Distributed Admission Control for Real-Time Communication over Wormhole-Routed Networks

Sharad Sundaresan Riccardo Bettati
Department of Computer Science
H.R. Bright Building
Texas A&M University
College Station, TX 77843-3112
tel: (409) 845-5469
fax: (409) 847-8578
{sharad,bettati}@cs.tamu.edu

Technical Report 96-015

July 10, 1996

Abstract

Wormhole networks provide a very-high-speed communication medium that is well suited for a large number of bandwidth demanding applications. Unfortunately, the lack of buffering in the switches causes blocked packets to transiently monopolize arbitrary large portions of the network, thus making it difficult to give real-time guarantees when connections contend for communication links. We present an efficient, fully distributed admission control scheme for time-critical connections over wormhole networks. We compare this scheme to a centralized algorithm in a suite of simulation experiments and show that its performance is comparable, in particular for highly loaded networks.

1 Introduction

The increased development and use of networked real-time applications in distributed multimedia, remote laboratoring, and distributed virtual reality has generated a large amount of interest in the development of real-time communication protocols to support such applications. Such protocols provide real-time guarantees for message deliveries to the network clients, typically in terms of bounded delay and jitter. This in turn makes it possible for the applications to meet their end-to-end timing requirements. In order to provide such guarantees on message delivery, resources in the network must be appropriately allocated. An admission control component of the protocol must ensure that enough resources are available in the system that a new connection can be in fact be guaranteed the required performance. Such admission control protocols have been extensively studied for packet-switched networks [4, 8, 11, 10].

Due to the increasing bandwidth requirements in a large number of applications (such as incorporation of uncompressed video streams in control loops for remote laboratoring), a number of very-high speed networking technologies are currently under investigation for their applicability to support hard-real-time communication, such as ATM [2], FDDI [1], HIPPI [3]. We are investigating the applicability of wormhole-routed network technologies for this class of applications.

As exemplified by recently developed technology, wormhole-routed networks are a promising approach for high-bandwidth, low-latency communication for small and medium-sized networks. For example, Myrinet switches [6, 7] have a link bandwidth of 640Mb/s with a worst-case routing latency of 550ns for a 8×8 switch. In the following discussion, we will be using Myrinet as a reference for the system model.

A well known problem with wormhole-routed packet networks is the potentially large amount of blocking that packets can experience because of contention for links. Because of the very limited amount of buffering in such networks, blocked packets "remain" in the network and keep using network resources. Thus, they may in turn cause other packets to be blocked. This may affect a large number of packets over a large portion of the network. Proper connection management strategies and connection protocols must be devised that ensure that blocking of packets due to link contention is bounded. Zhao *et. al* [5] have developed a transmission-control scheme that regulates the rate of connections at the entrance to the network. A worst-case achievable utilization of 50% could be proved, allowing for a simple admission control scheme: new connections are accepted as long as the utilization of the network does not exceed the worst-case achievable utilization.

In this paper we propose a distributed admission control scheme with the following two objectives:

1. Exploit the concurrency available in the network. Not always do connections conflict for links. With appropriate routing, network utilizations that go far beyond the bounds described in [5] can be achieved.

2. The admission control scheme should be distributed and affect only the switches that are on the route of the connection. Node failures and network partitions then affect the admission process only if they happen along the route of the connection.

The paper is organized as follows. In Section 2 we describe our model of packet transmission over wormhole-routed networks and derive simple worst-case bounds for the time packets can be blocked during transmission. We also describe the basic traffic model used in the protocol. We describe how the admission of a new connection affects existing connections. From this we derive a general admission control scheme. In Section 3 we describe our distributed admission control scheme based on residual time. Section 4 describes simulation results that illustrate that our scheme compares

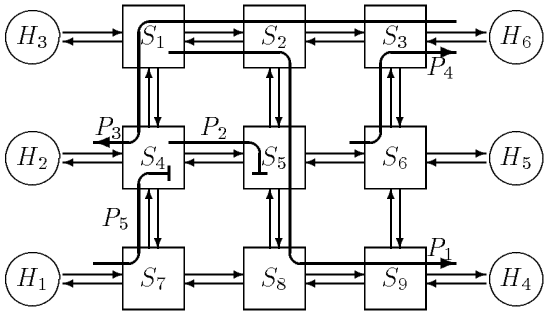


Figure 1: Snapshot of a Network with Packets in Transit

favorable against centralized schemes. We will conclude and give an outlook on future research in Section 5.

2 Real Time Communication over Wormhole Routed Networks

In the following we assume a general-topology network consisting of *hosts* and *switches* and unidirectional *links*. The links forward arbitrary-length packets, which are routed through the switches in a cut-through manner. If the outgoing link is not being used by another packet, the head of the packet can be forwarded to that link as soon as the routing information in the packet header is decoded.

In the eventuality that the output link is in use, the packet is *blocked* until the link becomes available. (If a packet is using a link that is blocked by another packet, we say that the packet is *directly blocked.*)

We assume the switches to be *perfect*, meaning that packets do not conflict at a node unless they are directed to the same outgoing link. There may be as many packets simultaneously traversing the switch as there are outgoing links.

Figure 1 shows a snapshot of a wormhole network with a number of packets in traffic. In this example, the network consists of six hosts $H_1, \ldots, H_6$, which are connected by nine 4×4 switches $S_1, \ldots, S_9$, forming a mesh. In this example, Packet P_2 is being blocked on the link between S_5 and S_8 by Packet P_1. On the other side, although P_3 and P_2 use the same Switch S_4, they do not affect each other. This example illustrates also how a packet can be *indirectly blocked* by another packet: although P_1 and P_5 do not directly contend for a link, P_1 indirectly blocks P_5 by blocking P_2, which in turn blocks P_5 on the link from S_4 to S_5.

Once the head of the packet reaches the destination, the remainder can be sent at full link bandwidth (of 640 Mb/s in the case of Myrinet). In order to bound packet delivery delay due to blocking during path setup, however, proper connection acceptance and connection management strategies must be devised. In the following, we assume connections that have a periodic traffic specification. Each connection is of type $C_i = (R_i, e_i, p_i)$, where R_i specifies the route followed by packets of C_i, e_i defines the worst-case message length (in time units), and p_i specifies the worst-case interarrival time of packets of C_i.

Assuming a fair service policy in the switches, we can determine an upper bound $B_{i,j}$ for the worst-case blocking time experienced by a packet of connection C_i at a single switch S_j because

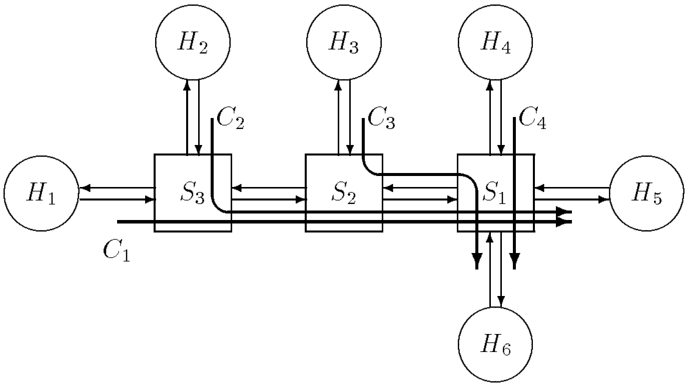


Figure 2: Example of Four Connection on Three Switches.

of contention for the outgoing link at S_j. In the worst case, the switch has to serve a packet of a connection that shares the output link with C_i for each input link other than the one used by C_i. Several connections may share a input/output link pair, in which case the packet of C_i has to wait for the longest of all such packets to be served. More formally, the worst-case blocking time $B_{i,j}$ in a network with one switch S_j is defined as follows:

$$B_{i,j} = \sum_{\substack{k \in L^I(j) \\ k \neq l^I(i,j)}} \max_{c \in \{\Gamma(k) \cap \Gamma(l^O(i,j))\}} (e_c) \tag{1}$$

where $L^I(j)$ and $L^O(j)$ denote the set of all input and output links to (from) S_j, $l^I(i,j)$ and $l^O(i,j)$ the input and output link used by connection C_i at switch S_j, and $\Gamma(l)$ the set of connections using link l, respectively.

In a network with multiple switches, a connection C_i with a route $R_i = (S_n, S_{n-1}, \ldots, S_2, S_1)$ has a worst-case blocking time at switch S_j of

$$B_{i,j} = \sum_{\substack{k \in L^I(j) \\ k \neq l^I(i,j)}} \max_{c \in \Gamma(k) \cap \Gamma(l^O(i,j))} \{B_{c,j-1} + e_c\} + B_{i,j-1} \tag{2}$$

Thus, the total blocking time experienced by a packet of C_i along its path is the time for which it is blocked at the source, since this is a cumulative sum of all the individual blocking times along the entire path.

The following example illustrates how worst-case blocking times are determined. Figure 2 shows a simple network with three switches and four connections. Our goal is to determine the worst case blocking time of connections C_3 and C_1 at their respective entries into the network (i.e. $B_{3,2}$ and $B_{1,3}$, respectively). For $B_{3,2}$ we have:

$$\begin{aligned} B_{3,2} &= \max(B_{1,1} + e_1, B_{2,1} + e_2) + B_{3,1} \\ &= \max(e_1, e_2) + B_{4,1} + e_4 \\ &= \max(e_1, e_2) + e_4 \quad . \end{aligned}$$

$B_{1,3}$ is determined as follows:

$$B_{1,3} \quad = \quad B_{2,2} + e_2 + B_{1,2}$$

$$
\begin{aligned}
&= B_{3,1} + e_3 + e_2 + B_{3,1} + e_3 \\
&= 2(B_{4,1} + e_4) + 2e_3 + e_2 \\
&= 2e_4 + 2e_3 + e_2 \quad .
\end{aligned}
$$

Equation 1 and Equation 2 allow for a very simple (though expensive!) centralized admission control scheme. With our simple periodic traffic model we can use the following formula to test whether packets of connection C_i can successfully be delivered before the end of the period:

$$\forall i : B_{i,X_i} + e_i \leq p_i \tag{3}$$

that is, whether for all connections, the worst-case blocking time at their entry X_i to the network plus the transmission time does not exceed the period.

For a new connection to be accepted, we centrally determine whether (1) the timing requirements of the new connection can be satisfied and whether (2) the timing requirements of existing connections are not violated. Since the admission of a new connection can potentially affect all previously established connections, the cost of this operation grows quadratically with the number of accepted connections.

In the following section, we describe a fully distributed admission control scheme that has a time overhead that grows linearly with the diameter of the network.

3 Residual Time Admission Control Algorithm

The optimality of a centralized scheme is the result of having access to the *global state* of the network at any given time. This allows the overall effect of allowing a new connection into the network to be determined *a priori* and a correct decision about the acceptance/rejection of this connection can be made. In order to map the optimality of such a centralized scheme to a distributed environment, each node in the network should be able to maintain a sufficient and pertinent information base of relevant connections. Relevant connections are either those which pass through that node or have dependencies with those passing through that node. The algorithm should then use and update this information in an appropriate manner so as to make an intelligent decision about resource allocation for incoming connections and also future connections.

The *Residual Time Algorithm* described in this section is one such distributed algorithm. Connection establishment decisions are made locally at each node in the connection's path. Only if each node accepts the connection, is it admitted into the network. As the name implies, this algorithm's basis for admission control is allocating a certain amount of residual time at each node in the network. This residual time, called the *Local Residual Time* (LRT), is a measure of the time any connection can be blocked at that particular node in addition to what it is previously blocked. This concept is elaborated in the forthcoming subsections.

3.1 Local Residual Time (LRT)

We consider our network workload model to be one where we accrue the *worst case* blocking time (as detailed earlier) for a connection. For any connection, we assume that it is blocked by *all other* possible connections at *every* node in its path. Assuming a deadlock-free collection of routes , each connection is thus blocked for a finite time in the network due to its dependencies (common paths) with other established connections. In accordance with our worst case scenario, we calculate this total blocking time (TBT_i) to be the worst case blocking time B_i, x_i at the entrance to the network. If connection C_i has a period p_i, that connection can be blocked by an additional $p_i - B_i, x_i - e_i$, before missing its deadline. We therefore call this the *Residual Time* (RT_i) left to C_i. This residual time can now be distributed in a fair manner to all the nodes in the path. This share of RT_i at a switch S_j, is the *local residual time*, LRT_{ij} for S_j. In addition, we define MAXBT to be the maximum time any connection can be blocked on a particular outgoing link and still meet its deadline. We note that $(MAXBT - LRT)$ defines the maximum amount of time a connection can be blocked at the next switch without missing its deadline. While performing the admission control checks on any future connections, if we constrain them so that the total blocking caused by the new connection upto switch S_j is within the MAXBT allocated to the outgoing link for that connection at that switch, we can safely restrict the cumulative increase in blocking time to ensure that the deadlines are met.

Having thus provided a broad overview of the Residual Time Algorithm, the next section explains the actual implementation along with a detailed explanation of the data maintained at each node, the checks performed during connection establishment and finally any updating of information that may necessary to maintain the correctness and consistency of the network and the set of established connections.

3.2 Description and Implementation

Any connection seeking establishment would initiate a *Connection Establishment* procedure at its source host. Although we limit ourselves to a source-initiated, source-routed scenario, this scheme can be adapted to any destination-initiated or adaptive routing scheme. The current version of the algorithm is implemented as a three-pass protocol. During the first pass, (Phase I of the algorithm), an *establishment request* message travels from the source of the connection along the path. In this pass, the local admission control components at the switches traversed by the request message are initialized appropriately. When the establishment request message reaches the destination, the second pass (Phase II of the algorithm) is invoked with an *establishment reply* message that travels from the destination back to the source along the route traveled in the first pass. During Phase II, the blocking time experienced by the new connection is calculated and the effect of the new connection on the residual times of existing connections is determined. If a switch along the path does not have enough local residual time to compensate for the additional blocking time caused by the new connection, the connection is rejected by masking the establishment reply message as a *reject* message. When the *establishment reply* message reaches the sender and this message is not masked as a *reject*, the test described in Equation 3 is performed. The third pass (Phase III), is initiated in which an *establishment-confirm* message is sent to the destination informing all the switches along the route whether the connection is accepted or rejected. The admission-control components in the switches update their state accordingly. In order to ensure serialization of establishment procedures, the admission-control components enter an *locked* mode upon receiving an *establishment-request* message, and continue processing subsequent request messages only after a *establishment-confirm* message marks the end of the establishment procedure. Appropriate mechanisms must be put in place to avoid deadlocks.

In the following, we describe the three phases of the algorithm in detail.

3.2.1 Phase I

This phase is initiated at the source switch of the incoming connection. Proceeding from the source to the destination, each switch is tagged as described below. In the following description,we say that another connection *branches in* if it shares the outgoing link with C_i, but not the incoming link. If it shares the incoming link with C_i but not the outgoing link, we say that the connection *branches out.*

- **BRIN** - Branch-In switch. At such a switch, there are connections that branch in and no connections that branch out.
- **BROUT** - Branch-Out switch. At such a switch, there are connections that branch out and no connections that branch in. output ports with other connections.
- **BRINOUT** - Branch-In-and-Out switch. At this switch, there are connections that branch in and connections that branch out.
- **NEITHER** - No connections branch in or branch out at this switch. In other words, connections either don't share incoming or outgoing links or share both with C_i.

The rationale behind this tagging is to estimate the worst case blocking time during Phase II.

3.2.2 Phase II

At the end of Phase I, the *establishment-request* message is at the destination switch. In Phase II, the worst case blocking time is calculated as the establishment reply message travels back to the sender, based on the type of the switches tagged in Phase I.

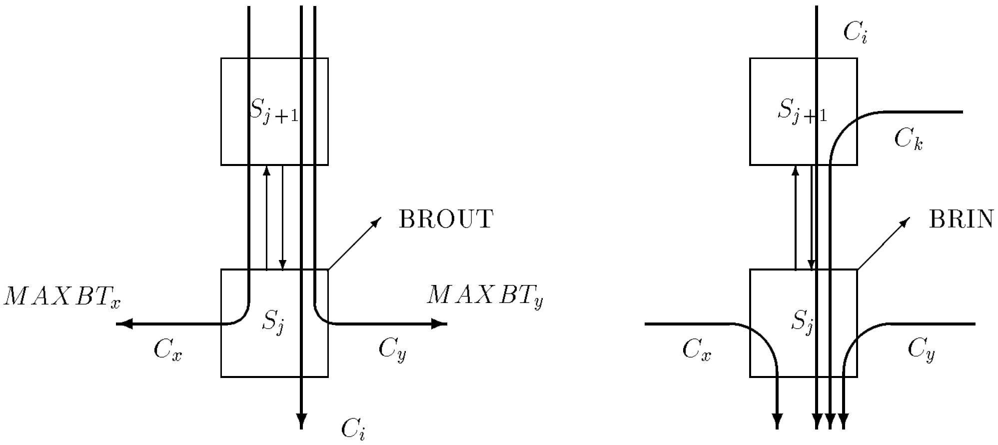


Figure 3: Connections at a BROUT and BRIN switch.

If the switch is a BROUT switch, the MAXBT values for the connections that are branching out are added to the blocking time of the new connection C_i. This is to estimate the maximum time C_i may be blocked at that particular switch. At the BROUT switch S_j, shown in Figure 3, C_i may be blocked at most by $MAXBT_x + MAXBT_y$. (The $MAXBTs$ were previously allocated to the respective outgoing links while establishing connections C_x and C_y. Allocation of $MAXBTs$ is explained in Phase III.) If we impose the restriction that C_x and C_y cannot be blocked for more than $MAXBT_x$ and $MAXBT_y$ respectively *beyond* switch S_j, then C_i will have a worst case blocking time of $B_{i,j} = B_{i,j-1} + MAXBT_x + MAXBT_y$ *at* switch S_j. This similarly holds for any switch of type BROUT.

Similarly, if the switch is a BRIN switch, we increment the blocking time by the sum of all the longest execution times of branching in connections at each incoming link, multiplied by one greater than the number of connections that share both the incoming and outgoing links with C_i at this switch. For the BRIN switch S_j shown in Figure 3, in the worst case, C_k has to wait for C_x and C_y, i.e be blocked for a time of $(e_x + e_y)$. C_i then has to wait for C_k at some *prior* switch (in this example at switch S_{j+1}) and for C_x and C_y at *this* switch, thus being blocked for a total of $2 * (e_x + e_y)$, as a result of sharing an outgoing link with C_x and C_y at switch S_j. If along with C_k there was another connection C_l, the blocking time would increase by another $(e_x + e_y)$.

For a BRINOUT switch, both the above procedures are performed.

Once the blocking time at switch S_j has been calculated, this is included in the payload of the Request packet, which is forwarded to the next switch S_{j+1}. At switch S_{j+1}, the blocking time accrued thus far is compared with the information at that switch to check whether the connection can be accommodated by that switch without disturbing existing connections. If any of the following checks fail, then the *establishment request* message is masked as a *reject* message and sent to the source host of the connection.

- Check 1: The accumulated blocking time $B_{i,j}$, should be less than MAXBT for the outgoing link.
- Check 2: MAXBT for the outgoing link should not be NULL.
- Check 3: The execution time e_i of the connection C_i should be less than MAXBT for the outgoing link.
- Check 4: The LRT should not be NULL.

These checks may need some additional explanation. If either MAXBT or LRT is NULL, then it indicates that the existing connections cannot tolerate any more delay, i.e they do not have any residual time to spare. If C_i's blocking time $B_{i,j}$ accumulated upto switch S_{j+1} is greater than MAXBT, then this will violate our policy of limiting the maximum blocking time suffered by other connections upto this switch. Since these values have been fixed prior to this connection, C_i has to adhere to these requirements. If the connection clears these checks then the blocking time is incremented as described in the earlier step.

In addition, if the switch is tagged as a BROUT or BRINOUT, then we include the MAXBTs from the outgoing links of the connections branching out, as BROUT_MAXBT. Since, we do not update any values along these links, we have to assume that in the worst case, we may be blocked by that much time by connections using those output links. This is again a cumulative value which is incremented at every BROUT or BRINOUT switch. This value is used in Phase III of the algorithm.

Finally, if the *Request* packet reaches the source without being rejected, then at the source, the blocking time is incremented by the sum of the execution times of those connections having that same host as their source. (This is similar to the increment at a BRIN switch.) The source host then checks to see if the sum of the total blocking time (TBT) and the execution time, e_i is less than the period p_i of C_i. If it is so, then this connection can be accepted, and Phase III is initiated.

3.2.3 Phase III

During this phase, the sender sends an *establishment-confirm* message, which can be either an *accept* or a *reject* message, to the receiver. A *reject* message is sent out if during Phase II, the connection has been rejected either at a switch or at the sender host. Upon receiving a *reject* message, a switch simply unlocks the admission control component, which now is ready to process the next *establishment-request* message. The *reject* message is then forwarded to the next switch on the route. If the connection was accepted at the end of Phase II, then the information at every switch (specifically the values for MAXBT and LRT on all the links along the route) must be updated to incorporate the new connection. An *accept* message is relayed to all the switches with the updating information.

- First, the initial values are calculated for the MAXBT at the entrance to the network and the local residual time that is allocated on each link along the route. The maximum blocking time that C_i can experience at the entrance to the network is equal to the available slack, therefore:
  $newMAXBT = p_i - e_i$
  The total residual time for this connection is equally distributed among the links on the route of C_i. The LRT's then become:
  $newLRT = (D_i - ei - B_i, x_i)/(N_i + 1)$
  where X_i denotes the entry switch to the network and N_i is the number of hops on the route for C_i.

  Finally, the host sends an *accept* message with the values (new MAXBT - new LRT, new LRT) to the first switch on C_i's route. Upon receiving an *accept* message with the values (MAXBT', LRT'), the switch performs the following actions, depending on how it has been tagged during Phase I of the algorithm.

- If the switch is a BRIN, then MAXBT and LRT for the outgoing link are recalculated to reflect the addition of this new connection. For the recalculation, the BROUT_MAXBT value (deposited during Phase II) is used.

$$\begin{aligned} MAXBT &:= min(MAXBT, MAXBT') \\ LRT &:= min((MAXBT - BROUT\_MAXBT)/N, LRT') \end{aligned}$$

  where N denotes the number of hops for C_i from the current switch to the destination. The recalculation of the local residual time for the remaining portion of the route is necessary because of the optimistic assumption made earlier at the source that the total residual time can be equally distributed over all links of the connection. This is not always true because the MAXBT value at an outgoing link maybe less than the value in the *accept* message. This is then the bounding value for all connections using that outgoing link, including C_i. The LRTs for the remaining switches are now calculated with this value after decrementing it with the BROUT_MAXBT value. (Since this is the sum of the MAXBTs at output links which cannot be remodified, we have to assume the worst case blocking due to connections using those links.)

- If the switch is a BROUT, then the MAXBT has to be *decremented* by an amount which has been allocated as the MAXBT to the outgoing links of the connections branching out. This prevents those connections from getting disturbed due to this new connection. So, the tuple values for this switch are,

$$\begin{aligned} LRT &= min(LRT, LRT') \\ MAXBT &= min(MAXBT, MAXBT' - \sum_{\substack{k \in outgoing\ links\ carrying \\ outbranching\ connections}} MAXBT_k) \end{aligned}$$

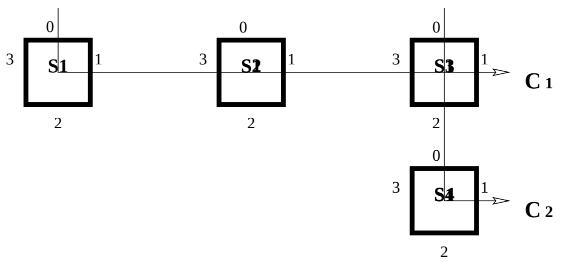
- At a BRINOUT switch, the procedure is a combination of that at a BROUT switch and a BRIN switch, in that order.
- At a NEITHER switch, the values are copied without any modifications.

After the values for MAXBT and LRT for the outgoing link have been updated at the switch, the *accept* message is forwarded to the next switch with the values $(MAXBT - LRT, LRT)$. Also, the switch unblocks the admission control component, which becomes free to process the next establishment request.

This is the basic implementation of the Residual Time algorithm. The entire algorithm is distributed and the only overhead is at connection-establishment time. In the next subsection, a brief example of the Residual Time algorithm is provided.

3.3 Residual Time Algorithm : Example

A small subset of a network is considered and each phase of the algorithm is detailed. The switches present in the subset have accepted connections earlier, and the existing values are assumed to have been calculated during previous invocations of the algorithm.



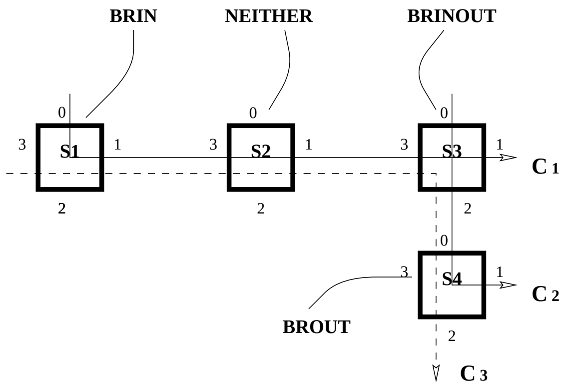
OP Port	Switch			
	S1	S2	S3	S4
0	-	-	-	-
1	(72, 24)	(48, 24)	(24, 24)	(32, 32)
2	-	-	(64, 32)	-
3	-	-	-	-

Figure 4: State of Network after C_1 and C_2

Figure 4 illustrates the state of the network after having accepted two connections - C_1 and C_2. Both C_1 and C_2 have an interarrival time of 100 and a message length of 4 time units (i.e $e_1 = e_2 = 4$ and $p_1 = p_2 = 100$). The values for MAXBT and LRT at each of the switches are given in the table alongside.

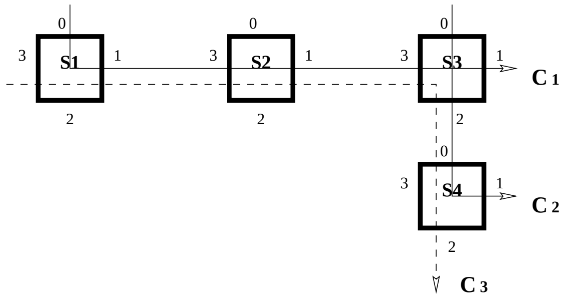
The Residual Time algorithm is applied for the establishment of connection C_3, with $p_3 = 100$ and $e_3 = 6$. During Phase I, the *establishment-request* message is sent from the source to the destination of C_3 and the switches in C_3's route are tagged. Figure 5 shows the path of C_3 and the type of switches, tagged after Phase I of the algorithm.

During Phase II, the *establishment reply* message is sent from the destination to the source of C_3. Along the route, the blocking time, B_3 experienced by C_3 at each switch is calculated. S_4 is tagged as BROUT, so, $B_{3,4} = 32$, which is the MAXBT value calculated for outgoing link 1 during the establishment of C_2. S_3 is tagged BRINOUT, so the blocking time is incremented by the MAXBT at outgoing link 1 (24, calculated during C_1's establishment) and the execution time e_2 (4) of C_2. So, $B_{3,3} = 32 + 24 + 4 = 60$. Since, S_2 is tagged NEITHER, there is no increment in the blocking time. At S_1, which is tagged BRIN, the blocking time is incremented by e_1. So, $B_{3,1} = B_{3,2} + e_1 = 60 + 4 = 64$. So, the final blocking time, B_{3,S_1} is 64. Figure 6 shows the



OP Port	Switch			
	S1	S2	S3	S4
0	-	-	-	-
1	(72, 24)	(48, 24)	(24, 24)	(32, 32)
2	-	-	(64, 32)	-
3	-	-	-	-

Figure 5: Introduction of Connection C_3 and Phase I



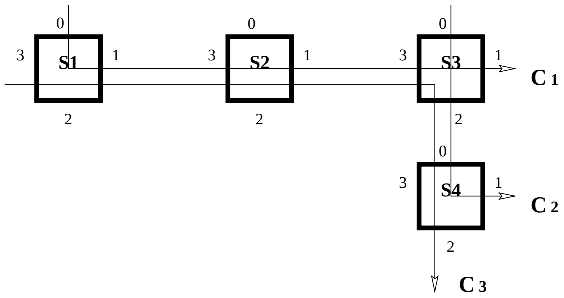
OP Port	Switch			
	S1	S2	S3	S4
0	-	-	-	-
1	(72, 24)	(48, 24)	(24, 24)	(32, 32)
2	-	-	(64, 32)	-
3	-	-	-	-

Value	Switch			
	S1	S2	S3	S4
BT	64	60	60	32
BROUT MAXBT	56	-	32	-

Figure 6: Conclusion of Phase II

blocking time of C_3 at each switch and also the BROUT_MAXBT values deposited at the BRIN and BRINOUT switches. This is at the end of Phase II of the Residual Time algorithm.

Figure 7 shows the network after C_3 has been accepted. Since, the sum of $B_{3,1}$ (64) and e_3 (6) is less than p_3 (100), C_3 can be accepted. At the source of C_3, the MAXBT and LRT are calculated to be $100 - 6 = 94$ and $100 - 6 - 64/5 = 6$ respectively. The MAXBT' and LRT' values in the *accept* message are thus $94 - 6 = 88$ and 6 respectively. At, S_1, there is a recalculation and the new values are $MAXBT = min(88, 72) = 72$, and $LRT = (72 - 56)/4 = 4$, where 56 is the BROUT_MAXBT value at S_1, deposited during Phase II. The new MAXBT' and LRT' values in the *accept* message are $72 - 4 = 68$ and 4. These are copied at S_2 and the MAXBT' and LRT' values changed to $68 - 4 = 64$ and 4. At S_3, which is tagged as BRINOUT, the new MAXBT value is first decremented by 24 (which is the MAXBT at outgoing link 1). So, MAXBT is now $\min(64 - 24, 64) = 40$. The LRT is recalculated similar to that at S_1. The new LRT $= \min((40 - 32)/2, 4) = 4$. The new MAXBT' and LRT' values in the *accept* message are $40 - 4 = 36$ and 4. At the BROUT switch S_4, the MAXBT $= 36 - 32 = 4$ (decremented in a similar manner as



OP Port	Switch			
	S1	S2	S3	S4
0	-	-	-	-
1	(72, 4)	(68, 4)	(24, 24)	(32, 32)
2	-	-	(40, 4)	(4, 4)
3	-	-	-	-

Figure 7: State of the Network after C_3 has been Accepted

switch S_3) and the LRT = 4. The table now shows the updated values at each of the switches, as a result of Phase III. We can see that there is a reduction in the value of LRT at the switches. This indicates that now the connections experience additional blocking due to the new connection, C_3.

4 Experiments

A suite of simulation experiments was made to determine the performance of the Residual-Time algorithm. For each of the experiments, we assumed a network topology and increased the load on the network by repeatedly generating random connections and trying to establish them. (For simplicity, we restricted ourselves to connections with identical traffic parameters.) As performance criteria, we measured the *acceptance ratio*, that is, the ratio of the number of accepted connections over the total number of generated connections. A related measure is the *average link utilization*, which shows how much the accepted connections utilize the existing links [1].

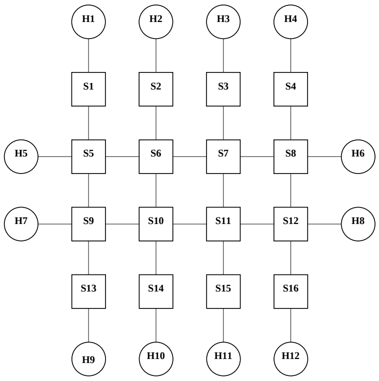
As a reference for comparison, we use the centralized admission control scheme described in Section 2. We recall that the centralized algorithm is an optimal admission control test, in that, given a set of connections, the test fails if and only if at least one of the connections can not meet its timing requirements. Both the Residual-Time and the centralized algorithm operate on-line, that is, they do not have knowledge about future connection arrivals.

In order to factor out routing effects, we assume a simple routing subsystem, which returns $x - y$ routes. This must be taken into consideration when analyzing the simulation results. Substantial performance increases can be achieved by using a load-sensitive routing algorithm. The 95% confidence intervals of all depicted results are below 1% absolute value.

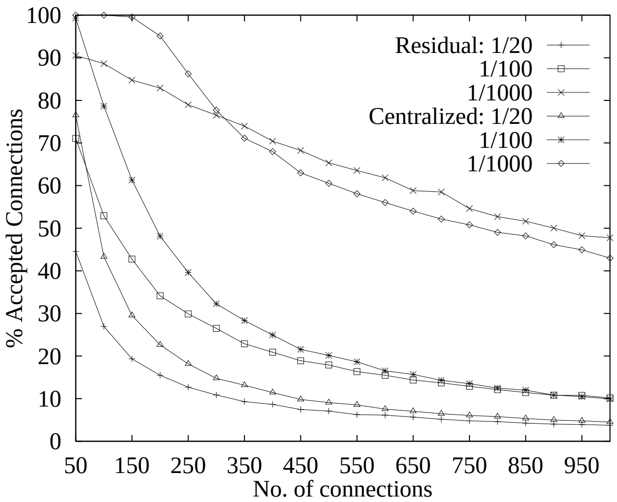
4.1 Results

Figure 8 shows the experiment results for a 4x4 Mesh topology with 16 4-port switches and 12 hosts. The results are for worst case message lengths of 1 time unit and varying interarrival times of 20, 100 and 1000 units. (The example of 1 time-unit packets every 1000 units, closely models a non-compressed CD-audio stream on a Myrinet network.) As to be expected, the acceptance ratio drops as the number of connections increase. It comes as no surprise that the high-bandwidth

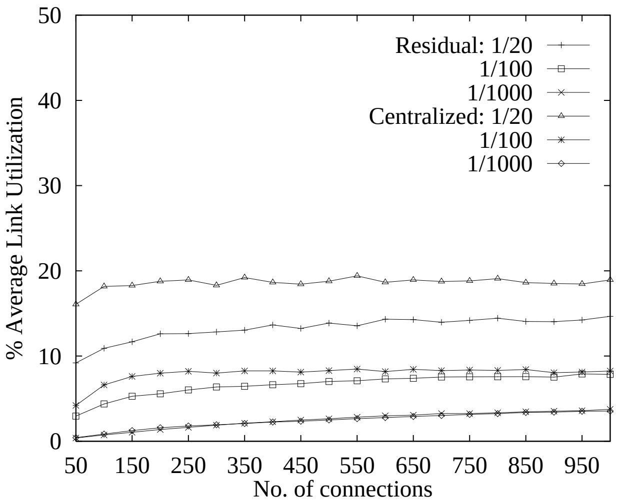
[1] An approximation of the latter measure can be easily derived from the acceptance ratio, but we add it for illustration purposes.



(a) 4x4 Mesh Topology



(b) Acceptance Ratio



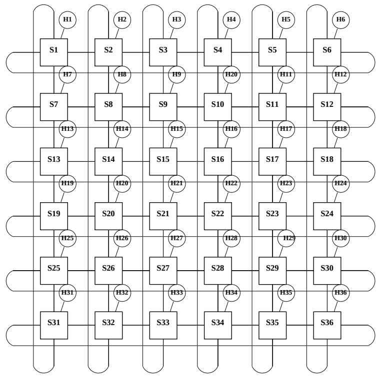
(c) Ave. Link Utilization

Figure 8: Experimental Results on a 4x4 Mesh

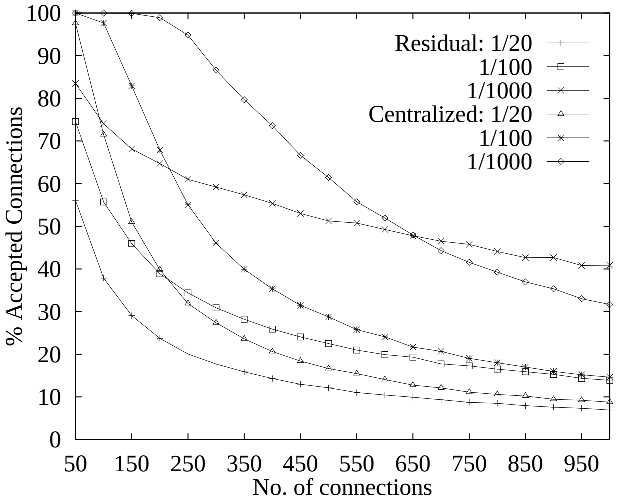
connections have lower acceptance ratios and link utilizations; this is because they generate higher blocking delays.

Although the centralized algorithm has a better acceptance ratio for light network loads, the Residual Time algorithm performs better for a heavily loaded network, where its performance equals that of the centralized algorithm. Especially for the case where the message length is 1 and the interarrival time is 1000, we see that the Residual Time algorithm has a better acceptance ratio from about 275 connections onward. It seems that the more conservative approach used in the Residual-Time algorithm retains network resources longer and so allows for better performance at high network loads. For example, the centralized algorithm may indiscriminately admit connections that traverse multiple links, thus causing a lot of blocking delay to other connections.

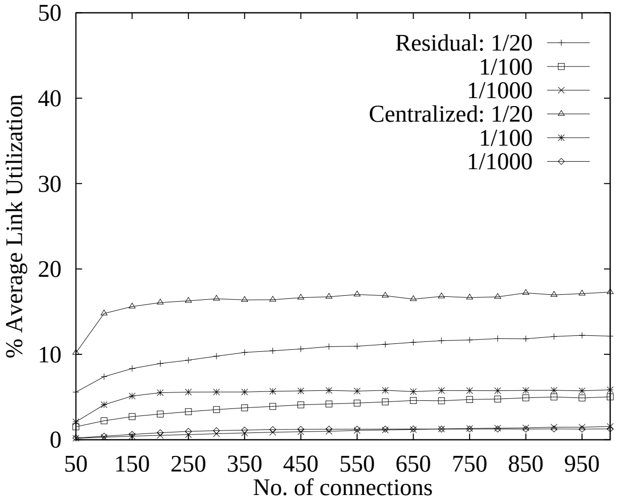
On the other hand, the Residual Time algorithm may reject such long connections and admit other connections instead, that use less resources. This indicates that although the Residual Time algorithm seems to be conservative initially, it actually performs better under increased network load. Figure 8 gives the percentage of average link utilization. We can see that, in spite of the acceptance ratio of the Residual Time algorithm being higher for higher network loads, the average



(a) 6x6 Torus Topology



(b) Acceptance Ratio



(c) Ave. Link Utilization

Figure 9: Experimental Results on a 6x6 Torus

link utilization for both the algorithms remains more or less the same. This result supports the theory stated above, of the centralized algorithm admitting a smaller number of long connections and the Residual Time algorithm's tendency to admit more shorter ones. To compare the performance of the algorithm for different network topologies, we performed a similar set of experiments on a 6x6 Torus topology, with 36 5-port switches and 36 hosts, as shown in Figure 9. This topology has a higher host-per-link ratio, and thus less contention for links among connections. Because of this, the performance difference for low load between the two algorithms is more marked as compared to the 4x4 mesh. For light network loads, the centralized algorithm performs substantially better, but again, the Residual Time algorithm catches up for heavy loads. This is all the more critical in a large network where the overhead of a centralized scheme is more pronounced. The corresponding link utilization results are also comparable.

5 Summary and Conclusion

Wormhole routing is a cost-effective way to provide very-high speed, very-low latency communication for emerging distributed real-time applications with high-bandwidth demands. Unfortunately, the lack of traffic buffering in the switches makes it difficult to give real-time guarantees to traffic streams that contend for communication links. In this paper we propose a simple blocking model for traffic on wormhole-routed networks and derive a fully distributed admission control scheme for periodic traffic streams. The scheme is based on a three-pass message exchange among the switches on the connection route. Experimental comparisons with a fully centralized algorithm show that this scheme performs very well, especially for highly loaded networks.

This distributed scheme has a number of non-obvious benefits. First, it can be seamlessly integrated into existing admission control schemes for store-and-forward networks (e.g. [8, 10]), and thus allow a well-integrated real-time service over hybrid (that is, store-and-forward and wormhole-routed) networks. Second, guaranteeing timing requirements subsumes deadlock avoidance. Our scheme can therefore support a distributed, adaptive routing scheme that guarantees deadlock-free routes. We are currently not taking advantage of this. We are currently investigating adaptive, load-sensitive scheduling schemes for real-time connections over wormhole-routed networks.

During connection establishment, resources can be allocated (in form of maximum blocking times) without involving nodes that are not on the connection route. We don't have a similar algorithm for resource deallocation during channel teardown. Resource deallocation must be done by network management, which periodically discovers allocated, but unused blocking times, and makes them available to new connections.

The only assumption we make about the switches in this work is that they be fair. There are current efforts (e.g. [9]) to study the benefits of special support for real-time traffic in wormhole-routed switches. We are investigating how such additional functionality (e.g. priorities for packets) affects the performance of our scheme.

References

[1] G. Agrawal, B. Chen, W. Zhao, and S. Davari, Guaranteeing Synchronous Message Deadlines with the Timed Token Protocol. In *IEEE Computer*, Vol. 43. No. 3, March 1994.

[2] A. Banerjea and S. Keshav. Queueing delays in Rate Controled ATM Networks. In *Proceedings of INFOCOM '93*, San Francisco, March-April 1993.

[3] Riccardo Bettati and Anisoara Nica. Real-Time Networking over HIPPI. In *Proceedings of the Workshop on Parallel and Distributed Real-Time Systems*, Santa Barbara, California, April 24-26, 1995.

[4] Riccardo Bettati, Domenico Ferrari, Amit Gupta, Wendy Heffner, Winnie Howe, Mark Moran, Quyen Nguyen and Raj Yavatkar. Connection Establishment for Multi-Party Real-Time Communication. In *Proceedings of the 5th International Workshop on Network and Operating System Support for Digital Audio and Video*, Durham, NH, April 1995.

[5] B. Chen, H. Li and W. Zhao. Meeting Delay Requirements in Computer Networks with Wormhole Routing. In *Proceedings of the IEEE International Conference on Distributed Computing*, Hong Kong, May 1996.

[6] N. J. Boden, D. Cohen, R. E. Felderman, A. E. Kulawik, C.L. Seitz, J. N. Seizovic and W.-K. Su. Myrinet – A Gigabit-per-Second Local-Area Network. In *IEEE-Micro*, Vol.15, No. 1, pp. 29-36, February 1995.

[7] R. Felderman, A. DeSchon, D. Cohen and F. Finn. ATOMIC: A High Speed Local Communication Architecture. In *Journal of High Speed Networks*, Vol. 3, No. 1., 1994, pp. 1-30.

[8] Domenico Ferrari, Anindo Banerjea and Hui Zhang. Network Support for Multimedia: A Discussion of the Tenet Approach. In Computer Network and ISDN Systems, pages 1267–1280, July 1994.

[9] J.-P. Li and M. W. Mutka. Priority Based Real-Time Communication for Large Scale Wormhole Networks. In *Proceedings of the IEEE IPPS '94, 8th International Parallel Processing Symposium*, pp. 433-438, April, 1994.

[10] Craig Partridge and Stephen Pink. An Implementation of the Revised Internet Stream Protocol (ST-2). In *Journal of Internetworking Research and Experience*, pages 27-54, 1992.

[11] Lixia Zhang, Steve Deering, Deborah Estrin, Scott Shenker and Daniel Zappala. RSVP: A New Resource Reservation Protocol. In *IEEE Networks Magazine*, Vol. 31, No. 9, pages 8-18, September 1993.