A Cooperative Congestion Management Scheme for Switched High-Speed Networks

Andrea F. Lobo and Adarshpal S. Sethi
Department of Computer and Information Sciences
University of Delaware
Newark, Delaware 19716, USA
lobo,sethi@cis.udel.edu

Abstract

In a wide-area network with high-speed links, congestion may lead to poor utilization and the degradation of the service provided to the users. This paper presents a Cooperative Congestion Management Scheme in which the intermediate nodes in the vicinity of a congested link share their resources to offload this link before packets are discarded. The scheme uses a progression of proactive control mechanisms on selected traffic bursts: From local rerouting, through upstream rerouting and quenching, to packet discarding. It complements existing open-loop mechanisms by allowing more liberal packet admission while preserving the underlying switching architecture. The performance evaluation results obtained via simulation show that its implementation reduces the probability of packet discarding by an order of magnitude.

1. Introduction

Typical existing and emerging broadband applications are highly bursty, and require connection-oriented service and Quality of Service (QoS) guarantees [1, 2]. Designers of large high-speed networks trade off efficient utilization of network resources and the flexibility to provide integrated services to different classes of users. Utilization can only be increased by admitting more traffic into the network. Higher loads of bursty traffic may lead to congestion and QoS violations. This paper presents the details of a Cooperative Congestion Management Scheme that complements existing open-loop control mechanisms and allows increased network loading while satisfying the QoS requirements of broadband applications.

Traffic and congestion management are particularly challenging in switched networks that have large bandwidth-delay products and bursty users. There are two main approaches to these problems in the literature. The first one advocates open-loop control mechanisms for admitting traffic into the network and simple processing at the intermediate nodes. They include rate-based control, the leaky bucket, and other forms of policing mechanisms [3, 4, 5]. These are sometimes combined with local traffic control mechanisms in the switches that discard traffic selectively [6] or that provide intelligent buffer management [7, 8, 9, 10, 11]. Potential downstream interaction between the bursty traffic from different entry nodes leads to inadequate or conservative admission policies since the management function at the intermediate nodes in the case of potential congestion is limited [12, 13]. The second approach uses a credit-based closed-loop control which is exercised per link, per connection, per packet [14, 15, 16]. A major disadvantage of this approach is that the control functions interfere with the switching functions at the intermediate nodes.

The Cooperative Congestion Management Scheme (CCMS) complements open-loop schemes by enhancing the funtionalities of the intermediate network nodes so that they may cooperate to prevent congestion. The entry nodes can implement more liberal traffic admission policies knowing that the network can quickly prevent a situation of potential congestion from escalating. Unlike the credit-based scheme, higher utilization is supported by the CCMS while em preserving the underlying switching architecture. The next section addresses the Cooperative Congestion Management strategy. Section 3 presents the details of the CCMS. The performance of the CCMS is evaluated in Section 4.

2. The Cooperative Congestion Management Strategy

When the traffic streams of bursty sources are allowed to enter the network with more freedom, their large peak rates can cause congestion to arise quickly at a link. This creates the need for quick control decisions in a network of large

0-8186-7453-9/96 $05.00 © 1996 IEEE

geographic expanse. The CCMS adopts a solution based on the proactive intervention of the intermediate nodes close to the overloaded area [17, 18]. The actions at the nodes include local and upstream rerouting of bursts of traffic onto alternate routes with available capacity.

When a node detects that one of its links is in a state of potential congestion, it attempts to offload the link by rerouting one of its active connections. Since connectivity in the networks of interest is moderate, it is likely that there exist one or more alternate paths from the node controlling a congested link to the destination of one of its active connections. If available resources exist along one of these alternate routes, they can be used to limit the impact of congestion by rerouting a selected burst around the congested link. If the local node does not succeed in rerouting, it informs an upstream node which in turn tries to reroute the ongoing burst. Any node that does not succeed in rerouting requests an upstream neighbor to do so. If none of the nodes can reroute, the reroute request message reaches the connection's entry node which then implements a more conservative packet admission policy for the remainder of the burst. Only as a last resort, if the congestion persists and exceeds a given threshold, will the congested node discard packets.

The scheme must preserve the underlying switching architecture while placing the necessary functionality at the intermediate nodes to control potential congestion as locally as possible. This is achieved by separating the switching and control functions at the intermediate nodes. Each intermediate node contains two entities. A switch performs the function of routing packets from each incoming link to the appropriate outgoing link. An *Integrated Network Control Agent* (INCA) [19] is responsible for the control functions at the node, including congestion management. The INCAs monitor the control variables of interest, exchange state information and requests for remote actions, and implement the control functions.

The CCMS is independent of a particular connection start-up mechanism but the discussion to follow assumes the concept of *Paths* is being implemented. This means that a mechanism for route selection and statistical bandwidth allocation is in place which allows the entry nodes to quickly set up a connection, yet does not provide enforcement or guarantee bandwidth availability. Paths are pre-established from each entry node to one or more exit nodes. The next section presents the details of the CCMS.

3. The Cooperative Congestion Management Scheme

The INCA at each network node manages the local outgoing links using a hierarchy of control mechanisms. As a link's congestion level increases, more drastic control actions are used to manage it. The congestion level can be either *low*, *moderate* or *high*. In the ideal situation of a low level of congestion, the node switches the connections that traverse this link along their usual Paths with small or non-existent queuing delays. If the congestion level is high, the agent informs the local switch to selectively discard any tagged packets arriving for transmission on the link. A moderate congestion level indicates the potential for congestion. In this state, the agent attempts to offload the link and bring its congestion level down to low. Proactive control by the agent during this state of potential congestion is one of the major contributions of this work and is the main focus of this section.

The local agent attempts to offload a moderately congested link by rerouting the ongoing burst of a connection. The agent selects an active connection and tries to find an alternate Path—from the local node to the destination of the connection—that avoids the overloaded link and supports the QoS requirements of the connection's ongoing burst. The mechanism used by an INCA for alternate Path selection is the same as that used by an entry node during connection setup. The rerouted burst constitutes a special type of "connection" on the alternate Path: one with a single, peak-rate burst. The connection management scheme relies on the concept of Paths to satisfy the QoS requirements of the rerouted bursts.

Whenever a node cannot find an adequate alternate Path locally, it instructs the corresponding upstream neighboring node to attempt to reroute. If the request to reroute propagates back to the entry node of the connection and rerouting fails at this node, the entry node implements a more conservative packet admission policy for the remainder of the burst. This quenches the source of the connection and reduces the load of the downstream moderately congested link.

When an agent successfully finds an alternate Path, it sets up a temporary connection along this Path. The agent informs the local switch to forward any remaining packets of the burst on this temporary connection. This is accomplished by temporarily changing one routing table entry at the rerouting node and does not require modifications to the underlying switching architecture at the intermediate nodes. When the rerouted burst ends, the agent informs the local switch to revert to the connection's original Path. The resources allocated to the temporary connection are used for rerouting other connections or are de-allocated.

When rerouting succeeds, the destination node of the rerouted connection receives two sub-bursts: The first part of the rerouted burst arrives on the original Path and connection, and the second part arrives on the alternate Path and temporary connection. To preserve packet ordering, the destination node concatenates the two sub-bursts. Implementation of the CCMS requires modification of the underlying protocol at the exit nodes since they must process the in-channel signaling messages needed for sub-burst concatenation. If the initial packets of the second sub-burst arrive at their exit node before the entire first sub-burst is received, it is desirable to have buffers at the exit node so that these packets may be stored until they can be delivered to the higher layer on the protocol stack.

The next section describes the parameters of the cooperative congestion management scheme. Section 3.2 introduces some notation that will be useful in the presentation of the sections that follow. The implementation of the hierarchy of control mechanisms at the intermediate nodes is presented in Sections 3.3 through 3.9. The management functions of the entry and exit nodes are described in Sections 3.11 and 3.12, respectively.

3.1. Parameters of the Cooperative Congestion Management Scheme

Five parameters determine the behavior of the cooperative congestion management scheme. The first two are used by each control agent to measure the time-averaged load of its local outgoing links as described in Section 3.3:

- *WeigHist* is the weight given to the history of the average link load. The value of *WeigHist* must be between 0 and 1. A value closer to 0 gives less weight to the older values of measured link load and more importance to the most recent observation of link utilization.
- *UpdaInte* denotes the number of slots between updates to the average link load, where a *slot* is the time it takes to transmit a fixed-size packet on a representative link.

The measured load of a link is compared against two thresholds to determine the link's level of congestion:

- *ThreLow* denotes the threshold between the *low* and *moderate* congestion levels. Its value is the target upper bound on the utilization of each link.
- *ThreHigh* is the threshold between the *moderate* and *high* congestion levels.

The fifth parameter tells the control agents how insistent they must be on local rerouting and is used, as described in Section 3.6, when selecting a connection for rerouting:

- *LocaPref* denotes the preference given by the control agents to local rerouting. When selecting a connection for offloading, a local agent will choose one that requires upstream rerouting only after unsuccessfully attempting to find a local alternate Path for these many connections.

Section 3.3 describes how a local control agent uses the first four parameters to compute each link's congestion level. The use of *LocaPref* when selecting a connection for rerouting is shown in Section 3.6.

3.2. Data Structures and Notation

This section describes the variables that are maintained at each agent. The notation used for each variable is also given. A variable that has invalid semantics at a node will be set to *nil*. Let e and x be the respective entry and exit nodes of a Path p, n be an intermediate node on p and l be the outgoing link of n that p traverses, a be the agent that manages n and s the switch at n, and c be n's local identifier for a connection that traverses p.

For each outgoing link of n, a keeps track of the following dynamic information about link load:

- $Cong_{n,l}$, the level of congestion of the link. $Cong_{n,l} \in \{low, moderate, high\}$.
- $Load_{n,l}$, the time-averaged measured load of the link. $Load_{n,l} \in [0, 1]$.
- $Busy_{n,l}$, the number of slots in the current load-measurement window where the link was busy transmitting. $Busy_{n,l} \in \{0, 1, \ldots, UpdaInte\}$.

The local agent also maintains information about the dynamic state of some connections that traverse each outgoing link as follows. The table $ReroCand_{n,l}$ identifies those connections that traverse l and whose QoS requirements can tolerate intentional rerouting. In addition to a connection identifier, each entry in $ReroCand_{n,l}$ contains an active bit, $ReroCand_{n,l}Acti_c$, that is set when connection c has an ongoing burst. If $ReroCand_{n,l}Acti_c$ is set, the table also contains a $ReroCand_{n,l}Stat_c$ field with information about the state of c's ongoing burst. There are four possible values for $ReroCand_{n,l}Stat_c$:

- *candidate* indicates that a has not considered a control action on the burst and it remains a candidate for rerouting or quenching;
- *rerouted* indicates that the burst was locally rerouted at n. In this case, a records in $ReroCand_{n,l}Temp_c$ the identifier of the temporary connection that is carrying the rerouted portion of the burst, and a number $ReroCand_{n,l}Numb_c$ that identifies each of the two sub-bursts;
- *quenched* indicates that the burst is being quenched at its entry node, and no further action will be required;
- *pending* indicates that a has sent an upstream reroute request and that action on the ongoing burst is pending.

Figure 1 illustrates the possible transitions between the states of a connection. When a connection becomes *active* it is a *candidate* for control actions. The connection is targeted for rerouting if the local agent receives a request to reroute it from downstream. It may also be targeted as a result of a local decision to offload its outgoing link. In either case, the local agent tries to find an adequate alternate Path for its

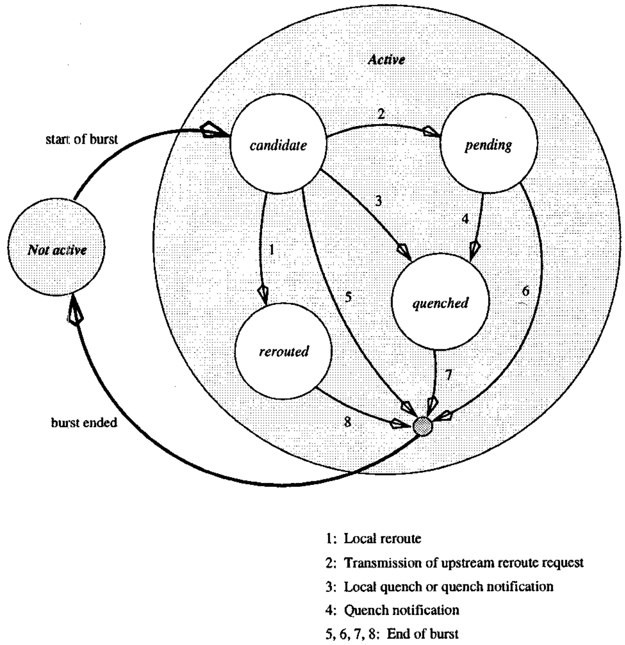

Figure 1. State diagram for a reroutable connection.

ongoing burst. If this succeeds (1), the connection is *rerouted.* If no alternate Path is found, the state of the connection depends on its original Path. If there is an upstream node (2), an upstream reroute message is sent and the connection is *pending.* If the local node is the connection's entry node (3), the connection is *quenched.* The connection enters the *quenched* state at its intermediate nodes when they are informed that quenching is in effect (3 and 4). The connection becomes *not active* when its current burst ends (5, 6, 7, and 8). The details of the transitions are presented below.

The local agent keeps the following data about the congestion control actions in effect for each outgoing link:

- $Disc_{n,l}$, a bit that is set if the local switch is discarding the tagged packets that arrive at l's buffer.
- $RequOffl_{n,l}$ is the sum of the peak rates of all the connections in $ReroCand_{n,l}$ such that $ReroCand_{n,l}Acti_c$ and $ReroCand_{n,l}Stat_c$ indicate that an upstream request is pending for the connection's ongoing burst. This is the expected reduction in the link's measured load if all outstanding upstream reroute requests succeed. $RequOffl_{n,l} \in [0, 1]$.
- Other variables as required by the connection admission mechanisms, such as link capacity, resources that have not been allocated to any Path, or number of connections of each class of traffic that have been admitted on Paths that traverse l.

The following sections to present the complete algorithms of the CCMS. The control agent at a node tracks the congestion level of its outgoing links using the algorithms of Section 3.3. An agent, a, manages its outgoing links using a hierarchy of control mechanisms. The decision by a to implement a control action on a link l is addressed in Section 3.4. The agent's proactive control actions are discussed in Sections 3.5 through 3.8. If previous measures fail to reduce the load on l and its congestion level becomes high, a decides to selectively discard the packets arriving for transmission on l. The algorithms relevant to selective discarding are presented in Section 3.4 and further discussed in Section 3.10. Quenching at the entry nodes is discussed in Sections 3.7, 3.9 and 3.11. The exit nodes preserve packet order via the algorithms of Section 3.12.

3.3. Congestion Level of a Link

Each agent monitors the load on the local outgoing links. Agent a simultaneously runs two algorithms to compute the congestion level of l. A passive monitor tracks the utilization of l during each slot. An averaging algorithm runs every *UpdaInte* slots. It uses the most recent observation on the utilization of l to update the time-averaged link load. The congestion level of l is abstracted from the updated link load. When a link's congestion level changes or remains above *low*, the local agent may decide to implement or cease a control action on the link. The minimum time between these decisions is *UpdaInte*, the time between consecutive end-of-window interrupts.

3.4. Decision to Initiate a Control Action

The decision mentioned above is made by a procedure that determines whether the local agent must implement a control action to manage the link. The appropriate control action depends on the congestion level of the link. If the link is no longer congested, no control action is necessary. If it is highly congested, the local agent decides to discard tagged packets while attempting to offload the link. If the link is moderately congested, the local agent may attempt to offload it.

3.5. Attempt to Offload a Link

To prevent excessive rerouting, an agent with outstanding upstream reroute requests bases its decision to attempt offloading on the quantitative measurement of link load, and on the previously requested offload level. When rerouting cannot be implemented at the local node, the local agent sends an upstream reroute request. The effect of this request will not be perceived by the local agent before a round-trip propagation delay to the rerouting node. Since *UpdaInte* is likely to be many orders of magnitude smaller than the propagation delay between the local agent and its upstream neighbors, there may be several pending upstream reroute requests. The local agent assumes that earlier requests are as likely to successfully offload the moderately congested link as are new requests. It attempts further offloading only if the link's measured load minus the requested offload exceeds the target *ThreLow*. Thus, offloading is attempted if the link load would entail a congestion level other than *low* in the event of all pending requests succeeding.

The local agent attempts to reduce the load on link l by rerouting one of its active bursts. The agent a computes the set C of connections that traverse l and are *candidate*s for rerouting when offloading is attempted. Alternatively, a would maintain an updated copy of C for each link with a moderate or high congestion level. The latter approach reduces the execution time of the attempt to offload but requires updating C when connections are set up and torn down, when bursts start and end, and when a congestion management decision affects the state of a burst.

3.6. Selection of a Connection for Rerouting and its Alternate Path

The local agent selects a connection *reroutingCand* as a candidate for rerouting from C. The selection process uses randomization to prevent oscillations and is biased towards connections that can be locally rerouted.

The agent determines if one of its own Paths can support the selected connection's ongoing burst. If so, the burst's subsequent packets will follow the connection's original route up to the local node, and the new route from this node to the exit node. In this manner, the local agent cooperates with its peer at the connection's entry node by temporarily sharing some of the local resources. The agent maintains information on each connection's final destination in $Exit_{n,c}$. This variable is redundant but it speeds up the process of finding an alternate Path since the rerouting node can retrieve the identity of the intended destination with a single table access.

An alternate Path must have the following characteristics: its entry node is the local node; its exit node is the destination of *reroutingCand*; supports a connection with an average load of *reroutingCand*'s peak rate since it will be carrying an ongoing burst; supports any other QoS requirements of *reroutingCand*;avoids the overloaded link; and its first link has a low congestion level. When selecting a Path for rerouting, it would be desirable to ask the connection admission mechanisms for a Path that does not contain the overloaded link, if they have provisions for such requests.

If an adequate alternate Path, *alternatePath*, is found, a temporary connection is set up along it to carry the tail of the burst. The time-consuming task of fabric (switch) configuration for the new connection can be avoided if the local agent sets up special connections along the potential alternate Paths for the purpose of rerouting. These connections are allocated bandwidth resources when they are assigned to reroute a sub-burst and repeatedly serve for rerouting. A temporary connection is not a candidate for rerouting. This simplifies the task of restoring packet order at the destination since it precludes the nodes on the alternate Path from further partitioning the burst. A second reroute of a burst may occur if a downstream node selects to reroute the remaining portion of the first sub-burst, or if an upstream node selects the burst for rerouting at any time.

3.7. Upstream Reroute Requests

The agent at n may not be able to locally reroute c. If this is the case and n is not the entry node of c, the local agent sends an upstream reroute request to its peer in the node that is immediately upstream with respect to c. If n is the entry node, this upstream neighbor does not exist and the local agent quenches c's ongoing burst. If this is the case and c's Path contains a downstream intermediate node, the local agent also sends a message to its downstream neighbor indicating that c has been quenched.

A downstream intermediate node processes an incoming quench control message as follows. If the receiving node has a pending upstream reroute request for c, it now knows that the attempt to reroute failed and updates the corresponding *RequOffl*. If c had been rerouted at the receiving node, the quench message produces no action because a rerouted burst cannot be rerouted again. The quench control message is passed on to the next intermediate node, if such a node exists.

When an upstream reroute request arrives at n, the local agent attempts to find a local alternate Path to reroute the ongoing peak-rate burst of c. The neighboring downstream node that sent the request was recently unable to reroute. The algorithm assumes that the most likely cause for this failure to reroute is the overload of the first links on the potential alternate Paths. Thus, an adequate local alternate Path must avoid the downstream node to bypass its overloaded links. The alternate Path must also avoid the overloaded node that originated the request to reroute c. An alternate Path that does not satisfy these conditions is rejected. A successful reroute immediately adds to the load on the alternate Path since a rerouted sub-burst is in progress. To prevent increasing local congestion, an alternate Path is also rejected if the current congestion level on its first link is not low. If an adequate alternate Path is not found, the local agent tries to forward the request further upstream. If n is c's entry point, this will result in a quench.

The decision to attempt to offload a link, as described in Section 3.5, depends on the link's measured load and its pending upstream reroute requests. While a connection is in the *pending* state of Figure 1, it contributes to the load and the *RequOffl* of its local outgoing link. During this time, the local agent assumes that a control action on the pending connection is imminent but the connection's packets continue to traverse the overloaded link.

A connection enters the *pending* state when the local agent transmits a request to reroute to the corresponding upstream agent (transition 2). It leaves this state when the local agent receives notification that the connection was quenched (transition 4) or when the local agent receives its next end of burst message (transition 6). The time spent in this state is the minimum of the remaining burst length as witnessed by the local agent, and the time before the arrival of a quench notification for this burst. Thus, the upper bound on the time that an upstream reroute request remains outstanding is the minimum of the remaining burst length and the maximum time before the arrival of a quench notification. In the case of an upstream reroute request that results in a quench, the control actions effectively reduce the link load if the round-trip propagation delay from the local node to the entry node plus the processing delays at the upstream nodes is shorter than the remaining length of the connection's natural burst. When an upstream reroute request results in a successful upstream reroute, the control actions reduce the link load if the round-trip propagation delay from the local node to the rerouting node plus the processing delays at the intermediate upstream nodes is shorter than the remaining length of the connection's natural burst. The latter case is illustrated in Figure 1 by transition number 6 since, as discussed in the following section, a successful reroute produces an artificial end of burst at the downstream intermediate nodes.

3.8. Rerouting

An agent locally reroutes a connection in response to its decision to offload a local outgoing link, or to a reroute request that arrives from downstream. The following algorithm is executed by the agent at n to reroute c onto temporary connection c' along alternate Path p'.

Before starting the transmission of packets on the alternate Path, the local agent sends an in-channel dummy start-of-burst message on c'. It also sends an in-channel start-of-second-sub-burst control message on c' to the connections' exit node. This message indicates that c' carries the second part of a rerouted burst, and specifies the identifier that will be received by the agent at the exit node at the end of the first part of the burst.

The agent informs the local switch to forward future packets of the rerouted connection onto c'. These rerouted packets are switched without additional overhead. They traverse their original Path up to the rerouting node, and the alternate Path from this node to their exit node. Although a rerouted connection does not flow through the nodes on its original Path that are downstream of the rerouting node, these nodes maintain their information about the connection so that it may be returned to its original Path when its current burst ends. Thus, the temporary connection only carries the rerouted sub-burst.

As soon as the switch stops forwarding packets on the original Path, the local agent sends an end-of-first-sub-burst control message on c. This indicates the end of the first part of the rerouted burst to the agent at the exit node, and specifies the same number that identified the start of the second part of the burst. The agent uses the identifiers in the control messages to concatenate the two sub-bursts as described in Section 3.12.

3.9. Processing Start- and End-of-Burst Control Messages

The intermediate nodes are informed about the start and the end of bursts of their rerouting candidates. The entry node of a connection transmits start-of-burst and end-of-burst control messages using in-channel signaling. A control message is forwarded to the next node on its Path but not handed to the upper layers at the exit node. A copy of the message is given to the local agent where it is processed. An end-of-burst message marks the end of a rerouted burst annd the rerouting agent returns the connection to its original Path. Since the information about the connection has been saved at all other nodes on its original Path, the routing agent re-establishes the original Path by restoring the local switch's routing table entry for the rerouted connection. The end-of-burst message is propagated down the alternate Path since the downstream nodes on the original Path have received the dummy end-of-burst message generated at the time of rerouting. If the connection is *pending*, the value of *RequOffl* for its outgoing link is updated. In all cases, the connection is no longer active and its state is reset to *candidate* to prepare for its next burst.

3.10. The Switch at Each Intermediate Node

There are three functions that the switches must support in addition to forwarding packets. They must be capable of implementing selective discarding during periods of high congestion. Since the control actions of the cooperative congestion management scheme are on bursts, the switches must be managed via operations that change the value of its control variables and routing tables within this real-time granularity. As discussed in Section 3.9, a switch must also recognize if a packet contains in-channel signaling. In this case, in addition to forwarding the packet, the switch makes a copy for the local agent.

3.11. The Entry Nodes

The entry nodes implement two policies for the admission of excess packets: A conservative one, comparable to that which would be used without the CCMS, and another that admits excess packets more liberally. The liberal packet admission policy is used for connections that are candidates for intentional rerouting only. The local agent must be able to specify which policy is to be used for each connection, and to request a change from liberal to conservative policy while a burst is in progress. In the latter case, the agent generates a control message informing its downstream peers that the current burst is being quenched. When the quenched burst ends, the packet admission policy for its connection reverts to liberal. The entry node of a connection also informs the agents on the connection's Path about the beginning and end of a burst of traffic by sending an in-channel start-of-burst and end-of-burst control message.

3.12. The Exit Nodes

Exit nodes terminate connections. The exit node x of connection c delivers the service data unit in each incoming packet to the next higher protocol layer via the $SAP_{x,c}$ service access point. A rerouted burst arrives at its exit node in two parts. The first sub-burst arrives on the connection's original Path, and is followed by an end-of-first-sub-burst control message. The second sub-burst arrives along a temporary connection on an alternate Path, and is preceded by a start-of-second-sub-burst control message. Both control messages contain a common magic number that is used by the exit node to match the corresponding pair of sub-bursts. Since packet order is maintained within each sub-burst, the exit node preserves global packet ordering and rerouting transparency by ensuring that all packets of the first sub-burst are delivered to the higher layer before the first packet of the second sub-burst.

An exit node x maintains a sub-burst concatenation table ($Conc_x$) that relates sub-burst pairs. The entries in $Conc_x$ contain five fields. *ConcNumb* denotes the magic number that is used on both sub-bursts. $Conc_x$ is indexed by *ConcNumb*. *ConcConn* denotes the original connection that was rerouted. *ConcTemp* denotes the temporary connection that traverses the alternate Path and transports the rerouted sub-burst. *ConcBufR* denotes the local concatenation buffer assigned to store any packets of the rerouted sub-burst that must wait before being delivered to the upper layer. *ConcBufO* denotes the local concatenation buffer assigned to store any packets that may arrive on the original connection before the end of the rerouted sub-burst. An entry of $Conc_x$ that contains a nil *ConcTemp* and a non-nil *ConcConn* represents a connection whose end-of-first-sub-burst control message has arrived at x, and whose matching start-of-second-sub-burst message is outstanding. In this case, the packets in the second sub-burst may be delivered to the higher layer as they arrive at x. When the control messages arrive in the reverse order, the corresponding entry is set up with a nil *ConcConn* and non-nil *ConcTemp*. In this case the packets of the second sub-burst must be buffered until the end of the first sub-burst. While the rerouted sub-burst is in progress, any packets that arrive on the original connection are stored in *ConcBufO*. They are delivered to the higher layer after the rerouted sub-burst ends. This preserves packet order in the unlikely case that the end of the rerouted sub-burst is delayed on the alternate Path beyond the beginning of the next burst. The maximum buffering requirements are obtained as above. The rare use of these buffers makes them excellent candidates for a shared buffer scheme.

4. Performance Evaluation

In a network with a single type of traffic, the behavior of the system is controlled by a single parameter: the threshold between low and moderate congestion levels, *ThreLow*. There are two performance indices of interest in this case. The effectiveness of the congestion management scheme is measured by the number of discarded packets. The cost of implementing the scheme is an increasing function of the number of bursts that are rerouted.

A *congestion spurt* at a link starts when the link load crosses from low to moderate; it persists until the link load returns to low. The effectiveness of the CCMS during one congestion spurt at a link is characterized by the C/S ratio, the ratio of the delay in offloading the link to the time elapsed until the first packet discard [20]. Once the link load crosses the threshold into the moderate level, the link must be offloaded before the first discard. Effective closed-loop congestion management requires that C be less than S. S denotes the time between the threshold crossing and the first discard at the link. The upper bound on C is less strict if the rate of congestion onset is slower or the threshold has a smaller value. The numerator, C, denotes the time between the threshold crossing and the instant when the link is offloaded. The dominant factor in the expression for C is the round-trip delay between the congested and the rerouting node. When local rerouting is not possible, the congestion management scheme will be less effective since C increases. Long link lengths have a negative impact on system performance because they increase the critical round-trip propagation delay. Shorter link lengths result in decreased propagation delays, thus decreasing C and enabling the congestion management scheme to be effective when the onset of congestion occurs more quickly.

The effectiveness of the CCMS was studied via simulation. The network topology is centered around two targeted

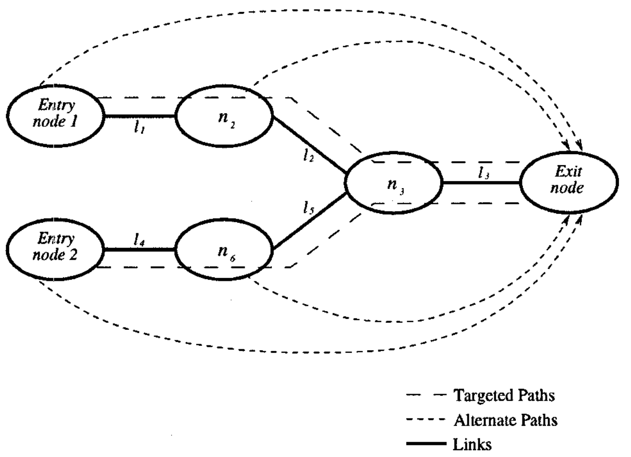

Figure 2. Topology for the simulations.

Paths called *TP1* and *TP2*. *TP1* carries traffic from the *Entry node1*, also called n_1, to the *Exit node*, also called n_4, as depicted in Figure 2. *TP1* traverses the forward direction of links l_1, l_2 and l_3, and the intermediate nodes n_2 and n_3. *TP2* carries traffic from the *Entry node2*, also called n_5, to the *Exit node*. *TP2* traverses the forward direction of links l_4, l_5 and l_3, and the intermediate nodes n_6 and n_3. Thus, *TP1* and *TP2* share their downstream-most link.

All links have equal capacity and length. As propagation delays between the congested and rerouting nodes increase beyond the average burst length, closed-loop control becomes infeasible. Link lengths are on the order of the average burst length so that the simulation experiments test the boundary of the scheme's operating region.

Each of n_1, n_2, n_5 and n_6 has an alternate Path to the destination node with capacity to support one burst. Alternate Paths are mutually disjoint, and also disjoint with *TP1* and *TP2*.

The traffic on each link consists of packets from the connections that traverse either *TP1* or *TP2*. All connections have the same traffic characteristics. The traffic of each connection is generated using a binary Markov model. A source of traffic is either idle and generating no traffic, or generating traffic at its peak transmission rate. The peak and average rate of a connection are 0.05 and 0.005 of link capacity, respectively. Thus, the burstiness ratio of a connection is 10 (0.05/0.005). The average burst length is 142,857 slots. Each targeted Path is loaded with 60 connections, bringing the average load of link l_3 to 0.60.

We run simulation experiments for 12 different cases. In all cases, *ThreHigh* is set to 1.0, *UpdaInte* to 50 slots, *WeigHist* to 0.3, and *LocaPref* to 5. Each case is specified by the value of the threshold between the low and moderate congestion levels, and the value of link length. The values of *ThreLow* are 0.60, 0.70 and 0.80. The values of link length are $1/8, 1/4, 1/2$ and 1 times the average burst length. The combinations of values for these two parameters yield the 12 test cases in the test suite. In those cases where link length is equal to half the average burst length, the network topology results in a best-case rerouting delay equal to the average burst length. Thus, the operating region of the CCMS is expected to range over values of link length that do not exceed 70,000 slots.

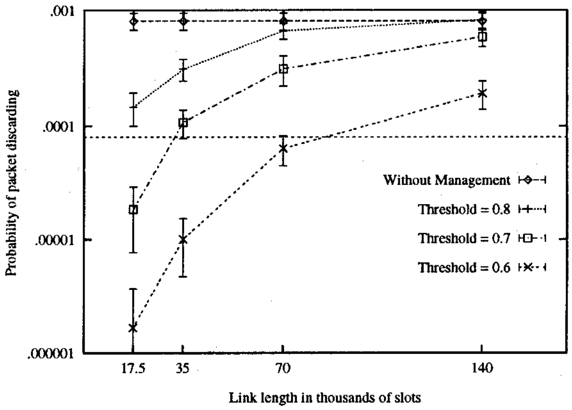

Figure 3. Probability of packet discarding as link length increases for different values of threshold.

Figure 3 shows the 95% confidence interval of the probability of discarding a packet for each link length and threshold pair. The test case for each pair was simulated for 300 million slots. Each reported value was calculated from 50 independent measurements that were obtained after disregarding a simulation startup transient of 10 million slots. Within the operating region of link lengths smaller than 70,000 slots, the results show that it is possible to obtain an order-of-magnitude performance improvement with the CCMS. Without congestion management, the probability of packet discarding is approximately 0.001 for a load of 0.60 on l_3. If this QoS does not meet the packet discard performance objectives, the connection admission procedures can be modified to decrease the network load. Alternatively, the implementation of the CCMS with an appropriate *ThreLow* can support the present load with a reduced packet discard probability.

Figure 4 plots the percentage of rerouted bursts for the same link lengths and threshold values of the previous plot. In the most aggressive rerouting scenario, with a threshold of 0.60 and link lengths of 17,500 slots, approximately 14% of all bursts were rerouted. For sufficiently small threshold and link lengths, the CCMS can effectively utilize the alter-

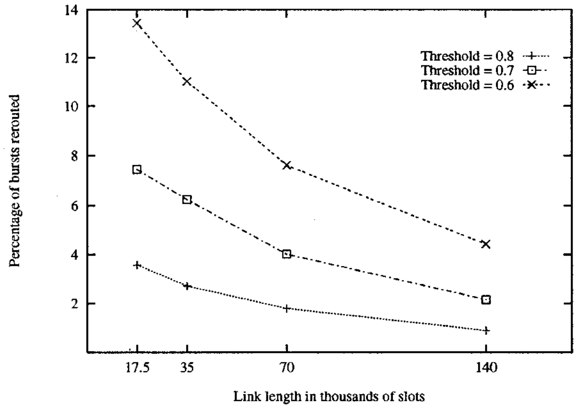

Figure 4. Percentage of bursts rerouted as link length increases for different values of threshold.

nate capacity since the alternate Paths provide 16.7% of the bandwidth to the *Exit node*. As expected, a better performance for a link length and threshold pair in Figure 3 comes at the cost of increased rerouting. For a given topology and desired probability of packet discard, a network manager can obtain the appropriate *ThreLow* value from Figure 3. The corresponding rerouting cost to implement the CCMS in this particular system can then be obtained from Figure 4. Recently obtained results [20] show that the negative impact of burstier sources on the performance of the CCMS is bounded.

References

[1] M. Ransom and D. Spears. Applications of public gigabit networks. *IEEE Network*, 6(2):30–40, March 1992.

[2] H. Fowler and W. Leland. Local area network traffic characteristics, with implications for broadband network congestion management. *IEEE Journal on Selected Areas in Communications*, SAC-9(7):1139–1149, September 1991.

[3] L. Roberts. Enhanced PRCA (proportional rate-control algorithm). Contribution ATM-Forum/94-0735R1, ATM Forum, August 1994.

[4] O. S. Aboul-Magd. Performance of link-by-link rate control for ABR services. Contribution ATM-Forum/94-0767, ATM Forum, September 1994.

[5] K. Bala, I. Cidon, and K. Sohraby. Congestion control for high speed packet switched networks. In *Proc. IEEE INFOCOM'90*, pages 520–526, San Francisco, California, June 1990.

[6] A. Eckberg, B. Doshi, and R. Zoccolillo. Controlling congestion in B-ISDN/ATM: Issues and strategies. *IEEE Communications Magazine*, 29(9):64–70, September 1991.

[7] A. Demers, S. Keshav, and S. Shenker. Analysis and simulation of a fair queueing algorithm. In *Proc. ACM SIGCOMM'89*, pages 1–12, Austin, Texas, September 1989.

[8] G. Bianchi and J. S. Turner. Improved queueing analysis of shared buffer switching networks. *IEEE/ACM Transactions on Networking*, 1(4):482–490, August 1993.

[9] J. Chao. A novel architecture for queue management in the ATM networks. *IEEE Journal on Selected Areas in Communications*, SAC-9(7):1110–1118, September 1991.

[10] E. Hahne. Round-robin scheduling for max-min fairness in data networks. *IEEE Journal on Selected Areas in Communications*, SAC-9(7):1024–1039, September 1991.

[11] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala. RSVP: A new resource reservation protocol. *IEEE Network*, 7(5):8–19, September 1993.

[12] K. Sohraby and M. Sidi. On the performance of bursty and correlated sources subject to leaky bucket rate–based access control schemes. *IEEE Transactions on Communications*, 42(2-4):477–487, February-April 1994.

[13] M. Butto, E. Cavallero, and A. Tonietti. Effectiveness of the "leaky bucket" policing mechanism in ATM networks. *IEEE Journal on Selected Areas in Communications*, SAC-9:335–342, April 1991.

[14] H. T. Kung and R. Morris. Credit-based flow control for ATM networks. *IEEE Network*, 9(2):40–48, March/April 1995.

[15] D. Hunt, R. Nair, et al. Flow controlled virtual connections proposal for ATM traffic management. Contribution ATM-Forum/94-0632R2, ATM Forum, September 1994.

[16] R. J. Simcoe. The great debate over ATM congestion control. *McGraw-Hill's Networking Technology Magazine*, 23(13):75–94, September 1994.

[17] A. F. Lobo and A. S. Sethi. A combination of mechanisms for effective congestion management in switched high-speed networks. In *2nd International Conference on Telecommunications Systems Modeling and Analysis*, pages 367–371, Nashville,Tennessee, March 1994.

[18] A. Lobo and A. Sethi. A distributed cooperative congestion management strategy for high-speed networks. In *Proc. 27th Annual Conference on Information Sciences and Systems (CISS '93)*, pages 707–712, Baltimore, Maryland, March 1993.

[19] A. Sethi and A. Lobo. An architecture for distributed, cooperative, integrated network control. In *Proc. IFIP/IEEE International Workshop on Distributed Systems: Operations and Management*, Santa Barbara, October 1991.

[20] A. F. Lobo. *A Cooperative Congestion Management Scheme for Switched High-Speed Networks.* PhD thesis, Department of Computer and Information Sciences, University of Delaware, Newark, DE, May 1996. Also Technical Report CIS 96-07.