arise: in the first, the decomposition of X does not contain F_5, while in the second, it does. In the first case, the number of buffers required is clearly upper bounded by $2\left[\left\lceil \frac{\lceil \log_\phi \sqrt{5} X \rceil - 1}{2} \right\rceil - 1\right]$. In the second case, from [9] we must have $X - F_{k_1} - \cdots - F_5 \leq 2$. If the remainder is 0, we have reduced the number of terms in the expansion by 2, while if it is 1 or 2, the number of terms is reduced by 1, as $F_1 = 1$ and $F_3 = 2$. It follows that when k_1 is odd, the bound can be tightened to $2\left[\left\lceil \frac{\lceil \log_\phi \sqrt{5} X \rceil - 1}{2} \right\rceil\right]$. If S sessions use a link, the total number of buffers required for that link is approximately

$$\sum_{i=1}^{S} \log_\phi \sqrt{5} X_i \leq S \log_\phi \frac{\sqrt{5} N}{S}. \tag{3}$$

The second step follows from the observation that the total buffer requirement is maximized by alloting the same number of slots to every session. The bound is reasonably tight when the number of sessions is large, but is rather loose when the number of sessions is small. As the buffer requirement for two or more session is subadditive, the total number of buffers required is maximized when each slot is allotted to a different session (in which case N buffers are required), and minimized when each slot is allotted to a single session (in which case only 1 buffer is required). Theorem 3 implies that intra-network delays are bounded, as a packet can wait for at most a finite number of buffers to empty at every node that it passes through. In [8] it is shown that the interpermit distance is at most F_{k-k_1+2} slots where F_k is the smallest Fibonacci number greater than or equal to N and F_{k_1} is the largest Fibonacci number less than or equal to X. The following corollary to Theorem 3 follows immediately.

Corollary: Under the same assumptions made in Theorem 3, if the length of a transmission cycle is T seconds, and the session passes over L links, the intra-network delay (excluding any delay at the source and all propagation delays on the links) is upper bounded by $(L-1) \times \left[2\left\lceil \frac{\lceil \log_\phi \sqrt{5} X \rceil - 1}{2} \right\rceil \times F_{k-k_1+2} + 1\right] \times T/N$. The one accounts for the transmission time of the packet. □

IV. Performance Analysis

The theorems presented in the last section assume either that the total number of slots is a Fibonacci number or that there is network wide frame synchronization. In many applications, however, these conditions may not be met. We cannot then prove analogs of the theorems presented earlier. The difficulty lies in the fact that the spacing between consecutive transmission permits can take on three different values [8], and the exact sequence of intervals formed cannot be easily determined.

Bounds, however, may still be calculated on the number of buffers required via the following "staircase" technique. Construct a rectangle of width 1 and height X. On the x axis, starting at $x = 0$, mark off the start of intervals allocated to session 1 as determined by the Golden Ratio Policy. Draw horizontal lines at $y = i + 1$ between the ith and the $i+1$st point on the x axis, and vertical lines to join the endpoints of the horizontal lines. The result is an uneven staircase that starts at $(0, 1)$ and ends at $(1, X)$, with jumps at multiples of $\frac{1}{N}$ as

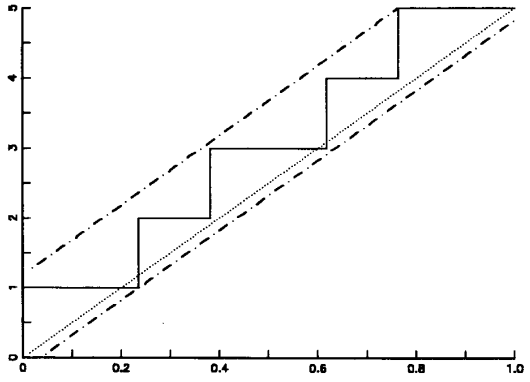

Fig. 1. The staircase for $X = 5$ and $N = 8$.

shown in Fig. 1. The staircase represents the flow of messages from the session over the course of a transmission cycle. The points at which the staircase jumps correspond to the transmission of a message (if the staircase is used to represent the transmission schedule on an edge) or the reception of one (if it is used to represent the transmission schedule on an upstream edge).

Any cyclic shift of the transmission sequence also generates a valid staircase. If two staircases are superimposed on each other, the maximum difference between the two is the number of buffers required to hold enqueued messages. This difference may be upper bounded by bounding the staircase between two straight lines of slope X. Let the equation of the first line be $y = X \times x + c_1$ and that of the second be $y = X \times x + c_2$. c_1 is chosen so that the second line touches the staircase at one or more corners alone and lies below the staircase at all other points. c_2 is chosen so that the second line touches the staircase at one or more corners alone and lies below the staircase at all other points. $2(c_1 - c_2)$ is an upper bound on the maximum distance between the two staircases (one representing incoming packets and the other representing outgoing packets), and, as the number of buffers required is an integer, $\lfloor 2(c_1 - c_2) \rfloor$ upper bounds the number of buffers required by a session at any intermediate node. It is interesting to examine the variation in the number of buffers required for a fixed value of N as X is varied, as shown in Fig. 2 for $N = 89$. The curve is jagged, but tends to increase with X as can be seen from its envelope. Sharp dips are seen at Fibonacci numbers. The same dips are seen if N is not a Fibonacci number.

Last, we analyze queueing delays at the source node. We assume that the arrival process of the packets is Poisson with parameter λ per slot. A session that is allotted X slots has a utilization of $\rho = \lambda \times N/X$. An exact analysis of the queuing delay for arbitrary arrival processes can be found in [7] where an algorithm to compute the delays is presented. The computation requirements are $O(X^3)$ and the storage requirements are $O(X^2)$. For large X, these can be prohibitive. For Poisson inputs, however, a simple approximation for the mean waiting time exists [14]. The computational complexity and storage requirements are both $O(X)$. To put this in perspective, it took over 12 h of CPU time

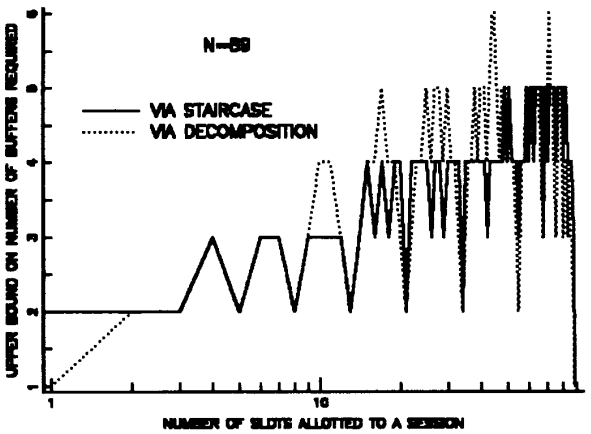

Fig. 2. The upper bound on the number of buffers required.

on an IBM 3090 mainframe to generate the exact solution for Figure &errfig., while the approximation took only a quarter of a second. Let the distances between successive permits to the session, measured in slots, be $d_1, \cdots, d_X$. Define the average interpermit distance $\bar{d} \triangleq \frac{N}{X}$. The utilization ρ is the mean number of arrivals in $\bar{d}$ slots. Then if $\rho < 1$, the mean waiting time of a packet is approximately

$$\overline{W}(X,N) \approx \left\{ (1+\rho) \sum_{i=1}^{X} \frac{d_i^2}{2N} + \frac{\bar{d}\rho^2}{2[1-\rho]} \right\} \frac{T}{N} \qquad (4)$$

where T is the length of transmission cycle (in seconds). The mean response time, $\overline{R}(X,N)$, is simply the sum of the mean waiting time and the service time (one slot time) so that

$$\overline{R}(X,N) \approx \left\{ (1+\rho) \sum_{i=1}^{X} \frac{d_i^2}{2N} + \frac{\bar{d}\rho^2}{2[1-\rho]} + 1 \right\} \frac{T}{N}. \qquad (5)$$

The approximation was compared to the exact solution in [7], and the error in the approximation is plotted in Fig. 3 for $N = 89$. The error is less than 2.1% over the entire range of X and for values of ρ between 0.1 and 0.9. This is low enough for most practical purposes. In the special case when all the interslot distances are equal, (4) is exact. If the interslot distances are unequal, the approximation becomes exact as $\rho \to 0$.

V. Design Considerations

We now present two extensions of the Golden Ratio Policy that reduce the buffer requirements and eliminate the wastage of unused slots. Let $B(X,N)$ be the buffer requirement of a session which is allotted X slots in a cycle of length N. If $X = \sum_{i=1}^{m} X_i$, then $B(X,N) \leq \sum_{i=1}^{m} B(X_i,N)$ as buffer requirements are at most additive. This implies that multiplexing sessions can reduce the total buffer requirement.

Given N, X and $(X_1, X_2, \cdots, X_m)$ such that $X = \sum_{i=1}^{m} X_i$, the reduction in the number of buffers required as a result of multiplexing is given by $\Delta B \triangleq \sum_{i=1}^{m} B(X_i,N) - B(X,N)$. If N and m are fixed, the average reduction in the buffer requirement can be computed as follows. For each X between 2 and $N-1$, compute ΔB for all possible partitions

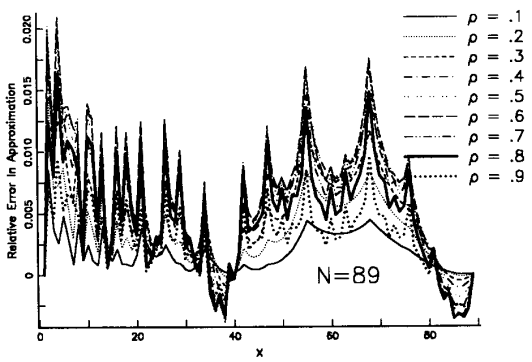

Fig. 3. Error in mean waiting time.

TABLE I
AVERAGE AMOUNT OF BUFFER REDUCTION FROM MULTIPLEXING

	Average Buffer Reduction (percentage)		
N	$m = 2$	$m = 3$	$m = 4$
89	2.8 (39%)	5.5 (55%)	8.1 (64%)
150	3.0 (40%)	8.1 (71%)	11.8 (79%)
300	3.5 (42%)	6.8 (58%)	10.1 (67%)

of X into m integers. Then find the average of ΔB over all possible partitions and all possible values of X, assuming each value of X and each partition to be equally likely. For example, if $X = 4$ and $m = 2$, the set of all possible partitions is $\{(1,3),(2,2)\}$. The mean reduction in the buffer requirement for $X = 4$ is

$$\frac{2 \times B(4,N) - B(1,N) - B(3,N) - 2 \times B(2,N)}{2}. \quad (6)$$

As an exact computation of the buffer requirements is infeasible, we have used the upper bounds on $B(X,N)$ derived from the staircase construction. Computational experience, however, shows these bounds to be very tight. Table I shows the average reduction in buffer requirements for three different values of N and m. Clearly, the reduction, which ranges from 39 to 79%, is significant. Also note that the effectiveness of multiplexing increases with m.

This immediately raises the question of which sessions are best multiplexed. Three criteria for deciding which sessions to multiplex are examined in [2], namely, the "path-based" scheme, the "route-based" scheme and the "hop-count" scheme. In the path-based scheme, two routes are combined into a single entity when they enter a node if both their destinations and subsequent routes are identical. In the route-based scheme, sessions are combined into a single entity when they have the same source, destination, and physical path. In the hop-count scheme, two sessions are combined if

1) They have traversed the same number of links from their respective sources, or
2) If they have the same number of links to traverse to reach their respective destinations.

The path-based scheme is shown to be better than the route-based scheme by a factor of the "average route length" (measured in hops), which is usually about 3 to 4 in typical networks. The hop-count scheme creates the fewest multiplexed entities, but is not desirable because of possible congestion unfairness.

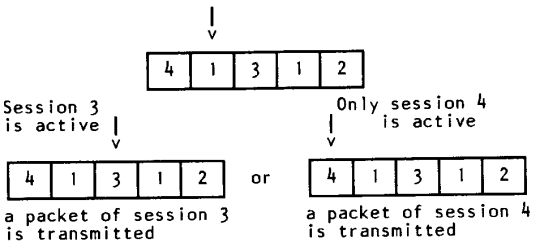

Fig. 4. An example of the single pointer scheme.

We now examine ways to utilize unused slots. The Golden Ratio Policy is a scheduled based scheme, and slots can be used only by the session assigned to them. This problem, which is typical of time division multiplexed (TDM) schemes can be solved by adopting statistical time division multiplexing (STDM). Two STDM schemes which are derived from the Golden Ratio Policy are next presented.

The first scheme, which we call "dynamic substitution," is based on a scheduling sequence and a replacement selection procedure. The scheduling sequence, say SEQ, is generated according to the Golden Ratio Policy. For example, $SEQ(i) = j$ implies that slot i is assigned to session j. At each slot, if the designated session is active, then one of its packets will be transmitted in the slot. If not, a replacement session will be selected to use the slot instead.

The selection of the replacement session can be done in many ways. For example, bursty sessions may be prioritized over stream oriented sessions, as the latter tend not to require additional bandwidth. Among bursty session, moreover, the ones with heavier load should be given higher priority. Another intuitively appealing idea is to select the session with the longest queue.

The second scheme, referred to as the "single pointer" scheme, also uses the Golden Ratio Policy to generate the scheduling sequence SEQ. Unlike the dynamic substitution scheme, however, there is no fixed association between the indexes in SEQ and the slot numbers. Instead, the actual transmission is completely governed by a single pointer in the following manner.

1) Move the pointer to the next entry of SEQ
2) If the session pointed to is active, its packet will be transmitted
3) Otherwise go to Step 1.

Take the simple configuration depicted in Fig. 4 as an example. There are four sessions and their scheduling sequences are $SEQ = (4,1,3,1,2)$. Let the transmission be for session 1, which is indicated by the position of the pointer. For the current slot, if session 3 is active, its packet will be transmitted. If on the other hand only session 4 is active, then a packet of session 4 will be transmitted, as depicted in the figure.

Both these schemes allocate bandwidth dynamically to sessions. They do, however, need an additional flow control policy to prevent buffer overflow. An analysis of the buffer requirement under these policies is beyond the scope of this paper. Finally, we discuss some properties of the single pointer scheme.

Thereom 4: Consider a Golden Ratio Policy with S sessions and let $\sum_{i=1}^{S} X_i \triangleq N$ where the $\{X_i\}$ and N need not be Fibonacci numbers. Suppose that sessions are sequentially assigned, i.e., session 1 is assigned its slots first, session 2 is assigned its slots next, and so on until all the slots have been allotted. Then when session S or 1 is idle, and the remaining $(S-1)$ sessions are always active, the Single Pointer scheme is a Golden Ratio Policy with $(S-1)$ sessions and cycle length $(N - X_s)$ or $(N - X_1)$.

Proof: When session S is idle, the entries of session S are essentially absent from the scheduling sequence. Equivalently, the points $(X_1 + \cdots + X_{S-1} + 1)\phi^{-1} \mod 1$ are removed from the circle. The remaining $(N - X_S)$ points are identical to those of a Golden Ratio Policy with $(S-1)$ sessions and a cycle length of $(N - X_S)$ slots. The correctness of the claim follows.

When session 1 is idle, the first X_1 marks on the unit circle are ignored. The remaining $(N - X_1)$ marks are a rotation of the set of marks generated for a Golden Ratio Policy with $(S-1)$ sessions and cycle length $(N - X_1)$ slots. Once again, the correctness of the claim follows.

Corollary: Consider a Golden Ratio Policy with S sessions, let $\sum_{i=1}^{S} X_i \triangleq N$, and suppose that sessions are sequentially assigned. Then if sessions $i, i+1, \cdots, 1+Z-1$ are the only active sessions, the single pointer scheme is identical to a Golden Ratio Policy with Z sessions and a cycle length of $X_i + X_{i+1} + \cdots + X_{i+Z-1}$.

Proof: This theorem can be proven by repeatedly applying Theorem 4 to each of the idle sessions. □

Another implementational issue that is of importance is the process by which sessions are added and deleted after the initial schedule has been computed. A session is easily deleted by disallowing transmissions in the slots assigned to it (possibly by marking its slots as unused). If a new session which requires X slots is to be added, and if a contiguous block (in Golden Ratio sequence) of X or more slots is available, some (or all) of these slots can be assigned to the session. If no such block is available, it is necessary to reallocate two or more active sessions into a single continguous block so as to free up enough slots for the new session. Doing so can cause a backlog of packets, thus requiring additional buffer space if packet loss is to be avoided. Provided that a session does not transmit continuously, this backlog will eventually be cleared. If the addition of sessions is done only infrequently, it may be possible to simply stop all transmissions for a short while, recompute the schedule, and then restart transmission. A backlog of packets may build up at the source of each session, but will be cleared if the sessions do not transmit continuously. No completely satisfactory solution (other than the temporary imposition of some form of window flow control) to this problem has been found, and we leave this as an important open problem.

VI. Conclusion

A schedule based flow control scheme constructed around the Golden Ratio policy that requires very few buffers and guarantees low end-to-end delays in a network has been presented. Various properties of the policy have been determined, and a number of implementational issues have been examined. Many open questions remain, however.

Tighter bounds on the buffer requirements when the number of slots assigned to a session is not a Fibonacci number would be of interest. Additionally, some understanding of the behavior of the interslot distances when the total number of slots is not a Fibonacci number would prove useful.

Two other topics merit further investigation.

1) A provably optimal (under some suitable measure of optimality) scheme for combining sessions to reduce the buffer requirements.

2) A policy for assigning idle slots to sessions with enqueued packets so as to minimize (over all the sessions) the maximum mean queue length. We conjecture that the optimal policy will be to serve the session with the longest queue, but no proof of this has been found.

Appendix

For convenience, we first produce Lemma 5.1 and Corollary 5.1 from [8].

Corollary 5.1: If $N = F_k$ and $X = F_{k_1}$, there are F_{k_1-2} interslot distances of length l_{k_1}, and F_{k_1-1} interslot distances of length l_{k_1-1}.

Lemma 5.1: If $N = F_k$, $l_m = F_{k+1-m}$.

Lemma 1: Consider a schedule of length $N = F_k$, $k \geq 3$. For a session assigned F_{k_1} slots, $k_1 < k$, the Golden Ratio schedule can be generated the following way. If the number of slots in the schedule are numbered 1 to F_k, then allot slot numbers $F_{k-1} \mod F_k$, $2 \times F_{k-1} \mod F_k, \cdots, F_{k_1} \times F_{k-1} \mod F_k$ to the session.

Proof: This follows from Lemma 5.1 in [8] with $m = 2$. Each new slot must be F_{k-1} slots clockwise from the last slot. □

Comment: Lemma 1 provides a practical way to allocate slots to sessions without any need to perform real arithmetic.

Lemma 2: $F_{k_1} \times F_{k-1} = (-1)^{k_1+1} F_{k-k_1} \mod F_k$, $k \geq 1$, $1 \leq k_1 < k$.

Proof: By induction on k_1. This relation is true for $k_1 = 1 (F_1 = 1)$ and $k_1 = 2 (F_2 = 1)$ for any $k \geq k_1$. Assume it to be true for all $k_1 < k$. Then

$$\begin{aligned} F_{k_1+1} \times F_{k-1} &= (F_{k_1-1} + F_{k_1}) \times F_{k-1} \\ &= (-1)^{k_1+1}[F_{k-k_1} - F_{k-k_1+1}] \mod F_k \\ &= (-1)^{k_1+2} F_{k-k_1-1} \mod F_k. \end{aligned}$$

□

Consider the schedule of a session assigned F_{k_1} slots in a schedule of length F_k. For a particular node in the sessions path, we will call the schedule on the incoming and outgoing links, schedule I and schedule O, respectively. Number the slots assigned to the session under schedules I and O $I_1, I_2, I_3, \cdots, I_{F_{k_1}}$, and $O_1, O_2, \cdots, O_{F_{k_1}}$ in the order they are generated by the Golden Ratio Policy. The relative positions

of these slots under schedules I and O are cyclic shifts of one another. The schedules need not be slot synchronized, i.e., I may be shifted from O by a fraction of a slot. The interslot distances, measured in slots, are either F_{k-k_1+1} or F_{k-k_1+2} for $k_1 \geq 1$ and $k \geq k_1$ (Lemma 5.1, Remark 5.1, [8]).

Consider schedule I, and denote in parentheses the length of the intervals, measured in slots, between successive slots assigned to the session, starting with the slot number F_{k-1} (i.e., the first slot assigned to the session in the clockwise direction. Thus, for $k_1 = 2$, we have $I_1(F_k)$, for $k_1 = 3$, we have $I_1(F_{k-1})I_2(F_{k-2})$, for $k_1 = 4$ we have $I_1(F_{k-3})I_3(F_{k-2})I_2(F_{k-2})$ and for $k_1 = 5$ we have $I_1(F_{k-3})I_3(F_{k-3})I_5(F_{k-4})I_2(F_{k-3})I_4(F_{k-4})$.

Lemma 3: If $X = F_{k_1}$ and $N = F_k$, the following hold.

1) For k_1 even, the intervals of length F_{k-k_1+2} are located in the schedule fragments $I_{F_{k_1-2}+1}(F_{k-k_1+2})I_1$, $I_{F_{k_1-2}+2}(F_{k-k_1+2})I_2, \cdots, I_{F_{k_1}}(F_{k-k_1+2})I_{F_{k_1-1}}$.

2) For k_1 even, the intervals of length F_{k-k_1+1} are located in the schedule fragments $I_1(F_{k-k_1+1})I_{F_{k_1-1}+1}$, $I_2(F_{k-k_1+1})I_{F_{k_1-1}+2}, \cdots, I_{F_{k_1-2}}(F_{k-k_1+1})I_{F_{k_1}}$.

3) For k_1 odd the intervals of length F_{k-k_1+2} are located in the schedule fragments $I_1(F_{k-k_1+2})I_{F_{k_1-2}+1}$, $I_2(F_{k-k_1+2})I_{F_{k_1-2}+2}, \cdots, I_{F_{k_1-2}}(F_{k-k_1+2})I_{F_{k_1}}$.

4) For k_1 odd the intervals of length F_{k-k_1+1} are located in the schedule fragments $I_{F_{k_1-1}+1}(F_{k-k_1+1})I_1$, $I_{F_{k_1-1}+2}(F_{k-k_1+1})I_2, \cdots, I_{F_{k_1}}(F_{k-k_1+2})I_{F_{k_1-2}}$.

Proof: By induction. For $k_1 = 2, 3, 4, 5$ this can be verified by writing the sequence out as done above. Assume that these properties hold for an odd value of $k_1 \geq 5$. We shall show that properties 1 and 2 hold for $k_1 + 1$. (An identical argument shows properties 3 and 4 to hold for $k_1 + 1$ when k_1 is even).

By Lemmas 1 and 2, $I_{F_{k_1+i}}$, $1 \leq i \leq F_{k_1-1}$ will fall F_{k-k_1+1} slots in the clockwise direction with respect to I_i, splitting intervals of length F_{k-k_1+2} into two intervals, one of length F_{k-k_1} and the other of length F_{k-k_1+1} in the following manner: $I_i(F_{k-k_1})I_{F_{k_1+i}}(F_{k-k_1+1})I_{F_{k_1-2}+i}$, $1 \leq i \leq F_{k_1-1}$. In addition, the remaining intervals of length F_{k-k_1+1} were created by the allocation of the first F_{k_1} slots. The manner in which intervals of length F_{k-k_1+1} and F_{k-k_1} occur satisfies properties 1 and 2 for $k_1 + 1$. □

Proof of Theorem 1: We use the notation $A \preccurlyeq B$ to indicate that slot B lies in a clockwise direction from slot A. Consider an interslot interval of length F_{k-k_1+1} between slots I_L and I_R, $1 \leq L$, $R \leq F_{k_1}$, $L \neq R$ on schedule I. Two cases may arise depending on the relative position of schedule I and schedule O.

Case 1: There lies a slot O_m, assigned to the session under schedule O, between the two, i.e., $I_l \preccurlyeq O_m \preccurlyeq I_R$, $1 \leq m \leq F_{k_1}$.

Case 2: There lie two slots $O_{L'}$ and $O_{R'}$, assigned to the session under schedule O, on either side of I_L and I_R, respectively, such that $O_{L'} \preccurlyeq I_L \preccurlyeq I_R \preccurlyeq O_{R'}$. This occurs when an interslot distance of F_{k-k_1+2}, under schedule O overlaps both I_L and I_R, $k \geq k_1 + 3$.

We will now define a procedure of associating, or pairing off, each slot from schedule I with a neighboring slot from schedule O. We begin by associating O_m or $O_{L'}$ or $O_{R'}$ with I_L or I_R. A "-" connects the associated pair.

Case 1:

Subcase 1: $m \geq L, R$.

k_1 odd: $O_m - I_L$.

k_1 even: $O_m - I_L$.

Subcase 2: $m \leq L, R$.

k_1 odd: $O_m - I_R$.

k_1 even: $O_m - I_L$.

Subcase 3: $L \leq m \leq R$.

k_1 odd: Cannot occur as $L \geq R$ by property 4 of Lemma 3.

k_1 even: $O_m - I_R$.

Subcase 4: $R \leq m \leq L$.

k_1 odd: $O_m - I_R$.

k_1 even: Cannot occur as $R \geq L$ by property 2 of Lemma 3.

Case 2: Lemma 3 can be used to show that only two possibilities exist.

Subcase 1: $L < L'$, $R > R'$

k_1 even: $O_{L'} - I_L$.

Subcase 2: $L > L'$, $R < R'$

k_1 odd: $O_{L'} - I_L$.

As an example consider what happens when O_m is paired off with I_R (Case 1, Subcase 2, k_1 odd). O_{m+1} can then be paired off with I_{R+1} since both are F_{k-1} slots away from O_m and I_R and therefore have the same relative position. We can continue this pairing off until we reach $I_{F_{k_1}}$, which is paired with $O_{m+F_{k_1}-R}$. At this point we next pair off I_1 with $O_{m+F_{k_1}-R+1}$. But I_1, by Lemma 2, is $F_{k-1} - F_{k-k_1}$ slots away from $I_{F_{k_1}}$. Therefore I_1 shifts by F_{k-k_1} slots in the anticlockwise direction in relation to $O_{m+F_{k_1}-R+1}$ as compared to the previously paired off slots. This will still ensure that I_1 and $O_{m+F_{k_1}-R+1}$ are still less than F_{k-k_1+1} slots away from each other. We can now complete this process by pairing off I_j with $O_{m+F_{k_1}-R+j}$ for $j \leq R - m$. $O_1, O_2, \cdots, O_{m-1}$ may then be paired off with $I_{R-m+1}, \cdots I_{R-1}$. In all subcases, the pairing off is done in such a way that the two slots paired off are neighbors. In all subcases, except for two, this is done by making sure that the two slots that are paired off are within F_{k-k_1+1} slots of each other. The two subcases where the paired off slots may be more than F_{k-k_1+1} slots away from each other, and yet can be shown to be neighbors, are as follows:

1) *Case 1, Subcase 3, k_1 even:*

We start the pairing off process with O_m and I_R. Continuing we have $O_{m+1} - I_{R+1}, \cdots, O_{m+F_{k_1}-R} - I_{F_{k_1}}$. Also we can pair off $O_1 - I_{R-m+1}, O_2 - I_{R-m+2} \cdots O_{m-1} - I_{R-1}$. Next, we continue with $O_{m+F_{k_1}-R+1}I_1$. This widens the distance between the two associated slots by F_{k-k_1} slots, and therefore the distance between the two can be at most $F_{k-k_1+2} = F_{k-k_1+1} + F_{k-k_1}$ slots. Also, I_1 will be in the clockwise direction relative to $O_{m+F_{k_1}-R+1}$. But since the interslot distances in the anticlockwise direction from $I_1, I_2, \cdots I_{F_{k_1-1}}$ is F_{k-k_1+2}, the paired off slots will continue to be neighbors. This will take care of all the remaining slots as long as $R - m \leq F_{k_1-1}$ or $m + F_{k_1} - R + F_{k_1-1} \geq F_{k_1}$, i.e., $R - m \leq F_{k_1-1}$. Since $L \leq m \leq R$ and since $R - L = F_{k_1-1}$ by property 2 of Lemma 3, this is always true.

2) *Case 1, Subcase 4, k_1 odd:*

Once again, a one-to-one pairing off between neighbors in schedule I and O can be constructed as in the previous example.

Define the discrepancy d between the two schedules as follows. Traverse the circle in a clockwise direction, starting from any slot. Initialize the discrepancy to 0. Each time a slot from schedule I is encountered, increment it by 1. Each time a slot from schedule O is encountered, decrement it by 1. If an I slot and O slot coincide, do not modify the discrepancy. At the end of one cycle we must have $d = 0$ (the two schedules contain an identical number of slots). Let the maximum value of the discrepancy over one cycle over all possible starting slots and over all possible relative shifts of schedules I and O be $d_{\max}$. Then the maximum number of buffers required will be $d_{\max}$. Now at the end of the cycle, d can be expressed as a running sum, i.e., $d = \{+1 - 1 + 1 - 1 - 1 + 1\} = 0$. By the pairing off process described earlier it is clear we can express d in the following form: $d = \{(+1 - 1) + (-1 + 1) + \cdots)\}$ or $d = \{+1) + (+1 - 1) + \ldots (-1)\}$. Note that each bracketed pair consists of a $+1$ and a -1. It is easily seen that the discrepancy at any point cannot exceed 2, i.e., $d_{\max} \leq 2$. It follows that at most two buffers are required.

To see that the bound is tight, consider any subcase of Case 2). As there are two slots on schedule I between consecutive slots on schedule O, two packets may arrive before one of them can be transferred to the transmission buffer for transmission. This necessitates at least two buffers. □

ACKNOWLEDGMENT

The statement and proof of Theorem 2 are due to E. Hahne. We gratefully acknowledge her contribution to this work. We are also greatly indebted to the referees: their unstinting efforts have significantly improved the presentation of this paper.

REFERENCES

[1] J. Bauwens and M. De Prycker, "Broadband experiment using asynchronous time division techniques" *Elect. Commun.* vol. 61, no. 1, 1987.
[2] M-S. Chen, B. Kadaba, and G. Grover, "Efficient hop by hop buffer class flow control schemes," in *Proc. GLOBECOM 87,* Tokyo, Japan, 1987.
[3] E. M. Gafni, and D. P. Bertsekas, "Dynamic control of session input rates in communication networks," *IEEE Trans. Automat. Contr.,* vol. AC-29, pp. 1009–1016, 1984.
[4] M. Gerla and L. Kleinrock, "Flow control: A comparative survey," *IEEE Trans. Commun.,* vol. COM-28, pp. 553–574, Apr. 1980.
[5] E. L. Hahne, "Round robin scheduling for fair flow control in data communication networks," Ph.D. dissertation, M.I.T, Cambridge, MA, 1986.
[6] G. Hebuterne, "STD switching in an ATD environment," in *Proc. INFOCOM 88,* pp. 449–458.
[7] A. Hofri and Z. Rosberg, "Packet delay under the golden ratio weighted TDM policy in a multiple-access channel," *IEEE Trans. Inform. Theory,* vol. IT-33, pp. 341–349, 1987.
[8] A. Itai and Z. Rosberg, "A golden ratio control policy for a multiple access channel," *IEEE Trans. Automat. Contr.,* vol. AC-29, pp. 712–718, 1984.
[9] D. E. Knuth, *The Art of Computer Programming, vol. 1.* Reading, MA: Addison-Wesley, 1973, p. 85.
[10] ——, *The Art of Computer Programming, Vol. 3.* Reading, MA: Addison-Wesley, 1973. pp. 506–544.
[11] H. Le Bris and M. Servel, "Integrated wideband networks using asynchronous time division techniques," in *Proc. ICC 1986,* pp. 1720–1724.
[12] N. F. Maxemchuk, and M. El Zarki, "Routing and flow control in high speed wide area networks," in *Proc. IEEE,* vol. 78, pp. 204–221, Jan. 1990.
[13] U. Mukherji, "A periodic scheduling problem in flow control for data communication networks," *IEEE Trans. Inform. Theory,* vol. 35, pp. 436–439, 1989.
[14] T. K. Philips and S. S. Panwar, "Approximating the mean waiting time under a TDMA schedule," in *Proc. 28th Allerton Conf. Commun. Conf.. Comput.,* 1990.
[15] Z. Rosberg and M. Sidi, "TDM policies in multi-station packet radio networks," IBM Res. Rep. RC 12781, May 1987.
[16] V. T. Sos, "On the distribution mod 1 of the sequence $n\alpha$," *Ann. Univ. Sci. Budapest Eotvos Sect. Math.,* vol. 1, pp. 127–134, 1958.
[17] S. Swierczkowski, "On successive settings of an arc on the circumference of a circle," *Fund. Math.,* vol. 46, pp. 187–189, 1958.
[18] J. Turner, "New directions in communications (or which way to the information age?)," *IEEE Commun. Mag.,* vol. 24, pp. 8–15, 1986.

Shivendra S. Panwar (S'82–M'85) was born in Delhi, India, on December 15, 1959. He received the B. Tech. degree in electrical engineering from the Indian Institute of Technology, Kanpur, in 1981, and the M.S. and Ph.D. degrees in electrical and computer engineering from the University of Massachusetts, Amherst, in 1983 and 1986, respectively.

From 1981 to 1985 he was a Research Assistant at the University of Massachusetts. He joined the Department of Electrical Engineering at the Polytechnic Institute of New York, Brooklyn (now Polytechnic University), where he is currently an Associate Professor. He spent the summer of 1987 as a Visiting Scientist at the IBM T. J. Watson Research Center, Yorktown Heights, NY, and has been a Consultant to AT&T Bell Laboratories, Holmdel, NJ. His research interests include the analysis and design of integrated networks, local area networks and multiaccess channels.

Dr. Panwar is a member of Tau Beta Pi and Sigma Xi. He is currently a member of the IEEE Communications Society Technical Committee on Computer Communications.

Thomas K. Philips received the B.E. degree from Benares Hindu University, Benares, India, and the M.S. and Ph.D. degrees from the University of Massachussets at Amherst in 1980, 1983 and 1986 respectively.

From 1985 to 1990 he was a member of the Research Staff at the IBM T.J. Watson Research Center, Yorktown Heights, NY, working on Graph Theory, Queuing Theory, and the Design of Algorithms.

He is currently on assignment at the IBM Retirement Fund, Stamford, CT.

Mon-Song Chen was born in Taipei, Taiwan, in 1956. He received the B.S. degree in electrical engineering from National Taiwan University, Taipei, Taiwan, in 1978, the M.S. degree in electrical engineering from University of Washington, Seattle, WA, in 1982, and the Ph.D. degree in system engineering from Polytechnic Institute of New York, Brooklyn, NY, in 1985.

He joined IBM T. J. Watson Research Center, Yorktown Heights, NY as a Research Staff Member in 1985. He worked on networking architecture design and analysis, protocol verification and conformance testing, and WDMA signaling protocols. He is currently the Manager of the High Bandwidth Applications group and working on multimedia teleconferencing systems and other bandwidth intensive applications.

Dr. Chen is a member of the IEEE Communications and Computer Societies.