*Journal of Network and Systems Management, Vol. 10, No. 3, September 2002 (© 2002)*

# Crankback Prediction in Hierarchical ATM Networks

**Eyal Felstaine,[1] Reuven Cohen,[1] and Ofer Hadar[1,2]**

When an ATM node discovers that it cannot continue the setup of a virtual channel under the requested Quality of Service (QoS), it initiates a backtracking procedure called "crankback." We propose a novel scheme, referred to as crankback prediction, that decreases the crankback overhead. Under the proposed scheme, nodes check during the connection admission control procedure whether the establishment of a virtual channel has a good chance to be admitted over the entire designated route. If this is not the case, crankback is initiated even before a particular QoS parameter is violated. The main idea behind the proposed scheme is to allocate a "quota" to the Peer Groups (PGs) along the message path, and then to suballocate this quota to the child PGs of these PGs. This process continues recursively until reaching the 1-level PG, which contains only physical nodes. The main advantage of the proposed scheme is that it lowers the setup delay and the processing and communication load imposed by signaling messages that establish unused portions of Virtual Channels (VCs)

**KEY WORDS:**

**Kindly provide keywords (3–8) for your article**

## 1. INTRODUCTION

In the future, global networks will consist of a hierarchy of subnetworks called domains. For reasons of both scalability and security, domains will not reveal details of their internal structure to other nodes. Instead, these domains will advertise only a summary, or aggregated view, of their internal structure, e.g., as proposed by the ATM (Asynchronous Transfer Mode) Forum PNNI (Public Network to Network Interface) standard.

In PNNI, every ATM switch is defined as a lower level node. Such a node maintains detailed topology information about the lowest cluster to which it belongs, namely about its PG (Peer Group). Such a PG usually contains a small number of switches and links. Hence, maintaining the detailed information for the

[1]Department of Computer Science, Technion, Haifa, Israel.
[2]To whom correspondence should be addressed at Department of Computer Science, Technion, Haifa 32000, Israel. E-mail: *hadar@bgumail.bgu.ac.il*

P1: Vendor/FZN
pp577-jons-378729 JONS.cls July 9, 2002 17:40

PG does not impose an excessive communication, storage, and processing burden on the node. In contrast, the node maintains only compressed topology information for its parent PG, because the latter contains a larger number of switches and links. Similarly, the node stores more aggressively compressed topology information about its grandparent PG, and so forth. A detailed description of the PNNI model is presented in Section 2.

When a virtual channel needs to be set up, the source node creates a hierarchical route consisting of a detailed path within the source node PG, a less detailed path within the source node's parent PG, and so on until reaching the lowest level PG, which is an ancestor of both the source and the destination nodes. When this PG is reached, a new "source" route is computed (not by the source node, but by some intermediate nodes) to descend to the final destination. A "source" route is also computed by intermediate border nodes in each hierarchy level, that need to determine the best way for crossing their PGs.

When a PNNI node discovers that it cannot continue the VC (Virtual Channel) setup process under the requested QoS (Quality of Service), it initiates a backtracking procedure called "crankback." To this end, it sends a RELEASE message that propagates backwards to the last node that has made a routing decision. The latter is required to compute an alternative route. If an alternative route does not exist, the node continues the roll back process, recursively, by sending a RELEASE message backwards to the originator of the route that was supposed to cross the parent PG.

We propose a novel scheme, referred to as crankback prediction, that decreases the crankback overhead. Under the proposed scheme, during the connection admission control procedure, nodes check whether the establishment of a virtual channel has a good chance to be admitted over the entire designated route. If this is not the case, crankback is initiated even before a certain QoS parameter is exceeded. As a simple example, suppose that the SETUP message spends 90% of the maximal allowed delay after traversing only 10% of the route. Hence it is very unlikely that the setup will be successfully completed over the designated route, and the next node will initiate crankback prediction even though not all the delay "allowance" has been exhausted.

The main idea behind the proposed scheme is to allocate a "quota" to the PGs along the message path, and then to suballocate this quota to the child PGs of these PGs. This process continues recursively until reaching the 1-level PG, which contains only physical nodes. The main advantage of the proposed scheme is that it lowers the setup delay and the processing and communication load imposed by signaling messages that establish unused portions of VCs.

The rest of the paper is organized as follows. Section 2 describes in more detail the hierarchical routing scheme of ATM PNNI. This section also outlines PNNI routing's related work. In Section 4, the concept of crankback and crankback prediction is described. Section 5 outlines the hierarchical quota subdivision

P1: Vendor/FZN
pp577-jons-378729 JONS.cls July 9, 2002 17:40

mechanism. Section 6 discusses several algorithms for computing the quota recursively allocated by each PG to its child PGs, and Section 7 presents a statistic approach for this computation. Section 8 presents simulation results of the benefit and loss of each algorithm. Finally, Section 10 concludes the paper.

## 2. ATM PNNI ROUTING

### 2.1. An Overview of ATM PNNI Routing

In computer networks, routing is a collection of algorithms that determine the routes that data packets will traverse until reaching their destination nodes. In order to make routing decisions, the network nodes should obtain topology information and maintain routing tables. There are two well-known approaches to perform this task distributedly. In the first approach, called *distance-vector routing* [1], each node sends its *neighboring nodes* its *entire* routing table. The receiving nodes use the received information to update their own routing tables, which they then send to their own neighbors. In the second approach, called *link-state routing* [1], each node broadcasts information to *all network nodes* regarding the status of its *local links only*. The network nodes use the received information to create and maintain an up-to-date network map, from which they deduce their routing tables.

The main drawback of the distance-vector algorithm is that it takes a long time to reconverge to alternate paths when a failure occurs in the network [1]. During that time, the routing tables may define loops that may cause congestion in the network. The link-state protocol responds much faster to topology changes. However, in large networks it lays an excessive communication, storage, and processing burden on the nodes. The concept of hierarchical routing introduced by Kleinrock and Kamoun [2] is usually employed in order to overcome this limitation of link-state routing. The idea of using hierarchical routing to avoid the excessive complexity in topology advertisement is discussed in [3–8]. The ATM Forum adopted a hierarchical routing algorithm for the ATM network [9].

According to the PNNI concept, the network nodes and links are organized hierarchically. At the lowest level of the hierarchy, each node represents an ATM switch and each link represents a physical link or an ATM VP (virtual path) [10]. The nodes and links of each level can be recursively aggregated into higher levels, such that a high-level node represents a collection of one or more lower level nodes, and a high-level link represents a collection of one or more lower level links. The OSPF (Open Shortest Path First) protocol [1], used for autonomous system routing in the Internet [1], has two levels of hierarchy. The ATM PNNI [9] is a hierarchical, dynamic link-state routing protocol, designed to scale to the largest possible ATM networks, encompassing thousands of switches. It can support a maximum of 105 hierarchy levels.

P1: Vendor/FZN
pp577-jons-378729 JONS.cls July 9, 2002 17:40

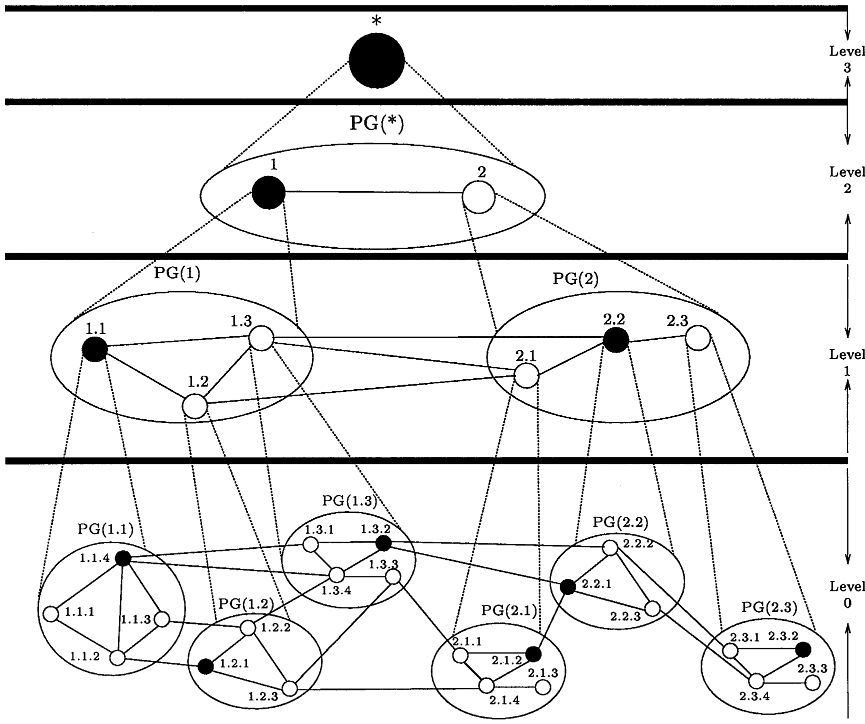


**Fig. 1.** An example for ATM PNNI hierarchy.

ATM PNNI uses hierarchical link-state routing with a maximum of 105 levels. To support this hierarchy, PNNI defines a uniform network model at each level, with a set of logical nodes connected by logical links. Each lowest level node represents a physical ATM switch with a unique ATM address. Nodes within a given level are grouped into sets of peer groups. A peer group (PG) is a collection of logical nodes that exchange link-state messages, called PTSPs (PNNI Topology State Packets), with other members of the group, such that all members maintain an identical view of the group. Each PG is assigned a unique PGID (peer group identifier) and is represented in its parent PG as a single node, called LGN (logical group node).

Figure 1 shows an example of ATM PNNI hierarchy. In this figure, a logical node whose identity is is of the form $x.y.z$ represents a physical node (ATM switch). Nodes 1.1.1–1.1.4 are grouped into a PG called PG(1.1), and in the same way additional five PGs are created. Hence, the network has 22 logical nodes in level 0 and 6 logical nodes in level 1. In level 2 there exist only two logical nodes: PG(1), which consists of LGNs (1.1)–(1.3), and PG(2), which consists of LGNs

P1: Vendor/FZN
pp577-jons-378729 JONS.cls July 9, 2002 17:40

(2.1)–(2.2). The upper level always has only one logical node (LGN $*$ in Fig. 1). The links between the physical (level 1) nodes, such as (1.1.1, 1.1.4) or (1.3.2, 2.2.1), represent physical links or VP (virtual path) connections. By contrast, the links between logical nodes in level 2 or above, such as (1.1, 1.3) or (1, 2), represent a set of physical links and VP connections.

As already indicated, each PG is represented in the next hierarchical level by a single LGN. The functions associated with an LGN are actually performed by the PGL (peer group leader). The PGL is determined by means of a distributed election process executed by all the LGNs in the PG (see [9] for more details). In Fig. 1, the PGLs are those LGNs represented by black circles. Note that physical node 1.1.4 functions as a PGL of PG(1.1). However, since LGN 1.1 is the PGL of PG(1) and LGN 1 is the PGL of PG($*$), the functions that LGNs 1.1 and 1 are supposed to perform as PGLs are actually implemented by physical node 1.1.4.

Topology information flows horizontally through a PG and downward into and through child PGs. A PGL within each PG has the responsibility of creating PTSPs that represent the status of the links connecting its PG with other PGs, and broadcasting these PTSPs to the other LGNs in its parent PG. In Fig. 1, LGN 1.1.4 is the PGL of PG(1.1); LGN 1.2.1 is the PGL of PG(1.2); and LGN 1.3.2 is the PGL of PG(1.3). Hence, these three LGNs exchange PTSPs in PG(1). The PTSPs created by 1.1.4 indicates the status of the links connecting PG(1.1) with other PGs: (1.1.4, 1.3.1), (1.1.4, 1.3.4), (1.1.3, 1.2.2), and (1.1.2, 1.2.1). This PTSP exchange may be thought of as the PGL feeding information up the hierarchy. It is necessary for creating the hierarchy and for distributing routing information about child PGs. Conversely, feeding information down the hierarchy is necessary to allow nodes in the lower level PGs to obtain knowledge about the full network hierarchy in order to select routes to destinations. When information is fed down from one level to a lower level, it is aggregated (summarized) [11–13]. Hence, at the lowest level each node has full information about its PG, aggregated information about its parent group, more aggregated information about its parent's parent group, and so on.

The physical and logical links are classified into two categories: inside links that connect two nodes within the same PG, and outside links that connect two nodes within two different PGs. Nodes connected by outside links are referred to as border nodes. For instance, in level 1 of Fig. 1, physical nodes 1.1.4 and 1.3.1 are border nodes and the physical link connecting them is an outside link. In level 2, nodes 1.3 and 2.2 are border nodes and the logical link (1.3, 2.2) connecting them, which represents the two physical links (1.3.2, 2.2.2) and (1.3.2, 2.2.1), is an outside link.

By exchanging topology information among nodes, each node obtains the information needed to create its "view of the world." Figure 2 depicts the local view obtained by the nodes in PG(2.2) of Fig. 1.

The solid lines at PG(2) and PG($*$) represent aggregated logical links, such as (2.1, 2.2), or physical links, such as (2.2.1, 2.2.2), inside PGs.

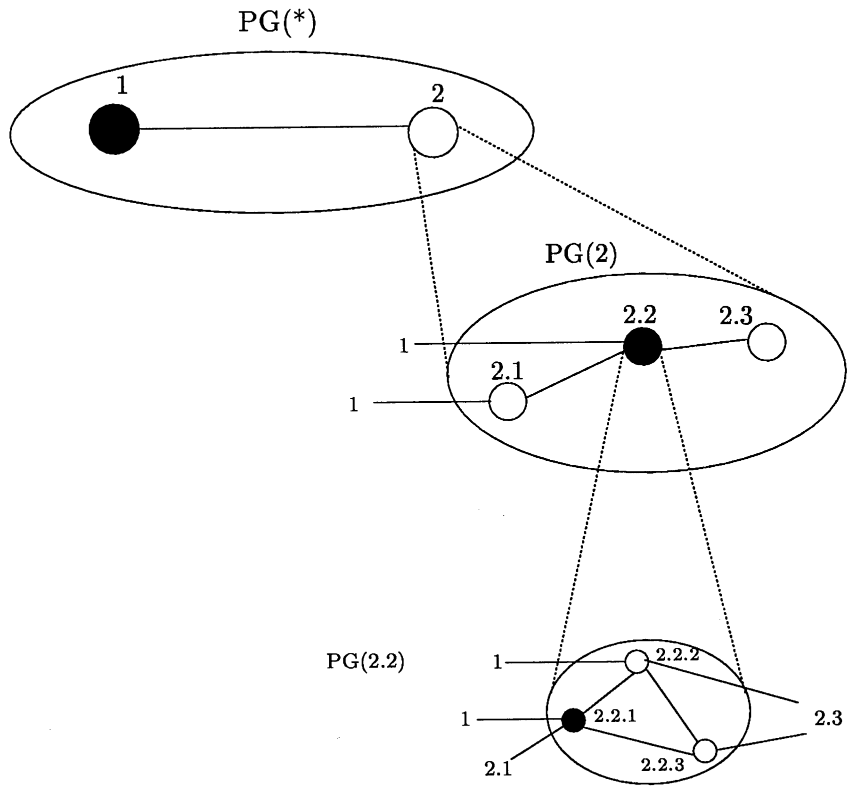


**Fig. 2.** Local view of the network from nodes in PG(2.2).

The ATM PNNI is supposed to provide sophisticated QoS routing while still allowing flexibility in the choice of route computation. Hence, each implementation is free to use its own algorithm. This gives rise to source routing, which does not require different switches to agree on the same computation. The switch of the source host creates a hierarchical route consisting of a detailed path within the source node PG, a less detailed path within the source node's parent PG, and so on until reaching the lowest level PG, which is an ancestor PG to both the source and destination nodes. When the LGN that contains the destination node in the lowest level common ancestor is reached, a new "source" route is computed to descend to the final destination. When necessary a "source" route is also computed by border nodes that determine the best way to cross their PG.

The path is encoded as a set of DTLs (Designated Transit Lists), which is explicitly included in a stack within the PNNI signaling call setup request. Each DTL contains the description of a path for one level in the hierarchy. It

P1: Vendor/FZN
pp577-jons-378729 JONS.cls July 9, 2002 17:40

explicitly specifies every LGN, and optionally every link, used to cross the PG. Each DTL is associated with a pointer that indicates the next element in the list to be processed.

## 2.2. Related Works on PNNI Routing

A major area for research in PNNI is the study of compact data structures that are designed for efficient topology aggregation. While PNNI defines a hierarchical structure and a flexible representation mechanism for performing topology aggregation, it does not specify any topology aggregation scheme. Several schemes have been proposed in the literature, most of which carry out the topology aggregation in two steps. In the first step, a fully connected mesh of border nodes of a PG is constructed, with a direct link between each pair. For a single QoS parameter, this step is "lossless" in the sense that it retains all the distances between the border nodes of the original graph. The second step involves mapping the full mesh into a more compact topology, such as a symmetric node (simple node), a star [13], a minimum spanning tree [12, 14], a hybrid [15], or a $t$-spanner [16, 17]. Graph reduction is then performed by pruning several links of a full mesh. Finally, the compact topology is represented as a complex node, which is broadcasted to the rest of the network.

One problem in the above approach is that the amount of lost information resulting from graph reduction is not known in advance and can vary depending on the actual value of the QoS parameters. To remedy this situation, some researchers proposed new topology aggregation approaches that minimize the average distortion in a least-squares sense. For example in [18] and [19] the authors present several algorithms for route selection in the presence of inaccurate topological information, including inaccuracies that are caused by topology aggregation or by the use of multiple QoS parameters. They suggest using shortest path algorithms with weights based on the probability that a link can satisfy the connection bandwidth requirements.

Other works focus on the effect of topology aggregation schemes on routing performance [20–22]. In [23–25], several different aggregation schemes are compared: cost matrix, star, minimum spanning tree, random spanning tree, and subgraphs of the internal PG graph topology. The authors introduce a logarithmic update scheme that determines when reaggregation should be computed. Another work suggests an enhancement to the PNNI standard based on random trees [11].

Another research area is PNNI network partitioning. It was shown in [26] that when topology aggregation is used, different hierarchical structures imposed on the same topology yield significantly different routing performance. An extensive survey of the graph partitioning literature can be found in [27]. In this paper the authors also present a self-structuring graph partition algorithm that addresses PNNI

P1: Vendor/FZN
pp577-jons-378729 JONS.cls July 9, 2002 17:40

related optimizations such as traffic localization. In [28], a top-down hierarchy construction algorithm is proposed that requires less CPU resources for hierarchy maintenance and also improves total network efficiency.

Research has also been conducted on ways to reduce the communication overhead associated with topology advertisement. There are two generic approaches to reduce update frequency. In the first approach, an update triggering mechanism triggers an update at the "right" time [29]. The second approach is to use appropriate routing algorithms to tolerate less frequent updates [30–32]. In the context of PNNI, an approach that focuses on reducing topology distribution overhead is described in [15]. In [33], the authors present a scheme where topology information is advertised only to those parts of the network where it is most relevant. Thus, the signaling and storage overhead is reduced while the accuracy of the information remains unaffected. The scheme is based on the area-based link vector-algorithm (ALVA) for hierarchical routing in the Internet [34]. Reduction in routing communication overhead is obtained in [35, 36], by an "adaptive" source routing scheme. The "viewserver hierarchy" framework [37] shows that merging the views of several domain-level servers can yield accurate knowledge of the hierarchical network topology.

## 3. "NORMAL" VS. "PREDICTED" CRANKBACK

### 3.1. PNNI Crankback

The PNNI protocol supports the notion of crankback. Crankback means that the setup of a blocked connection is rolled back to the intermediate node that has created and inserted a DTL into the SETUP message. Recall that such a node must be physically connected to another node outside the blocked PG. This intermediate node attempts to discover an alternative route toward the final destination. If it finds such a route, it creates a new DTL and sends the new SETUP message to the next node along the new route. If an alternative route is not found, the SETUP message is cranked back once again to the previous node that has inserted a DTL.

Setup blocking may occur owing to one or more of the following reasons:

1. A link or a node fails before it is traversed by the SETUP message.
2. A nonadditive metric associated with a link or a PG is not sufficiently large for the requested QoS.
3. The aggregated value of an additive metric exceeds the maximum specified by the calling host.

The distinction between additive and nonadditive metrics is as follows. Let $\mu$ be a QoS routing metric, and $\mu(i, j)$ be the value of $\mu$ over the link $(i, j)$. Consider a path $P$ established over the links $(i_1, i_2)\ (i_2, i_3) \cdots (i_{N-1}, i_N)$. Let $\mu(P)$ be the value of $\mu$ over path $P$. Then

- metric $\mu$ is said to be *additive* (*or cumulative*) if $\mu(P) = \sum_{j=1}^{N-1} \mu(i_j, i_{j+1})$. Examples for such a metric are delay and cost.
- metric $\mu$ is said to be *nonadditive* (*or concave*) if $\mu(P) = \text{MIN}_{j=1}^{N-1}\mu(i_j, i_{j+1})$ or if $\mu(P) = \text{MAX}_{j=1}^{N-1}\mu(i_j, i_{j+1})$. An example for such a metric is bandwidth.

To satisfy the constraint imposed by an additive metric $\mu$, a route $P$ should fulfill $\mu(P) \leq I$, where $I$ is some integer or real number. One exception is when $\mu$ represents cost, in which case it is usually required that $\mu(P) \leq \mu(P')$ for every route $P'$ between the same source/destination pair.

As an example, consider again the setup of a VC between a host of 2.2.3 and a host of 1.1.2 in Fig. 1 as discussed in Section 2. Assume that when the SETUP message reaches node 1.3.4, after crossing 2.2.1 and 1.3.2, the accumulated delay is equal to the maximum specified by the calling user. Hence, the message is considered to be *blocked* at 1.3.4 and is cranked back to 1.3.2, the originator of the topmost DTL [1.3.2, 1.3.4]. Node 1.3.2 tries to find an alternative route to PG(1.1). Suppose that such a route is not found, e.g., because the sum of the delay of link (1.3.2, 1.3.1) and the delay accumulated so far is higher than the maximum allowed. Therefore the setup is blocked at 1.3.2 as well, and the SETUP message is cranked back again to the originator 2.2.3, which is the previous node to compute a DTL.

### 3.2. Predicted Crankback

The main motivation for predicted crankback is to alleviate the crankback overhead by decreasing

1. the setup delay;
2. the processing and communication load imposed by signaling messages, which establish the unused portions of a VC;
3. the bandwidth and buffer resources allocated to the unused portions of the VC. These resources are wasted during the time period between the receipt of a forward SETUP message and the receipt of a backward RELEASE message.

The overhead of these factors increases with the length of the crankback segment. Since crankback prediction reduces the length of the crankback segment, it reduces this overhead as well.

Crankbacks that are related to link or node failure or to insufficient nonadditive metrics can be predicted by checking the designated route against the network topology using the local view of intermediate nodes, and observing that not all the PGs along the rest of the route can support the required QoS. The local view of intermediate nodes is more detailed and updated than that of the node that has designated the route, since the former are closer in time and location to the PG where crankback may occur.

However, predicting a crankback due to insufficient additive metrics is more complicated. This requires nodes to realize that the quota associated with an additive metrics is running out too fast, before it is exhausted. One may think that "additive" crankbacks can be avoided by simply using more precise aggregation schemes. However, it is shown [11, 23, 24] that deviations between the PG actual topology and its aggregated representation are inevitable.

To this end, we propose that each node along the route traversed by the SETUP message will invoke crankback if there exists an additive metric whose remaining quota is predicted to be insufficient for the rest of the routing through the specified DTLs. The observation we make between "predicted" crankback and "normal" crankback has to do only with the reachild for invoking crankback. If crankback is invoked because no route is found for the specified QoS parameters, it is considered to be "normal." If crankback is invoked because an additive metric is running out too fast, it is considered to be "predicted." In both cases, the node that invokes crankback performs the same procedure.

As an example for crankback prediction, consider Fig. 3 and suppose that a VC has to be set up between a physical node in PG(E) and A.1.2. To simplify the example, suppose that the only QoS requirement is that the cell transfer delay (CTD) will not be larger than 100 ms. Suppose that the aggregated delay for crossing PG(A), PG(B), PG(C), and PG(D) as advertised by their PGLs is 40, 50, 30, and 30 ms, respectively. Recall that each of these numbers is only an estimate. This estimate might be incorrect either because of unreported changes or because of the selected aggregation scheme. Suppose that the physical source node located in PG(E) has a route with negligible delay to a node in PG(B) and to a node in PG(D). This source is likely to route the message through PGLs E, D, C, and A rather than through PGLs E, B, C, and A, because the delay associated with the former route is $0 + 30 + 30 + \frac{40}{2} = 80$ ms, whereas the delay associated with the latter one is $0 + 50 + 30 + \frac{40}{2} = 100$ ms. Note that in both cases reaching the destination in PG(A) is estimated as crossing half the PG.

Suppose that the actual delay for crossing PG(D) is 80 ms, as opposed to the advertised cost of 30 ms. When C.3.3, the physical border node of PG(C), receives the SETUP message, it knows that the message has so far traversed only $\frac{30}{80}$ of its intended route, but has spent $\frac{80}{100}$ of the maximum delay. Therefore, it is unlikely that the remaining 20 ms will be sufficient for crossing the rest of the designated route. Hence, C.3.3 may decide to invoke the predicted crankback procedure.

If, despite of this observation, C.3.3 does not invoke the crankback procedure, it will continue the routing process by adding the DTLs [C.3.3, C.3.2, C.3.1] and

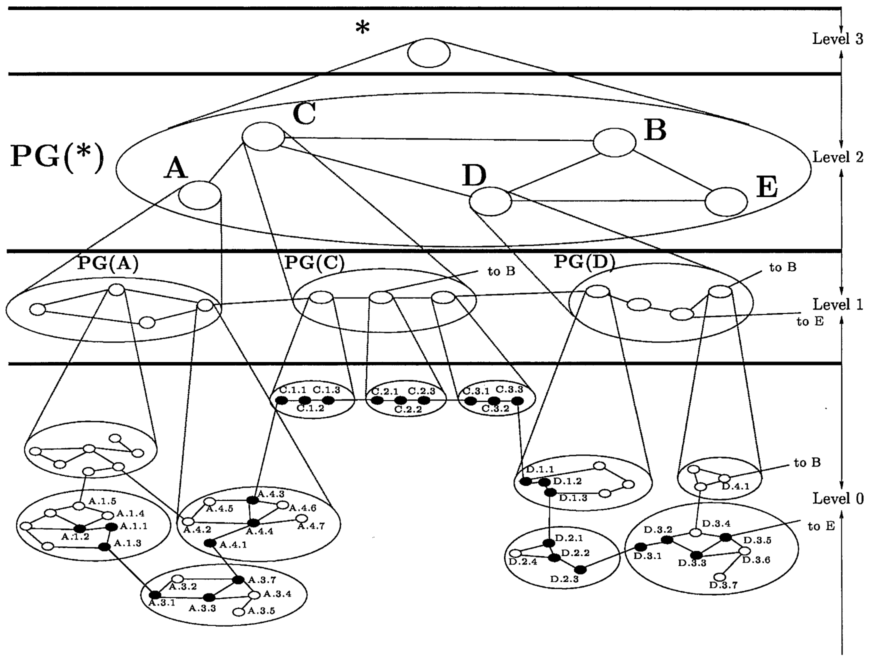


**Fig. 3.** An example of VC setup.

[C.3, C.2.C.1] only if the remaining quota of 20 ms is sufficiently large for crossing PG(C), based on the knowledge it has on PG(C). In such a case, it is likely that the SETUP message will enter PG(A), and will be received by A.4.3 with insufficient quota for the remaining trip. Hence, node A.4.3 will have to invoke a "normal" crankback.

## 4. A SCHEME FOR CRANKBACK PREDICTION

In this section we present a scheme for crankback prediction. The proposed scheme allows the nodes on the message path to initiate a crankback *before* the quota of an additive QoS metric is completely exhausted, based on an estimate that the quota is running out too fast. As already explained, this might happen because the route traversed so far falls short of expectation, and sooner or later crankback is likely to be invoked.

**Definition 4.1.** At a certain time, and for a certain SETUP message, an active $i$-level PG is the $i$-level PG being crossed by the message.

For example, if a message is in node A.4.4, the active 1-level PG is PG(A.4) and the active 2-level PG is PG(A).

The main idea behind the proposed scheme is to allocate a quota for every additive metric to the active high-level PG along the message path, then to suballocate this quota to the active child of the high-level PG, and to continue this process recursively until reaching the active 1-level PG. When a physical border node of a PG receives a message, it uses its local view of the network in order to find a route for crossing the PG or for reaching the destination within the PG, using no more then the specified quota. If such a route does not exist, the predicted crankback procedure is invoked.

To implement this approach, for every additive metric $j$ two new parameters are added to an $i$-level DTL in the DTL stack:

- $\alpha_j^i$ – indicates the metric $j$ quota assigned for crossing the *active* $i$-level PG. Recall that every DTL has a pointer that points toward the active PG.
- $\sigma_j^i$ – indicates the total amount of metric $j$ quota spent so far for crossing the $i$-level PG.

To simplify our notation, in the following we shall concentrate on one metric, and therefore use $\alpha^i$ and $\sigma^i$ instead of $\alpha_j^i$ and $\sigma_j^i$. By definition, if the topmost level the message traverses is $N$, then $\alpha^N$ indicates the quota assigned for the whole route, namely the maximum cost specified by the calling host. For every other $i$, $1 \leq i < N$, the value of $\alpha^i$ is determined by the physical border node of the *active* $i$-level PG, based on $\sigma^{i+1}$, $\alpha^{i+1}$ and other parameters, as discussed in the following.

Recall from Section 2 that upon receiving a SETUP message, a physical border node of an $i$-level PG pushes a DTL entry into the DTL stack. Our scheme requires that such a physical border node will compute the values of $\alpha^i$ and $\sigma^i$ for each $i$, using the algorithm depicted in Fig. 4. A physical border node of an $i$-level

```
A) Upon inserting an $i$-level DTL to the stack, for $1 \leq i \leq N$:
    A.1) $\sigma^i \leftarrow 0$
    A.2) determine a new value for $\alpha^i$.
    A.3) check if there exists a path that uses a quota of less than $\alpha^i$ for crossing the active $i$-level
         PG
        A.3.1) if such a path exists, push a new DTL into the DTL stack
        A.3.2) if such a path does not exist - invoke a *predicted crankback*
B) Upon removing an $i$-level DTL from the stack, for $1 \leq i \leq N$:
    B.1) $\sigma^{i+1} \leftarrow \sigma^{i+1} + \sigma^i$
C) When a physical node $a$ receives a SETUP message and the next 1-level physical node is $b$, where
   the cost of $(a, b)$ is $\sigma^0(a, b)$.
    C.1) $\sigma^1 = \sigma^1 + \sigma^0(a, b)$
    C.2) if $\sigma^1 > \alpha^1$ invoke a *predicted crankback.*
```

**Fig. 4.** A predicted crankback algorithm, and the recursive computation of $\alpha$ and $\sigma$.

P1: Vendor/FZN
pp577-jons-378729 JONS.cls July 9, 2002 17:40

PG is also a physical border node of lower level PGs. Hence, such a node needs to perform steps A and B of the algorithm i times. Step C, in contrast, is performed only once.

To demonstrate this algorithm, consider again the setup depicted in Fig. 3. When the SETUP message reaches C.3.3, the DTL stack contains only one DTL: $[E, D, \underline{C}, A]$. At this time, the value of $\alpha^3$ represents the total quota allocated to the whole connection, say $\alpha(*)$, whereas $\sigma^3$ represents the total quota spent so far $\sigma(D) + \sigma(E)$. Since the receiving node C.3.3 is a 2-level physical border node, it performs as follows (see Fig. 4):

$A$.1) It sets $\sigma^2$ to 0, since the quota spent so far for crossing PG(C) (the new 2-level PG) is zero.
$A$.2) It determines the quota $\alpha^2$ for crossing PG(C), say $\alpha(C)$, based on the remaining quota $\alpha^3 - \sigma^3$ and on other parameters as discussed in the sequel.
$A$.3) It then searches for a route of $\alpha^2$ or less for crossing PG(C). If such a route is not found, predicted crankback is invoked. If such a route is found, a new DTL is pushed into the DTL stack. Suppose that C.3.3 finds that route [<u>C.3</u>, C.2, C.1] can be used for crossing PG(C). Hence, it performs step A.3.1 and the SETUP message now looks as follows:
level 2: $[\underline{C.3}, C.2, C.1], \alpha^2 = \sigma(C), \sigma^2 = 0.$
level 3: $[E, D, \underline{C}, A], \alpha^3 = \alpha(*), \sigma^3 = \sigma(D) + \sigma(E).$

Next, C.3.3 has to fulfill its role as a 1-level physical border node. Hence, it needs to perform as follows:

$A$.1) It sets $\sigma^1$ to 0, since the quota spent so far for crossing PG(C.3) (the new 1-level PG) is zero.
$A$.2) It determines a quota $\alpha^1$ for crossing PG(C.3), say $\alpha(C.3)$, based on $\alpha^2$ and $\sigma^2$ whose values are $\alpha(C)$ and 0 respectively, and on other parameters.
$A$.3) It searches for a route of $\alpha^1$ or less for crossing PG(C.3). If such a route is not found, predicted crankback is invoked. If it is found, a new DTL is pushed to the DTL stack. Suppose that C.3.3 finds that route [C.3.3, C.3.2, C.3.1] is suitable. After C.3.3 finishes its role as a 1-level physical border node, the DTL stack looks as follows:
level 1: $[\underline{C.3.3}, C.3.2, C.3.1], \alpha^1 = \alpha(C.3), \sigma^1 = 0.$
level 2: $[\underline{C.3}, C.2, C.1], \alpha^2 = \alpha(C), \sigma^2 = 0.$
level 3: $[E, D, \underline{C}, A], \alpha^3 = \alpha(*), \sigma^3 = \sigma(D) = \sigma(E).$

Finally, C.3.3 has to fulfill its role as a 0-level node, namely, to route the message to C.3.2 over the link (C.3.3, C.3.2). Let the cost of this link be $\sigma^0(C.3.3, C.3.2)$. Before forwarding the message, C.3.3 has to perform as follows:

$C$.1) $\sigma^1 \leftarrow 0 + \sigma^0(C.3.3, C.3.2).$

Hence, the SETUP message received by C.3.2 contains

level 1: $[C.3.3, \underline{C.3.2}, C.3.1], \alpha^1 = \alpha(C.3), \sigma^1 = \sigma^0(C.3.3, C, 3.2)$.
level 2: $[\underline{C.3}, C.2, C.1], \alpha^2 = \alpha(C), \sigma^2 = 0$.
level 3: $[E, D, \underline{C}, A], \alpha^3 = \alpha(*), \sigma^3 = \sigma(D) + \sigma(E)$.

Assuming that $\sigma^0(C.3.2, C.3.1) + \sigma^0(C.3.3, C.3.2) \leq \alpha(C.3)$, node C.3.2 forwards the SETUP to C.3.1 with the same 3-level and 2-level DTLs, and with

level 1: $[C.3.3, C.3.2, \underline{C.3.1}], \alpha^1 = \alpha(C.3), \sigma^1 = \sigma^0(C.3.3, C.3.2) + \sigma^0(C.3.2, C.3.1)$.

C.3.1 is the last node on the 1-level DTL. It deduces that the SETUP message should be forwarded to PG(C.2) over a link (or VP) whose cost is not larger than $\alpha(C.3) - \sigma^0(C.3.3, C.3.2) - \sigma^0(C.3.2, C.3.1)$. Suppose that link (C.3.1, C.2.3) fulfills this requirement. Hence C.3.1 performs

$C.1)\ \sigma^1 \leftarrow \sigma^0(C.3.3, C.3.2) + \sigma^0(C.3.2, C.3.1) + \sigma^0(C.3.1, C.2.3)$.

However, since the 1-level DTL has to be removed, C.3.1 performs

$B.1)\ \sigma^2 \leftarrow \sigma^2 + \sigma^1$

Consequently, the SETUP sent by C.3.1 contains

level 2: $[C.3, \underline{C.2}, C.1], \quad \alpha^2 = \alpha(C), \quad \sigma^2 = \sigma^0(C.3.3, C.3.2) + \sigma^0(C.3.2, C.3.1) + \sigma^0(C.3.1, C.2.3)$.
level 3: $[E, D, \underline{C}, A], \alpha^3 = \alpha(*), \sigma^3 = \sigma(D) + \sigma(E)$.

## 5. CRANKBACK PREDICTION THRESHOLD AND QUOTA SUBDIVISION

In this section we define the functions that are used in step A.2 of the algorithm specified in Fig. 4 for predicting crankback and deriving the quota distributed to the child PGs. We start by defining the following terms:

- $\epsilon_x$ = The estimated delay for crossing PG($X$), as advertised by the PGL of PG($X$).
- $\epsilon^i = \{\epsilon_1, \epsilon_2, \ldots, \epsilon_n\}$ = An ordered list of estimated delays for crossing the child PGs of the active $i$-level PG that are found in the $i$-level DTL. The $k$th element in this list is denoted $\epsilon^i_k$.
- $p^i$ = The sequence number of the active ($i - 1$)-level PG in the $i$-level DTL stack.

For example, consider again the setup discussed in the network of Fig. 3. Assume that inside PG(D) the child PGs PG(D.1), PG(D.2), PG(D.3), and PG(D.4) advertised a crossing delay cost of 10, 20, 30, and 40 ms, respectively. Hence, $\epsilon_{\text{D.1}} = 10$, $\epsilon_{\text{D.2}} = 20$, $\epsilon_{\text{D.3}} = 30$, and $\epsilon_{\text{D.4}} = 40$. When PG(D) becomes the active 2-level PG and the 2-level DTL is [D.3, <u>D.2</u>, D.1], then $\epsilon^2 = \{\epsilon_{\text{D.3}}, \epsilon_{\text{D.2}}, \epsilon_{\text{D.1}}\}$ and $p^2 = 2$. Note that at this time, the estimated delay for the portion of PG(D) that needs to be traversed is

$$\sum_{j=p^2}^{|\epsilon^2|} \epsilon_j^2 = \epsilon_{\text{D.2}} + \epsilon_{\text{D.1}} = 30 \text{ ms}.$$

Note that $\alpha^2$ and $\epsilon^2$ do not change between the time the message enters PG(D) (the active 2-level PG) and the time it leaves PG(D). In fact, the value of $\epsilon^2$ is not even transmitted with the DTL, but is flooded to the participating nodes using the PNNI normal topology advertisement mechanism. The parameters that do change while the message is routed through the PG are $\sigma^2$ and $p^2$.

We now define a new function $f$ that uses $\alpha^i$, $\epsilon^i$, $p^i$, and other parameters for determining whether or not a crankback is likely to occur under a given set of parameters. At no point in time should the quota already spent in the active $i$-level PG, namely $\sigma^i$, exceed the threshold computed by $f$. We start with a simple function, referred to as $f_{\text{LIN}}$, defined as follows:

$$f_{\text{LIN}} = \frac{\sum_{j=1}^{p^i} \epsilon_j^i}{\sum_{j=1}^{|\epsilon^i|} \epsilon_j^i} \times \alpha^i. \tag{5.1}$$

This function multiplies the allocated quota for the active $i$-level PG, $\alpha^i$, by a fraction that represents the relative quota that should have been spent according to the advertised values. To demonstrate this function, consider again the network in Fig. 3 and suppose that $\alpha^2 = 66$. Hence,

$$f_{\text{LIN}} = \frac{\epsilon_1^2 + \epsilon_2^2}{\epsilon_1^2 + \epsilon_2^2 + \epsilon_3^2} \times \alpha^2 = \frac{30 + 20}{30 + 20 + 10} \times 66 = 55. \tag{5.2}$$

Hence, according to $f_{\text{LIN}}$, predicted crankback should be invoked if the quota used for crossing PG(D.3) and PG(D.2) is more than 55.

The main weakness of $f_{\text{LIN}}$ is that linear subdivision of the quota is less tolerant to the variance in the ratio between the advertised and actual costs. Consequently, the probability for false prediction, namely unnecessary invocation of crankback, increases.

The second $f$ function we propose is $f_{\text{DECAY}}$, defined as follows:

$$f_{\text{DECEY}} = f_{\text{LIN}} \times g\left(\frac{\sum_{j=1}^{p^i} \epsilon_j^i}{\sum_{j=1}^{|\epsilon^i|} \epsilon_j^i}\right), \tag{5.3}$$

where $g(z)$ is a monotonically decreasing function, defined for $0 \leq z \leq 1$, and yielding a value $\geq 1$. The closer the current node is to the beginning of the route, the greater $g$'s value would be. Function $g$ converges to 1 as $z$ approaches 1, namely, when the SETUP message reaches the end of the PG.

$f_{\mathrm{DECAY}}$ aims at solving the drawback of $f_{\mathrm{LIN}}$. It is more tolerant to higher costs than expected at the beginning of the path, and less tolerant as the message reaches the end of the PG. Section 8 discusses the performance of the crankback prediction mechanism under three different types of $g(z)$ functions: concave, convex, and linear.

The last function we propose is $f_{\mathrm{CONV}}$. Its aim is to find out whether the chances to cross the rest of the PG using the allocated quota, based on the aggregated estimations for the rest of the route, are high enough with respect to a given threshold. The properties of $f_{\mathrm{CONV}}$ are explained in Section 7.

The $f$ functions are used to compute and recursively divide the quota to the PGs along the planned route. This is done using the fact that at no point in time should the quota already spent in the $(i+1)$-level PG, namely $\sigma^{i+1}$, exceed the threshold computed by $f$. Given function $f$, line A.2 of the algorithm in Fig. 4 will be as follows:

$$\text{A.2)} \quad \alpha^i \leftarrow f(\alpha^{i+1}, \epsilon^{i+1}, p^{i+1}, \ldots) - \sigma^{i+1}. \tag{5.4}$$

To demonstrate the operation of the algorithm with $f$, suppose that when the message enters D.2.3 in PG(D.2) and step A.2 of the algorithm is about to be performed, $\sigma^2 = 27$ holds, namely 27 ms were spent for crossing PG(D.3). Suppose also that function $f_{\mathrm{LIN}}$ defined in Eq. (5.1) is considered. By Eq. (5.2) $f(\alpha^2 = 60, \epsilon^2 = \{\epsilon_{\mathrm{D.3}}, \epsilon_{\mathrm{D.2}}, \epsilon_{\mathrm{D.1}}\}, p^2 = 55$. This implies that crossing both PG(D.3) and PG(D.2) must cost no more than 55. Hence a quota of $\alpha^1 = f(\alpha^2, \epsilon^2, p^2, \ldots) - \sigma^2 = 55 - 27 = 28$ ms, should be allocated for crossing PG(D.2) in step A.2. If PG(D.2) is crossed using less than 28 ms, predicted crankback will not be performed as the message leaves PG(D.2) and gets to D.1.3.

## 6. CRANKBACK PREDICTION ACCORDING TO STATISTIC MEASURES

This section presents a crankback mechanism based on statistic rather than deterministic analysis. The idea is that every child PG of the active $i$ level PG advertises the *probability density function* (*pdf*) of its crossing delay. The advertised statistic information is then used for crankback prediction during each step of the setup procedure. The *pdf* statistics parameters can be advertised by the PNNI topology advertisement mechanism. For independent, but identically distributed delay *pdfs*, the end-to-end distribution function can be derived by convolution. The correlation coefficient of traffic on successive network links has been calculated in [38]. While some correlation exists, the degree of which seems to be small, the independence assumption appears valid for the cases where the target connection occupies only a small fraction of the total link capacity.

The work in [38], uses the accumulation algorithm to compute the end-to-end CTD. In this section we suggest to use the accumulative CTD function for deriving the crankback probability at each stage of the setup process. We propose an algorithm that evaluiates the probability for crossing the PG, or reaching the destination within the PG, using no more than the specified quota.

In what follows, we derive the statistical condition for crankback based on the *pdf* advertised by every node along the route. We consider an $i$-level PG that contains $N = |\epsilon^i|$ child PGs. The allocated quota for crossing this PG, namely $\alpha^i$, is determined on the basis of the algorithm in Fig. 4. For simplicity, we omit the level superscript $i$ notation from all expressions in the following. Using accumulative *pdf*, we derive the predicted crankback probability at each node along the route. If the calculated probability is higher than some threshold, predicted crankback is invoked. We start by defining the following terms:

- $pdf(j) =$ the probability density function advertised by the $j$th child of the considered $i$-level PG. This child will be referred to as PG($j$).
- $E(pdf(j))$ and $\mathrm{VAR}(pdf(j)) =$ the average and variance of $pdf(j)$.
- $\sigma(j) =$ the total amount of delay spent for crossing PG($j$).
- $\tau =$ the threshold probability that determines whether or not to invoke the predicted crankback.
- $\sigma(1 \cdots j) = \sum_{k=1}^{j} \sigma(k) =$ the actual accumulative delay from the first child PG of the considered $i$-level PG to the $j$th one. Hence, $\alpha - \sigma(1 \cdots j)$ indicates the quota left for crossing the rest of the active $i$-level PG after crossing its $j$th child PG.

The proposed algorithm uses convolution in order to derive the distribution function of the accumulative delay at each node for the rest of the route. To this end, we define for every $j$, $1 \leq j \leq N$,

$$pdf(j \cdots N) = pdf(j) \otimes pdf(j+1) \otimes \cdots \otimes pdf(N).$$

The method for deriving the crankback probability at each node along the path is described in Fig. 5. The average value of $pdf(j \cdots N)$ at each stage after crossing the $(j-1)$th child PG is given by

$$E(pdf(j \cdots N)) = \sum_{k=j}^{N} E(pdf(k)).$$

The solid line in Fig. 5 represents the actual quota left for crossing the rest of the route, namely, $\alpha - \sigma(1 \cdots (j-1))$, whereas the dashed line represents the

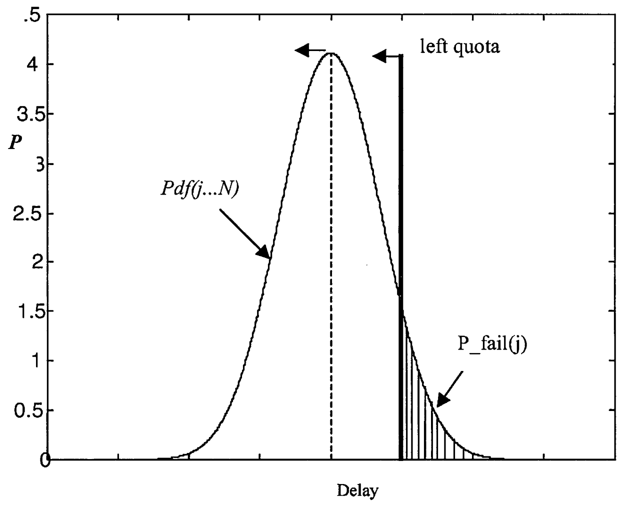


**Fig. 5.** Deriving the probability for routing failure from the accumulative *pdf*.

center of the accumulative *pdf*. The probability, $P\_\text{fail}(j)$, for consuming more than the remaining quota which would lead to crankback is given by the area bounded by the solid-line $pdf(j \cdots N)$. Hence, the following holds:

$$P\_\text{fail}(j) = \int_{\alpha-\sigma(1\cdots(j-1))}^{\infty} pdf(j \cdots N)(x)\,dx. \tag{6.1}$$

For a symmetrical *pdf* function this probability is larger than 0.5 when the solid-line is located left to the central line. Crankback prediction is invoked before PG($j$) is traversed if

$$P\_\text{fail}(j) \geq \tau$$

holds. As discussed later, selecting the right value of $\tau$ has a great impact on the success of the algorithm.

Equation (6.1) holds for a general distribution function. However, the algorithm can be simplified for a Gaussian distribution, in which case the *pdf* can be derived analytically because of the assumption that the delays in successive links along the route are independent. With this assumption, the variance of the accumulative *pdf* can be represented as the sum of all variances, namely,

$$\text{VAR}(pdf(j \cdots N)) = \sum_{k=j}^{N} \text{VAR}(pdf(k)). \tag{6.2}$$

By replacing the general *pdf* in Eq. (6.1) with the Gaussian distribution function and using Eq. (6.2) we get

$$\begin{aligned} P\_\text{fail}(j) &= \frac{1}{\sqrt{2\pi\ \text{VAR}(pdf(j\cdots N))}} \\ &\quad\times \int_{\alpha-\sigma(1\cdots(j-1))}^{\infty} \exp\left(\frac{(x-E(pdf(j\cdots N)))^2}{2\ \text{VAR}(pdf(j\cdots N))}\right)dx \\ &= 1-\Phi\left(\frac{\alpha-\sigma(1\cdots(j-1))-E(pdf(j\cdots N))}{2\ \text{VAR}(pdf(j\cdots N))}\right). \qquad (6.3) \end{aligned}$$

The results for Eq. (6.3) can be derived from the common probabilistic tables of the complementary error function.

Figure 6 is a sketch that demonstrates three representative scenarios that could take place during the operation of the crankback mechanism: the case when

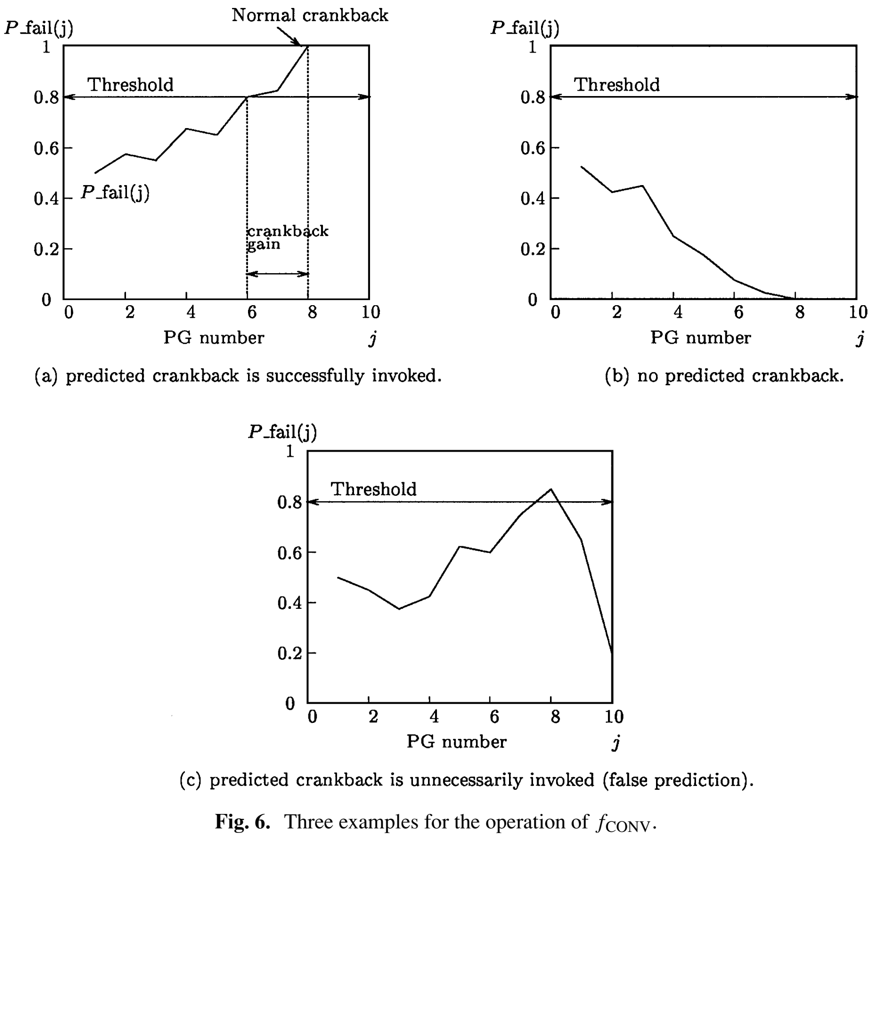


**Fig. 6.** Three examples for the operation of $f_{\text{CONV}}$.

predicted crankback is invoked successfully (a); the case when predicted crankback is not invoked (b); and the case when it is unnecessarily invoked (c). In these figures, the calculated value of $P\_\text{fail}(j)$ is presented as a function of the PG number along a path of 10 PGs. Figure 6(a) demonstrates a case of successful predicted crankback that takes place at the 6th PG when $P\_\text{fail}(j)$ reaches the threshold ($\tau$) which equals to 0.8. This saves the need to traverse PG 7 and PG 8 before a nonpredicted crankback is invoked. Figure 6(b) presents the case when the procedure makes a correct decision not to invoke crankback because $P\_\text{fail}(j)$ for $j = 1, \ldots, 10$ is smaller than the threshold $\tau$. Finally Fig. 6(c) depicts the case of false prediction (at the 8th PG).

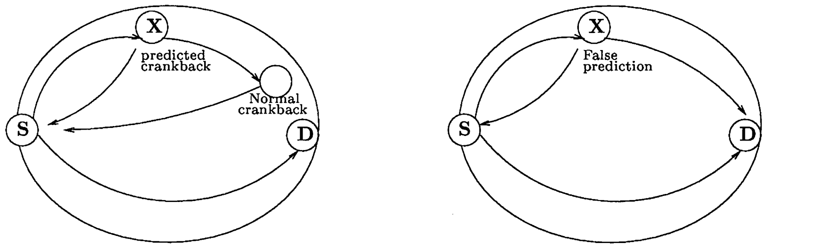
## 7. SIMULATION RESULTS

To study the performance of the proposed algorithms, we ran simulations on a randomly generated 3-level hierarchical network. In the simulation, each PG represents a fully connected graph that contains on the average 36 child PGs. These PGs are connected by randomly selected edges whose average number is 54, according to the following algorithm proposed in [39]. The nodes in the network are placed at random points in a two-dimensional grid. A link is randomly added between a pair of nodes using a probability that is proportional to the Euclidean distance between those nodes. After connectivity is achieved within a PG, more physical links are added in a manner that leaves the nodes connectivity degree as uniform as possible. The fraction of border nodes that are connected to neighboring PGs is kept fixed in every PG [40, 41]. The additive metric associated with every link is proportional to the Euclidean distance between the two end nodes.

The aggregation mechanism for additive metrics in our simulation model is as follows. We compute the average delay required for crossing each PG between all border node pairs. In order to simulate topology changes that are not reflected in the average delay, the advertised delay for that PG is then normally distributed around the computed delay.

We start with an advertised delay that is normally distributed around the computed average delay $\mu$, with a variance of 10% of that delay ($\sigma^2 = 0.1\ \mu$), and then continue with variances of 20% and 30% of the average delay. Note that a higher variance represents nonaccurate aggregation, attributed to not-yet-advertised topological changes or to lossy aggregation. Confidence intervals of 97% are used to assure the convergence of the simulation results.

To understand how we measure the performance of our scheme, consider a message routed from node $S$ to node $D$ in the PG depicted in Fig. 7. Let the first route selected by the routing algorithm be the upper one. Assume that node $X$ performs crankback prediction after $a$ links are traversed by the message. The setup message returns back to its originator, after traversing the same $a$ links in the reverse direction. It is then routed over an alternative route that consists of $c$



(a) Gain $= 2 \cdot b$ due to correct prediction. (b) Loss $= a + c - d \simeq 2 \cdot a$ due to false alarm.

**Fig. 7.** Illustration of gain and loss computation.

links. As shown in Fig. 7(a), without crankback prediction $2b$ extra links would be traversed if regular crankback is detected. Figure 7(b) illustrates the case where the route from $S$ to $D$ via node $X$ is a valid route, and crankback prediction leads to false alarm. The penalty in such a case is $a + c - d$. Since $c \simeq a + b$, the penalty is approximately $2a$.

We measured the gain and loss of crankback prediction by generating random source/destination pairs. We use *CPG(X)* to denote the *crankback prediction gained* when function $X$ is used to determine the threshold. *CPG(X)* is computed as a function of the gain over the total cost without prediction. This function is multiplied by 100 to represent percentage. Therefore, assuming that every source destination pair $i(S_i, D_i)$ is associated with a triple $\{a_i, b_i, c_i\}$,

$$CPG(X) = 100 \times \frac{\sum_i 2b_i}{\sum_i 2(a_i + b_i) + c_i}.$$

In the same way, *FPL(X)* denotes the *false prediction loss* under function $X$. Namely,

$$FPL(X) = 100 \times \frac{\sum_i (a_i + c_i - d_i)}{\sum_i a_i + d_i} \simeq 100 \times \frac{\sum_i 2a_i}{\sum_i a_i + d_i}.$$

Note, however, that in practice there are cases when several predictions and false alarms take place during the same routing session. These cases are addressed in our simulation program although they are not reflected in the above two equations. The absolute gain from the new scheme under function $X$ is *CPG(X) − FPL(X)*. However, by distinguishing between *CPG* and *FPL*, we will be able to analyze interesting trends that are not reflected by the absolute gain.

We used in our simulations three functions for $f_{\mathrm{DECAY}}$ (recall Eq. (5.3), where $g(z)$ is discussed):

$$g_1(z) = M - (M-1)z, \quad g_2(z) = M - (M-1)z^3, \quad \text{and}$$
$$g_3(z) = 1 + (M-1)(1-z)^3.$$

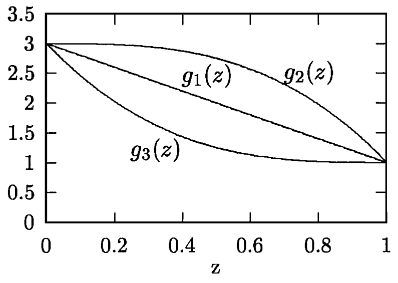


**Fig. 8.** The value of $g_i(z)$ for the particular case where $M = 3$.

These functions are referred to as $f_{\text{DECAY\_1}}$, $f_{\text{DECAY\_2}}$, and $f_{\text{DECAY\_3}}$ respectively. They are depicted in Fig. 8. The parameter $M$ represents the tolerance factor and it equals to $g_i(0)$. For example, $M = 3$ holds for all the three functions in Fig. 8.

We used values between $M = 1$ and $M = 4$ in $f_{\text{DECAY}}$. Note that when $M = 1$, $f_{\text{DECAY}}$ is equivalent to $f_{\text{LIN}}$, since for $0 \leq z \leq 1$, $g(z) = 1$. The results for the case where the aggregated delay value is normally distributed around the average delay with a variance of 10% of that delay are depicted in Fig. 9(a)–(c) for $f_{\text{DECAY\_1}}$, $f_{\text{DECAY\_2}}$, and $f_{\text{DECAY\_3}}$ respectively. For the case where the aggregation variance is 30% of the mean delay, the results are depicted in Fig. 10(a)–(c). Figure 11(a) compares the performance of $f_{\text{DECAY\_1}}$ in the cases where the advertised delay is normally distributed around the average computed delay with a variance of 10%, 20%, and 30% of that delay.

As expected, for the three considered functions, there is a higher tolerance to higher than advertised costs at the beginning of the route. This tolerance declines as $z$ grows, namely, as the message advance along the path. However, $f_{\text{DECAY\_3}}$ allows tolerance only at the very beginning and declines faster than $f_{\text{DECAY\_1}}$ and $f_{\text{DECAY\_2}}$ when the routing proceeds (namely, for $\frac{1}{2} \leq z < 1$, $1 \simeq g_3(z) < g_1(z) < g_2(z)$). On the other hand, a crankback mechanism that uses $f_{\text{DECAY\_2}}$ remains tolerant even when a significant portion of the route is reached (namely, for $\frac{1}{2} \leq z \leq \frac{4}{5}$, $1 < g_2(z) > g_1(z) > g_3(z)$). Consequentially,

$$CPG(f_{\text{DECAY\_3}}) \geq CPG(f_{\text{DECAY\_1}}) \geq CPG(f_{\text{DECAY\_2}}).$$

The reason is that $f_{\text{DECAY\_3}}$ is the most aggressive in terms of crankback prediction. There are cases where it invokes predicted crankback while the other two functions do not, but there are not opposite cases. For similar reasons,

$$FPL(f_{\text{DECAY\_3}}) \geq FPL(f_{\text{DECAY\_1}}) \geq FPL(f_{\text{DECAY\_2}}).$$

In a similar manner, we simulated the crankback prediction mechanism using $f_{\text{CONV}}$ as described in Section 7 with threshold values ($\tau$) ranging from 0.05 to 1.

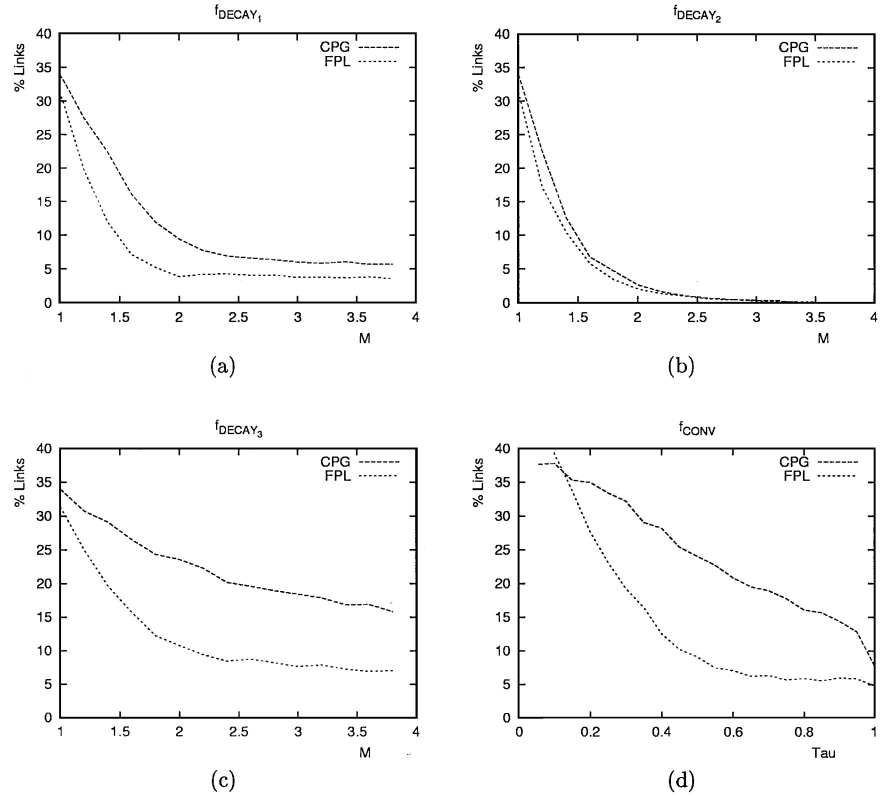


**Fig. 9.** Benefit of crankback prediction where aggregation variance is 10% of the delay.

Crankback is performed as soon as $P\_\text{fail}(j) \geq \tau$. Note that threshold $\tau = 0$ has no physical meaning since this implies that crankback prediction is performed in any case. The results for this function are presented in Figs. 9(d) and 10(d). Figure 11(b) compares the performance of $f_{\text{CONV}}$ for the three different aggregation modes.

As expected, and can easily be observed from the graphs, for a certain value of $M$ or $\tau$, of $f_{\text{DECAY}}$ and $f_{\text{CONV}}$ respectively, when CPG increases FPL increases as well. This occurs since in cases where predicted crankback is intolerant to higher-than-expected cost, false predictions are more likely to take place. When $M$ and $\tau$ are too small, the intolerance to higher-than-expected costs is high and the probability of false prediction increases.

From Figs. 9 and 10 it is evident that regardless of the crankback function used, CPG dramatically increases when topology information is not precise. This is not surprising since crankback prediction reduces the penalty of aggregation fault. As can be seen clearly in Fig. 11(a), when the aggregation method becomes

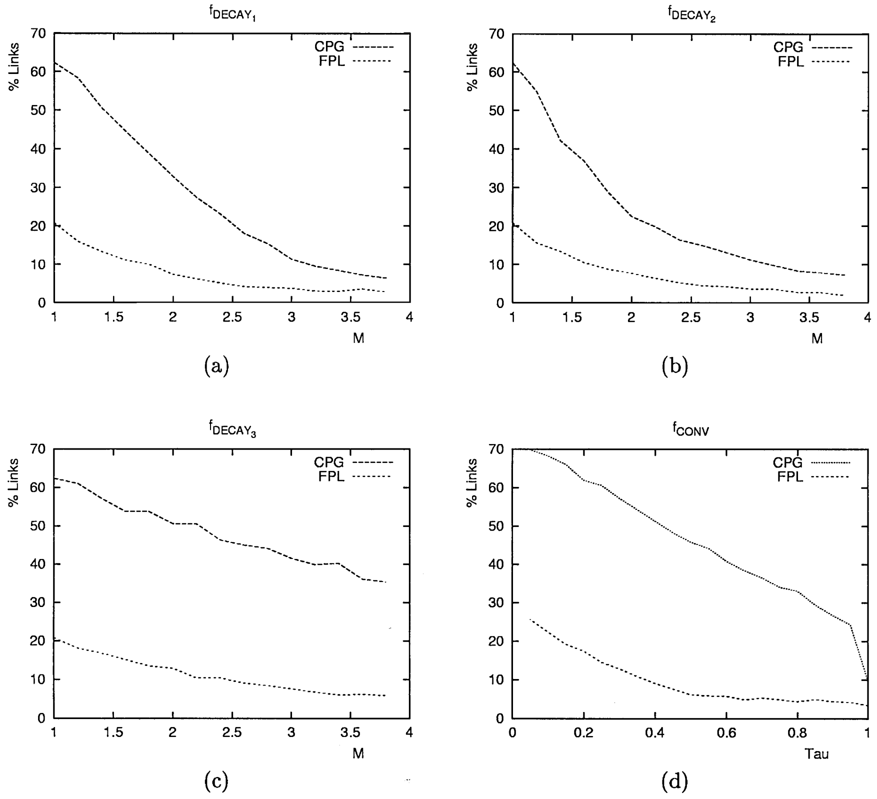


**Fig. 10.** Benefit of crankback prediction where aggregation variance is 30% of the delay.

more inaccurate, FPL decreases slightly for small values of $M$. We explain this by the fact that when nonaccurate metrics are involved and when the tolerance is low ($M \simeq 1$), predicted crankbacks, including false ones, are likely to be triggered at early stages. Since the penalty of false prediction is proportional to the portion of the route already traversed, as described in Fig. 7(b), early triggered false predictions are less harmful. Hence, when the aggregation is not accurate, the optimal decay function should be completely intolerant, and $M = 1$ should be used.

On the other hand, when the aggregation is accurate, as in Fig. 9, FPL is higher when the tolerance is low ($M \simeq 1$). Regardless of the aggregation scheme, $CPG(f_{\text{DECAY\_1}})$ and $CPG(f_{\text{DECAY\_2}})$ decline faster than $CPG(f_{\text{DECAY\_3}})$ for large values of $M$, because for such values the benefits of crankback prediction is reduced as the setup proceeds toward the destination. Hence, for $2 \leq M \leq 4$, $CPG(f_{\text{DECAY\_3}})$ is much larger than $CPG(f_{\text{DECAY\_1}})$ and $CPG(f_{\text{DECAY\_2}})$. In contrast, for large values of $M$, $FPL(f_{\text{DECAY\_3}})$ is not that much higher than

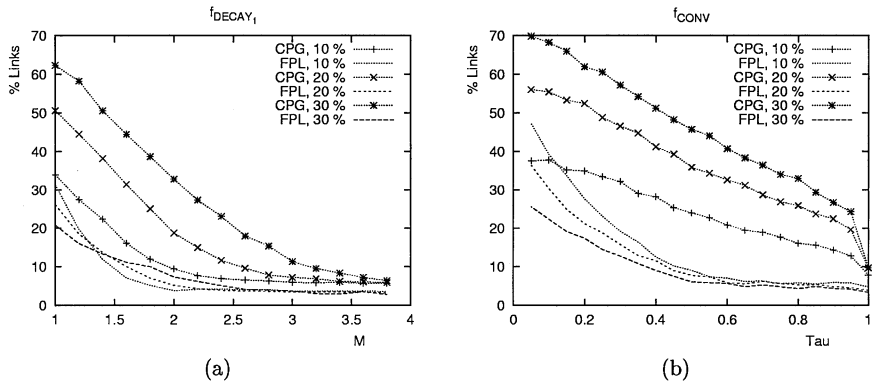


**Fig. 11.** The benefit of crankback prediction under several aggregation variances.

$FPL(f_{\text{DECAY\_2}})$ and $FPL(f_{\text{DECAY\_1}})$. Therefore, the absolute gain from the decay prediction mechanism, namely CPG minus FPL, is obtained by using $f_{\text{DECAY\_3}}$ with $M \geq 2$. Therefore, *in networks where the aggregation is accurate, a deterministic (decay) prediction function should use a steep (convex) tolerance decay function. When the aggregation is lossy, a deterministic prediction function should use a constant tolerance without any decay function.*

The results for $f_{\text{CONV}}$ are very similar to those of $f_{\text{DECAY}}$. However, *since* $f_{\text{CONV}}$ *uses advertised information that is not used* in $f_{\text{DECAY}}$, *it achieves the best results*. The benefit of $f_{\text{CONV}}$ over $f_{\text{DECAY\_3}}$ is reduced in cases of lossy aggregation since the prediction function uses a convolution of inaccurate *pdf*s.

We did not observe any significant changes in CPG and FPL by changing the topology creation algorithm. We do believe, however, that because the inaccuracy is aggregated over the hierarchy levels [11, 23, 24], for larger networks the advantage of our scheme increases. However, such large-scale simulation requires much more processing power than we have today.

## 8. CONCLUSIONS AND FURTHER WORK

This paper proposed a new scheme for crankback prediction in hierarchical networks. The proposed scheme triggers crankback before the QoS quota is really exhausted. The main purpose of the scheme is to reduce the connection setup delay, the processing time and the communication load associated with a setup that cannot be completed. In addition, the scheme reduces the bandwidth and buffer resources allocated to the unused portions of the VC during the connection setup time.

In order to predict crankback, the scheme subdivides the QoS quota recursively among the network nodes. When the allocated fraction of the quota is exhausted, crankback is predicted. Several functions for quota subdivision were presented and discussed. Three functions use deterministic considerations, while the forth one uses convolution-based probabilistic considerations by computing the probability density function of the required quota for the remaining portion

We investigated by simulations the performances of the hierarchical crankback algorithm using these functions, and found that the probabilistic approach yields the best performance since it requires the PNNI nodes to advertise statistic measures. We found that our scheme always reduces the connection setup delay and the associated processing and communication load, and that the results are more significant for the networks that use lossy aggregation.

In this paper we introduced the concept of crankback prediction in the context of hierarchical networks, specifically PNNI, which is the most comprehensive model for hierarchical routing. Nevertheless, the scope of the scheme could be extended to IP/MPLS networks. One of the design goals of MPLS tag-switching is to improve the scaling properties of the IP routing system. The routing architecture used today in the Internet models the networks as a collection of routing domains, where routing within each domain is provided by means of an intradomain routing protocol, like RIP and OSPF, and routing across multiple domains is provided by means of an interdomain routing protocols (BGP). To support forwarding in the presence of a hierarchcal network organization, MPLS allows a packet to carry multiple tags, arranged in a stack. A possible way to use this stack is as follows. When a packet is forwarded from one IP routing domain to another, the tag stack carried by the packet will contain a single tag. When a packet is forwarded through a transit routing domain, the stack will contain two tags: the top one for the intradomain routing, and the bottom one for the interdomain routing.

Extensions for crankback routing in both LDP and RSVP environments are currently being proposed to the IETF [42–44]. In these proposals, a request can be retried on an alternate path that detours around upon a setup failure. The crankback prediction mechanism proposed in this paper can be extended to support MPLS crankback as well. However, we believe that the extra complexity can be justified only when the full concept of hierarchcal routing is adopted for IP routing, something which many researchers believe is unavoidable.

## REFERENCES

**Au: Kindly update this reference (if possible)**

1. C. Huitema, *Routing in the Internet*, Prentice Hall, 1995.
2. L. Kleinrock and F. Kamoun, Hierarchical routing for large networks. *Computer Networks*, Vol. 1, No. 3, pp. 155–174, 1977.
3. B. Awerbuch, A. Bar-Noy, N. Linial, and D. Peleg, Compact distributed data structures for adaptive routing (extended abstract). In *Proceedings of the Twenty First Annual ACM Symposium on Theory of Computing*, Seattle, WA, May 15–17, 1989, pp. 479–489.
4. B. Awerbuch, A. Bar-Noy, N. Linial, and D. Peleg, Improved routing strategies with succinct tables. *Journal of Algorithms*, Vol. 11, No. 3, pp. 307–341, 1990.

P1: Vendor/FZN
pp577-jons-378729 JONS.cls July 9, 2002 17:40

5. B. Awerbuch and D. Peleg, Routing with polynomial communication-space tradeoff. Technical Memo MIT/LCS/TM-411, Massachusetts Institute of Technology, Laboratory for Computer Science, July 1989.
6. B. Awerbuch and D. Peleg, Sparse partitions. In IEEE (ed.), *Proceedings of the 31st Annual Symposium on Foundations of Computer Science*, St. Louis, MS, Oct. 1990, IEEE Computer Society Press, Silver Spring, MD, pp. 503–513.
7. R. Cohen, E. Felstaine and R. Emek, A framework for multicast routing in hierarchical ATM networks. In *IEEE INFOCOM*, 2000.
8. D. Peleg and E. Upfal, A trade-off between space and efficiency for routing tables. *JACM*, Vol. 36, No. 3, pp. 510–530, 1989.
9. ATM Forum PNNI SWG 94-0471R13, *ATM Forum PNNI Draft Specifications*, Mar. 1996.
10. R. Cohen and A. Segall, Connection management and rerouting in ATM networks. In *IEEE INFOCOM*, 1994.
11. B. Awerbuch and Y. Shavitt, Topology aggregation for directed graph. Technical Report 98-14, DIMACS, Feb. 23, 1998.
12. W. C. Lee, Spanning tree method for link state aggregation in large communication networks. In *IEEE INFOCOM*, 1995.
13. W. C. Lee, Topology aggregation for hierarchical routing in ATM networks. In *ACM SIGCOMM*, April 1995.
14. W. C. Lee, Minimum equivalent subspanner algorithms for topology aggregation in ATM networks. In *2nd International Conference on ATM (ICATM '99)*, Colmar, France, June 1999, pp. 351–359.
15. F. Hao and E. W. Zegura, On scalable QoS routing: Performance evaluation of topology aggregation. In *IEEE INFOCOM*, Tel Aviv, Israel, Mar. 2000.
16. I. Althofer, G. Das, D. Dobkin, D. Joseph, and J. Soares, On sparse spanners of weighted graphs, *GEOMETRY: Discrete and Computational Geometry*, Vol. 9, No. 1, pp. 81–100, 1993.
17. D. Peleg and A. A. Schaffer, Graph spanners. *Journal of Graph Theory*, Vol. 13, No. 1, pp. 99–116, 1989.
18. R. Guerin and A. Orda, Qos-based routing in networks with inaccurate information: Theory and algorithms. In *IEEE INFOCOM*, Kobe, Japan, Apr. 1997, pp 75–83.
19. D. Lorenz and A. Orda, QoS routing in networks with uncertain parameters. *IEEE/ACM Transactions on Networking*, Vol. 6, No. 6, pp. 768–778, 1998.
20. L. Guo and I. Matta, On state aggregation for scalable qos routing. In *Proceedings of the 1998 IEEE ATM Workshop*, IEEE, May 1998, pp. 306–314.
21. I. Ilias, Optimal PNNI complex node representations for restrictive costs and minimal path computation time. *IEEE/ACM Transactions on Networking*, Vol. 8, No. 4, pp. 493–506, 2000.
22. P. van Mieghem, Topology information condensation in hierarchical networks. *Computer Networks*, Vol. 31, No. 20, pp. 2115–2137, 1999.
23. B. Awerbuch, Y. Du, B. Khan, and Y. Shavitt, Routing through networks with hierarchical topology aggregation. *Journal of High Speed Networks*, Vol. 7, No. 1, 1998.
24. B. Awerbuch, Y. Du, B. Khan, and Y. Shavitt, Routing through teranode networks with topology aggregation. In *ISCCS'98*, June 1998.
25. B. Awerbuch, Y. Du, and Y. Shavitt, Stars: A simulator for performance study of aggregation based hierarchical routing. In *SCS/IEEE SPECTS'98*, July 1998.
26. B. Awerbuch, Y. Du, and Y. Shavitt, The effect of network hierarchy structure on performance of ATM PNNI hierarchical routing. *Computer Communications*, Vol. 23, No. 10, pp. 980–986, 2000.
27. R. Krishnan, R. Ramanathan, and M. Steenstrup, Optimization algorithms for large self-structuring networks. In *IEEE INFOCOM*, New York, Mar. 1999.
28. D. G. Thaler and C. V. Ravishanka, Distributed top-down hierarchy construction. In *IEEE INFOCOM*, San Francisco, CA, Mar./Apr. 1998, p. 693.

**Au: Kindly provide the place where this proceed was hold. Also provide the page no(s)., if possible. (Also see all such reference.)**

29. G. Apostolopoulos, R. Guerin, S. Kamat, and S. K. Tripathi, Quality of service based routing: A performance perspective. In *ACM SIGCOMM*, 1998.
30. S. Bahk and M. El Zarki, Dynamic multi-path routing and how it compares with other dynamic routing algorithms for high speed wide area networks. In *ACM SIGCOMM*, Baltimore, MD, Aug. 1992. *Computer Communication Review*, Vol. 22, No. 4, pp. 53–64, 1992.
31. I. Cidon, R. Rom, and Y. Shavitt, Multi-path routing combined with resource reservation. In *IEEE INFOCOM*, Kobe, Japan, Apr. 1997, pp. 92–100.
32. I. Cidon, R. Rom, and Y. Shavitt, Analysis of multi-path routing. *IEEE/ACM Transactions on Networking*, Vol. 7, No. 6, pp. 885–896, 1999.
33. T. Korkmaz and M. Krunz, Source-oriented topology aggregation with multiple qos parameters in hierarchical atm networks. *ACM Transactions on Modeling and Computer Simulation*, Vol. 10, No. 4, pp. 295–325, 2000.
34. J. Garcia-Luna-Aceves and J. Behrens, Distributed, scalable routing based on vectors of link states. *IEEE Journal on Selected Areas of Communication*, Vol. 13, No. 8, 1995.
35. M. Montgomery and G. de Veciana, Hierarchical source routing through clouds. In *IEEE INFOCOM*, San Francisco, CA, Mar./Apr. 1998, p. 685.
36. M. Montgomery and G. de Veciana, Hierarchical source routing using implied costs. *Computer Networks*, Vol. 34, No. 3, pp. 379–397, 2000.

**Au: Kindly provid the page no(s) in all journals type refs.**

37. C. Alaettinoglu and A. Shankar, The viewserver hierarchy for interdomain routing: Protocols and evaluation, *IEEE Journal on Selected Areas of Communication*, Vol. 13, No. 8, 1995.
38. S. Wright, D. Jarrett, D. Kataria, and Y. Viniotis, Accumulation algorithm for CDV. Technical Report 95-0556, ATM Forum, 1995.
39. B. Waxman, Routing of multipoint connections. *IEEE Journal on Selected Areas of Communication*, Vol. 1, 1988.
40. E. Felstaine and R. Cohen, On the distribution of routing computation in hierarchical ATM networks. *IEEE/ACM Transactions on Networking*, Vol. 7, No. 6, 1999.

**Au: 18, 25, and 33 (new refs. 45, 46, 47 respectively) are not cited anywhere in the text kindly cite them at the appropriate place as delete them from the reference list.**
**Au: Kindly update 42, 43, 44, 46 Refs. if possible.**

41. E. W. Zegura, K. Calvert, and S. Bhattacharjee, How to model an internetwork. In *IEEE INFOCOM*, Apr. 1996.
42. A. Iwata, N. Fujita, G. Ash, and A. Farrel, Crankback routing extensions for MPLS signaling. Internet Draft, Internet Engineering Task Force, Nov. 2001, work in progress.
43. A. Iwata, N. Fujita, G. Ash, and A. Farrel, Crankback routing extensions for MPLS signaling with RSVP-TE. Internet Draft, Internet Engineering Task Force, Jan. 2001, work in progress.
44. S. Venkatachalam, D. Papadimitriou, and S. Dharanikota, A framework for the LSP setup across IGP areas for MPLS traffic engineering. Internet Draft, Internet Engineering Task Force, Nov. 2001, work in progress.
45. D. Farinacci, Y. Rekhter, D. Meyer, P. Lothberg, H. Kilmer, and J. Hall, Multicast Source Discovery Protocol (MSDP). Internet Draft, draft-itef-msdpspec*.txt, Feb. 2000.
46. B. Hinden, M. O'Dell, and S. Deering, An IPv6 aggregatable global unicast address format. Internet Draft, Internet Engineering Task Force, July 1998, work in progress.
47. S. Kumart, P. Radoslavov, D. G. Thaler, C. V. Alaettinoglu, D. Estrin, and M. Handley, The MASC/BGMP architecture for inter-domain multicast routing. In *ACM SIGCOMM*, Sept. 1998.

**Eyal Felstaine** received the BSc degree in electrical engineering and MSc degree in Computer Science from the Technion – Israel Institute of Technology, Haifa, in 1995 and 1998, respectively, where he is currently completing his PhD degree in the Department of Computer Science. He is currently the President of SANRAD, which he founded in 2000. His research interests include hierarchical networks, scalable multicast routing, QoS, policy-based networking, and algorithms for packet classification. He has published numerous papers and was the recipient of the Gutwirth and the Intel-Technion Award for research excellence in 1998 and 1999.

**Reuven Cohen** received the BSc, MSc, and PhD degrees in Computer Science from the Technion, Israel Institute of Technology in 1986, 1988, and 1991, respectively. From 1991 to 1993, he was with IBM T.J. Watchild Research Center, working on protocols for high-speed networks. Since 1993, he has been with the Department of Computer Science at the Technion, where he is now an associate processor. He has also consulted for numerous companies, including Hewlett-Packard, ECI Telecom, and Terayon, mainly in the context of protocols and architectures for broadband access networks. His most recent work focuses on the design and evaluation of routing, multicast and transport protocols in networks, and on technologies for broadband access networks.

**Ofer Hadar** received the BSc, MSc (cum laude), and PhD degrees from Ben-Gurion University of the Negev, Israel, in 1990, 1992, and 1997, respectively, all in electrical and computer engineering. From October 1997 to March 1999, he was a Postdoctoral Fellow in the Department of Computer Science at the Technion, working on high-speed networks. Currently he is a faculty member at the Communication Systems Engineering Department at Ben-Gurion University of the Negev. His research interests include image compression, video compression, routing in ATM networks, packet video, transmission of video over IP networks, and video rate smoothing and multiplexing. He also works as a consultant for Scopus Ltd. in the area of video compression and transmission over satellite network.