An Investigation into Fault Recovery in Guaranteed Performance Service Connections†

Colin J. Parris and Anindo Banerjea.

Tenet Group
Computer Science Division, UC Berkeley
and
International Computer Science Institute,
1947 Center St. , Suite 600
Berkeley, CA 94704-1105.
Tel: (510)-642-8905 Fax:(510)-643-7684
E-mail: {parris,banerjea}@tenet.berkeley.edu

SUPERCOMM/ICC '94
Technical Topic: Management of Advanced Networks

ABSTRACT

As high speed networks are starting to provide guaranteed performance service, it is imperative to revise fault recovery techniques to support this new service. In this paper we investigate one aspect of fault recovery in this context, the rerouting of guaranteed performance connections affected by link faults in the network. Recovery is achieved by rerouting the affected connection so as to avoid the failed link while ensuring that the traffic and performance guarantees made along the previous route are satisfied along the new route. The goal of the rerouting schemes is to reroute as much of the affected traffic as quickly and efficiently as possible. We investigate rerouting along the lines of two orthogonal components: the *locus of reroute*, which determines the node that does route selection and the new route selected; and the *timing* component, which determines when the individual reroute attempts are initiated. Within each of these two components we examine approaches that span the spectrum of that component. We compare all possible combinations of these approaches under a cross-section of network workloads, using in our comparisons a novel metric, the *Queuing Delay Load Index*, that captures both the bandwidth and delay resources required by a connection. Extensive simulation experiments were conducted on the various combinations and their results and analysis are presented in the paper.

†This research was supported by the National Science Foundation and the Defense Advanced Research Projects Agency (DARPA) under Cooperative Agreement NCR-8919038 with the Corporation for National Research Initiatives, by AT&T Bell Laboratories, Hitachi, Ltd., Hitachi America, Ltd., Pacific Bell, the University of California under a MICRO grant, and the International Computer Science Institute. The views and conclusions contained in this document are those of the authors, and should not be interpreted as representing official policies, either expressed or implied, of the U.S. Government or any of the sponsoring organizations.

1. Introduction

The need for guaranteed performance service communication in the emerging high-speed packet/cell switched networks has been widely recognized in research and industry [23]. This is usually defined as communication with bounds on performance parameters such as bandwidth, delay, jitter, and loss rates. A significant amount of work has been done to investigate ways of providing such services in packet/cell switched store-and-forward networks [1, 2, 3, 4]. While many investigators have proposed solutions to the above problem, all these solutions presume adequate fault-free service from the underlying network, in order to provide the guarantees. It is impossible to build networks which perform perfectly and meet all service guarantees under all fault conditions; therefore it is of paramount importance to incorporate fault recovery into guaranteed service schemes.

While fault recovery for general computer and telecommunications networks has long been an important topic of research, we have not come across any literature describing research that extend this all important feature to guaranteed performance service networks. This is probably due to the fact that such services have been very recently implemented, and high level management functionalities have not yet been considered. However, the basic ideas behind guaranteed performance communication have crystallized to the point where it is feasible to model the mechanisms which provide such services abstractly, in order to design fault recovery and other schemes which would work with any guaranteed performance communication scheme.

This paper examines rerouting techniques (one aspect of fault recovery) which can be used to restore guaranteed performance connections, using the remaining capacity of the network, after the occurrence of a link fault. We assume the existence of redundant connectivity (multiple paths between any host pairs) in the network. We investigate rerouting techniques by identifying two important orthogonal components or dimensions along which all rerouting schemes can be classified. The investigation examines the importance of the components as well as compares specific schemes along those dimensions, in terms of a number of performance indices which quantify the speed and efficiency of the recovery process. The objective is to identify techniques which allow connections with performance guarantees to be restored efficiently and quickly after the occurrence of a link fault.

The paper is organized as follows. In the next section, Section 2, we look at related work in the area of fault management for guaranteed performance service communication. We note that there is an absence of previous research into providing fault recovery for guaranteed performance connections. In Section 3 we motivate the investigation by considering why recovery is desirable. We define the communication model, fault model and recovery model in Section 4. In Section 5 we discuss our experimental design and describe a new load index. We present and analyze our results in Section 6. Finally we summarize our results and mention directions for future research in Section 7.

2. Related Work

Several approaches to the problem of providing guaranteed performance service communication have been proposed. For reasons of brevity we will mention just a few of the well known solutions. [1] describes the approach of the Tenet group [1] to provide such a service in a heterogeneous internetwork. This approach is based on a general parameterized client-network interface to specify traffic descriptions and performance requirements. Our work builds on the Tenet approach that will be described in greater detail in Section 4. [2] proposes dividing delay sensitive communication into predictive service and guaranteed service classes. The guaranteed service class is similar to that of the Tenet approach, however, the predictive service class does not meet our definition of guaranteed service communication as the service varies with the network load. The Session Reservation Protocol (SRP) [3] provides deterministic guarantees and is based on an approach similar to that of Tenet. The Asynchronous Time Sharing (ATS) [4] approach provides a fixed menu of Quality-of-service classes, from which the client can choose one which matches his performance and traffic needs. The Heidelberg Resource and Administration Technique (HeiRAT) [5] uses the Stream Protocol version-II to reserve bandwidth and delay resources for unicast and multicast connections.

All these schemes share two common properties. First, the network needs to know about and impose restrictions on the traffic that can be sent by clients and, second, it needs to maintain information about the performance requirements of each guaranteed service connection existing in the network, perhaps encoded in the form of service priority. These two properties are used to examine each new connection before it is accepted to ensure that the performance guarantees provided to any existing connections may not be violated because of this new connection. Failure recovery is not supported in any of the schemes presented above. However, their fundamental similarity lead us to believe that most of the conclusions of this paper could apply to designing fault recovery schemes for most guaranteed service communication networks.

Fault recovery has been studied under the context of telecommunication networks and conventional data networks. [6] presents the issues of fault recovery in telecommunication networks against a four layer model of the transport network: the switched layer, the cross-connect layer, the multiplex layer and the physical layer. The highest two layers are of the most interest. At the *switched layer* the units of

[1] At the University of California and the International Computer Science Institute at Berkeley

communication and recovery are calls. The switches of this layer use routing tables to route new calls. The recovery action taken at this layer is to simply update the routing database to correctly route calls around the fault in the network [7, 8]. Existing calls are lost and have to be redialed by the end systems. At the *cross-connect layer* the units of communication are trunks, such as DS1 or DS3 links, which in turn appear as the links of the next higher layer (the *switched layer*). These connections have fixed bandwidth requirements and the recovery action is to recompute the network state information as well as reroute affected trunks on the remaining network so as to meet the bandwidth requirements of the flows. Techniques used include pre-computing (e.g. solving integer flow problems to find near optimal solutions) and storing configurations for all or a subset of failure states [9, 10, 11], and running dynamic distributed algorithms to find short routes [12, 13, 14, 15]

Fault recovery in conventional data networks is usually concerned with recomputing the routing information so as to correctly route new data. In the Internet, the network layer protocol (IP) is connection-less, so updating routing tables to reflect the changed network state is sufficient to maintain connectivity [16]. While no performance guarantees are given, the protocols attempt to reduce congestion and network instabilities by cooperation between the routers [17, 18]. AN1 [19] from DEC SRC is a packet switched local area network designed for high survivability. On the detection of network fault, the switches run a distributed algorithm which computes the new topology and sets the routing tables. Data is not forwarded during route computation. AN2 is an ATM based connection oriented LAN still in the design stage at DEC. The basic technique in AN2 is to stop data forwarding and perform the network state acquisition and routing table computation. After that, reconnection of affected circuits is triggered by the arrival of the first data packets and uses the new routing information to bypass the fault.

3. Motivation.

While fault recovery is important for telecommunications networks and best-effort data networks, it is apparent that in the context of guaranteed performance service communications the need for such schemes is even greater. Since the applications are sensitive to performance degradations, interruption of service is all the more undesirable.

Fault recovery in a guaranteed performance service network is different from that in a conventional data network because the goals are different. Since the data delivery at the network layer of conventional data networks does not have performance bounds, recovery schemes do not have to worry about strictly controlling the resource state within the network. Rather the focus of such schemes is on maintaining the existence of a "good" route between any two points in the network. As a result, only the routing state which governs the selection of future routes is changed. There is usually no other state to change since the networks are typically connectionless. In contrast, the fault recovery in telecommunication networks, especially at the cross-connect layer, shows much of the same requirements as for a guaranteed performance service network. The network state contains information about the bandwidth requirements of each of the call groups, which must be preserved during the recovery. In addition, the call groups themselves follow stable routes, which are changed during recovery if the fault affects the route. The difference lies in the fact that for the cross-connect layer, all the flows have only bandwidth information associated with them, and belong to a few limited classes (such as DS1 or DS3). In a guaranteed service network, the connection's traffic specification and performance requirements cover a much larger number of parameters and these connections cannot be classified into a small number of classes. Thus the techniques used in the telecommunication network such as solving linear programming problems or pre-computing configurations for each network state are not feasible.

Fault recovery can be divided into the tasks of detection, instigation, rerouting and restorative reconfiguration [6]. In our opinion, the most difficult problems of failure recovery occur in the rerouting task. Our work will focus on the rerouting task. Rerouting has two aspects which are sensitive to time, and hence are of interest in the context of guaranteed performance service communication. One is the calculation and dissemination of new routing information which takes the fault into account, so that new and rerouted connections can be routed around the failed component. The other is the initiation of reroutes which move affected connections from the failed component to other routes which can continue to support the connections. The first aspect is very similar to operations that are done in a best-effort data network, or for the switched layer of the telecommunication network, and the reader is referred to [16, 7] for more details. This paper focuses on the second aspect.

A simple way to deal with rerouting is to calculate and disseminate new routing information to guide the routing of new connections, and tear down all the affected connections. The applications would then notice the failure and attempt to reconnect. This reconnection would follow a different route, since the routing information has been updated, and avoid the failure. This solution, while simple and elegant, has the problem of causing human end users to be involved in the recovery action, leading to slow recovery and dissatisfaction. Alternatively the recovery action might be built into the application software, making the application more complex.

A more compelling argument against application initiated recovery is that after a fault all affected applications will attempt rerouting within a small time interval, especially if they have requirements for quick recovery. Thus all affected connections will compete for resources during the rerouting phase, causing *route collisions*[2], which lead to lower success rates. If the reroutes are controlled (in terms of reroute starting time and route selection) such that they prevent competition then there will be higher success rates. This kind of cooperation is not possible at the application level, since network state information is not available to the application. In addition, much advantage can be gained by reusing common portions of the route, since the state information is already present. This network state information is not available at the application level. Over the common portion of the route, we already have resources reserved so that there will be less likelihood of route collisions due to the reroute. In addition, some schemes may make it possible to exchange messages only over the new portion of the route, making rerouting faster. Finally, the lower the level at which rerouting is performed, the faster the response. If all the rerouting action can be performed on the nodes adjacent to the failed component, recovery will be fastest. On the other hand, if the information about the failure has to be presented to the application level, a certain amount of lag cannot be avoided.

A scheme for rerouting connections within the network could also be used to move traffic off a link with high error rate. In this case, when the error rate for a link approaches the error rate bounds specified in the performance requirements of the connection, the connection could be rerouted before the application notices a problem. Only the fault detection mechanism would need to be changed to introduce this service. Thus there are strong reasons to believe that placing recovery intelligence within the network (and making the applications simple and dumb) is the correct design decision.

4. Model.

In this section we present our network model: the model of the guaranteed performance service and the basic scheme for data delivery and resource reservation which makes this service possible; the proposed model of the fault recovery scheme, to add recoverability to such a network; and the underlying support mechanisms that we assume from the network. We make and justify some design decisions in the choice of the fault recovery model, but leave other components open for experimentation.

4.1. The Tenet Scheme

In the Tenet Scheme, a guaranteed performance service connection (*a.k.a. a real-time channel*) is a communication abstraction that defines real-time communication services associated with traffic and performance parameters in a packet-switched network[1]. A channel's real-time traffic is characterized by the

[2] Route collisions are discussed in detail in Section 4.2.

following parameters: $X_{\min}$, the minimum packet inter-arrival time, X_{ave}, the average packet inter-arrival time over an averaging interval, I, the averaging interval, and, $S_{\max}$, the maximum packet size. The performance requirements available to a channel are: D,the maximum delay permissible from the source to the destination, J, the maximum delay jitter[3], Z, the probability that the delay of the packet is smaller than the delay bound, and, W, the buffer overflow probability. A channel is *established* before data transfer. This channel establishment is achieved, in the following manner: a real-time client specifies its traffic characteristics and end-to-end performance requirements to the network; the network determines the most suitable route for a channel with these traffic characteristics and performance requirements; it then translates the end-to-end parameters into local parameters at each node, and attempts to reserves resources at these nodes accordingly. This establishment attempt is accomplished in two passes (i.e. the round trip from the source host to the destination host). Along the forward pass, enough resources are reserved at each local node such that a resource deficiency at a later node along the path can be accommodated. This may cause the resource reservation on the forward pass to be higher than the minimum required to meet the end to end performance requirements. At the destination node, if the resulting end-to-end performance does not meet the requirements, the channel is rejected. Otherwise, along the reverse pass the reserved resources are *relaxed* such that resource reservations reflect the exact traffic and performance requirements of the connection.

The Tenet algorithms or admissions tests, which are based on the service discipline in the nodes and the traffic model, are used to determine whether a node has sufficient resources to accommodate a channel request. During the data transfer phase the local performance requirements of each packet are met using the appropriate scheduling and rate control, thereby satisfying the client specified end-to-end performance guarantees. While many service disciplines (subject to specified constraints [20]) can be employed in the Tenet framework, in this work we have chosen Rate Controlled Static Priority (RCSP) queuing as it decouples the bandwidth and delay resources while providing simplified admissions control tests. An overview of RCSP is presented below. A detailed description can be found in [22].

4.1.1. Rate Controlled Static Priority Queuing and Admission Control

The RCSP service discipline can be thought of as logically consisting of a rate controller and a scheduler. The rate controller takes the input from the network and ensures that all sources obey their traffic description. If at any point a packet violates its traffic description, this packet is considered to have

[3] Jitter is defined as the difference between the delays experienced by any two packets on the same connection.

arrived too early and held until such time that its traffic description is not violated. The packets are then passed on to the scheduler which maintains several levels or priorities and serves packets First-Come-First-Serve (FCFS) within each priority starting at the highest priority level queue at which packets are waiting and ending at the Non-Real-Time queue. Packets in the Non-Real-Time queue are only served after the lowest level RCSP queue's packets have been served. The separation of the rate controller and the scheduler provides the decoupling of the bandwidth and delay resources.

In order to provide guaranteed performance service communication in a network of servers with RCSP scheduling, the following admission control test needs to be performed during channel establishment. For a request with the traffic specification $(X_{\min}, X_{ave,} I, S_{\max})$ and a delay bound requirement D_k (where $D_1, D_2, D_3, .., D_l$ are the delays associated with each of the l priority levels in the switch), if for all priority levels p greater than or equal to k:

$$\sum_{j=1}^{j=c_p} \left\lceil \frac{D_p}{X_{\min_j}} \right\rceil * S_{\max_j} + \left\lceil \frac{D_p}{X_{\min}} \right\rceil * S_{\max} + S_{maxP} \le D_p * L \qquad (1)$$

then the new channel can be accepted (at the priority level k). c_p is the current number of channels at or above the priority level p, $X_{\min_j}$ is the minimum interarrival time of packets corresponding to the jth connection at that priority level, S_{maxP} is the largest packet size that can be transmitted over the link, and L is the link speed.

The test ensures that at any level p, when a packet arrives, the maximum amount of time that it could possibly wait before it can be sent out is bounded by the delay bound D_p corresponding to that level. This bound must take into account any packets at the same level which could be there before the packet arrived, and any packets which could come in at any higher level before it is sent out. Thus any time a channel is added to level k at a node, we have to ensure that for any level $p \ge k$ (i.e equal or lower priority levels), the amount of work (in terms of transmission time of the packet) which can come in during a time period D_p is less than D_p. In other words, adding a channel at level k adds a certain amount of *work* to all equal and lower priority levels $p \ge k$. The admission control tests ensure that the total *work* at any level p does not exceed the delay bound D_p for that level, where *work* at level p is defined as

$$W_p = \sum_{j=1}^{j=c_p} \left\lceil \frac{D_p}{X_{\min_j}} \right\rceil * \frac{S_{\max_j}}{L} + \frac{S_{maxP}}{L} \qquad (2)$$

4.2. The Failure Recovery Model.

Our model of fault recovery is based on our model of the scheme used to provide guaranteed performance service. In this section we first define the set of tasks which need to be performed on the occurrence of the fault. Thereafter we make and justify a number of design decisions which restrict the solution space that we need to search for good failure recovery schemes. This solution space defines our model, since all the possible schemes which can fit this model are candidate solutions which need to be explored. We further restrict our search (for this initial study) by choosing a subset of the components of rerouting that we will examine. We then organize our search of the remaining solution space along the lines of two orthogonal dimensions or components.

As mentioned in Section 3 fault recovery can be divided into the tasks of detection, instigation, rerouting and restorative reconfiguration. In this work we will investigate the task of rerouting which can be further subdivided into *route selection* and *alternate establishment*. The *route selection* requires the most intelligence in the part of the network, since optimal solutions have exponential computational complexity, and all other solutions involve trade-offs between time and goodness of the solution. The process of *alternate establishment*, during which the resource state in the network is changed to reflect these new routes, involves cooperation, since network resources must be shared. The approaches to rerouting can be broadly classified along three axes: *Centralized vs. Distributed schemes*, *Link rerouting vs End-To-End rerouting*, and *Pre-computation vs dynamic computation of routes*. In our environment, i.e. guaranteed performance service connections in integrated high-speed packet-switched networks, we believe our most efficient rerouting approach to be a distributed, dynamic computation approach. The distributed approach is best as it scales easily, does not presume a reliable central control unit or a separate reliable control network, and does not introduce the control traffic congestion and computational bottleneck generally observed at the central controller in centralized schemes. It is our expectation that future workloads will be highly dynamic, especially with the increased presence of multimedia applications, therefore a pre-computed scheme will not be effective. Hence we believe that dynamically computed reroutes are the best approach. We will compare link and end-to-end rerouting approaches in our experiments.

In our model, fault detection is performed by the node upstream of the faulty link. This node reports the failure to all other nodes in the network and instigates recovery of the connections affected by this fault. This instigation takes the form of a reroute message which is disseminated to all nodes affected by the fault and provides them with rerouting information. The timing and contents of the message sent depends on the particular rerouting scheme within this model. The network then performs route computations and attempts to reroute as many of the affected connections as possible, such that all rerouted connections meet their initial traffic and performance requirements.

Within this framework an infinite number of failure recovery schemes are possible. The components of the model which can be varied to construct various recovery schemes include: the method used for fault detection; the routing algorithm used to find a new route for a given channel; the mechanism for cooperation between the various nodes to share the network resources; the timing of the alternate establishment; the constraints on the route selection; and whether retries are performed. For this investigation, we will focus on the components of timing and the constraints of the route selection. We fix or make simplifying assumptions about the other factors. For example we do not perform retries. The assumptions about the fault detection mechanisms and the choice of the routing algorithm are discussed in the next section on underlying mechanisms.

We have chosen to examine rerouting along the lines of two orthogonal components: *the locus of reroute* and, *reroute timing*. The purpose of this work is to determine the effect that each of these components exert on rerouting. Towards that end we have chosen easily analyzed yet broadly encompassing instances of these components with which to examine each subspace. The *locus of reroute* component indicates the segment of the current route traversed by the connection upon which a reroute attempt will be made, and the node which performs reroute selection. Three locus of reroutes types are considered: *global reroutes*, *local reroutes*, and *hybrid reroutes*. With a global reroute, the path of the rerouted connection is recalculated at the source node given the current network load information and the traffic and performance characteristics of the connection. The current load information incorporates the new topology caused by the failed link. Since all available information and possible routes are considered, this scheme would tend to distribute load evenly through out the network. With a local reroute, the rerouted connection traverses the original route with the exception that the failed link is bypassed. The route computation is performed by the node upstream of the failed link, and consists of finding a new path from this node to the node downstream of the failed link. This path is combined with the unaffected portion of the old route, removing any redundant links or loops. This alternate establishment attempt has the advantage of reusing the maximum number of previously reserved resources, and in a highly loaded network may have a higher probability of success than a global reroute. However the route selection selection is from a much smaller set of alternatives, and is less likely to succeed than the global reroute. Of course all of the traffic and performance characteristics of the connection must be maintained after the reroute. With a hybrid reroute, a local reroute is attempted on an affected connection, and if the route selection fails a global reroute is then attempted from the source node. It is expected that this type of reroute should be the most successful of the three as it combines both of the above types. Experiments will test our assertion.

The *reroute timing* component indicates the starting time of the reroute attempt (i.e. the time at which subsequent reroutings begin). As the resource reservation schemes reserve more resources on the forward pass, there is the probability that connections attempting reroutes fail due to *route collisions*. Route collisions occur when an establishment request, during the forward pass along a link, consumes a large fraction of the remaining resources on the link, thus causing an immediately preceding forward pass establishment request to fail due to lack of resources, even though on the reverse pass the initial establishment request would have relaxed its resource reservations such that request immediately preceding could have had its resource request honored. Avoiding route collisions increases the probability of establishing a connection. In addition to route collisions, simultaneous reroute attempts (or attempts within a small time interval with each other) can also interfere due to temporary inconsistencies in the routing data base. Had there been sufficient time between connection reroute attempts the routing database may have reflected a more accurate network load and a more suitable (i.e. having greater probability of success) route would have been chosen. Thus, cooperation between network nodes, to spread the reroute attempts in time and space, would increase the success rates of the alternate establishments.

We have chosen to consider three possible approaches to determining the time of a reroute: *Immediate*, *Random(n)*, and *Sequential*. With the immediate timing approach all of the reroutes attempts are initiated by the controlling node as soon as the failure is known to it. This corresponds to the scenario with the least cooperation and possibly the most expected interference. In the Random(n) approach the reroute attempt time is determined by generating a random value from a uniform distribution over an interval of length n and using this value added to the time of notification as the reroute starting time. This approach does introduce some level of cooperation as all controlling nodes use the same algorithm. In the Sequential approach all reroutes attempts are handled sequentially, with only one controlling node initiating a reroute attempt at any given time. When all the reroute attempts of a single affected connection are completed only then are reroute attempts made on another connection. This method would reduce route collisions to the minimum, maximizing the success rates of the reroute attempts, but at the cost of much higher average time to reroute.

These schemes (i.e. combinations of approaches) are fairly simple and may not translate to good practical implementations. However the objective of our investigation is to look at the significance of the components of the recovery model by choosing approaches which span the entire spectrum of the component. Sequential timing, while it may not be practical from the stand point of the time to reroute, will give us an idea of the best possible performance, in terms of rerouting success, which may be achieved by a perfect timing strategy. It gives us a standard against which we can compare our other more practical timing schemes. In our experiments we will explore the solution space of our recovery model by examining all combinations of the two components.

4.3. Underlying Mechanisms to Support Failure Recovery

There are two mechanisms that are used by the rerouting schemes that we will study. These are part of our overall model of fault recovery and we do not change them in our experiments. These mechanisms are a *fault detection and control mechanism* and a *routing mechanism*. The fault detection and control mechanism determines if a fault has occurred and sends a fault message to the source nodes of the affected connections. The process of fault detection is fairly complex and we do not investigate this further in this paper. We assume that when a link is characterized as faulty, some low level validation has been performed to eliminate transient faults and oscillations. The fault message contains information about the topology change, the resource state on the affected components, and other information which depends on the particular scheme.

The routing mechanism is based on the DCM Routing Algorithm [21] and determines a route from the source node to the destination node taking into consideration the traffic and performance characteristics of the connection, the current network load (including the failed link or links), and the locus of control. The DCM Routing Algorithm provides source routing and is achieved by using a modified, constrained, version of the Bellman-Ford algorithm. The goals of the routing algorithm were to maximize throughput, balance the network load, obtain routes in a timely manner, and to maximize the probability that the route provided will be successfully established (i.e. the route will be established with the traffic and performance specifications given by the client). The routing algorithm calculates a minimal-cost route, in accordance with the criteria presented above, where the cost of the route is the sum of the costs of the links comprising the route. The cost of a link is a delay value, which is the sum of the minimum queuing delay offered by the starting node to a real-time channel with these traffic characteristics, the transmission delay, and the propagation delay along the output link. While the transmission and propagation delay and are fixed costs, the queuing delay experienced in the RCSP scheduler is variable, and is dependent on the current channel resource reservations on the corresponding output link and the traffic characteristics of this new channel. An overview of the equation used to obtain this variable value was presented in Section 4.2 above.

The DCM routing algorithm proceeds in two steps. In the first step a directed graph is created in which the nodes correspond to switches and hosts in the network and the edges to the links connecting these switches and hosts. The weights attributed to each edge represent the link costs. The link cost are computed just prior to applying the algorithm thereby using the most recent link information obtained from routing update messages. In the second step a constrained, modified version of the Bellman-Ford[4] algorithm is then applied to this graph to determine a possible route. In this algorithm consecutive searches are

[4] The fundamental Bellman-Ford algorithm [22] searches for the shortest paths between a specified source and destination node starting from all possible one-hop paths and continuing until the N-2-hop paths are examined.

performed on all 1, 2, .., N-2-hop paths from the source to the destination node until the delay condition $\sum_{l(s,d)} d_l \le D$ is satisfied, where D is the delay bound of the channel, d_l is the weight of link l, (s, d) is the path from the source s to the destination d, and N is the number of nodes in the network. A *constraint* is placed on the number of possible searches by stopping at the first *hoplevel*[5] at which the delay bound condition is satisfied, and choosing the minimum cost path within the same level.

The DCM routing algorithm seeks to maximize throughput by minimizing the number of intermediate nodes encountered along the path from the source to the destination host. By minimizing the hop count, there will be less contention for resources among channel requests which will result in a corresponding rise in throughput. The load balancing criterion also reduces call blocking by distributing the load more evenly throughout the network. The algorithm limits its search space, thus reducing its computation time, thereby providing routes in a timely manner. It also increases the probability that channel establishment will be successful as it determines the link queuing delays based on the traffic characteristics of the channel and the most recent resource reservation information.

Routing updates are currently done on a per-channel-establishment basis. This is accomplished by having every node broadcast the load values of its links to all other nodes after it sends the reverse channel establishment message to the previous node on the new channels route; each receiving node updates its local link-state table. This broadcast is done along a minimum spanning tree.

5. Experimental Design

We propose to explore rerouting along the lines of the components of *locus of reroute*, and *reroute timing* as defined in the previous section. In this section we will describe our experimental framework by presenting a network load index which we use both to characterize the traffic on the network and to compare the success rate of different schemes; a set of performance criteria or metrics which we use to gauge the performance of our schemes; and the other factors (such as network topology and workload) which need to be fixed in order to compare our rerouting schemes under identical conditions.

5.1. Load Index

One of the problems facing any investigation which compares schemes involving guaranteed performance connections is the one of choosing an index which can characterize the load imposed on the network by a single or any number of guaranteed performance connections. This is important, for instance, in

[5] This *hoplevel* is the number of hops from the source to the destination node.

studying the effect of network load on the behavior of the rerouting schemes. This also affects how one measures the success of reroute actions. To quantify the amount of traffic successfully rerouted one must take the number of channels, their bandwidth specifications, their delay requirements, and the number of links affected affected by each of these channels into account. A general solution to this problem would be quite complex, however, we have devised a simple index which meets the above requirements within the context of the RCSP (Rate Controlled Static Priority) scheduling discipline.

The description of the RCSP admission control scheme, in section 4.1, motivates the choice of the load index in a network using this scheduling strategy at all nodes. Introducing a channel into the network at level *l* adds work to all lower levels. This is why a low delay (high priority) channel consumes more resources in the network, since it uses a high-priority level (small *l*) and it increases call blocking probability at all levels $m > l$. If the delay requirement is larger, it goes into a lower priority level (larger *l*) and effects fewer levels. This prompts us to consider the sum of the *work* (as specified by equation 2) across all levels as an index of the load on the node. A high priority channel will cause a big change in this index as it will increase the *work* at all levels, while a channel at the lowest priority level will only effect one level. We call this index of the load on the network the Queuing Delay Index.

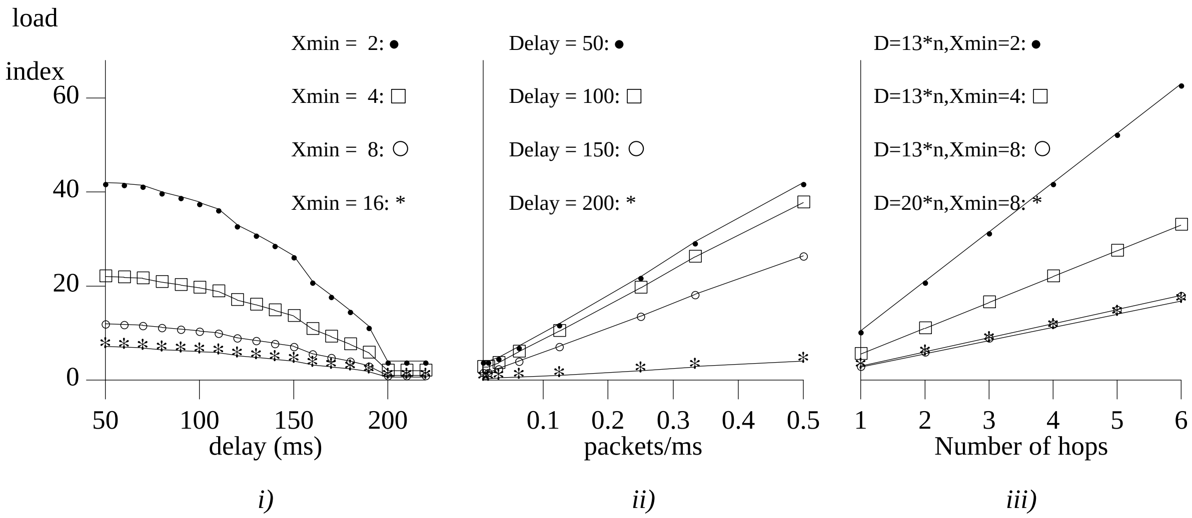

Figure 1. Queueing Delay index vs. other load indices: Graph *i)* shows that the proposed load index is monotonically decreasing as a function of delay if all other factors are constant. Graph *ii)* shows that it is linear in bandwidth and graph *iii)* shows that it is linear in the path length of the route provided the delay requirement per hop remains constant.

Figure 1 shows how successful this index is at capturing bandwidth, delay and path-length as components in the overall load on the network. Graph *i)* shows how the index changes as a function of the delay requirement of a channel on an otherwise empty network. We can see that as the delay requirement becomes slacker (higher end-to-end delay bounds) the load index decreases. The graph flattens out beyond

200ms because at this point the channel is at the lowest level in the RCSP queues and further increases in the delay bound do not effect the priority level of the channel. Thus, bandwidth and path-length remaining the same, the index is monotonically decreasing as a function of the end to end delay bound. Graph *ii)* shows that the load index is linear in the bandwidth of the traffic, with the slope of the line dependent on the delay bound of the channel. Graph *iii)* show that the index increases linearly with the path length, as long as the delay bound per link remains constant. We believe that this index is a good characterization of the overall load on the network. We will use this index to characterize the load on the network in order to study how the behavior of the schemes change as a function of the network load.

This index also makes it possible to compare channels which have different bandwidth, delay and path lengths. We define the Empty Network Delay index (END index) of a channel as the load index measured by establishing the channel in an otherwise empty network. By measuring this in an empty network we eliminate second order effects such as interference with other channels which might cause the channel under consideration to take a longer path. Under the assumption that the routing scheme always picks the shortest available path and the admission scheme always puts the channel in the lowest RCSP queue level which would suffice, this is the smallest increase in the overall load index which can be caused by setting up this channel successfully under any condition. We will use the END index to compare schemes which successfully reroute different sets of channels under identical conditions, by comparing the sum of the END indices of one set against the other.

5.2. Metrics

The metrics by which we compare the performance of the rerouting schemes is significant as it must provide us with a useful assessment of the scheme. To this end our criteria is based on the effect to both the client and the network. The metrics are: the rerouting success ratio; the average and maximum time to reroute a connection; the average and maximum packets lost before reroute completion; and the excess load consumed by the reroute.

The fraction of the affected traffic which could be successfully restored by the rerouting scheme is important to both the network and the client. This is measured in our simulations by summing the END indices (defined above) of the successfully rerouted channels, and expressing this value as a percentage of the total affected traffic. This *success ratio* allows us to compare channels with different bandwidth, delay, and path length on a single metric. Looking at the ratio rather than the absolute value of the amount of traffic successfully rerouted is also important because it makes our graphs independent of the load. The effect on the client is captured by the average and maximum time to recover the affected connection of a client (provided it can be rerouted), as well as the average and maximum number of packets lost before reroute completion. This is measured in our simulation over the set of successfully rerouted channels, by keeping a record of the simulation time at the time of the fault and the finish of each channel reroute. We also record the number of packets lost over the lifetime of the connection.

The effect on the network is captured by the excess of resources that are consumed by the successfully rerouted channels, as compared to the minimum amount of resource required to support this set of channels. The actual resources consumed by the successfully rerouted channels is measured in the simulation by calculating the network load index (as defined in Section 4.4.) once after the rerouting is complete and again after tearing down the set of successfully rerouted channels, and taking the difference between the two values. The minimum resources required to support this set of channels can be approximated by the sum of the END indices, since the END index of a channel represents the amount of resources required if the routing and relaxation is not constrained by the presence of other channels. The excess of resources can thus be expressed as a percentage by $\frac{(required - minimum)}{minimum} * 100$.

5.3. Other Factors

The factors that we considered significant in our experiment design were the network topology, the routing scheme (i.e. the routing algorithm and the routing update mechanism), the workload, the rerouting scheme, the number of retries permitted on an affected connection, and the fault model. In the experiment set discussed in this paper we fixed the network topology, the routing scheme, the number of retries, and the fault model. The network topology used was that of a simple 5x5 square *mesh* as it is allows us to manually verify results fairly easily, while at the same time providing a diverse number of possible routes between any host pairs. The routing scheme used was described in Section 4.2. In these experiments no retries were permitted since retries would interact with the different rerouting schemes and prevent us from isolating the effects of one from the other. A single link fault was allowed during each recovery cycle, which is the time interval between the occurrence of a fault and the complete reconfiguration of the network. This complete reconfiguration includes the reroute of all possible affected connections and the reconfiguration of the routing databases of all nodes in the network. In other subsequent experiments (to be published in another paper) the number of retries and the fault model will also be varied.

In the experiments described in this paper the workload and rerouting schemes are the only factors that we vary. This allows us to look at the effects of these factors without allowing the other factors to influence these effects. The workload is composed of the set of channels which were established prior to the fault in the network. To study how the performance of the rerouting schemes changes over load we ran our simulations at four different workload levels. In terms of the network load index defined in Section 4.3, the load levels were *low* = 270, *medium* = 620, *medium-high* = 790, and *high* = 858. To give the reader an idea of the corresponding bandwidth utilization, sample values from our experiments are presented here. In one set of workloads the *low* workload level corresponded to a 15.8% average bandwidth utilization and a 63.3% bandwidth utilization (maximum) on the heaviest loaded link; *medium* corresponded to a 32.0% average and a 84.7% maximum; *medium-high* corresponded to a 41.2% average and a 87.3% maximum; and *high* corresponded to a 44.5% average and a 86.7% maximum; The numbers shown above correspond to only one of a number of workloads which we generated and used for our simulations at each workload level. It should also be stressed that the network load index captures more than the bandwidth information, and a set of channels with identical bandwidth requirements but different delay requirements would give us different load index values.

The workloads themselves were statistically generated with the following characteristics. The channels in the workload come from three classes, with bandwidth requirements corresponding to *A)* a one way low quality video conference channel, *B)* a CD quality audio channel, *C)* and a telephone quality audio channel. The distribution of these is chosen so that 30% of the channels belong to class *A*, 30% to class *B*, the the rest to class *C*. The host pairs were selected randomly to lie along the periphery of the mesh. The delay requirements of the channels were also generated statistically, to lie uniformly in the range $[x+50, x+100]$ ms, where x is the one way propagation delay between the source and the destination of the channel. Thereafter different numbers of channels were generated with the above statistical properties to get different values of the workload index. The specific values of the workload index we use were chosen by studying the main criteria (routing success ratio) across a much more densely chosen set of workload points and looking for regions in which the behavior was relatively stable. The workload points chosen represent the midpoints of these stable regions. Thereafter a number of experiments were performed at workloads at or very close to each of the four values shown, and their results averaged (or the maximum across the set taken in the case of metrics such as max time to reroute) to increase confidence in our experiments. We did not calculate the statistical confidence because we use these results for rough comparisons and not to provide precise numbers. Thus we merely reassured ourselves that the variation between experiments at the same workload level were small compared to the differences which we use to draw conclusions.

The other factor we vary in our experiments is the rerouting schemes. We have already described the two orthogonal dimensions along which we plan to explore the scheme space. Within the Random(n) approach we look at three randomization intervals: 300, 1500, and 3000 ms. These three values were chosen from a larger set of randomization intervals with which we experimented. They span the range over which such a technique is reasonable in terms of the time to reroute, and are also fairly representative of the results we saw from other values of the randomization interval.

6. Results and Analysis.

The results of our experiments are shown in Figures 2 - 5 as bar charts. Each figure contains eight graphs showing a particular measure of the performance of the rerouting schemes across four different values of network load. In each set, graphs *i) -v)* keep the timing component fixed within each graph, and compare the different locus of rerouting approaches across different load values. For example, graph *i)* of each set keeps the timing fixed at Immediate, and shows the Global, Local, and Hybrid locus of rerouting approaches across different network loads. In graphs *vi) - viii)* the locus of rerouting is fixed within each graph and the different timing approaches can be compared. They are: Immediate timing, Sequential timing, and Random(n) timing over three randomization intervals. In the graphs R_1, R_2, and R_3 stand for Random over 300, 1500, and 3000 ms respectively.

Figure 2 shows the rerouting success ratio (SR) across the set of workload indices. Graphs *i) - v)* each keep the timing component fixed to allow us to compare the locus of rerouting approaches. Global and Hybrid rerouting seem to be comparable. Local rerouting reroutes far fewer channels at all load levels. Further the performance of Local rerouting deteriorates more rapidly with increasing load. The reason is that Local rerouting considers a far fewer set of routes during route selection. Graphs *vi) - viii)* examine the effect of the timing component, keeping the locus of rerouting component constant. As expected, Sequential performs the best in terms of the number of rerouted channels. Immediate tends to perform the worst since we have the most route collisions when all channels are trying to reroute simultaneously. We see that increasing the interval over which the reroutes happen improves the success rate even with random timing. From this we conclude that a more intelligent mechanism for preventing collisions by spacing out reroutes based on common links would perform even better. Sequential represents in a sense the best performance which can be hoped for after eliminating all collisions, i.e. with perfect cooperation between all nodes.

The next set of graphs (Figure 3) shows the time to reroute for different work loads. The lower (solid) bar shows the average time to reroute over all successful channels, the higher (dashed) bar shows the maximum. As before graphs *i) - v)* fix the timing component. We see that in this respect, the locus of rerouting approaches are all comparable. This is because the major cost of the rerouting comes from the round trip for channel establishment, which is the same across the three approaches. The maximum time to reroute is about 150 ms for Immediate timing across all locus of reroute approaches, which is close to the round trip time between the farthest set of host pairs in our load model. For *Random* (n) timing the maximum time to reroute is about n+150 ms, which is as expected since the last reroute attempted would start near the end of n ms and would take about 150 ms to complete. The maximum time with Sequential timing changes with the load. Initially, it increases with load since there are more channels affected and they are tried in sequence. But at high load it drops off since less channels can be successfully rerouted, and the maximum and average times are computed over the set of successful channels. Since the later ones are more likely to fail (as the earlier reroutes consume the remaining resources) the maximum time to reroute (corresponding to the last successfully rerouted channel) is lower for load=858 than for load=790. Definitely at this point the network is approaching saturation, in terms of the amount of guaranteed performance connections that can be supported. With the exception of Immediate, the average time to reroute is about half the maximum since the reroutes occur uniformly over the interval considered. For Immediate the distribution of path lengths determines where the average is compared to the maximum. If we consider the data from the other perspective, i.e. we keep the locus of reroute constant and compare the different timing approaches (see graphs *vi) - viii)*) we note significant differences. As expected Immediate is the fastest, faster than Sequential by an order of magnitude. This is the reason why Sequential is only an academic ideal to compare the success of reroutes, and not a practical scheme.

Figure 4 shows the packet loss in the same format as in Figure 3. The shape of the graphs is similar to that of the previous set. This is because the number of packets lost depends on the time taken to reroute. However the number of packets lost for Local at higher loads, seems small compared to the time to reroute. This is because at that high load only the low resource connections were rerouted. This again points to the inability of Local rerouting to deal with high load conditions.

The last set of graphs (Figure 5) show the excess resources (as described in Section 5.2) consumed by the rerouted requests across different network loads. These graphs are best interpreted in conjunction with Figure 2 where the success of the reroute in terms of the amount of work rerouted are plotted. Without this additional information we might look at the smaller excess resources consumed by Local (graphs *i) - v)*) at load=858 and conclude that it is more efficient than Global at high load. However we note from Figure 2 that Local successfully rerouted much less work so that it is not unreasonable that it was more efficient about it. In other words we can *only* call a approach more efficient on the basis of a lower value of excess resources, *if* it is also comparable or better in rerouting success (Figure 2). Thus we can conclude that while its rerouting success is comparable to that of Global, Local has less efficiency in terms of excess resources used. At high loads we have already seen it to have a poorer rerouting success. Thus we can say that it is consistently worse than Global rerouting. We can also compare Global and Hybrid rerouting, since their success rates are similar throughout all ranges of loads. We note that while at low loads the efficiency of Hybrid is less (because of the longer routes used by the local reroutes), at high loads its use of resources is better than Global. Since its success ratio at high loads is equal, this gives us a slight reason to believe that Hybrid offers some advantage over Global. However the evidence is not really conclusive on either side. Looking at the data from the perspective of graphs *vi) - viii)* does not give us much insight. We note that at high loads Sequential, while rerouting more work than the other approaches, still shows higher efficiency. Thus increasing the level of cooperation pays off not only in terms of a larger amount of traffic rerouted, but also because it uses the network resources less wastefully. This encourages us to search for good cooperation schemes which are faster than Sequential but give us the same benefits. We believe that more intelligent schemes which trade off time to reroute for higher success rates in more sophisticated ways than Random(n) will pay off.

7. Conclusion.

In this paper we investigated approaches for the rerouting of guaranteed performance connections affected by faults in the network. In addition to avoiding the faulty link, rerouting must ensure that the traffic and performance guarantees made along the previous routes will be satisfied along the new routes. The goal of a rerouting scheme is to reroute as much of the resources (i.e the resource reservations affected by the fault) as possible, as quickly as possible, and minimize the effect of the fault on the client and the network. To this end we choose criteria, or metrics of success, that reflect these goals. From the client perspective, the criteria of interest were the average and maximum time taken to establish the successfully rerouted connections, and the average and maximum number of packets lost during rerouting. From the network perspective, the criteria of interest were the amount of affected traffic that was recovered by the scheme, the success ratio (SR), and the amount of network resources which was wasted in the process. To measure these resources we used a *Queuing Delay Load* index that captures both the bandwidth and delay resources reserved by a connection. The flattening of these two resources along one dimension allows us to better compare schemes.

We examined rerouting along two orthogonal components; the *locus of reroute* and the *timing* component. The locus of reroute determines the node which selects the new route and the constraints under which this route is selected, and the timing component determines the time at which the recovery attempt should begin. There are three locus of reroute approaches; Local, Hybrid, and Global. There are also three different reroute timings approaches: Immediate, Random and Sequential. The cross product of the approaches in the two components span the complete rerouting scheme space.

All rerouting schemes were examined across a cross-section of workloads and the approaches were compared along the lines of the criteria mentioned above. These comparisons were achieved by using extensive simulation experiments. With our assumptions of topology and experimental configuration, our analysis shows that the Global and Hybrid reroute type provide comparable success ratios (SR), across all loads, with Local providing the worst. Local also performs worse in terms of efficient use of resources. Global seems to use resources more efficiently than Hybrid at low loads, but Hybrid matches and surpasses Global at very high loads. Sequential timing provides the best success ratio (SR) across all loads, as well as the most efficient usage of network resources. However Sequential is at most a theoretical "best" method, because the average and maximum time to reroute is an order of magnitude higher than Immediate. We conclude that more intelligent approaches for cooperation would provide significant gains over Immediate (i.e. no cooperation). Even very simple randomization approaches can provide significant gains over approaches with no cooperation, but more intelligent approaches should be investigated since the theoretical best is still well beyond what can be achieved by randomization within reasonable intervals.

Global, Local and Hybrid perform comparably in terms of time to reroute and packets lost during rerouting. Using Immediate timing they take about 150 ms at most to reroute. This, of course, is depends on the topology, link speeds and distribution of channel path lengths. In general, the fastest maximum time to reroute is slightly larger than the round trip time. Cooperation between nodes increase the maximum and average time to reroute and also the number of packets lost. Sequential rerouting is too slow and expensive in terms of lost packets to be used in practice. Randomization over a small interval takes a maximum of the interval plus one round trip time to complete.

There are many other areas of investigation to be explored in rerouting. Currently we are investigating the behavior of our approaches in the presence of multiple faults within a recovery cycle, the effect of multiple retries on the approaches, and the effect of different routing algorithms on the approaches.

Bibliography

[1] Domenico Ferrari, Anindo Banerjea, and Hui Zhang, ‘‘Network Support for Multimedia: A discussion of the tenet approach’’, Technical Report No. TR-92-072, International Computer Science Institute, Berkeley, CA, November 1992.

[2] David Clark, Scott Shenker, and Lixia Zhang, ‘‘Supporting Real-Time applications in an Integrated Services Packet Network: Architecture and Mechanisms’’, In *Proceedings of ACM SIGCOMM’92*, pages 14-26, Baltimore, Maryland, August 1992.

[3] David P. Anderson, Ralf Guido Herrtwich, and Carl Schaeffer, ‘‘SRP: A Resource Reservation Protocol for Guaranteed Performance Communication in Internet.’’, Technical Report TR-90-006, International Computer Science Institute, Berkeley, CA, February 1990.

[4] J. Hyman and A. Lazar, ‘‘MARS: The Magnet II Real-Time Scheduling Algorithm’’, In *Proceedings of ACM SIGCOMM’91*, pages 285-293, Zurich Switzerland, September 1991.

[5] C. Vogt, R. Herrtwich, and R. Nagaragan, ‘‘HeiRAT - The Heidelberg Resource and Administration Technique: Design Philosophy and Goals’’, IBM Technical Report No. 43.9243, IBM ENC, Heidelberg, 1992.

[6] Robert Doverspike, ‘‘A Multi-Layered Model for Survivability in Intra-LATA Transport Networks’’, In *Proceedings of IEEE GLOBECOM ’91*, Phoenix, Arizona, pp. 2025-2031, December 1991.

[7] K.R. Krishnan and T.J. Ott, ‘‘Forward Looking Routing: A New State Dependent Routing Scheme’’, In *Proceedings of the 12th International Teletraffic Congress* (ITC), Torino, Italy, June 1988.

[8] V.P. Chaudhary, K.R. Krishnan, and C.D. Pack, ‘‘Implementing Dynamic Routing in the Local Telephone Networks of USA’’, Teletraffic and Datatraffic in a Period of Change, ITC-13, A. Jensen and V.B. Iversen (Editors), Elseviers Science Publishers B.V. (North Holland).

[9] Brian A. Coan, Will E. Leland, Mario P. Vecchi, Abel Weinrib, and Liang T. Wu, ‘‘Using Distributed Topology Update and Preplanned Configuration to Achieve Trunk Network Survivability’’, In *IEEE Transactions on Reliability*, Volume 49, Number 4, October 1991.

[10] M. Barezzani, E. Pedrinelli, and M. Gerla, ‘‘Protection Planning in Transmission Networks’’, In *Proceedings of SUPERCOMM / International Conference on Communications ’92*, Chicago, June 1992.

[11] Makiko Yoshida and Hiroyuki Okazaki, ‘‘Cooperative control over logical and physical networks for multiservice environments’’, In *IEICE Transactions*, Vol.E.74, No.12. December 1991.

[12] W.D. Grover, ‘‘The Self-Healing Network: A Fast Distributed Restoration Technique for Networks using Digital Cross-connect Machines’’, In *Proceedings of IEEE Global Telecommunications Conference*, pp. 28.2.1-28.2.6, December 1987.

[13] C. Han Yang and S. Hasegawa, ‘‘FITNESS: Failure Immunization Technology for Network Service Survivability’’, In *Proceedings of IEEE Global Telecommunications Conference*, pp. 47.3.1-47.3.6, November/December 1988.

[14] H. Komine, T. Chuja, T. Ogura, K. Miyazaki, and T. Soejima, ‘‘A Distributed Restoration Algorithm for Multiple Link and Node Failures of Transport Networks’’, In *Proceedings of IEEE Global telecommunications Conference*, pp. 403.4.1-403.4.5, December 1990.

[15] H. Sakauchi, Y. Nishimura, and S. Hasegawa, ‘‘A Self-healing Network with an Economic Spare-Channel Assignment’’, In *Proceedings of IEEE Global Telecommunications Conference*, pp. 403.1.1-403.1.6, December 1990.

[16] J. McQuillan et al., ‘‘The New Routing Algorithm for the ARPANET’’, In *IEEE Transactions on Communications*, Vol. COM-28, May 1980.

[17] William T.Zaumen and J.J. Garcia-Luna-Aceves, ‘‘Dymamics of Distributed Shortest-Path Routing Algorithms’’, In *Proceedings of the 21st ACM SIGCOMM Conference’91*, Zurich, Switzerland, September 3-6, 1991, *Computer Communications Review*, Vol. 21, No. 4, ACM Press.

[18] J.J. Garcia-Luna-Aceves, ‘‘A Unified Approach for Loop-Free Routing Using Link States or Distance Vectors’’, In *ACM Computer Communication Review*, Vol. 19, No. 4, pp. 212-223, 1989.

[19] M. Schroeder, A. Birell, M. Burrows, H. Murray, R. Needham, T. Rodeheffer, E. Satterthwaite, and C. Thacker, , ‘‘Autonet: A High-Speed Self-Configuring Local Area Network Using Point-To-Point Links’’, *IEEE Journal Selected Areas in Communication*, Vol. 9, No. 8, pp. 1318-1335, October 1991.

[20] Domenico Ferrari, ‘‘Real-time Communication in an Internetwork’’ *Journal of High Speed Networks*, Vol. 1, No. 1, pp. 79-103.

[21] Colin Parris and Domenico Ferrari, ‘‘A Dynamic Connection Management Scheme for Guaranteed Performance Services in Packet-Switching Integrated Services Networks’’, Technical Report TR-93-005, Jan 1993, International Computer Science Institute, Berkeley, CA.

[22] Hui Zhang and Domenico Ferrari, ‘‘Rate-Controlled Static Priority Queueing’’, In *Proceeding of the IEEE INFOCOM’93*, April 1993, San Francisco, CA.

[22] R. Bellman, ‘‘On a routing problem’’, In *Quarterly of Applied Mathematics*, Vol. 16, 1958, pp. 87-90.

[23] Domenico Ferrari, ‘‘Client requirements for real-time communication services’’, In *IEEE Communications Magazine*, Vol. 28, No. 4, pp. 65-72, November 1990.

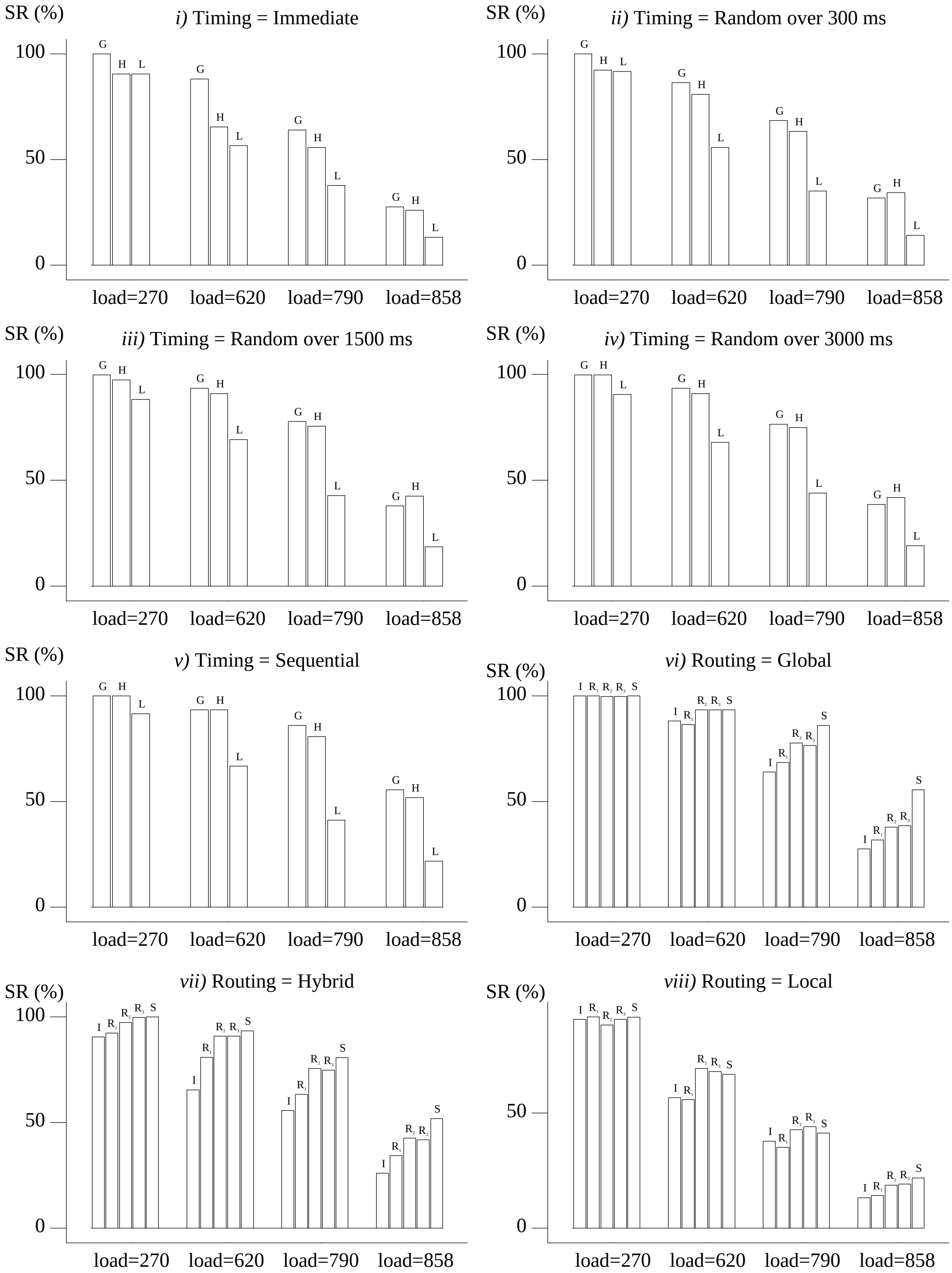

Figure 2. Routing success ratio vs. load : The sum of the END indices of the successfully rerouted channels expressed as a percentage of the sum of the END indices of the affected channels is used as a measure of the success of rerouting. Graphs *i) - v)* compare the routing approaches keeping the timing component constant. Graphs *vi) - viii)* compare the different timing approaches keeping routing constant.

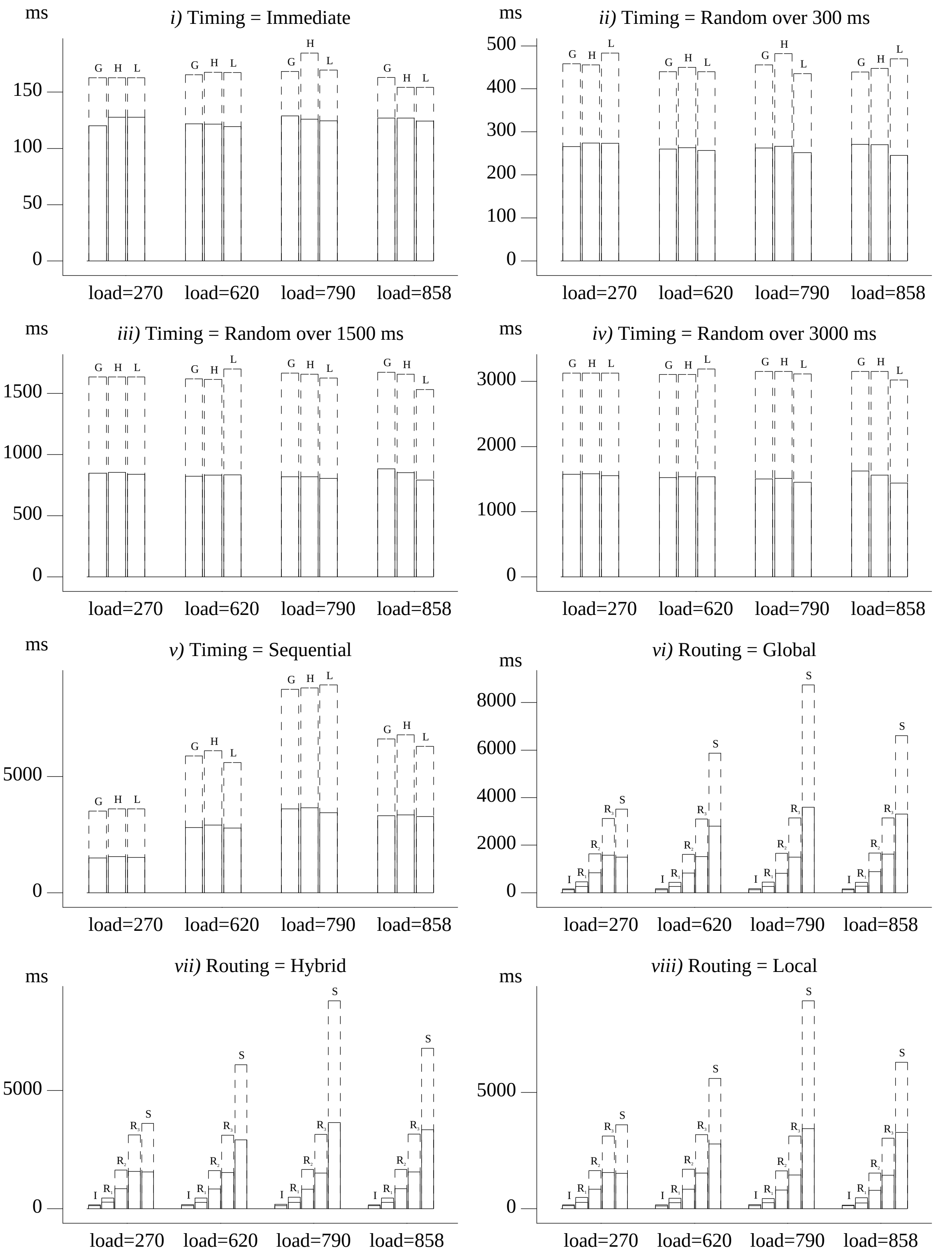

Figure 3. Time to reroute vs. load : The lower solid box shows the average time to reroute, the higher dashed box show the maximum time to reroute over all successfully rerouted channels. Graphs *i) - v)* compare the routing approaches keeping the timing component constant. Graphs *vi) - viii)* compare the different timing approaches keeping routing constant.

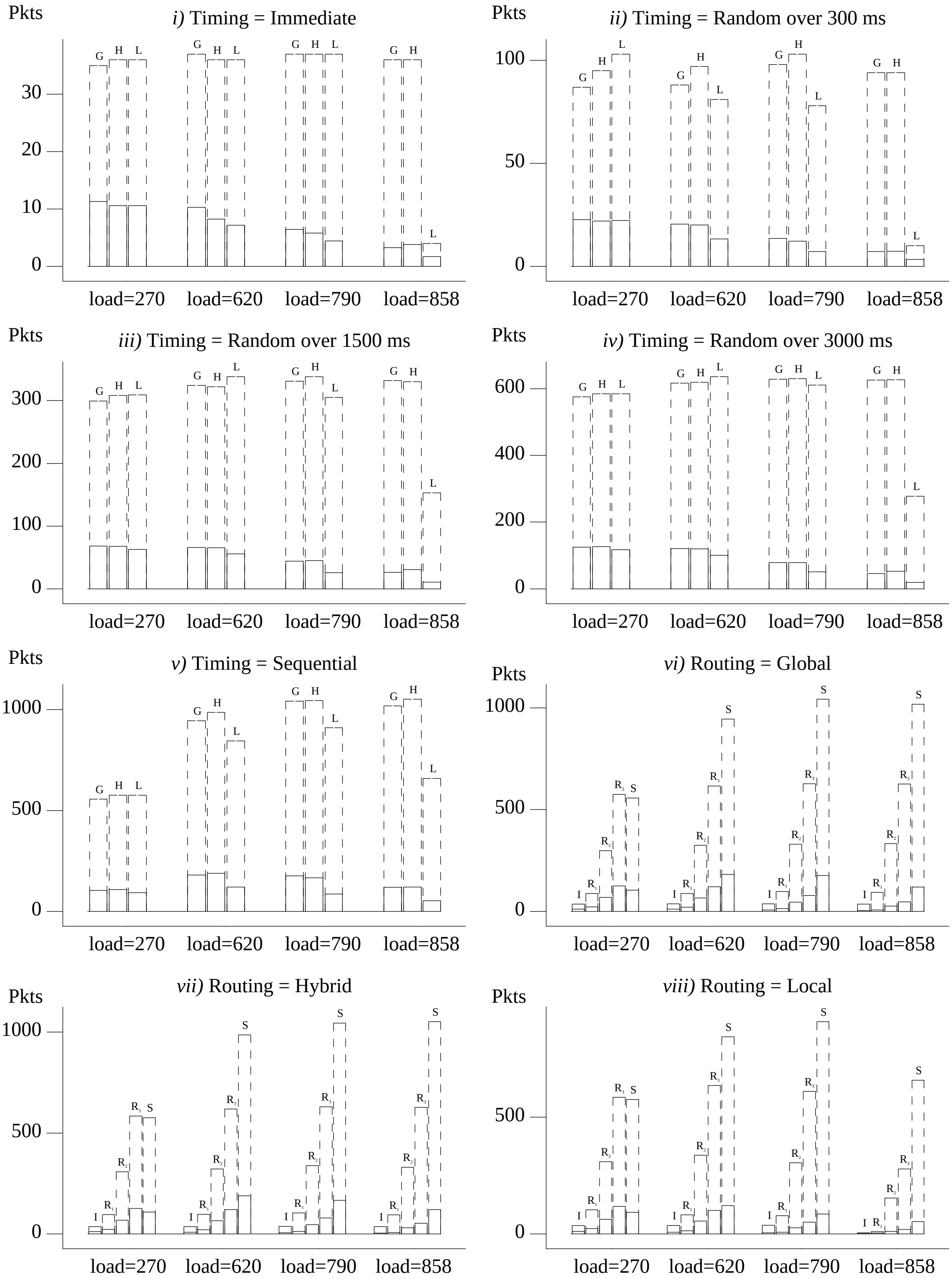

Figure 4. Packet loss vs. load : The lower solid box shows the average packet loss, the higher dashed box show the maximum packet loss over all successfully rerouted channels. Graphs *i) - v)* compare the routing approaches keeping the timing component constant. Graphs *vi) - viii)* compare the different timing approaches keeping routing constant.

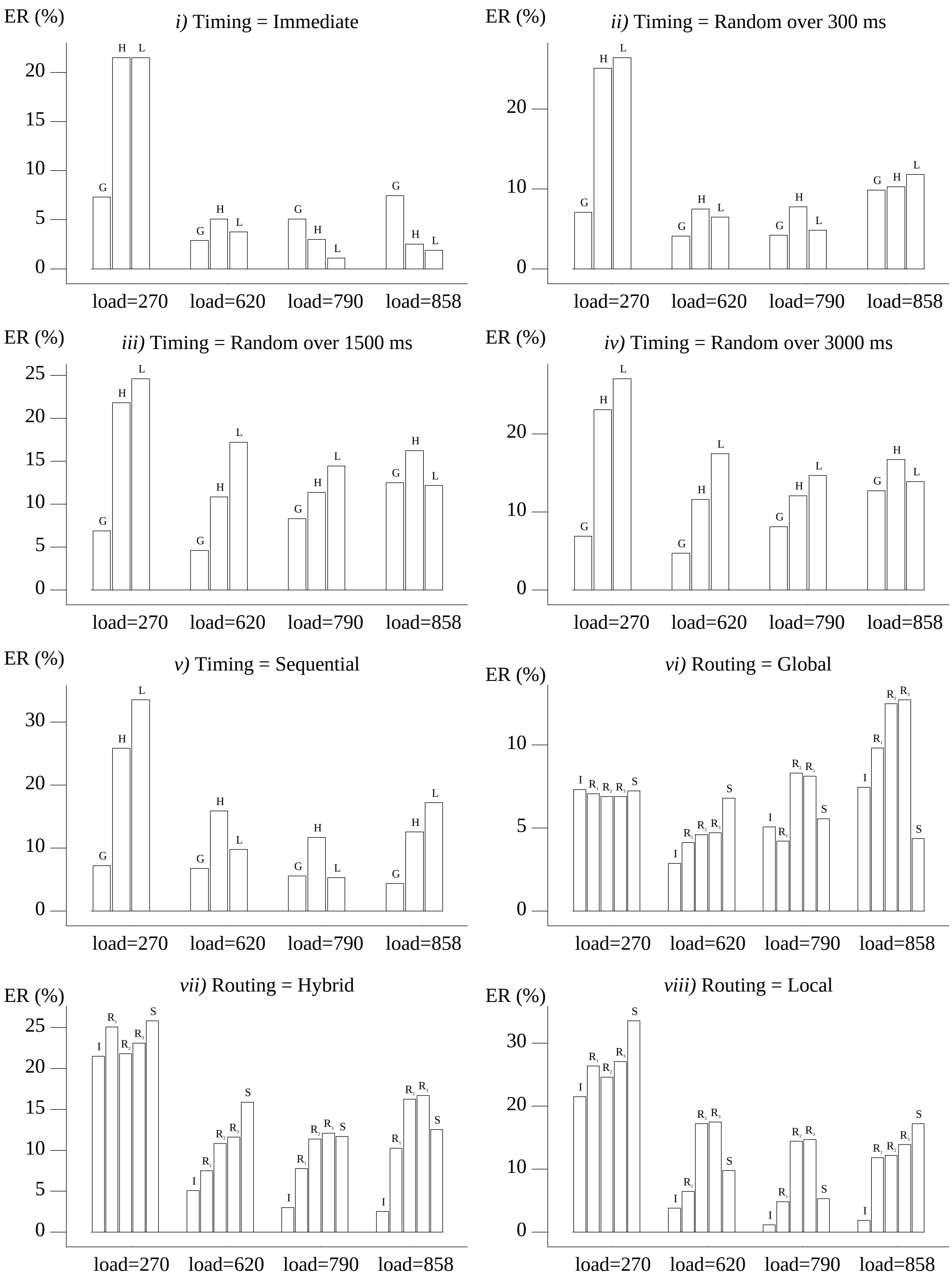

Figure 5. Excess resources vs. load : The amount of network load attributable to the successfully rerouted channels is expressed as a percentage difference from the sum of END indices of the successfully rerouted channels. Graphs *i) - v)* compare the routing approaches keeping the timing component constant. Graphs *vi) - viii)* compare the different timing approaches keeping routing constant.