An Overview of Quality-of-Service Routing for the Next Generation High-Speed Networks: Problems and Solutions *

Shigang Chen, Klara Nahrstedt †
Department of Computer Science
University of Illinois at Urbana-Champaign
{s-chen5, klara}@cs.uiuc.edu

Abstract

The up-coming Gbps high-speed networks are expected to support a wide range of communication-intensive, real-time multimedia applications. The quality-of-service (QoS) requirements for the timely delivery of digitized audio-visual information raise new challenges for the development of integrated-service broadband networks. One of the key issues is QoS routing, which allows selecting network routes with sufficient resources for requested QoS parameters. The goal of QoS routing solutions is two-folded: satisfying the QoS requirements for every admitted connection and achieving global efficiency in resource utilization. In this paper we give an overview of the QoS routing problem as well as existing solutions. Many unicast/multicast QoS routing algorithms were published recently, and they work with a variety of QoS requirements and resource constraints. Overall they can be partitioned into three broad classes: (1) source routing, (2) distributed routing and (3) hierarchical routing algorithms, based on different network state models. We present the strengths and weaknesses of different routing strategies, outline the challenges, discuss basic algorithms in each routing class, classify and compare them, and point out possible future directions in the QoS routing area.

1 Introduction

The timely delivery of digitized audio-visual information over local or wide area networks is now becoming realistic, thanks to the fruitful research in high-speed networks, image processing, and vedio/audio compression. Communication-intensive, real-time multimedia applications raise new challenges for the network research and development. In the current Internet, data packets of a session may follow different paths to the destination. The network resources, e.g. switch buffer and link bandwidth, are fairly shared by packets from different sessions. However, this architecture does not meet the requirements of the future integrated-service networks that carry heterogeneous data traffic. First, it does not support resource reservation which is vital for the provision of guaranteed end-to-end performance (bounded delay, delay jitter and loss ratio). Second, data packets may experience unpredictable delay and arrive at the destination out of order which is undesirable for continuous real-time media such as audio and video. Hence, the next generation of high-speed wide-area networks will be *connection-oriented* [1]. This paper focuses on the routing problem of the connection establishment, i.e., finding a network path (tree) that meets the requirement of a connection and makes the efficient use of the resources.

The notion of Quality-of-Service (QoS) has been proposed

*This work was supported by the Airforce grant under contract number F30602-97-2-0121 and the National Science Foundation Career grant under contract number NSF CCR 96-23867.

†Please address all correspondences to Klara Nahrstedt at Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL 61801, phone: (217) 244-6624, fax: (217) 244-6869.

[1] At the transport layer, a connection (call) means the logical association of the end users and the correct, ordered delivery of data. At the network layer, a unicast (multicast) connection means a network path (tree) consisting of switches and links which connects two (a group of) end users. Data packets (or cells) of the same connection are sent along the path (tree) in the FIFO order.

to capture the qualitatively or quantitatively defined performance contract between the service provider and the user applications. The QoS requirement of a connection is given as a set of constraints, which can be either *link constraints* or *path (tree) constraints* [29]. A *link constraint* specifies the restriction on the use of links. For instance, a *bandwidth constraint* of a unicast connection requires that the links composing the path must have certain amount of free bandwidth available. On the other hand, a *path (tree) constraint* specifies the requirement on the entire path (multicast tree). For instance, a *delay constraint* of a multicast connection requires that the longest end-to-end delay from the sender to any receiver in the tree must not exceed an upper bound.

A *feasible path (tree)* is one that has sufficient residual (unused) resources to satisfy the QoS constraints of a connection. The basic function of QoS routing is to find such a feasible path. In addition, most QoS routing algorithms consider the optimization of resource utilization, measured by an abstract metric *cost*. The cost of a link can be defined in dollars or as a function of the buffer or bandwidth utilization. The cost of a path (tree) is the total cost of all links on the path (tree). The optimization problem is to find the least-cost path (tree) among all feasible paths (trees).

The problem of QoS routing is difficult due to a number of reasons. First, distributed applications such as teleconference, video-on-demand, Internet phone and web-based games have very diverse QoS constraints on delay, delay jitter, lose ratio, bandwidth, etc. Multiple constraints often make the routing problem intractable. In particular, finding the least-cost path with one path constraint or finding a feasible path with two independent path constraints is NP-complete [18]. Second, any future integrated-service network is likely to carry both QoS traffic and best-effort traffic, which makes the issue of performance optimization complicated. It is hard to determine the best operating point for both types of traffic if their distributions are independent. Although the QoS traffic will not be affected due to resource reservation, the throughput of the best-effort traffic will suffer if the overall traffic distribution is misjudged. Third, the network state changes dynamically due to transient load fluctuation, connections in and out, and links up and down. The growing network size makes it increasingly difficult to gather up-to-date state information in a dynamic environment, particularly when wireless communication is involved. The performance of a QoS routing algorithm can be seriously degraded if the state information being used is outdated.

Many QoS routing algorithms have been proposed recently with a variety of constraints considered. The purpose of this paper is to provide a survey on the recent development in this area. In the following, we present different routing problems, their challenges, the routing strategies, the classification and comparison of the existing routing algorithms, and possible future directions. We refer to the QoS routing simply by "routing", unless it is necessary to make clear distinguishment from the best-effort routing.

2 Weighted Graph Model

A network can be modeled as a graph $\langle V, E \rangle$. Nodes (V) of the graph represent switches, routers and hosts. Edges (E) represent the communication links. The edges are undirected only if the communication links are always symmetric, i.e., each link has the same properties (capacity, propagation delay, etc) and the same traffic volume in both directions. For real networks where the communication links are asymmetric, every link is represented by two directed edges in opposite directions. It should be noted that, though the examples in this paper use the undirected graphs for fewer edges, most routing algorithms under discussion were designed for asymmetric networks.

Every link has a state measured by the QoS metrics of concern. In Figure 1, the link state is a triple consisting of residual bandwidth, delay and cost. Every node also has a state, which can be either measured independently or, as it does in this paper, combined into the state of the adjacent links. For the latter case, the residual bandwidth is the minimal of the link bandwidth and the CPU bandwidth in terms

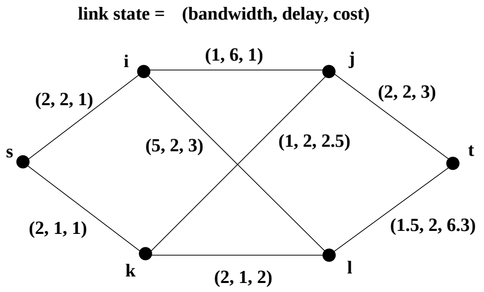


Figure 1: Network state

destination	i	j	k	l	t
bandwidth	2	1.5	2	2	1.5
next-hop	i	k	k	i	i

destination	i	j	k	l	t
delay	2	3	1	2	4
next-hop	i	k	k	k	k

destination	i	j	k	l	t
cost	1	2	1	3	5
next-hop	i	i	k	k	i

Figure 2: Global state in distance vectors at node s

of the maximum rate at which the node can pump data into the link; the delay of a link consists of the link propagation delay and the queueing delay at the node; the cost of a link is determined by the total resource consumption at the link and the node.

3 Maintenance of State Information

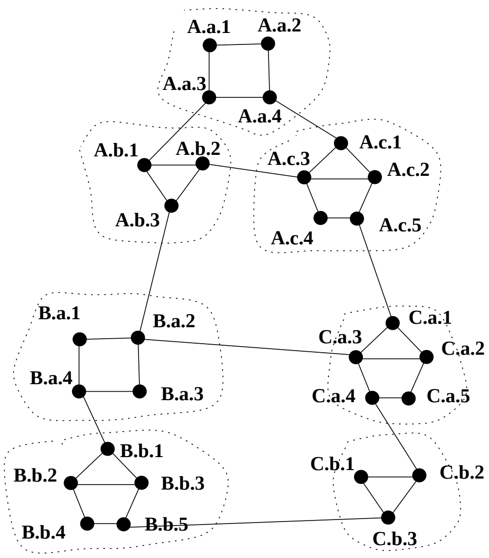
Routing consists of two basic tasks. The first task is to collect the state information and keep it up-to-date. The second task is to find a feasible path for the new connection based on the collected information. The performance of any routing algorithm directly depends on how well the first task is solved.

Local state: Each node is assumed to maintain its up-to-date local state, including the queueing and propagation delay, residual bandwidth of the outgoing links, and the availabilities of other resources, from which the QoS parameters can be calculated.

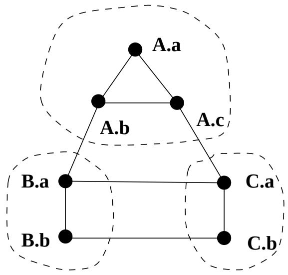
Global state: The combination of the local states of all nodes is called a global state. Every node is able to maintain the global state by either a link-state protocol [36, 34, 48] or a distance-vector protocol [21, 37, 31], which exchanges the local states among the nodes periodically. The link-state protocols broadcast the local state of every node to every other node so that each node knows the topology of the network and the state of every link (Figure 1). The distance-vector protocols periodically exchange the distance vectors among the adjacent nodes. A distance vector has an entry for every possible destination, consisting of the property of the best path and the next node on the best path (Figure 2).

The global state kept by a node can be viewed as the result of a distributed snapshot of the network state. It is always an approximation of the current network state due to the unnegligible delay of propagating local states. As the network size grows, the imprecision increases.

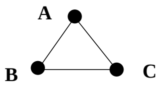
Aggregated (partial) global state: A common approach of achieving scalability is to reduce the size of the global state by aggregating information according to the hierarchical structure of large networks. Figure 3 shows the hierarchical model used by [16, 20]. In Figure 3 (a), nodes are clustered into the first-level groups. The nodes with at least one link crossing two groups are called *border nodes*. In Figure 3 (b), each group is represented by a *logical node*. A physical node in the group is elected to act on behalf of the logical node and store the higher-level state information. The links connecting logical nodes are *logical links*. The logical nodes are further clustered to form higher-level groups, which are abstracted by higher-level logical nodes, as shown in Figure 3 (c). Figure 3 (d) presents the overall clustering. On each hierarchy level, the nodes in a group are called the *children* of the logical node representing the group; the logical node is called *the parent*. An ancestor of a node is either its parent or an ancestor's parent. We have used the simplest topology aggregation, which abstracts a group by a single logical node. There are other types of aggregation using different simple topologies to replace a group, and their performance was studied in [3].



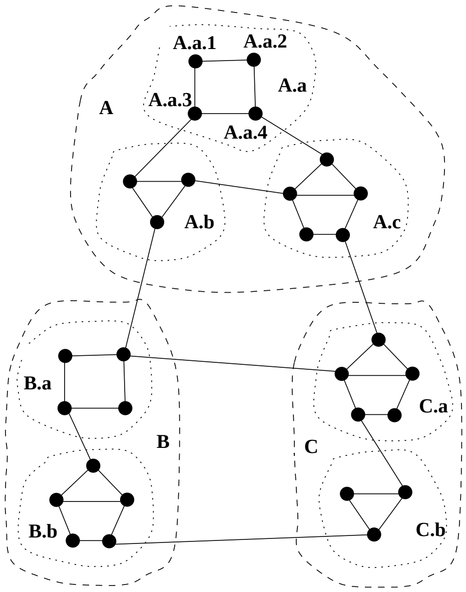
(a) Physical network



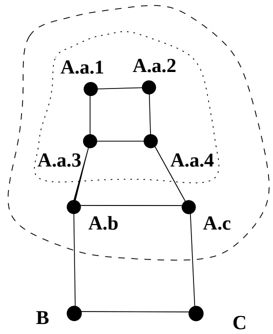
(b) First-level abstraction



(c) Second-level abstraction



(d) clustering



(e) the network image viewed by node A.a.1

Figure 3: Hierarchical network model

Each physical node maintains an *aggregated network image*, which stores different portions of the network in different details. More specifically, the image is derived by starting from the highest hierarchy level and recursively replacing the ancestor of the node with the corresponding lower-level group. The image maintained at node $A.a.1$ is shown in Figure 3 (e). As the network topology is aggregated, the state information is aggregated as well. The state of each logical link is the combination of the states of many lower-level links. The link-state algorithm can be extended to collect the aggregated state information for every node [16]. As the state is aggregated, the imprecision is also aggregated.

4 Routing Problems

The routing problems can be divided into two major classes: *unicast routing* and *multicast routing*. The unicast routing problem is defined as, given a source node s, a destination node t, a set of QoS constraints C and possibly an optimization goal, find the best feasible path from s to t, which satisfies C. The multicast routing problem is defined as, given a source node s, a set R of destination nodes, a set of constraints C and possibly an optimization goal, find the best feasible tree covering s and all nodes in R, which satisfies C. The two classes of routing problems are closely related, and the multicast routing can be viewed as a generalization of the unicast routing in many cases. These two problem classes can be further partitioned into sub-classes as follows.

4.1 Unicast Routing

For some QoS metrics such as residual bandwidth and residual buffer space, the state of a path is determined by the state of the bottleneck link. For example in Figure 1, the bandwidth of path $s \to i \to j \to t$ is 1, determined by the bandwidth of the bottleneck link (i, j). For these QoS metrics, two basic routing problems can be defined. One is called *link-optimization routing*, e.g., bandwidth-optimization routing, which is to find a path that has the largest bandwidth on the bottleneck link. Such a path is called the *widest path*. The other problem is called *link-constrained routing*, e.g., bandwidth-constrained routing, which is to find a path whose bottleneck bandwidth is above a required value. The link-optimization routing problem can be solved by a slightly modified Dijkstra's (or Bellman-Ford [5]) algorithm [14]. The link-constrained routing problem can be easily reduced to the link-optimization problem.

For other QoS metrics such as delay, delay jitter and cost, the state of a path is determined by the combined state over all links of the path. For example in Figure 1, the delay of path $s \to i \to j \to t$ is 10, which is the total delay of all links on the path. Two basic routing problems can be defined for this type of QoS metrics. One is called *path-optimization routing*, e.g., least-cost routing, which is to find a path whose total cost is minimized. The other problem is called *path-constrained routing*, e.g., delay-constrained routing, which is to find a path whose delay is bounded by a required value. Both problems can be directly solved by the Dijkstra's (or Bellman-Ford) algorithm.

Many composite routing problems can be derived from the above four basic problems (Figure 4). Bandwidth-constrained least-delay routing belongs to the *link-constrained path-optimization routing* problem class. It is to find the least-delay path that has the required bandwidth. The problem can be solved by a shortest path algorithm on the graph where the links violating the bandwidth constraint have been removed. The other four problem classes that are solvable in polynomial time by a modified shortest path algorithm are *link-constrained link-optimization routing*, *multi-link-constrained routing*, *link-constrained path-constrained routing*, and *path-constrained link-optimization routing*. Figure 4 gives an example for each of them.

There are two NP-complete problem classes, *path-constrained path-optimization routing* (PCPO) and *multi-path-constrained routing* (MPC), which are of particular interest. An example of PCPO is delay-constrained least-cost

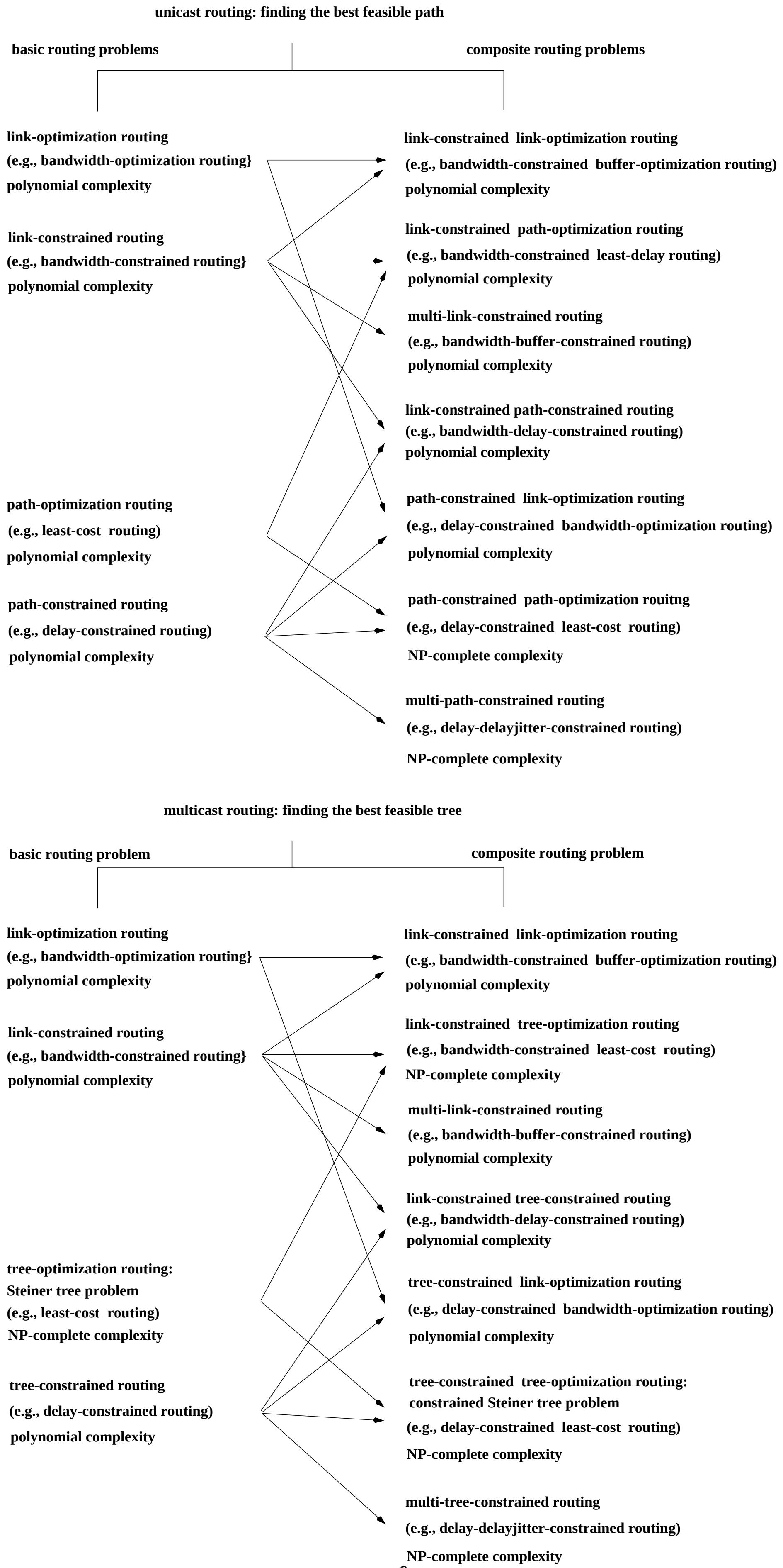


Figure 4: Routing problems

routing, which finds the least-cost path with bounded delay. An example of MPC is *delay–delay-jitter–constrained routing*, which finds a path with both bounded delay and bounded delay jitter. For the above problems to be NP-complete, we have two assumptions: (1) the QoS metrics are independent, and (2) they are allowed to be real numbers or unbounded interger numbers. If all metrics except one take bounded integer values, then the problems are solvable in polynomial time by running an extended Dijkstra's (or Bellman-Ford) algorithm [10].[2] If all metrics are dependent on a common metric, e.g., the worst-case delay and delay jitter are functions of bandwidth in networks using the WFQ scheduling, then the problems may also be solvable in polynomial time [32].

4.2 Multicast routing

The hierarchy of multicast routing problems are defined similarly in Figure 4. The difference is that an optimization or a constraint must be applied to the entire tree instead of a single path. For examples, the bandwidth-optimization routing asks to maximize the bandwidth of the bottleneck link of the tree, and the delay-constrained routing finds a tree in which the end-to-end delay from the sender to any destination is bounded by a given value.

There are several well-known multicast routing problems. The *Steiner tree problem* is to find a tree covering a group of destinations with the minimum total cost over all links (i.e., the least-cost tree). It is also called the *least-cost multicast routing* problem, belonging to the *tree-optimization routing* problem class (Figure 4). The *constrained Steiner tree problem* is to find the least-cost tree with the bounded delay. It is also called the *delay-constrained least-cost routing* problem, belonging to the *tree-constrained tree-optimization routing* problem class. Finding either Steiner tree or constrained Steiner tree is NP-complete [44]. The *multi-tree-constrained routing*, e.g., delay–delay-jitter–constrained multicast routing, is also NP-complete [42], with the assumptions that (1) the metrics under constraints take real numbers (or unbounded interger numbers) and (2) they are independent. However, the problem is solvable in polynomial time if all metrics except one take bounded integer values. If all metrics are dependent on a common metric, then the problem may also be solvable in polynomial time. Figure 5 gives examples of constrained paths and constrained trees.

4.3 QoS routing and other network components

QoS routing v.s. best-effort routing: The QoS routing is different from the traditional best-effort routing. The former is normally connection-oriented with resource reservation to provide the guaranteed service, whereas the latter can be either connection-oriented or connectionless with a dynamic performance subject to the current availability of shared resources. Meeting the QoS requirement of each individual connection and reducing the call-blocking rate are important for the QoS routing, while the fairness, overall throughput and average response time are the essential issues for the traditional routing.

QoS routing and resource reservation: The QoS routing and the resource reservation [15, 55] are two important, closely related network components. In order to provide the guaranteed services, the required resources (CPU time, buffer, bandwidth, etc.) must be reserved when a QoS connection is established, so that the data transmission will not be affected by the traffic dynamics of other connections sharing the common links. Before the reservation can be done, a path with the best chance to satisfy the resource requirement must be selected, which is the job of routing. While routing is decoupled from resource reservation in most existing schemes, some resent proposals combine routing and resource reservation in a single multi-path message pass from the source to the destination [11].

QoS routing and admission control: The task of *admission control* is to determine whether a connection request

[2] If all metrics (e.g., delay) except one take unbounded integer values but the maximum constraints (e.g., delay bound requirement) are bounded, then the problems are also solvable in polynomial time.

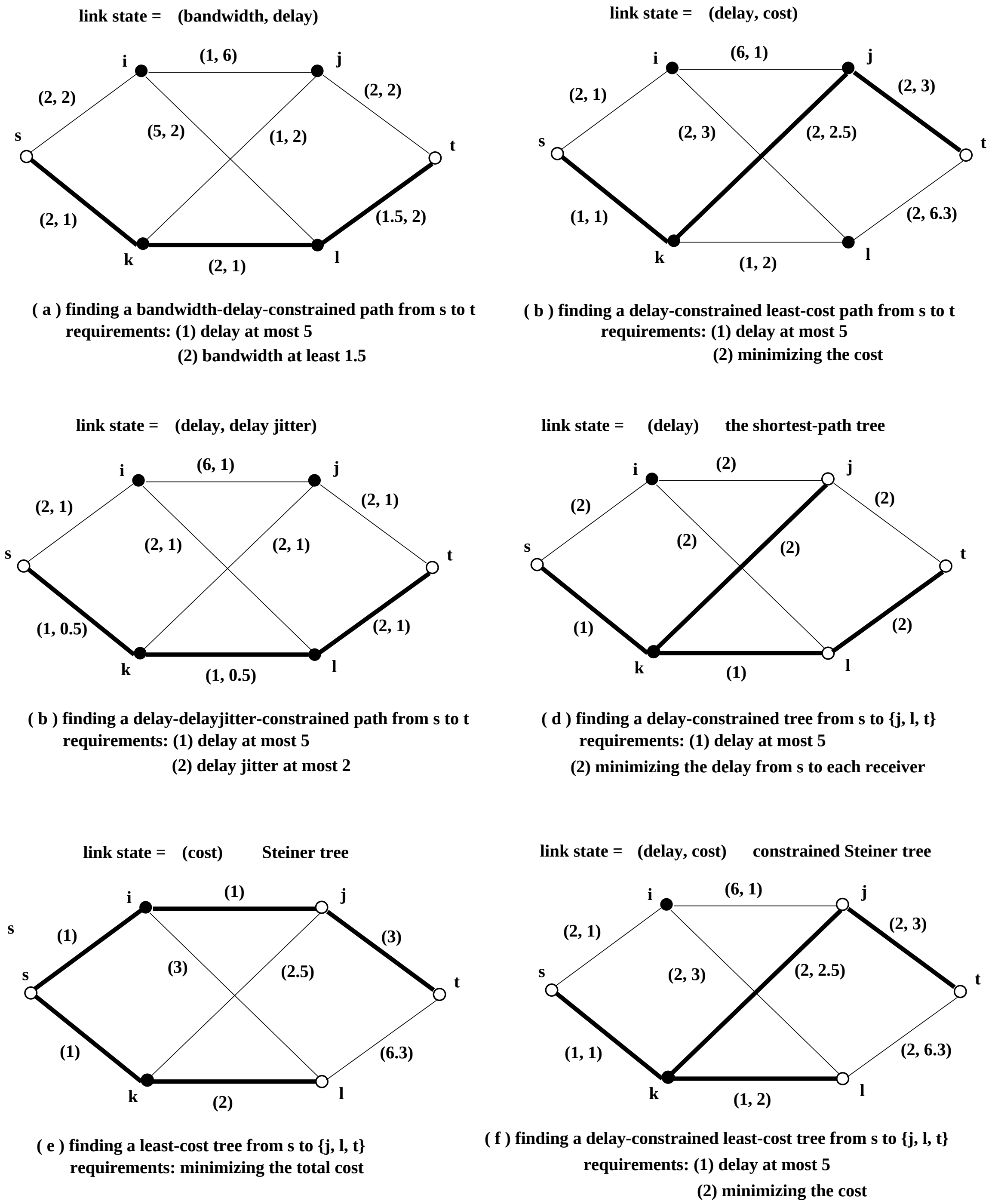


Figure 5: Constrained paths and constrained trees

should be accepted or rejected. Once a request is accepted, the required resources must be guaranteed. The admission control is often considered as a by-product of QoS routing and resource reservation. If the resource reservation is successfully done along the route(s) selected by the routing algorithm, the connection request is accepted; otherwise, the request is rejected.

QoS routing and QoS negotiation: A QoS routing algorithm may fail to find a feasible path for a new connection, either because there does not exist a feasible path, or because the searching space of a heuristic approach does not cover any existing feasible path. When this happens, the system can either reject the connection or negotiate with the application for a loosed QoS constraint. The QoS routing can assist the negotiation by finding the best available path and returning the QoS bounds that can be supported. If the negotiation is successful according to the provided bounds, the best available path can be used immediately.

5 Routing Strategies

Routing involves two basic tasks: (1) collecting the state information and keeping it up-to-date, and (2) searching the state information for a feasible path. In order to find an optimal path which satisfies the constraints, the state information about the intermediate links between the source and the destination(s) must be known. The search of feasible paths greatly depends on how the state information is collected and where the information is stored.

There are three routing strategies, *source routing, distributed routing* and *hierarchical routing*, classified according to the way how the state information is maintained and how the search of feasible paths is carried out. In the source routing, each node maintains the complete global state, including the network topology and the state information of every link. Based on the global state, a feasible path is locally computed at the source node. A control message is then sent out along the selected path to inform the intermediate nodes of their precedent and successive nodes. A link-state protocol is used to update the global state at every node. In the distributed routing, the path is computed by a distributed computation during which control messages are exchanged among the nodes and the state information kept at each node is collectively used for the path search. Most distributed routing algorithms need a distance-vector protocol (or a link-state protocol) to maintain a global state in form of distance vectors (Figure 2) at every node. Based on the distance vectors, the routing is done in a hop-by-hop basis. In the hierarchical routing, nodes are clustered into groups, which are recursively clustered into higher-level groups, creating a multi-level hierarchy. Each physical node maintains an aggregated global state (Section 3), which contains the detailed state information about the nodes in the same group and the aggregated state information about the other groups. The source routing is used to find a feasible path on which some nodes are *logical nodes* representing groups. A control message is then sent along this path to establish the connection. When the border node of a group represented by a logical node receives the message, it uses the source routing to expend the path through the group.

5.1 Strengths and weaknesses of source routing

The source routing achieves its simplicity by transforming a distributed problem into a centralized one. By maintaining a complete global state, the source node calculates the entire path locally. It avoids dealing with the distributed computing problems such as distributed state snapshot, deadlock detection and resolution, and distributed termination problem. Many source algorithms are conceptually simple and easy to implement, evaluate, debug and upgrade. In addition, it is much easier to design centralized heuristics for some NP-complete routing problems than to design distributed ones.

The source routing has several problems. First, the global network state maintained at every node has to be updated frequently enough to cope with the dynamics of network parameters such as bandwidth and delay, which makes the communication overhead excessively high for large scale networks. Second, the link-state algorithm can only provide *approximate* global state due to the overhead concern and non-negligible propagation delay of state messages. As a consequence, the imprecision in the global state can cause the QoS routing fail [45]. Third, the computation overhead at the source is excessively high, especially in the case of multicast routing or when multiple constraints are involved, considering that the QoS routing is typically done on a per connection basis. In summary, the source routing has the scalability problem. It is impractical for any single node to have access to detailed state information about all nodes and all links in large networks [20].

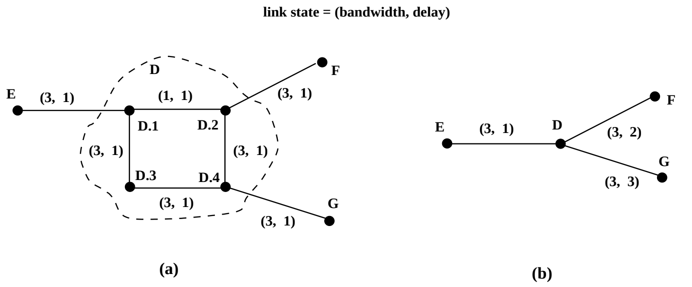


Figure 6: (a) the internal state, (b) an incorrect aggregation on link (D, F)

5.2 Strengths and weaknesses of distributed routing

In distributed routing, the path computation is distributed among the intermediate nodes between the source and the destination. Hence, the routing response time can be made shorter and the algorithm is more scalable. Searching multiple paths in parallel for a feasible one is made possible, which increases the chance of success. Most existing distributed routing algorithms [43, 50, 52] require each node to maintain a global network state (distance vectors), based on which the routing decision is made on a hop-by-hop basis. Some flooding-based algorithms do not require any global state to be maintained. The routing decision and optimization is done entirely based on the local states [9, 47].

The distributed routing algorithms which depend on the global state share more or less the same problems of the source routing algorithms. The distributed algorithms which do not need any global state tend to send more messages. It is also very difficult to design efficient distributed heuristics for the NP-complete routing problems, especially in the case of multicast routing, because there is no detailed topology and link-state information available. In addition, when the global states at different nodes are inconsistent, loops may occur. A loop can be easily detected when the routing message is received by a node for the second time. However, loops generally make the routing fail because the distance vectors do not provide sufficient information for an alternative path.

5.3 Strength and weaknesses of hierarchical routing

The hierarchical routing has long been used to cope with the scalability problem of source routing in large internetworks [3, 4]. The PNNI (Private Network-Network Interface) [16] standard for routing in ATM networks is also hierarchical. The hierarchical routing scales well because each node only maintains a partial global state where groups of nodes are aggregated into logical nodes. The size of such an aggregated state is logarithmic in the size of the complete global state. The well-studied source routing algorithms are directly used at each hierarchical level to find feasible paths based on the aggregated states maintained at nodes. Hence, the hierarchical routing retains many advantages of source routing. It has also some advantages of the distributed routing because the routing computation is shared by many nodes.

However, As the network state is aggregated, additional imprecision is introduced, which has a significant negative impact on the QoS routing [20]. Recall that a logical node in an aggregated network image may represent a large subnet with complex internal structure and a logical link may be the abstraction of multiple physical links. Consider the aggregated network image in Figure 3 (e). It is hard to estimate the end-to-end delay from $A.a.1$ to a node in the group represented by C, because the internal structure of C is hidden. More specifically, although the actual delay between physical nodes in $A.c$ and physical nodes in C may vary, there is a single delay from $A.c$ to C in the aggregated state. Such an abstraction inevitably results in imprecision. The same thing happens to all other logical links, $(A.a.3, A.b), (A.a.4, A.c), (A.b, A.c), (A.b, B)$ and (B, C).

The problem becomes more complicated when multiple QoS constraints are involved. Figure 6 shows an example. Two QoS metrics, *bandwidth* and *delay*, are considered. The pair of numbers beside a link are the residual bandwidth and the delay of the link, respectively. Four nodes, $D.1$, $D.2$, $D.3$ and $D.4$, form a group D. Suppose after aggregation the internal bandwidth and delay of D are *merged* into those of links (D, F) and (D, G). Consider the problem of determining the bandwidth and delay of link (D, F). A naive way is to find the path with the largest bandwidth from $D.1$ to $D.2$, which is $P_1 = D.1 \rightarrow D.3 \rightarrow D.4 \rightarrow D.2$ with bandwidth 3. The bandwidth of (D, F) is the minimum of 3 and the bandwidth of $(D.2, F)$, and the result is 3. Similarly, find the path with the smallest delay, which is $P_2 = D.1 \rightarrow D.2$ with delay 1. The delay of (D, F) is the summation of 1 and the delay of $(D.2, F)$, and the result is 2. Such an optimistic approach is however incorrect because P_1 and P_2 are not the same path. In general, there exist many different paths between two border nodes of a group. Some paths have better bandwidth availability and some others have smaller delay. There may not exist a path with the best properties in both terms. How to aggregate such information is still an open problem.

6 Unicast Routing Algorithms

We describe the unicast source, distributed and hierarchical routing algorithms in this section. We discuss the problems and solutions, basic ideas, compare them and discuss their pros and cons. See Table 1 for a summarizing comparison. Algorithms are referred by the authors' names and a reference to their paper.

6.1 Source routing algorithms

Wang-Crowcroft algorithm [52]: Wang-Crowcroft algorithm finds a *bandwidth-delay-constrained path* by the Dijkstra's shortest-path algorithm. First, all links with a bandwidth less than the requirement are eliminated so that any paths in the resulting graph will satisfy the bandwidth constraint. Then, the shortest path in terms of delay is found. The path is feasible if and only if it satisfies the delay constraint.

Ma-Steenkiste algorithm [32]: Ma and Steenkiste showed that when a class of WFQ-like (Weighted Fair Queueing) scheduling algorithms [6, 13, 19, 54] are used, the end-to-end delay, delay-jitter, and buffer space bounds are not independent but are functions of the reserved bandwidth, the selected path and the traffic characteristics. Therefore, the problem of finding a path satisfying bandwidth, delay, delay-jitter and buffer space constraints, which is NP-complete in general [18, 52], can be simplified by taking these functional relationships into consideration and thus can be solved by a modified version of Bellman-Ford algorithm in polynomial time. A much further study of QoS routing on rate-based scheduling networks was done recently by Orda [35].

Guerin-Orda algorithm [20] [3]: Guerin and Orda studied the bandwidth-constrained and delay-constrained source routing with imprecise network states. The model of imprecision is based on the probability distribution functions. Every node maintains, for each link l, the probability $p_l(w)$ of every feasible bandwidth availability, $w \in [0..c_l]$, where c_l is the capacity of the link. The goal of the bandwidth-constrained routing is to find the path that has the highest probability to accommodate a new connection with a given bandwidth requirement x. This problem can be solved by a standard shortest path algorithm with each link l weighted by $(-log p_l(x))$.

The goal of the delay-constrained routing is to find a path that has the highest probability to satisfy a given end-to-end

[3] Guerin-Orda algorithm was designed to be used in hierarchical routing, though we present it as an independent source routing algorithm in this paper.

Table 1: Unicast algorithms

alorithm	solving problem	routing strategy	time complexity	Comm. complexity	
				maintaining state	routing
Wang-Crowcroft [52]	bandwidth-delay-constrained r.	source	$O(v\log v + e)$	global	Zero
Ma-Steenkiste [32]	bandwidth-constrained r.	source	$O(v\log v + e)$	global	Zero
	multi-constrained r. (1)	source	$O(kve)$ (1)	global	Zero
Guerin-Orda [20] (2.1)	bandwidth-constrained r.	source	$O(v\log v + e)$	imprecise global	Zero
	delay-constrained r.	source	polynomial (2.2)	imprecise global	Zero
Chen-Nahrstedt [10]	bandwidth-cost-constrained r.	source	$O(xve)$	global	Zero
Wang-Crowcroft [52]	bandwidth-optimization r.	distributed	$O(ve)$	global	$O(v)$
Salama et al [43]	delay-constrained least-cost r.	distributed	$O(v^3)$	global	$O(v^3)$ (6)
Sun-Landgendorfer [50]	delay-constrained least-cost r.	distributed	$O(v)$	global	$O(v)$
Cidon et al [11]	generic r. (3)	distributed	$O(e)$	global	$O(e)$ (7)
Shin-Chou algorithm [47]	delay-constrained r.	distributed	$O(e)$	local	$O(e)$
Chen-Nahrstedt [9]	generic r. (3)	distributed	$O(e)$	local	$O(e)$
PNNI [16]	generic r. (3)	hierarchical	polynomial (5)	aggregated	$O(v)$

(*) v is the number of nodes and e is the number of edges.
(**) After a source routing algorithm selects a path, a control message needs to be sent along the path to establish the connection, which has a worst-case communication overhead of $O(v)$.
(1) Ma and Steenkiste studied routing with constraits on delay, delay jitter and buffer space in rate-based scheduling networks. k in the time complexity is the number of all possible residual bandwidth that a link may have.
(2) 2.1) Guerin-Orda algorithm was designed to be used in hierarchical routing, though we present it as an independent source routing algorithm in this paper. Recall that routing at each hierarchy level in the hierarchical routing is typically done by a source routing algorithm. 2.2) Heuristics with different assumptions have different polynormial time complexities.
(3) A routing framework was proposed, from which algorithms on different QoS constraints can be derived.
(4) x is a constant in the algorithm. A larger x results in a higher probability of finding a feasible path and a higher overhead.
(5) The time complexity of a hierachical routing algorithm depends on what source routing algorithm is used to route the connection through every group.
(6) It was shown that the average overhead is substantially less than the worst-case overhead.
(7) Variants of the algorithm may have higher worst-case overhead.

Table 2: Multicast algorithms

alorithm	solving problem	routing strategy	time complexity	Comm. complexity	
				maintaining state	routing
MOSPF [33]	least-delay r.	source	$O(v\log v)$	global	Zero
Kou et al [28]	least-cost r.	source	$O(gv^2)$	global	Zero
Takahashi-Matsuyama [51]	least-cost r.	source	$O(gv^2)$	global	Zero
Kompella et al [26]	delay-constrained least-cost r.	source	$O(v^3\Delta)$ (1)	global	Zero
Sun-Landgendorfer [49]	delay-constrained least-cost r.	source	$O(v\log v + e)$	global	Zero
Widyono [53]	delay-constrained least-cost r.	source	exponential (2)	global	Zero
Zhu et al [56]	delay-constrained least-cost r.	source	$O(kv^3\log v)$	global	Zero
Rouskas-Baldine [42]	delay-constrained least-cost r.	source	$O(klgv^4)$ (3)	global	Zero
Kompella et al [26]	delay-constrained least-cost r.	distributed	$O(gv^3)$	global	$O(v^3)$
Chen-Nahrstedt [9]	generic r.	distributed	$O(ge)$	local	$O(e)$

(*) v is the number of nodes, e is the number of edges, and g is the number of destinations.
(**) After a source routing algorithm constructs a multicast tree, a control message needs to be sent down the tree to establish the connection, which has a worst-case communication overhead of $O(v)$.
(1) Δ is the delay requirement. The time complexity is polynomial if Δ is a bounded integer.
(2) Widyono algorithm uses the constrained Bellman-Ford (CBF) algorithm. Widyno pointed out that there are cases where the running time of CBF grows exponentially. However, simulation shows that its average performance is comparable to other algorithms that construct constrained Steiner trees [44].
(3) k and l are constants in the algorithm. A larger k (or l) results in a higher probability of finding a feasible tree and a higher overhead.

delay requirement. Suppose every node maintains, for each link l, the probability $p_l(d)$ of link l having a delay of d units, where d ranges from zero to maximum possible value. Finding the path that has the best probability of satisfying a given delay bound is NP-hard, though various special cases (e.g., symmetric networks and tight constraints) can be solved in polynomial time. Heuristic algorithms were also proposed for this problem. The idea is to transforming a global constraint into local constraints. More specifically, it splits the end-to-end delay constraint among the intermediate links in such a way that every link in the path has an equal probability of satisfying its local constraint. The heuristics then try to find the path with the best multiplicative probability over all links.

Guerin-Orda algorithm works with imprecise information and is suitable to be used in hierarchical routing. One of the above heuristic algorithms was extended by the authors to make routing based on the aggregated network state of the hierarchical model (Section 3). A further study of QoS routing with imprecise state based on the probability model was done by Lorenz and Orda [30].

Chen-Nahrstedt algorithm [10]: Chen and Nahrstedt proposed a heuristic algorithm for the NP-complete multi-path-constrained routing problem. We have already known in Section 4.1 that if all metrics except one take bounded integer values, then multi-path-constrained routing is solvable in polynomial time. Consider the delay-cost-constrained routing. The idea is to map the costs (or delays) of the links from real numbers to bounded integers. This reduces the original NP-complete problem to a simpler problem solvable in polynomial time. Let C be the cost requirement and x be a small integer. The algorithm first maps the cost of every link to an integer bounded by $x+1$. Real numbers in $[0, C]$ are mapped into integers in $[0..x]$, real numbers in $[C, \infty]$ are mapped to $x+1$, and the cost bound C is mapped to x. See Figure 7 for an example. The new problem with link costs bounded by $x+1$ can be solved in polynomial time by an extended Dijkstra's algorithm (EDSP) or an extended Bellman-Ford algorithm (EBF) [10]. It was proved that a feasible path of the new problem must also be a feasible path of the original problem. The performance of the algorithm is tunable since choosing a larger x results in a larger probability of finding a feasible path and a larger overhead.

Awerbuch et al algorithm [1]: Awerbuch et al proposed a *throughput-competitive routing* algorithm for bandwidth-constrained connections. The algorithm tries to maximize the amortized (average) throughput of the network over time. It combines the functions of admission control and routing. Every link is associated with a cost function that is exponential to the bandwidth utilization. A new connection is admitted into the network only if there exists a path whose *accumulated cost over the duration of the connection* does not exceed the *profit* that is measured by the bandwidth-duration product of the connection. It was proved that such a path satisfies the bandwidth constraint. The algorithm achieves a throughput that is within $O(\log vT)$ factor of the highest possible throughput achieved by the best off-line algorithm that are assumed to know all of the connection requests in advance, where T is the maximum connection duration and v is the number of nodes in the network. Competitive routing for connections with unknown duration was studied in [2]. A survey for competitive routing algorithms was done by Plotkin [40].

Summary: All the above algorithms require a global state to be maintained at every node. Most algorithms transform the routing problem to a shortest path problem and then solve it by Dijkstra's or Bellman-Ford algorithm. We summarize the distinctive properties of some algorithms as follows: Ma-Steenkiste algorithm provides a routing solution to rate-based networks; Guerin-Orda algorithm works with imprecision information and hence is suitable to be used in hierarchical routing; the performance of Chen-Nahrstedt algorithm is tunable by trading overhead for success probability; the adimission control and routing in Awerbuch et al algorithm takes the connection duration into account, which allows more precise cost-profit comparison. All the

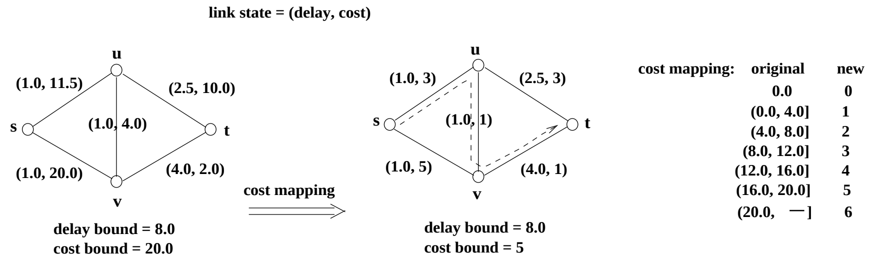


Figure 7: Chen-Nahrstedt heuristic ($C = 20.0$ and $x = 5$)

above algorithms is executed at connection arrival on a per-connection basis, which may cause the overall compuational overhead excessively high. *Path precomputation* and *caching* were studied to make a tradeoff between processing overhead and the routing performance [7, 24, 39, 46].

6.2 Distributed routing algorithms

Wang-Crowcroft algorithm [52]: Wang and Crowcroft proposed a *hop-by-hop* distributed routing scheme, in which every node pre-computes a forwarding entry for every possible destination. The forwarding entry, which is updated periodically, stores the *next hop* on the routing path to the destination. After the forwarding entries at every node are computed, the actual routing is simply to follow the entries.

A path with the maximum bandwidth between two nodes is called the *widest path*. If there are several such path, the one with the smallest delay is called the *shortest-widest path*. A link-state protocol is used to maintain at every node a complete global state, based on which the forwarding entries for the *shortest-widest* path is computed by a modified Bellman-Ford (or Dijkstra's) algorithm [52]. A routing path is the combination of the forwarding entries indexed by the same destination at all intermediate nodes. The path is loop-free if the state information at all nodes is consistent. However, in a dynamic network, the path may have a loop due to contradicting state information at different nodes.

Salama et al algorithm [43]: Salama et al proposed a distributed heuristic algorithm for the NP-complete delay-constrained least-cost routing problem. A *cost vector* and a *delay vector* are maintained at every node by a distance-vector protocol. The cost (delay) vector contains for every destination the next node on the least-cost (least-delay) path. A control message is sent from the source toward the destination to construct a delay-constrained path. Any node i at the end of the partially-constructed path can select one of only two alternative outgoing links. One link (i, j) is on the least-cost path directed by the cost vector, and the other (i, k) is on the least-delay path directed by the delay vector. Link (i, j) has the priority to be chosen, as long as adding the least-delay path from j to the destination does not violate the delay constraint.

Loops may occur as the control message chooses the least-cost path and the least-cost path alternatively. A loop is detected if the control message visits a node twice. Whenever it happens, the routing process is rolled back until reaching a node from which the least-cost path was followed. The routing process resumes from there by changing the next hop along the least-delay path. It was proved that such a mechanism removes all the loops, provided that delay and cost vectors at all nodes are up-to-date (or at least consistent), a condition that does not hold sometimes in a dynamic network.

Sun-Landgendorfer algorithm [50]: Sun and Langendorfer [50] improved the worst-case performance of Salama et al algorithm by avoiding loops instead of detecting and removing loops. A control message is sent to construct the routing path. The message travels along the least-delay path until reaching a node from which the delay of the least-cost path satisfies the delay constraint. From that node on, the message travels along the least-cost path all the way to the destination. The difference between Sun-Landgendorfer algorithm and Salama et al algorithm is illustrated in Figure

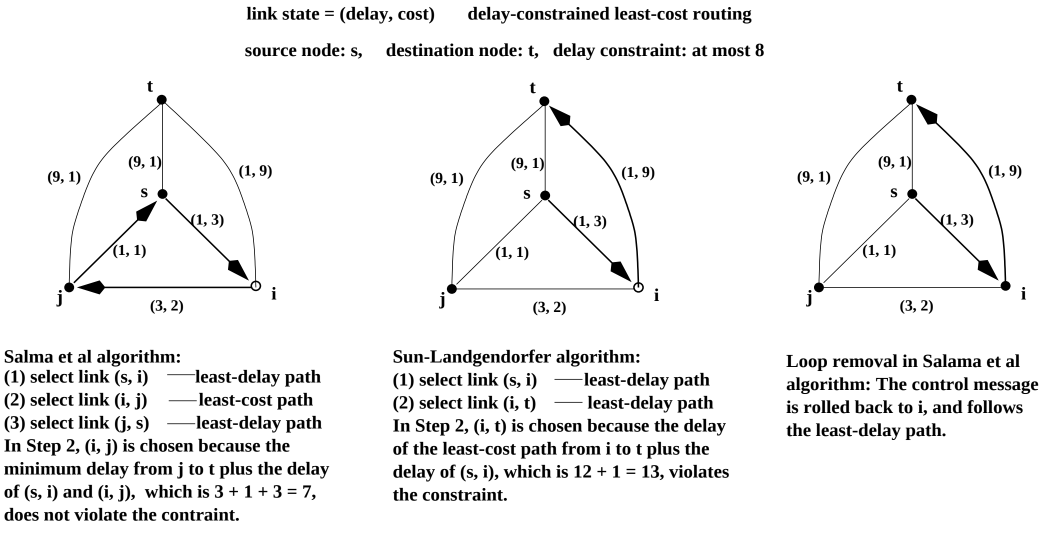


Figure 8: Salama et al algorithm v.s. Sun-Landgendorfer algorithm

8. It was proved that the algorithm constructs loop-free paths provided that the state information at all nodes is updated (or consistent). In a dynamic network, when different nodes have inconsistent information, the least-cost (least-delay) path computed based on contradicting information may contain a loop, which makes the control message not able to reach the destination.

Cidon et al. algorithm [11]: The distributed multi-path routing algorithms proposed by Cidon et al combine the process of routing and resource reservation. Every node maintains the topology of the network and the cost of every link. When a node wishes to establish a connection with QoS constraints, it finds a subgraph of the network which contains links that lead to the destination with a "reasonable" cost. Such a subgraph is called a *diroute*. A link is *eligible* if it has the required resources. Reservation messages are flooded along the eligible links in the diroute toward the destination and reserve resources along different paths in parallel. When the destination receives a reservation message, a routing path is established. The algorithm releases resources from segments of the diroute as soon as it learns that these segments are inferior to another segment where reservation was made. Variants of the above algorithm were proposed to make tradeoff between routing time and path optimality. Reserving resources on multiple paths makes the routing faster and more resilient. However, it also increases the level of resource contention.

Shin-Chou algorithm [47]: Shin and Chou proposed a distributed route-selection scheme for establishing delay-constrained connections. No global state is required to be maintained at any node. The algorithms flood routing messages from the source toward the destination. Each message accumulates the total delay of the path it has traversed so far. When a routing message is received by an intermediate node, if it is the first such message received or it carries a better accumulated delay than the previously received one, the message will be forwarded along the links whose delay plus the message's accumulated delay does not exceed the end-to-end delay requirement. Once a message reaches the destination, it finds a delay-constrained path, which is the one it has traversed. It was shown that, when certain scheduling policies [25] are used and the routing messages are set to appropriate priority, there will be at most one message sent along every link. Another flooding-based routing algorithm was proposed by Hou [22] to route virtual circuits with delay requirements in ATM networks.

Chen-Nahrstedt algorithm [9]: 1) selective probing Chen and Nahrstedt proposed a distributed routing framework based on selective probing. After a connection request arrives, probes are flooded selectively along those paths which satisfy QoS and optimization requirements. Every node only maintains its local state, based on which routing and optimization decisions are made collectively in the process of probing. As in Shin-Chou algorithm, each probe arriving at the destination detects a feasible path. Algorithms were derived from the framework to route connections with a veriety of QoS constraints on bandwidth, delay, delay jitter, cost and their combinations. Several techniques were developed to overcome the high communication overhead of Shin-Chou algorithm. First, probes are only allowed to be forwarded to a *subset* of outgoing links determined based on the topological distances to the destination. Second, *iterative probing* is used to further reduce the overhead. At the first iteration, probes are sent only along the shortest paths. If the first iteration fails, probes are allowed to be sent along paths with increasing maximum lengths in the following iterations. Simulation shows that with two iterations Chen-Nahrstedt algorithm achieves substantial overhead reduction.

2) ticket-based probing If every node maintains a global state, which is allowed to be imprecise, the ticket-based probing is used to improve the performance of selective probing. Certain number of tickets are issued at the source according to the contention level of network resources. Each probe must contain at least one ticket in order to be valid. Hence, the maximum number of probes is bounded by the total number of tickets, which limits the maximum number of paths to be searched. The algorithm utilizes the imprecise state at intermediate nodes to guide the limited tickets (the probes carrying them) along the best possible paths to the destination, so that the probability of finding a feasible path is maximized with limited probing overhead.

Summary: The distinctive properties of the above algorithms are summarized as follows: (1) Salama et al algorithm and Sun-landgendorfer algorithm provide efficient distributed solutions to the NP-complete delay-constrained least-cost routing problem. (2) Cidon et al algorithm, Shi-Chou algorithm and Chen-Nahrstedt algorithm are multi-path routing algorithms [4]. (3) Cidon et al algorithm combines routing with resource reservation. (4) Shi-Chou algorithm and Chen-Nahrstedt's selective probing algorithm require only local state to be maintained at each node. (5) Chen-Nahrstedt's iterative probing substantially reduces the routing overhead at the cost of longer routing time.

6.3 Hierarchical routing algorithms

PNNI (Private Network-Network Interface) [17]: PNNI is a hierarchical link-state routing protocol. Its hierarchial model has been discussed in Section 3. We uses an example to illustrates the routing process. The network in Figure 9 (a) has a two-level hierarchy with three groups. The aggregated topology maintained at $A.1$, $B.1$ and $C.1$ are shown in Figure 9 (b), (c) and (d), respectively. Suppose every link has an available bandwidth of one. Consider a connection request arriving at $A.1$ with a destination $C.2$. Let the bandwidth requirement be one. The routing process is described as follows. Based on the aggregated state, the source node $A.1$ finds a path ($A.1 \rightarrow A.2$) within its group and a logical path ($A \rightarrow B \rightarrow C$) on the higher hierarchy level. The logical path, together with the destination $C.2$, is sent to the next group B on the path. When the border node $B.1$ receives the information, it selects a path ($B.1 \rightarrow B.2 \rightarrow B.3$) within its group and then passes the logical path and the destination to group C. Finally, the border node $C.1$ of the destination group completes the routing by selecting $C.1 \rightarrow C.2$. It may happen that a link on the selected path does not have sufficient resources, as shown in Figure 9 (e), where link $B.3 \rightarrow B.2$ does not have enough bandwidth for the connection due to traffic dymanics. In this case, the routing process is cranked back to $B.1$ and resumes with an alternative path $B.1 \rightarrow B.2$.

[4] Search multiple paths for a feasible one.

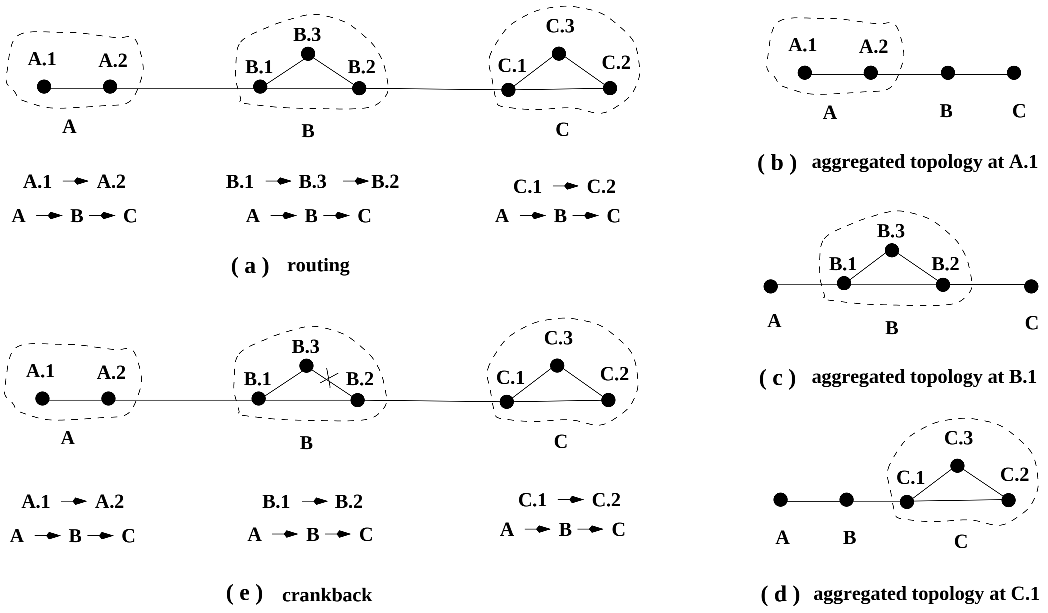


Figure 9: An example of PNNI routing

7 Multicast Routing Algorithms

Most existing work on multicast routing focuses on the following problems: bandwidth-constrained multicast routing, delay-constrained multicast routing, delay-constrained least-cost multicast routing (constrained Steiner tree problem), and delay–delay-jitter–constrained multicast routing. We describe the algorithms in this section. A summarizing comparison can be found in Table 2.

7.1 Source routing algorithms

MOSPF [33]: MOSPF is a multicast extension of the link-state unicast protocol OSFP [34]. It was based on Deering's work [12]. At every node, the protocol maintains, in addition to a global state, the membership information for every multicast group in the routing domain. Changes of group membership on a subnetwork are detected by a local router, and that router broadcasts the information to all other nodes. Given the full knowledge of network state and group membership, any node can compute the shortest-path multicast tree from a source to a group of destinations, using Dijkstra's algorithm. Such a protocol can be easily used for delay-constrained multicast routing.

Steiner tree problem A Steiner tree is the least-cost tree that spans a given subset of nodes. Strictly speaking, finding a Steiner tree is not a QoS routing problem. However, the heuristics for constructing Steiner trees have direct impact on constructing the Constrained Steiner tree. In the following, we briefly discuss two algorithms. A nice survey on the Steiner problem can be found in [23].

1) Kou et al algorithm [28]: In Kou et al algorithm, a network is abstracted to a complete graph, where the nodes represent the source and the destinations, and the edges represent the shortest paths between these nodes. The Prim's algorithm [41] is used to construct a minimum spanning tree in the complete graph, and the Steiner tree of the original network is obtained by replacing the edges in the minimum spanning tree by the shortest paths and removing the loops.

2) Takahashi-Matsuyama algorithm [51]: Takahashi-Matsuyama algorithm finds a Steiner tree by an incremental approach called *nearest destination first (NDF).* Initially, the nearest destination (in terms of cost) to the source is founded and the least-cost path between them is selected. Then at each iteration the *nearest* unconnected destination to the partially constructed tree is found and added into the tree. This process is repeated until all destinations are included in the tree.

Constrained Steiner tree problem The problem of finding a delay-bounded least-cost multicast tree, called a *constrained Steiner tree*, is NP-complete [26]. Heuristic source routing algorithms were proposed for this problem [26, 49, 53, 56]. A performance evaluation of the algorithms was done by Salama et al through extensive simulation [44].

1) Kompella et al algorithm [26]: A source routing heuristic was proposed by Kompella et al to construct a constrained Steiner tree. The algorithm first creates a complete graph, where the nodes represent the source and the destinations, and the edges represent the delay-constrained least-cost paths between these nodes. The link delays are assumed to be integers and the delay-bound requirement is assumed to be always bounded, so that such a graph can be constructed in polynomial time (Section 4.1). The second step is to construct a delay-constrained spanning tree of the complete gragh. Starting with the source node, the tree is incrementally expanded by adding an edge each time until every destination node is included. The selected edge is the one which (1) connects a node in the tree and a node outside of the tree, (2) does not violate the delay constraint, and (3) minimizes a selection function. Two selection functions are considered. One is simply the cost of the edge, the other tries to make a tradeoff between minimizing the cost and minimizing the delay. The third step is to expend the edges of the constrained spanning tree into the delay-constrained least-cost paths they represent. Any loops caused by this expansion are removed.

2) Sun-Langendoerfer algorithm [49]: Sun and Langendoerfer proposed an algorithm which constructs approximated constrained Steiner trees by Dijkstra's algorithm. It first computes the shortest path tree in terms of cost, i.e., the cost of every path in the tree from the source to a destination is minimized. If the end-to-end delay to any destination violates the delay constraint, the minimum-delay path is used to replace the minimum-cost path. The advantage of the algorithm is its low time complexity, $O(vlogv)$, which is the same complexity of Dijkstra's algorithm.

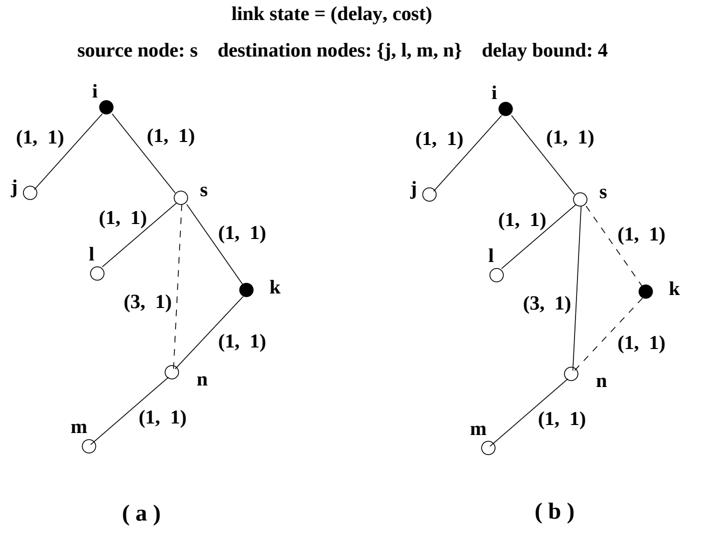


Figure 10: Solid links are in the multicast tree. Dashed links are not in the tree. Path $k \to m \to n$ in (a) is replaced by path $k \to n$ in (b). The cost is reduced by 1.

3) Widyono algorithm [53]: Widyono proposed several heuristic algorithms for the constrained Steiner tree problem. The one with the best performance is called *constrained adaptive ordering heuristic*. At each time, a constrained Bellman-Ford algorithm is used to find a delay-constrained least-cost path from the source to a destination that is not yet in the tree. The found path as well as the destination is then inserted into the tree. The cost of links in the tree is set to zero. The above process repeats until the tree covers all destinations.

4) Zhu et al algorithm [56]: Zhu et al proposed a source routing heuristic to construct constrained Steiner trees. The algorithm allows variable delay bounds on destinations. A shortest-path tree in terms of delay is first constructed by Dijkstra's algorithm. If the delay constraint cannot be satisfied for any destination, it must be re-negotiated. Otherwise, the algorithm proceeds to iteratively refine the tree for lower cost. The basic idea is to replace a path in the tree by another path with lower cost unless such a replacement can not be found. Figure 10 gives an example of replacement. Heuristics were proposed for finding such a replacement. The algorithm always finds a delay-constrained tree (probably not least-cost), if one exists, because it starts with a shortest path tree.

Rouskas-Baldine algorithm [42]: Rouskas and Baldine proposed a heuristic for constructing a delay–delay-jitter–constrained multicast tree. The tree must have (1) bounded delay along the paths from the source to the destinations and (2) bounded variation among the delays along these paths. The shortest path tree T_0 in terms of delay is first constructed by Dijkstra's algorithm. If the tree does not meet the delay-jitter constraint, the algorithm finds out the largest-delay path in T_0 from the source to a destination, and starts from the path to incrementally construct a feasible tree. At each iteration, a "good" path from a node in the tree to a destination out of the tree is found and added into the tree. The path must be completely disjoint from the tree, and must not cause the tree to violate the constraints. The above process repeats until all destinations are included in the tree. The authors showed that the heuristic demonstrates good average-case behavior in terms of the maximum inter-destination delay variation.

Summary: All the above algorithms require a global state to be maintained at every node. Most heuristic algorithms for NP-complete multicast routing problems construct a constrained tree incrementally by adding one destination into the tree each time based on certain selection criteria. Kou et al algorithm and Kompella et al algorithm reduce the original problem to a spanning tree problem by constructing a logical complete graph among the source node and the destination nodes. Zhu et al algorithm iteratively refines the multicast tree by replacing paths in the tree for lower cost. Rouskas-Baldine algorithm constructs a multicast tree with both bounded delay and bounded delay-jitter, which is very useful in interactive audio-visual communication such as teleconference. Among the four algorithms for the constrained Steiner tree problem, Salama's simulation [44] showed that (1) Zhu et al algorithm achieves the best average performance in minimizing the cost of the tree, and (2) Sun-Langendoerfer algorithm has the least execution time .

7.2 Distributed routing algorithms

Kompella et al algorithm [27]: Kompella et al proposed a distributed heuristic algorithm for constructing the constrained Steiner tree. The algorithm requires every node to maintain a distance vector storing the minimum delay to every other node. Starting with the source node, the algorithm constructs the multicast tree iteratively by adding a link into the tree each time. Each iteration of the algorithm consists of three phases of message passing. In the first phase, the source node broadcasts a Find message down the partially constructed tree. When a node receives the message, it finds out the adjacent link which (1) leads to a destination out of the tree, (2) does not violate the delay constraint and (3) minimize a selection function of the cost. In the second phase, the selected links are sent to the source node, where the best link l which minimizes the selection function is chosen. In the third phase, an ADD message is sent to add l to the tree. This procedure continues until every destination is included in the tree. The above algorithm requires intensive multi-pass message exchange, which results in a worst-case message complexity of $O(v^3)$.

Chen-Nahrstedt algorithm [9]: Chen and Nahrstedt extended their distributed unicast routing algorithms [9] (Section 6.2) for multicast routing. Probes (routing messages) are flooded from the source toward the destinations of a multicast group. A probe proceeds along a path only if the path leads to at least one destination and has sufficient resources to guarantee the end-to-end QoS. As probes traverse toward a group of destinations, a multicast tree is built in a distributed manner. Every node maintains only its local state. The worst-case message complexity is $O(e)$. This approach only works for multicast groups whose memberships are fixed and *a priori* known. The dynamic membership problem is handled by receiver-initiated probing. When a new destination joins in a multicast group, it sends probes toward the multicast tree. Probes proceed only along the paths which do not violate QoS and optimization requirements. Once a probe reaches any node in the multicast tree, a feasible extension of the tree is found.

Summary: Kompella et al algorithm provides a distributed solution to the NP-complete constrained Steiner tree problem. Its communication overhead is high and requires every node to maintain a global state. Chen-Nahrstedt algorithm requires only local state to be maintained at every node. It is suitable to construct the shortest path tree but not constrained Steiner tree since probes search shortest paths individually without cooperation to reduce the overall cost.

8 Future Directions

Efficient routing algorithms: Most source heuristic algorithms for the NP-complete routing problems (Figure 4) are not scalable due to prohibitively high time complexity. That is especially true in the case of multicast routing. New efficient algorithms are required to make a good balance between computation time and connection-success ratio, so that the time complexity can be reduced to the shortest-path computation range while the success ratio is still acceptable [44].

Routing with imprecise state information: Most existing routing algorithms assume the availability of precise state information. However, the state information is inheritly imprecise in a distributed network environment. The imprecision directly affects the routing performance. Therefore, the design of routing algorithms for large networks should take the information imprecision into consideration [9, 20, 30].

Distributed and Hierarchical Routing: Source routing based on complete global state is generally not scalable because of the following reasons. The communication overhead to maintain the global state is proportional to the size of the network and the frequency of broadcasting local states. The storage overhead to store the state is proportional to the size of the network. The computation overhead to calculate the feasible paths is polynomial to the size of the network and proportional to the arrival frequency of connection requests. The precision of the global state at a node is in inverse proportion to the diagonal of the network and the frequency of broadcasting local states. As networks grow large, the communication, storage and computation overhead grow accordingly. Reducing the updating frequency does not solve the problem because the precision of the global state will decrease.

Distributed and hierarchical algorithms offer solutions for the scalability problem. In particular, the distributed algorithm based on selective probing [8] uses only local states, and no shortest-path computation is conducted at a single node. The ticket-based probing algorithm [8] works with imprecise state information, which allows relative infrequent state updates. The hierarchical routing provides a clean solution to the scalability problem. It maintains an aggregate global state whose size is logarithmic to the network size if the (logical) nodes are clustered into groups with roughly uniform sizes. However, the state aggregation leads to further imprecision, especially when multiple QoS metrics are involved (Section 5.3). The design and evaluation of hierarchical routing algorithms should take this into account.

Multipath routing: When the traffic load is light, the network resources are readily available. The QoS routing is of less importance in terms of searching feasible paths but of more importance in terms of balancing the traffic in order to increase the call-admission ratio of future connections and to improve the responsive time of the best-effort traffic. However, when the network load is heavy and dynamic, efficient algorithms for finding feasible paths are critical. *Multipath routing* can be used to increase the probability of accepting a connection under resource contention. There are two interpretations for multipath routing.

One interpretation is to search multiple paths for a feasible one. PNNI [16] uses *crankback* to search multiple paths *sequentially*. When the selected path does not meet the requirement, the routing process is cranked back and resumes with an alternative path. This approach works well with network dynamics. The disadvantage is longer routing time. On the other hand, in *parallel* multipath routing, routing messages are sent along multiple paths in parallel and reserve resources along the way. If more than one message arrive at the destination, the best path is selected and resources reserved on the other paths are released. An alternative approach is to reserve resources only on a primary path. The messages sent along the other (secondary) paths only check the resource availability. If the reservation on the primary path fails, a secondary path is picked for resource reservation.

The other interpretation of multipath routing is to select a set of paths instead of a single one for a connection. When there does not exist a feasible path with sufficient resources, the algorithm tries to find multiple paths whose combined resources satisfy the requirement. Transmitting contiguous data (audio and video) along multiple paths arises the problem of synchronization and demands more buffer space at the receiving end to absorb the delay jitter between different paths.

Routing QoS and best-effort traffic: QoS traffic and best-effort traffic co-exist in most real-world networks. A primary task of routing is to maximize the *resource efficiency*, which is measured by two goals. One goal is to maximize the number of QoS flows that are admitted into the network, which is equivalent to minimize the *call-blocking ratio*. The other goal is to optimize the throughput and responsiveness of best-effort traffic. The two goals may contradict each other. That is because (1) the first goal considers only QoS traffic, (2) the second goal considers only best-effort traffic and (3) however the two types of traffic may have very different distributions. Generally speaking, the QoS traffic will not be affected by the best-effort traffic due to resource reservation. However, the throughput of the best-effort traffic will suffer if the overall traffic is misjudged. For example, links with light QoS traffic may have heavy best-effort traffic, and by many QoS routing algorithms, these links are often considered as good candidates for new QoS flows, which however causes the already congested best-effort traffic even more congested.

Re-routing: There are a number of situations where re-routing is desired. First, the routes of the connections are typically selected based on the network resource availability at the times when the requests arrive. Long paths are often assigned when resource contention occurs. However, as new connections are established and existing connections are torn down upon completion, the network state changes locally and globally, which makes the routes of the remaining connections less optimal [38]. Routes with light (heavy) traffic at the beginning may become congested (lightly loaded) later. Shorter paths for some connections may become available. Re-routing helps to balance the network traffic on the fly and improves the resource efficiency. Second, when there does not exist a feasible path for a new connection, instead of rejecting the connection, it is often possible to re-route some existing connections in order to make room for the new one. Re-routing is especially useful when connections have different priorities. A new connection with a higher priority will preempt the resources held by the existing connections. Instead of throwing the preempted connections out of the network, we can re-route them to other paths. The re-routing should not be done too frequently in order to avoid the excessive overhead and the oscillation of shifting the traffic from one part of the network to another.

Integration with other network components: Routing must work with other network components in order to provide guaranteed services. The design of routing algorithms must consider how the global state is maintained, how resources are reserved and how data packets are scheduled. Different scheduling policies make different requirements on routing algorithms, and often provide special properties to simplify the routing problems [32]. For example, when the rate-based scheduling policies are used, the end-to-end delay constraint can be transformed into a bandwidth constraint. In an integrated network system, the following properties are desired for the routing component.

Generality: Multimedia applications tend to have diverse QoS requirements on bandwidth, delay, delay jitter, cost, path length, etc. From a network designer's point of view, it would be beneficial to develop a *generic* routing algorithm, instead of implementing different routing algorithms for different types of QoS requirements independently. The generic algorithm captures the common messaging and computational structure, from which various concrete algorithms are derived by specifying the QoS-dependent open components [9].

Extensibility: As the network infrastructure evolves and the capacity increases, new applications are made possible, which require the routing algorithms to adapt in order to accommodate new service types. Designing the extensible algorithms and making them adapt to new applications are important as the networks become increasely complex and the deployment of new routing algorithms are very costly and problem-prone.

Simplicity: The simplicity of a routing algorithm in terms of time/logical complexity and state information requirement often allows efficient implementation, debugging and evaluation. It also makes the algorithm easier to be understood, maintained and upgraded.

9 Summary

The QoS routing is a key network function for the provision of guaranteed services in future high-speed networks. It has two objectives: finding routes that satisfy the QoS constraints and making the efficient use of the network resources. Based on the way the state information is maintained, the existing unicast/multicast routing algorithms can be divided into three classes: (1) source routing, (2) distributed routing and (3) hierarchical routing algorithms. The source routing algorithms are most thoroughly investigated. They simplify the path selection problem by locally computing a feasible path based on a global state that is maintained at every node. The responsibility of path selection is shared by intermediate nodes in distributed routing. Most existing distributed routing algorithms also require the maintenance of a global state. Limited work has been done on the hierarchical routing, especially for the NP-complete routing problems.

The polynomial-complexity routing problems (Figure 4) were well solved by the shortest path based algorithms. Heuristics were proposed for the NP-complete routing problems (Figure 4) with close-to-optimal results. However, there are still problems remaining. Most source and distributed algorithms do not scale well due to the need of maintaining up-to-date global states which is difficult in large networks with dynamic data traffic. In addition, the source heuristic algorithms often have prohibitively high time complexities, which limits their practical values. The hierarchical routing provides a solution which is scalable because the path selection is based on the aggregated state information whose size is much reduced. However, information imprecision is an issue of concern due to the state aggregation. Furthermore, it is an unsolved problem to aggregate the state of a subnet measured by multiple QoS metrics (Section 5.3, Figure 6). Future research should focus on efficient heuristic algorithms for the NP-complete routing problems, state aggregation with multiple QoS metrics, hierarchical routing with imprecise information, multipath routing, integrated routing of QoS and best-effort traffic, rerouting for dynamic traffic load, and efficient routing algorithms based on specific network models such as the rate-based scheduling network.

References

[1] B. Awerbuch, Y. Azar, S. Plotkin, and O. Waarts. Throughput-competitive on-line routing. *Proceedings of 34th Annual Symposium on Foundations of Computer Science, Palo Alto, California*, November 1993.

[2] B. Awerbuch, Y. Azar, S. Plotkin, and O. Waarts. Competitive routing of virtual circuits with unknown duration. *the 5th ACM-SIAM Symposium on Discrete Algorithms*, January 1994.

[3] B. Awerbuch, Y. Du, B. Khan, and Y. Shavitt. Routing through teranode networks with topology aggregation. *ISCC'98, Athens, Greece*, June 1998.

[4] J. Behrens and J. J. Garcia-Luna-Aceves. Hierarchical routing using link vectors. *IEEE INFOCOM'98*, March 29 - April 2 1998.

[5] R. E. Bellman. *Dynamic Programming.* Princeton University Press, Princeton, N.J., 1957.

[6] J. Bennett and H. Zhang. Hierarchical packet fair queueing algorithms. *ACM SIGCOMM'96*, 1996.

[7] J.-Y. Le Boudec and T. Przygienda. A route precomputation algorithm for integrated services networks. *Journal of Network and Systems Management*, 3(4):427–449, 1995.

[8] S. Chen and K. Nahrstedt. Distributed qos routing. *Tech. Report UIUCDCS-R-97-2017, Department of Computer Science, University of Illinois at Urbana-Champaign*, July 1997.

[9] S. Chen and K. Nahrstedt. Distributed quality-of-service routing in high-speed networks based on selective probing. *Technical Report, University of Illinois at Urbana-Champaign, Department of Computer Science*, 1998.

[10] S. Chen and K. Nahrstedt. On finding multi-constrained paths. *IEEE International Conference on Communications*, June 1998.

[11] I. Cidon, R. Rom, and Y. Shavitt. Multi-path routing combined with resource reservation. *Infocom'97, Japan*, pages 92–100, April 1997.

[12] S. Deering and D. Cheriton. Multicast routing in datagram internetworks and extended lans. *ACM Transactions on Computer Systems*, pages 85–111, May 1990.

[13] A. Demers, S. Keshav, and S. Shenker. Analysis and simulation of a fair queueing algorithm. *ACM SIGCOMM'89*, pages 3–12, 1989.

[14] E. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[15] D. Ferrari and D. C. Verma. A scheme for real-time channel establishment in wide-area networks. *IEEE JSAC*, 8(3):368–379, April 1990.

[16] ATM Forum. Private network network interface (pnni) v1.0 specifications. June 1996.

[17] ATM Forum. Private network network interface (pnni) v1.0 specifications. May 1996.

[18] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* New York: W.H. Freeman and Co., 1979.

[19] S. Golestani. A self-clocked fair queueing scheme for broadband applications. *Proceedings of IEEE INFOCOM'94*, pages 636–646, June 1994.

[20] R. Guerin and A. Orda. Qos-based routing in networks with inaccurate information: Theory and algorithms. *Infocom'97, Japan*, April 1997.

[21] C. Hedrick. Routing information protocol. *RFC 1058, Network Information Center, SRI Internation, Menlo Park, CA*, June 1988.

[22] C. Hou. Routing virtual circuits with timing requirements in virtual path based atm networks. *INFOCOM'96*, 1996.

[23] F. K. Hwang and D. S. Richards. Steiner tree problems. *Networks*, 22:55–89, 1992.

[24] A. Iwata, R. Izmailov, H. Suzuki, and B. Sengupta. Pnni routing algorithms for multimedia atm internet. *NEC Research and Development*, 38, 1997.

[25] D. D. Kandlur, K. G. Shin, and D. Ferrari. Real-time communication in multi-hop networks. *Eleventh Int'l Conf. on Distributed Computing Systems*, pages 300–307, 1991.

[26] V. P. Kompella, J. C. Pasquale, and G. C. Polyzos. Multicast routing for multimedia communication. *IEEE/ACM Transactions on Networking*, June 1993.

[27] V. P. Kompella, J. C. Pasquale, and G. C. Polyzos. Two distributed algorithms for multicasting multimedia information. *Proceedings of the 2nd International Conference on Computer Communications and Networks (IC-CCN), San Diego, CA*, pages 343–349, June 1993.

[28] L. Kou, G. Markowsky, and L. Berman. A fast algorithm for steiner tree. *Acta Informatica 15*, pages 141–145, 1981.

[29] W. C. Lee, M. G. Hluchyi, and P. A. Humblet. Routing subject to quality of service constraints in integrated communication networks. *IEEE Network*, July/August 1995.

[30] D. H. Lorenz and A. Orda. Qos routing in networks with uncertain parameters. *Infocom'98*, March 1998.

[31] K. Lougheed and Y. Rekhter. Border gateway protocol 3 (bgp-3). *RFC 1267, SRI International, Menlo Park, CA*, October 1991.

[32] Q. Ma and P. Steenkiste. Quality-of-service routing with performance guarantees. *Proceedings of the 4th International IFIP Workshop on Quality of Service*, May 1997.

[33] J. Moy. Multicast extension to ospf. *Internet Draft*, September 1992.

[34] J. Moy. Ospf version 2. *Network Working Group Internet Draft*, November 1992.

[35] A. Orda. Routing with end to end qos guarantees in broadband networks. *INFOCOM'98*, March 1998.

[36] International Standard Organization. Intra-domain is-is routing protocol. *ISO/IEC/JTC1/SC6 WG2 N323*, September 1989.

[37] International Standards Organization. Protocol for exchange of inter-domain routing information among intermediate systems to support forwarding of iso 8473 pdus. *ISO/IEC/JTC1/SC6 CD10747.*

[38] C. Parris, H. Zhang, and D. Ferrari. A mechanism for dynamic re-routing of real-time channels. *Technical Report TR-92-053, International Computer Science Institute, Berkeley, CA*, April 1992.

[39] M. Peyravian and A. D. Kshemkalyani. Network path cashing: Issues, algorithms and a simulation study. *Computer Communications*, 20:605–614, 1997.

[40] S. Plotkin. Competitive routing of virtual circuits in atm networks. *IEEE Journal on Selected Areas in Communications*, 13:1128–1136, August 1995.

[41] R. Prim. Shortest connection networks and some generalizations. *Bell Systems Tech. J.*, 36:1389–1401, 1957.

[42] G. N. Rouskas and I. Baldine. Multicast routing with end-to-end delay and delay variation constraints. *IEEE JSAC*, 15(3):346–356, April 1997.

[43] H. F. Salama, D. S. Reeves, and Y. Viniotis. A distributed algorithm for delay-constrained unicast routing. *INFOCOM'97, Japan*, April 1997.

[44] Hussein F. Salama, Douglas S. Reeves, and Yannis Viniotis. Evaluation of multicast routing algorithms for real-time communication on high-speed networks. *IEEE Journal on Selected Areas in Communications*, 15(3):332–345, April 1997.

[45] A. Shaikh, J. Rexford, and K. Shin. Dynamics of quality-of-service routing with inaccurate link-state information. *U. of Michigan Tech. Report CSE-TR-350-97*, Nov. 1997.

[46] A. Shaikh, J. Rexford, and K. Shin. Efficient precomputation of quality-of-service routes. *Workshop on Network and Operating Systems Support for Digital Audio and Video,*, July 1998.

[47] K. G. Shin and C.-C. Chou. A distributed route-selection scheme for establishing real-time channel. *Sixth IFIP Int'l Conf. on High Performance Networking Conf. (HPN'95)*, pages 319–329, Sep. 1995.

[48] M. Steenstrup. Inter-domain policy routing protocol specification: Version 1. *Internet Draft*, May 1992.

[49] Q. Sun and H. Langendorfer. A new distributed routing algorithm with end-to-end delay guarantee. *Second Workshop Protocols Multimedia Systems (PROMS'95)*, pages 452–458, Oct. 1995.

[50] Q. Sun and H. Langendorfer. A new distributed routing algorithm with end-to-end delay guarantee. *Unpublished paper*, 1997.

[51] H. Takahashi and A. Matsuyama. An approximate solution for the steiner tree problem in graphs. *Mathematica Japonica*, 1980.

[52] Z. Wang and J. Crowcroft. Qos routing for supporting resource reservation. *JSAC*, September 1996.

[53] R. Widyono. The design and evaluation of routing algorithms for real-time channels. *Technical Report, ICSI TR-94-024, University of California at Berkeley International Computer Science Institute*, June 1994.

[54] L. Zhang. Virtual clock: A new traffic control algorithm for packet switching networks. *ACM SIGCOMM'90*, pages 19–29, September 1990.

[55] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala. Rsvp: A new resource reservation protocol. *IEEE Network*, September 1993.

[56] Q. Zhu, M. Parsa, and J.J. Garcia-Luna-Aceves. A source-based algorithm for delay-constrained minimum-cost multicasting. *Proc. IEEE INFOCOM 95, Boston, MA*, April 1995.