Setup-Trail-Avoidance Routing Algorithm

Matthew T. Busche
Lucent Technologies
11900 North Pecos Street, Room 32Z76
Westminster, CO 80234
busche@lucent.com

Christopher J. Olszewski
AT&T
101 Crawfords Corner Road, Room 4L-403
Holmdel, NJ 07733
colszewski@att.com

Abstract

The setup-trail-avoidance routing (STAR) algorithm is a new simple distributed routing algorithm for connection-oriented mesh networks. In this algorithm, signaling messages used to set up a service (or end-to-end connection) contain a list of nodes, called the setup trail, that have participated in routing the service to its destination. A node receiving such a signaling message will then calculate a shortest-path route to the service's destination using a network topology specific to the type of service, after first removing from the topology other nodes in the service's setup trail. STAR adapts to changing network conditions, choosing efficient routes for services (e.g., asynchronous transfer mode [ATM] virtual paths) while at the same time quickly terminating service setup requests that have little or no chance of being successfully completed. Simulated restoration studies show STAR has the best performance characteristics among many distributed restoration algorithms.

1. Introduction

1.1. Characteristics of the algorithm

The setup-trail-avoidance routing (STAR) algorithm is a simple distributed routing algorithm developed for connection-oriented networks. This paper describes the algorithm and presents simulated restoration results. Comparisons with other distributed routing algorithms show that STAR has the best performance of the algorithms compared.

In the STAR algorithm, signaling messages used to set up a service are sent, node by node, along the intended restoration path of the connection. These messages contain a list of nodes, called the setup trail, that have already been requested to help set up the service. In choosing the next node on the route, a node removes from its map of the network topology (which may be specific to the type of service being set up) nodes in the setup trail other than itself. A shortest-path-pair calculation then determines the next node to forward the service setup request.

STAR could be used for initial service setup, optimization of existing service routes or restoration of failed services. STAR exhibits the greatest performance advantage in networks stressed by load or damage. In this paper, we show that STAR provides an extremely fast and efficient way to reestablish services in a restoration environment. Several investigations have been made into fast distributed restoration [1-10]. This topic becomes increasingly important with the continued deployment of high-capacity switching and transmission equipment, and the concomitant need for speedy restoration [11].

STAR-based restoration procedures are distributed among the nodes in the network. In contrast, restoration in a centralized system is directed by a central restoration controller. A centralized system should achieve a higher restoration efficiency, but would be slower and not as robust as a distributed system [12].

STAR makes routing decisions based on knowledge of a network's topology. Some recent innovative distributed restoration techniques do not rely at all on this kind of knowledge [1-7]. In these flood-search algorithms, setup requests are sent in all available directions. STAR uses a network topology to choose the single best route on which to forward a service setup request, resulting in much lighter signaling loads that often make STAR-based restoration faster. The particular mechanism for distributing topology information is not critical: for example, the topology state information proposed for the ATM private network-to-network interface (PNNI) would suffice [13].

STAR can be used with standard connection control protocols like the narrowband integrated services digital network (ISDN) user part (N-ISUP) [14] or the broadband ISUP (B-ISUP) [15]. These protocols would, however, need extensions to include the setup trail as an information

0-8186-7453-9/96 $05.00

element in some message types. In contrast, flood-search algorithms typically need a specialized signaling protocol to allow one incoming setup request message to spawn multiple outgoing messages, and to clear those branches of the routing tree that are no longer needed [1-7].

STAR is used to effect path-based restoration wherein a new end-to-end path is found from the origination node of a failed service to its destination node. Path-based restoration schemes offer higher efficiencies than link-based schemes which attempt to patch disrupted services around a point of failure [1-4]. Link-based restoration schemes are also often unable to restore services affected by node failures or multiple link failures [11, pp. 4-6–4-8].

STAR is a progressive hop-by-hop routing algorithm. During a setup procedure, each node determines only the next node in the path of a service. Should a setup request be blocked at a node, responsibility for further routing of that service is "cranked-back" to the previous node, which can then forward the setup request to a different neighboring node. This hop-by-hop routing approach contrasts with source routing techniques like those used in the existing draft specifications for routing ATM virtual channels (VCs) [13], in which the origination node determines the entire path for a service to its destination. If the setup request is blocked at any point along the specified path, control for the setup procedure is returned to the origination node and the procedure can be retried. In a restoration environment, link availability can change rapidly and we believe that source routing would needlessly delay the establishment of restoration routes.

STAR is a dynamic rules-based procedure. Some rapid restoration approaches use preplanned restoration routes to best utilize surviving network resources [9, 10]. We believe that these approaches are cumbersome and unnecessary. Preplanning is unneeded, since simple protection mechanism or SONET ring-switching procedure of the transport infrastructure can cope with simple failures. Additionally, provisioning a service and the service's restoration routes can realistically cause restoration routes for other services to be relocated in a cascading fashion throughout the network. Finally, it is probably impossible to plan for all possible complex or multiple failure events. In these cases, a robust, distributed and dynamic restoration algorithm should perform well. Indeed, many of the precursor ideas of STAR evolved under such considerations [16].

STAR uses no routing tables: each routing decision is based on a shortest-path-pair calculation made in real time on a service-by-service basis. Numerous routing policies can be practically implemented by varying the input topology as a function of service information. To obtain the same flexibility with conventional routing tables would require that a different routing table be maintained for each routing policy.

1.2. Terminology

A *network* is a set of *nodes* interconnected by *links* having a mesh topology. A link is a uni- or bidirectional transmission system between two nodes. A *node ID* is any unique identifier for a node in the network and a *link ID* is any unique identifier for a link in the network. The bandwidth of a link is allocated to *circuits*, each circuit identified by a *circuit ID* unique within the link. Each node also has *access circuits* associated with *end systems* which terminate network services. Each node is a (virtual) circuit switch that forms connections between circuits. Node *i* is a *neighbor* of node *j* if there is a link from node *j* to node *i*. A link from node *i* to any node *j* is a *local link* of node *i*.

A *service* is uni- or bidirectional, information transport path between two end-systems. A *service path* is formed by a series of circuits connected together at nodes. The first and last circuit in a complete service path are access circuits. All other circuits in the service path are on links between nodes in the network. The first node in a complete service path is the *origination node* for the service; the last node is the *destination node*; all intermediate nodes are *via nodes*. For any node in the service path, the circuit directed towards the end system at the origination node is the *incoming circuit*; the circuit directed towards the end-system at the destination node is the *outgoing circuit*.

A *setup procedure* is followed by nodes within the network to establish a service path. A setup procedure begins at the origination node for a service in response to a *setup request*. Nodes communicate using *signaling messages* from a *signaling protocol*. Two messages from the signaling protocol are of primary importance: the *setup message* and the *crankback message*. The function of the setup message is to reserve an outgoing circuit on a link to a neighboring node and to forward the setup request to the neighboring node (i.e., signal the neighboring node to continue the setup procedure for the service.) A crankback message is sent back to the previous node in a service path to free the previously reserved incoming circuit and to relinquish responsibility for continuation of the setup procedure back to the previous node. A *setup trail* is a list of node IDs for every node that has originated either a setup message or a crankback message during the setup procedure for a service, and is included in both setup and crankback messages.

2. Overview

Figure 1 is an information flow diagram for STAR and for two other processes within a switching system related

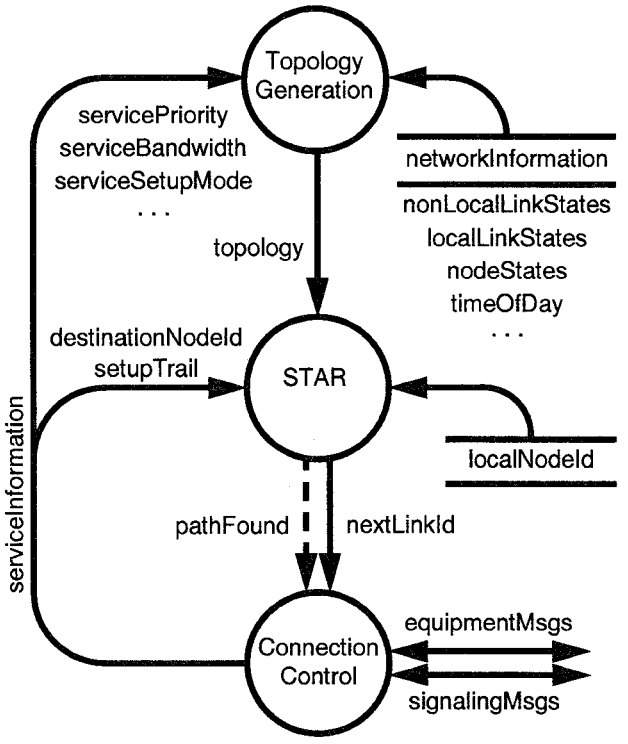

Figure 1. Information flow for the STAR algorithm and related processes.

to the setup procedure: connection control and topology generation. Actions begin when a setup request is received by connection control. Various service and network information is combined to generate a network topology relevant to the service being set up. STAR then combines this topology with the setup trail and local and destination node IDs, to determine the next direction to route the service. This information is then passed to connection control which transmits an appropriate signaling message to the proper node. The following subsections describe these three processes in more detail.

2.1. The STAR algorithm

This section defines the STAR routing algorithm in terms of input and output information and the process by which input is transformed to output. This specification is not intended to restrict the structure of information, nor detail STAR processing.

2.1.1. STAR inputs.

localNodeId: The node ID of the node routing a service.

destinationNodeId: The node ID of the service destination node.

topology: The list of *link data structures* associated with network links available to the service, where each link data structure includes the following information:

- *linkId*: The link ID of the link.
- *fromNodeId*: The node ID of one of the two nodes connected by the link. If the link is unidirectional, *fromNodeId* is the ID of the link's transmitting node.
- *toNodeId*: The node ID of the other node connected by the link.
- *linkLength*: Any positive number representing the relative cost for the service to use the link.

setupTrail: The service setup trail.

2.1.2. STAR outputs.

pathFound: A boolean indication. If true, then a setup-trail-avoiding path was found to node *destinationNodeId*. If false, then no setup-trail-avoiding path was found to node *destinationNodeId*.

nextLinkId: If *pathFound* is true, then *nextLinkId* is the link ID of the first link in the shortest-setup-trail-avoiding path from node *localNodeId* to node *destinationNodeId*.

2.1.3. STAR processing.

STEP 1: Reduce *topology*: for each node *i* listed in *setupTrail*, such that *i* is not equal to *localNodeId*, remove all link data structures for which either *fromNodeId* or *toNodeId* equals *i*. (If links are unidirectional, it is sufficient to remove either all links for which *fromNodeId* equals *i*, or all links for which *toNodeId* equals *i*.)

STEP 2: Find the shortest path from node *localNodeId* to node *destinationNodeId* in the graph defined by the reduced topology. If a path is found, return with *pathFound* equal to true, and *nextLinkId* equal to the link ID of the first link in the path. If no path is found, return with *pathFound* equal to false.

2.2. Connection control

The connection control process carries out the higher-level procedures needed in the establishment and removal of services. Only those aspects of this process relevant to understanding the operation of STAR are described here.

A setup procedure begins at the origination node for a service in response to a setup request which may arrive in the form of a setup message from an end system or an indication that a service has failed and should be restored. (Such failure indications might be received from local switching or transmission equipment or from another node along the path of a service.) During the setup procedure the setup request is passed from node to node via setup and crankback messages. When a connection control process receives a setup request, it requests a routing direction for the service from STAR. To carry out this routing request, service information is provided to topology generation and to STAR as shown in Figure 1. Connection control uses the results of the STAR calculation to determine the actions to take in routing a service to its destination.

If STAR finds a shortest-setup-trail-avoiding path from the local node to the service destination node, STAR returns the first link in this path. Connection control then sends a setup message to the neighboring node on the other side of this local link, allocating an outgoing circuit on the link and forwarding routing responsibility to the neighboring node. If, however, STAR fails and the local node is a via node, a crankback message is sent back to the previous node in the service path, freeing the previously allocated circuit and passing routing responsibility back to the previous node. Otherwise, if STAR fails at the origination node, the service setup procedure ends.

At some point in the service setup procedure, commands must be sent to the switching device to implement the type of connection called for in the setup request.

Aside from requiring the functionality of a setup and crankback message as described above, STAR does not otherwise define a signaling protocol. Information included in these messages could include the circuit ID of the circuit used by the service, the destination node ID, parameters characterizing the type of service, and, of course, the setup trail.

Additional messages may be needed in the signaling protocol for complete connection control but STAR places no requirements on these messages. STAR could be used with signaling protocols like N-ISUP [14] or B-ISUP [15], if the setup trail were added as an information element to the initial address message (IAM) (the ISUP name for a setup message) and crankback message (CBK).

2.3. Topology generation

The STAR algorithm requires a network topology as input as shown in Figure 1. This section gives a general description for generating this topology. The topology consists of a set of network links available to a service and defines a nonnegative length for each link, as specified in Section 2.1. The set of links and link-lengths could be dependent upon many input parameters. Parameters affecting the topology can be classified into two groups: parameters independent of the service being routed, and parameters dependent upon the specific service being routed. For simplicity, the former will be referred to as *network information* (although it may include things not related specifically to the network, e.g., time of day), and the latter will be referred to as *service information.* Examples of members of both groups are listed below.

Network information that may affect the topology:

- *nonLocalLinkStates*: Link state information (connectivity, failure, utilization, etc.) on non-local links from a link state database
- *localLinkStates:* Authoritative state information on local links from the local node
- *nodeStates*: State of critical resources (processing, memory, etc.) at nodes
- *timeOfDay*: Time of day.

Service information that may affect the topology:

- *servicePriority*: The priority of the service
- *serviceBandwidth*: The bandwidth required by the service
- *serviceSetupMode:* The circumstances under which the setup procedure was initiated (e.g., provision, restoration, or reroute.)

The only required inputs are *nonLocalLinkStates* and *localLinkStates*. At a minimum, *nonLocalLinkStates* must contain connectivity information for the network. *nonLocalLinkStates* is derived from a link state database containing information about all links in the network. Each node has its own copy of the link state database. The database could be static or may be maintained by a dynamic link state update protocol, such as those in [13, 17, 18]. STAR is robust to uncertainty in the availability of non-local links included in the topology: a service is deflected onto a different link or cranked back from a link that cannot support the service.

Proper operation of STAR requires that a local link be included in the input topology only if the link can actually be used to set up the service. *localLinkStates* provides sufficient authoritative status information on local links to determine the availability of local links for the type of service being routed.

3. STAR execution time

One possible disadvantage of STAR is that it can be more computationally intensive than other algorithms: each routing decision requires topology reduction and a shortest-path-pair calculation. If the network is to support many different routing policies, the initial topology might also need to be generated for each service. Because of these considerations, one might expect STAR to be impractical in large networks or in systems where processing is a premium.

Perhaps surprisingly, STAR can execute very quickly, even for relatively large networks. One of us (M.T. Busche) has written software for two versions of the STAR algorithm: the first supports integer link lengths; the second restricts link lengths to the constant value of 1 (i.e., a simple hop count metric). Both were compiled and executed on a Sun SPARCstation 10 running SunOS 4.1.3. The network simulated in this study had more than 300 nodes and over 500 links. (See Section 4.2.2.) The execution times for the two versions of STAR, averaged over all node pairs in this network, were 1.06 ms and 180 μs respectively. Although we believe this software to be efficient, no claims are made of its optimality. In

particular, the first algorithm uses a straight forward implementation of Dijkstra's algorithm having $O(N^2)$. Shortest path algorithms exist with lower orders of computational complexity that are expected to have faster execution times for large networks [19].

The second STAR implementation (i.e., the simpler hop count version) was used for the simulation in Section 4. For the simulated network (detailed in Section 4.2), STAR processing contributed only about 15% to the total processing time of setup and crankback messages. After accounting for transmission and insertion delays, the STAR calculation represented only 3% of the time for a node to receive and process a setup or crankback message. Furthermore, for many systems, the actual service connection time is not limited by the speed of the signaling system but by the time it takes a switch to make a connection. For example, the cross connect time for AT&T's Digital Access and Cross Connect System III (DACS III) is about 500 ms, more than two orders of magnitude larger than the message processing time of the node controller. With the speed of currently available microprocessors and expected future improvements, the trade off between the increased message processing time and the superior performance and flexibility of STAR should become increasingly favorable to STAR.

4. Performance assessment of STAR in a DCS-based restoration system

Simulation was used to compare the performance of STAR with a similar distributed routing algorithm in a DCS-based restoration system. A restoration system offers a stress test for the algorithms since during restoration a large number of services simultaneously require routing through a region of the network which has just lost usable capacity. The performance measures considered are restoration efficiency, signaling load and speed.

The algorithm we are comparing STAR against is the dynamic k-shortest path (DKSP) restoration algorithm [8, 16]. An independent simulation-based study of various distributed restoration algorithms conducted by Bellcore found that, of the algorithms they investigated [1, 2, 3, 7, 8], DKSP offered the best combination of speed and efficiency [11, p. 7-28]. By comparing STAR's performance with that of DKSP, we are thus comparing STAR to an independently-claimed leading distributed restoration algorithm.

Although the comparison is based on the results of a DCS-based restoration system, any connection-oriented restoration system employing the same routing algorithms and a like signaling protocol should give similar results with regards to efficiency and signaling load. The time needed to complete signaling procedures will depend on the speed of the signaling channels and the speed with which nodes can process signaling messages. The speed with which end-to-end data-carrying connections can be formed once a service path has been identified will also depend on the time it takes circuit connections to be formed and broken by the particular switches used in the network.

4.1. The DKSP algorithm

The setup procedure for DKSP is nearly identical to that described for STAR in Section 2.2: only the process that a node uses to select the next link in a service path is different. In the DKSP system, each node maintains a separate routing table for each other node in the network. If node v has k neighbors, then every routing table at node v will have k entries. Each entry of the routing table at v for some destination node d has two elements: the node ID of a neighbor of v and the length of the shortest path to d through that neighbor, with the restriction that the path from the neighbor to d does not pass through v. If no path exists under this restriction, the distance is considered infinite. The k entries in the routing table are sorted by distance so that the neighbor offering the shortest path to d appears first in the table.

The next-hop node in the path of a service with destination d is then the neighboring node listed in the first entry in the routing table for destination d that satisfies (at a minimum) these constraints:

1) The neighboring node is not listed in the setup trail.
2) The distance is not infinite.
3) A link to the neighboring node can support the service.

If no satisfactory entry is found, then the routing procedure fails.

4.2. Simulation description

Although not highly dependent upon exact details, the simulation is briefly described to put the discussion of the restoration performance into context. This description includes the node architecture used in the network, a gross depiction of the simulated network, a brief summary of the topology update and signaling protocols, and a listing of the timing parameters used in the simulation.

4.2.1. Node architecture. The node architecture for the distributed restoration system modeled by the simulation is shown in Figure 2. More complete descriptions can be found elsewhere [8, 16].

4.2.2. Simulated network. The network simulated is based on an extract of AT&T's national digital signal level 3 (DS3) facility network as of 1991. A DS3 is a digital communications channel with a data rate of

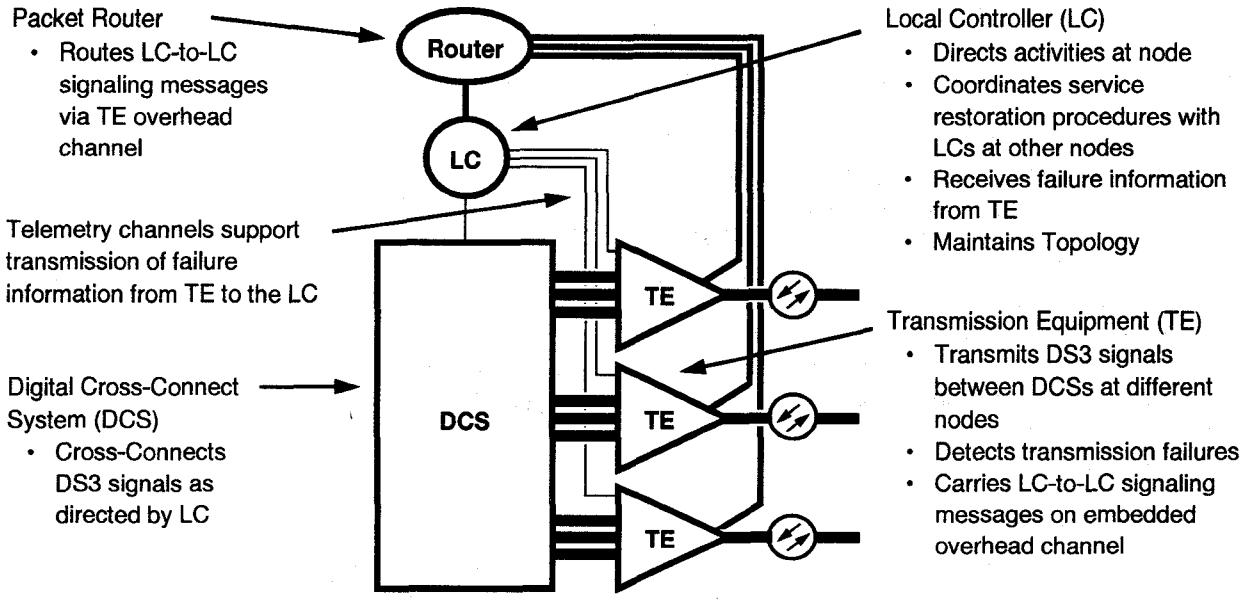

Figure 2. Node architecture of the simulated distributed restoration system.

44.736 Mbps. The physical path of a DS3 service is defined by a sequence of facility *spans*. A DACS III is assumed to exist at span endpoints which are selected so that in the resulting network, the links out of each DACS III node follow at least three different physical paths. A single span failure can fully or partially disrupt multiple links. The resulting network contained over 500 links and over 300 nodes, and was traversed by almost 9,000 services. Restoration capacity constituted about 30% of the circuits in the network.

4.2.3. Topology update protocol. The trunk topology update protocol (TTUP), a simple flooding protocol similar to [17], was used to maintain link-state connectivity. In this protocol, a node sends messages regarding the state of its links to its neighbors. Nodes receiving a new TTUP message forward it to all their neighbors, except the neighbor from which the message was received. This procedure is repeated until the TTUP message is received by all connected nodes in the network.

Two timers are associated with changes in the network's topology. When an LC detects a local failure, it delays any response by a *cutWaitTime* interval, to allow other protection mechanisms (e.g., a SONET ring restoration) to operate. If the failure still exists at the end of this interval, the node sends out TTUP messages and initiates any needed restoration procedures. Each time a node sends or receives a new TTUP message indicating failure, it inhibits further routing decisions for an *updateWaitTime* interval. This interval allows for the reception of all TTUP messages associated with the same failure event so that services are not routed based on a topology that includes failed links.

Two models were used regarding the information about links in the network. These were:

One Link State: A link can be in only one state: all that is known about non-local links is that they exist. No information about the failure status or busyness of non-local links is known. All non-local links are always included in the topology used by STAR and DKSP. All links have unit length. TTUP is effectively inactive.

Two Link State: A link can be in two states: failed or in-service. A link is considered failed if all of its circuits have failed, otherwise it is in-service. No information about the busyness of non-local links is known. Only in-service links are included in the topology used by STAR and DKSP. All links have unit length.

The static, one-link-state model, was included in the performance assessment as a cogent demonstration of the exceptional stability and efficiency of the STAR algorithm.

4.2.4. Signaling protocol. The signaling protocol used in the simulations was based on the ISUP of Signaling System 7 (SS7) [14]. Fundamental ISUP call control functions are provided by six messages. The function and usage of these six messages in the context of the simulation are briefly described here so that the simulation results can be understood.

The *Initial Address Message (IAM)* acts as the setup message described in Section 2.2. The content of the IAM is extended to include the setup trail.

The *Crankback Message (CBK)* acts as the crankback message described in Section 2.2. The content of the CBK is extended to include the setup trail.

An *Address Complete Message (ACM)* is generated when an IAM reaches the destination node of a service, and

is sent back along the service path. Via nodes receiving an ACM send an ACM to the previous node in the service path and initiate connection procedures between the incoming and outgoing circuits. When the origination node receives the ACM, the service is considered *search-completed.*

An *Answer Message (ANM)* is generated when the destination node connects the incoming circuit to the end system and is sent back along the service path. Each via node sends an ANM to the previous node in the service path after both an ANM has been received and the local circuit connection procedure has been completed.

The *Release Message (REL)* is sent by a node to release a circuit used by a service, and includes a cause code indicating why the service is being torn down. An origination node for a service receiving a REL with a cause code indicating a network failure may begin restoration procedures for that service.

A *Release Complete Message (RLC)* is sent in response to a REL.

4.2.5. Simulation timing parameters. Table 1 defines all timing parameters used by the simulation. All LC processing times were based on laboratory measurements of LC software running on a Sun SPARCstation 10 [20]. Router processing times were based on laboratory measurements of a Cisco AGS+ router. The internode communications channel speed of 500 kbps is similar to the 576 kbps channel available on the SONET Line DCC. The cross-connect time of 0.5 s is roughly the cross-connect time of an AT&T DACS III.

4.3. Simulated performance

The results presented here represent simulated failures of 51 individual spans and 31 individual nodes (or about 10% of the links and nodes in the network.) The number of services disrupted ranged from ten to several hundreds. The 82 span and node failures were simulated twice for each algorithm. The two sets of runs differed in the kind of topology information made available to the algorithms, either the one-link state or the two-link state, as described in Section 4.2.3. The tables and figures in this section give the aggregate results for the 82 simulations made for each combination of routing algorithms and link state models.

Table 1. Simulation timing parameters.

Parameter	Value
LC processing time of most messages	1.0 ms
LC processing time of all other messages, which require little activity (e.g., RLCs and DCS messages)	0.5 ms
Cut wait time	200.0 ms
Update wait time	50.0 ms
LC time to reinitiate routing for a service halted due to an active update wait timer	0.5 ms
Star routing time	0.2 ms
DKSP routing table calculation time	9.0 ms
DCS cross-connect time	500.0 ms
Router processing time	0.1 ms
LC to router communication rate	5 Mbps
LC to DCS communication rate	10 kbps
Router to router communication rate	500 kbps
Message size	800 bits

4.3.1. Restoration efficiency. The efficiencies of the routing algorithms were measured in terms of the number of services successfully restored after a failure. The number of services restored by STAR vs. the number of services restored by DKSP for the 51 span and 31 node failures are displayed in Figure 3(a) for the one link state system and in Figure 3(b) for the two link state system. Points above the 45 degree reference line in these plots indicate failures for which STAR was more efficient than DKSP. In both plots, many more points appear above the reference line than below, showing STAR's superior efficiency.

To independently quantify the performance of the two algorithms, a simple centralized algorithm was used to estimate an upper bound for the number of services that could be restored for each failure. This algorithm simply routed failed services on shortest paths one at a time until no more services could be restored. The centralized algorithm was able to restore a total of 13,273 services for all 82 span and node failures. The total number of services restored by each distributed algorithm was compared to this number to estimate the percent efficiency. Table 2 shows the number of services restored and the percent

Table 2. Comparison of the number of services restored by DKSP and STAR and the corresponding percent efficiencies relative to a centralized algorithm which restored 13,273 services.

Algorithm	One Link State System		Two Link State System	
	Number of Services Restored	Percent Efficiency	Number of Services Restored	Percent Efficiency
DKSP	12,056	90.8%	12,806	96.5%
STAR	12,747	96.0%	13,071	98.5%

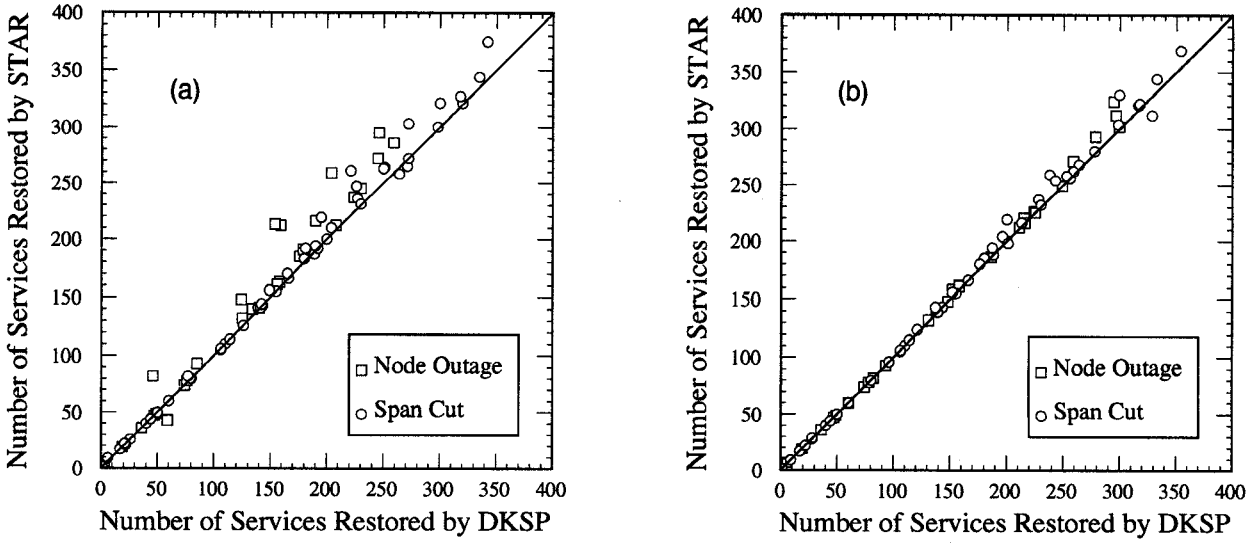

Figure 3. Comparison of the number of services restored by STAR and DKSP.
(a) One link state system.
(b) Two link state system.

efficiency for both STAR and DKSP for the one and two link state systems. The superior efficiency of STAR in the one link state case illustrates its capability to route services intelligently, even in the face of necessarily outdated topology information. Distributing just the failure information as in the two-link state case improved the efficiency of both routing approaches, but also narrowed the margin of difference between them. The efficiency of both STAR and DKSP improve with better topology information, but STAR makes better use of information that is available.

Most of the difference between the efficiencies of STAR and the centralized algorithm was due to congestion during the restoration. During the distributed restoration procedures, there were a large number of services simultaneously competing for the limited idle capacity in the network. In some cases, the competition ended in a stalemate leaving a small amount of idle capacity unclaimed. Reattempting the service setup procedure for services that were not initially successfully restored resulted in the restoration of additional services. Although the results are not otherwise detailed in this paper, including reattempts in the simulation increased STAR's efficiency to 98.8% relative to the centralized algorithm for the one link state system, and to 99.6% for the two link system. This increase in efficiency suggests that reattempts are more important to achieving high efficiencies than topology updates.

4.3.2. Signaling load. Perhaps the most striking performance characteristic of STAR is its ability to reduce internode signaling. Table 3 compares the number of IAMs and CBKs for STAR and DKSP for both the one and two link state systems. The numbers of IAMs and CBKs indicate the level of network activity in searching for restoration routes. Even though DKSP engaged in three times as much searching activity, it restored fewer services than STAR.

STAR reduces signaling load by recognizing situations where the nodes in the setup trail form an impassible boundary blocking the service from its destination. In such situations STAR fails and a crankback is generated. In this way, services that cannot be completed are quickly terminated, making STAR an extremely stable algorithm.

Table 3. Comparison of the signaling load for STAR and DKSP.

Message Type	One Link State System			Two Link State System		
	DKSP	STAR	Percent Reduction	DKSP	STAR	Percent Reduction
IAM+CBK	2,299,824	751,658	67.3%	1,698,832	534,466	68.5%
TTUP	0	0	0.0%	266,208	266,208	0.0%
Other	627,082	622,238	0.8%	625,788	607,810	2.9%
Total	2,926,906	1,373,896	53.1%	2,590,828	1,408,484	45.6%

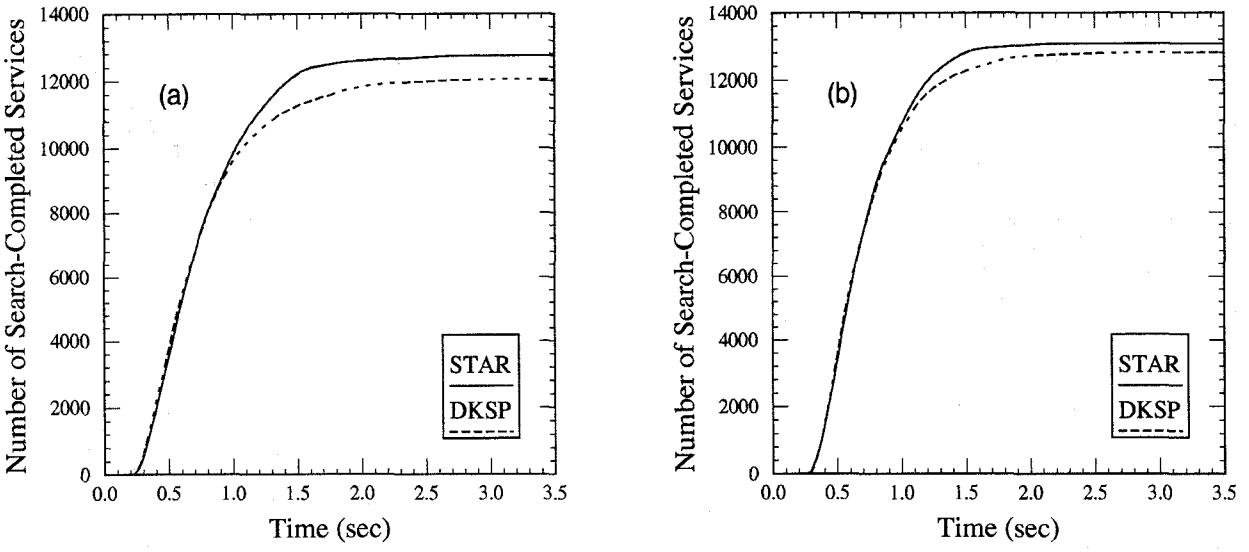

Figure 4. Cumulative distribution of the service completion times for STAR and DKSP. (a) One link state system. (b) Two link state system.

Because DKSP routing tables are not constructed with knowledge of the setup trail, a DKSP-routed service may wander to every node in an isolated portion of the network before cranking back.

4.3.3. Restoration speed. For all the simulations, the failure event was assumed to take place at time zero. Figure 4 shows the search completion time distributions for both STAR and DKSP for (a) the one link state system and (b) the two link state system. As can be seen from these distributions, and from the average search completion times shown in Table 4, there is not much difference in the speeds with which STAR and DKSP completed services, despite the extra time STAR needs for routing calculations.

The initial immediate lack of service completions was due to the 200 ms cut wait timer. (See Section 4.2.3.) In an actual system, the cut wait time should be adjusted to reflect the speed of protection switching and any delays in the detection of failures and protection switches.

The update wait timer in combination with the extra load TTUP messages placed on node processors and the signaling network made the two link state systems slightly slower during the early periods of restoration; however, the two link state systems surpassed the one link state systems after about 0.5 seconds and were slightly faster overall, as shown by Table 4. Thus, flooding limited topology information not only improved the efficiency of both the STAR and DKSP restoration systems, but also improved their overall speeds.

Signaling procedures for 90% of services routed by STAR were completed in about 1.3 seconds for the one link state system and in about 1.2 seconds for the two link state system. These times fall well within a target restoration time of two seconds identified by Bellcore for telecommunications networks [11]. The two-second target could be achieved with the simulated signaling system if it were combined with a fast enough switching vehicle. Such a switch would be required to process on the order of hundreds of switching commands per second.

Table 4. Comparison of the average search completion times for all services restored by DKSP and STAR.

System	DKSP (sec)	STAR (sec)	Percent Reduction
One Link State	0.752	0.763	-1.5%
Two Link State	0.732	0.725	1.0%
Percent Reduction	2.3%	5.0%	

5. Conclusions

The setup-trail-avoidance routing algorithm is a new simple distributed routing algorithm for connection-oriented mesh networks. STAR adapts its routing function based on the setup trail of each individual service to achieve exceptional performance even with inaccurate topology information in rapidly changing environments. STAR's compatibility with existing communication, topology, and signaling protocols indicates that the development of specialized protocols is unnecessary.

STAR calculations for a nationwide network with more than 300 nodes and 500 links have been measured on a

SUN SPARCstation 10 to take less than 200 μs. This time interval, for reasonable assumptions of signaling bandwidth, is small (~3%) compared to the total time needed for message insertion, transmission and processing.

STAR does not depend upon nor generate routing tables, but instead relies on a real-time, hop-by-hop decision of the best direction to set up a service. STAR can thus support many routing policies based on service properties like bandwidth or priority without a corresponding increase in processing or memory demand.

This paper demonstrates that STAR performs better than many previously proposed algorithms for directing distributed restoration. In an independent study by Bellcore comparing five other distributed restoration algorithms, they determined that DKSP had the best performance [11]. Simulation results in this paper show that STAR is as fast as DKSP and has better performance in terms of efficiency, signaling load and stability. Therefore, STAR should be considered a leading contender among currently proposed restoration algorithms.

Using only static topology information for the simulated network, STAR completed signaling procedures for 90% of restored services in 1.2 seconds and achieved 96.0% restoration efficiency relative to a centralized algorithm heuristic (not proven optimal) having total network knowledge. When topology changes were disseminated, both STAR's efficiency and speed improved (to 98.5% and to 1.1 seconds to complete signaling procedures for 90% of the restored services). Reattempting services that had not been initially restored further increased STAR's efficiency to 99.6%.

STAR's efficiency, fast actions, distributed nature, and stability can enhance the robustness and integrity of connection-oriented communications networks including those based on DCSs or ATM switches.

6. Acknowledgments

We wish to thank our former supervisor, Clayton Lockhart, for all the insights he has offered into the analysis of the system. The reviews and comments of Jim Stroud, Joe Pisano, Joe Scholl and Mr. Busche's wife, Tammy Banks, have greatly improved the quality of this paper.

References

[1] W. D. Grover, et al. Development and Performance Assessment of a Distributed Asynchronous Protocol for Real-Time Network Restoration. *IEEE Journal on Selected Areas in Communications*, 9(1):112-125, January 1991.

[2] H. Sakauchi, Y. Nishimura, and S. Hasegawa. A Self-Healing Network with an Economical Spare-channel Assignment. *Proceedings of IEEE Globecom 90*, pp. 438-443, December 1990.

[3] H. Komine et al. A Distributed Restoration Algorithm for Multiple-link and Node Failures of Transport Networks. *Proceedings of IEEE Globecom 90*, pp. 459-463, December 1990.

[4] C. Edward Chow, et al. A Fast Distributed Network Restoration Algorithm. *Proceedings of 12th International Phoenix Conference on Computers and Communications*, pp. 261-267, March 1993.

[5] R. Kawamura, K. Sato and I. Tokizawa. Self-Healing ATM Network Techniques Utilizing Virtual Paths. *Proceedings of the Fifth International Network Planning Symposium*, pp. 129-134, May 1992.

[6] H. Fujii and N. Yoshikai. Restoration Message Transfer Mechanism and Restoration Characteristics of Double-Search Self-Healing ATM Network. *IEEE Journal on Selected Areas in Communications*, 12(1):149-158, January 1994.

[7] B. G. Cortez, private communication, 1992.

[8] M. T. Busche, C. M. Lockhart and C. Olszewski. Dynamic K-Shortest Path (DKSP) Facility Restoration Algorithm. *Proceedings of IEEE Globecom 94*, pp. 536-542, November 1994.

[9] J. Anderson, et al. Fast Restoration of ATM Networks. *IEEE Journal on Selected Areas in Communications*, 12(1):128-138, January 1994.

[10] B. Edmaier. Protection mechanisms in ATM Networks. *Proceedings of the 12th Annual European Fiber Optics Communications and Networks Conference*, pp. 79-83, June 1994.

[11] Bellcore Special Report SR-NWT-002514, Issue 1, *Digital Cross-Connect Systems in Transport Network Survivability*, January 1993.

[12] Tsong-Ho Wu. *Fiber Network Service Survivability.* Artec House, Norwood, Massachusetts, 1992.

[13] ATM Forum 94-0471R10, *PNNI Draft Specification*, December 1995.

[14] ITU-T Recommendations Q.761, Q.762, Q.763, Q.764, *ISDN User Part (ISUP).*

[15] ITU-T Draft Recommendations Q.2761, Q.2762, Q.763, Q.2764, *Broadband Integrated Services Digital Network (B-ISDN) - Signalling System No. 7 B-ISDN User Part (B-ISUP)*, September 1994.

[16] National Communications System, *Network Management of Stressed Facility Networks, Level II Report Extension*, Technical Information Bulletin 92-14, September 1992.

[17] P. A. Hosein and D. C. Schmidt. Broadcasting Topology Information in the AT&T Survivable Signaling Network. *Conference Record, MILCOM 91*, November 1991.

[18] J. Moy. OSPF version 2. *Request For Comments (RFC) 1583*, March 1994.

[19] R. K. Ahuja, J. B. Orlin, K. Mehlhorn and R. E. Tarjan. Faster Algorithms for the Shortest Path Problem. *Journal of the Association for Computing Machinery*, 37(2):213-223, April 1990.

[20] National Communications System, *Distributed Restoration System Protocol Simulation Capability, DRS Protocol Test Report*, Contract DCA100-93-C-0340 (dated September 27, 1993), CDRL 003, November 10, 1994.