Unicast QoS Routing in Overlay Networks

Dragos Ilie[1] and Adrian Popescu[2]

[1] dragos.ilie@gmail.com
[2] Blekinge Institute of Technology, Sweden
adrian.popescu@bth.se

Abstract. The goal of quality of service (QoS) routing in overlay networks is to address deficiencies in today's Internet Protocol (IP) routing. This is achieved by application-layer protocols executed on end-nodes, which search for alternate paths that can provide better QoS for the overlay hosts.
In the first part of this paper we introduce fundamental concepts of QoS routing and the current state-of-the-art in overlay networks for QoS. In the remaining part of the paper we report performance results for the Overlay Routing Protocol (ORP) framework developed at Blekinge Institute of Technology (BTH) in Karlskrona, Sweden. The results show that QoS paths can be established and maintained as long as one is willing to accept a protocol overhead of maximum 1.5 % of the network capacity.

1 Introduction

One of Internet's characterizing features is its pervasive nature. Online banking, online shopping, voice over IP (VoIP), IP Television (IPTV), video on demand (VoD) and social networking (*e. g.*, Facebook and Twitter) are just a few examples of Internet services that have become an integral part of our lives. Some of these services, in particular those making heavy use of video and audio streams, have strict requirements on how the media streams must be handled during transit in the network. The requirements are typically expressed in the form of constraints on bandwidth[3], packet delay, delay jitter and packet loss. A network that meets these requirements is said to provide QoS [1]. A key issue in providing QoS is that of selecting paths for network traffic such that the stream requirements are satisfied. This can be done by QoS routing, which is a mechanism for optimizing network performance by constrained-path selection and traffic flow allocation.

To be more specific, QoS routing solves the following problem. A *source node* must transfer data over a network to a set of *destination nodes*. The two sets can be overlapping in the sense that some nodes are both senders and receivers. The data is in the form of packets. These packets are grouped into *flows*, where a flow consists of packets sharing the same source and destination address as well as a

[3] In the field of computer networking, the term *bandwidth* is used to denote data rate or capacity, unless specified otherwise.

set of common QoS requirements called the *flow demand*. The problem is how to satisfy all flow demands simultaneously. Two sub-problems must be solved in order to meet the flow demands. First, one must find at least one path that connects each source node to the corresponding destination node. Second, the flows must be allocated to these paths without violating their QoS requirements. A more far reaching objective is to solve these two problems while maximizing network throughput and minimizing congestion.

The network architect relies on a number of low-level building blocks in order to design a network with support for QoS. Examples of QoS low-level building blocks routing algorithms and protocols, resource reservation protocols, traffic shapers scheduling algorithms, admission control mechanisms, congestion avoidance and congestion control techniques. A *QoS architecture* defines how building blocks are combined in order to provide QoS.

Integrated Services (IntServ) is the first proposed QoS architecture for IP-based networks [2]. In IntServ, resources are allocated along the path by using the Resource Reservation Protocol (RSVP) [3]. IntServ performs *per-flow* resource management. This has led to skepticism towards IntServ's ability to scale, since core routers in the Internet must handle several hundred thousands flows simultaneously [4].

A second QoS architecture called Differentiated Services (DiffServ) [5] was developed, due to concerns about IntServ's scalability. DiffServ attempts to solve the scalability problem by dividing the traffic into separate forwarding classes. Each forwarding class is allocated resources as stipulated in the service level agreement (SLA) between provider and customer. Packets are classified and mapped to a specific forwarding class at the edge of the network. Inside the core, routers handle the packets according to their forwarding class. Since routers do not have to store state information for every flow, but only have to inspect certain fields in the packet header, it is expected that DiffServ scales much better than IntServ. A major problem with the DiffServ architecture has to do with end-to-end QoS provisioning over multiple DiffServ domains. Premium services cannot be offered unless bilateral SLAs exist between peering domains over the entire end-to-end path. Currently, technical difficulties coupled with the providers' lack of incentive to engage in bilateral SLAs has prevented wide-spread deployment of DiffServ [6, 7].

Both architectures are of benefit to services and users located in the same network (*e. g.* , the corporate network), but fail to address a more heterogeneous scenario, where the service provider and users are scattered across the Internet. The main reason for this situation is because of the lack of interaction between network providers or difficulties to align premium services to a common denominator among the providers [8].

Furthermore, another issue to consider in the context of QoS is that of inter-domain routing. From a hierarchical point of view, Internet consists of a large number of autonomous systems (ASs). Interconnected ASs exchange routing information using the Border Gateway Protocol (BGP) [9]. An AS connects to other ASs through peering agreements. A peering agreement is typically a business contract stipulating the cost of routing traffic across an AS along with other policies to be maintained. When there are several routes to a destination the peering agreements force an AS to prefer certain routes over others. For example, given two paths to a destination where the first one is shorter (in terms of hops) and the second one is cheaper, the AS will tend to select the cheaper path. This is called *policy routing* and is one of the reasons for suboptimal routing. With the commercialization of the Internet it is unlikely that problems related to policy routing will disappear in the near future.

In summary, the Internet landscape today consists of a number of “islands” where QoS is provided by IntServ, DiffServ or other customized architectures. As long as peering agreements exist QoS services can be extended beyond the “island” borders. Otherwise, QoS is limited to IP’s best-effort service. Consequently, some researchers are investigating the possibility to deploy QoS in overlay networks on top of IP. Overlay networks are more flexible to change architecturally, less dependent on network providers and cheaper to deploy. On the other hand, they face many challenges in the form of fluctuating QoS resources, coordination and scalability.

The remainder of this article is structured as follows. An overview of the current state-of-the-art in overlay networks for QoS is presented in Section 2. That is followed by a brief presentation of essential principals of QoS routing in Section 3. Section 4 introduces basic models for path selection and flow allocation. Simulation results for overlay routing protocols developed at BTH in Sweden are described in Section 5. We conclude with a brief summary and suggestions for future work. This article is based on the work presented in [10].

2 Overlay Networks for QoS

An overlay network utilizes the services of an existing network in an attempt to implement new or better services. Each overlay node establishes *virtual links* with a subset of its peers according to rules specific to the overlay. A virtual link consists of a sequence of one or more physical links (*i. e.* , a physical path). The physical path is selected by inter-domain (*e. g.* , BGP) and intra-domain routing protocols such as the Routing Information Protocol (RIP) and the Open Shortest Path First (OSPF). *Virtual paths* are selected by overlay routing protocols.

An example of an overlay network is shown in Figure 1. The physical interconnections of three ASs are depicted at the bottom of the figure. The gray circles denote nodes that use the physical interconnections to construct virtual paths used by the overlay network at the top of the figure.

The nodes participating in the overlay network perform active measurements to discover the QoS metrics associated with the virtual paths. As an example, assume that an overlay node in AS1 wishes to communicate with another overlay node in AS2. Assume further that AS1 always routes packets to AS2 by using the direct link between them, due to some policy or performance metric. The overlay node in AS1 may discover through active measurements that the path crossing AS3 can actually provide better QoS (*e. g.* , smaller delay), than the direct link

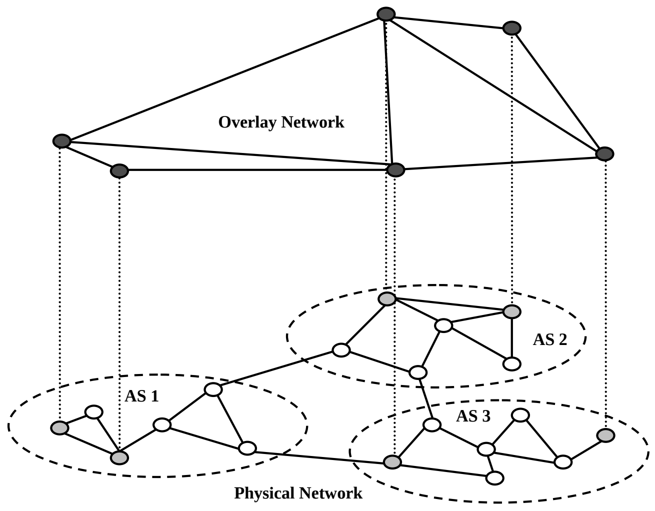


Fig. 1. Overlay network.

to AS2. In this specific case, the AS1 node forwards its traffic to the AS3 node, which in turn forwards the traffic to the destination node (or to the next node on the path if multiple hops are necessary). It is worth to emphasis that the overlay network can choose its routes automatically, without involving the autonomous systems into its decisions. This is the basic argument to motivate routing in overlay networks. Examples of such overlays are the Resilient Overlay Network (RON), OverQoS, the QoS-aware routing protocol for overlay networks (QRON) and the QoS overlay network (QSON).

In RONs [11], strategically placed nodes in the Internet are organized in an application-layer overlay. Nodes belonging to the overlay aid each other in routing packets in such a way as to avoid path failures in the Internet. Each RON node carefully monitors the quality of Internet paths to his neighbors through active measurements. In order to discover the RON topology, RON-nodes exchange routing tables and various quality metrics (*e. g.*, latency, packet loss rate, throughput) using a link-state routing protocol. The path selection is done at the source, which signals to nodes downstream the chosen path. Nodes along the path signal to the source nodes information about link failures pertaining to the selected path. Results involving thirteen sites scattered widely over Internet showed the feasibility of this solution. RON's routing mechanism was able to detect and route around all 32 outages that occurred during the time frame for the experiment, 1 % of the transfers doubled their Transmission Control Protocol (TCP) throughput and 5 % had their loss rate reduced with 5 %.

Following the success of RONs, the authors of [12] propose OverQoS, an overlay-based QoS architecture for enhancing Internet QoS. The key part of the architecture is the controlled-loss virtual link (CLVL) abstraction, which provides statistical loss guarantees to a traffic aggregate between two overlay nodes in the presence of changing traffic dynamics. They demonstrate that their architecture can supply the following QoS enhancements with as little as 5 % bandwidth overhead: smoothing losses, packet prioritization, as well as statistical bandwidth and loss guarantees.

Another approach involving strategically placed nodes in the Internet is presented in [13]. The authors propose an architecture where each AS has one or more overlay brokers. The overlay brokers are organized into clusters that interconnect with each other to form an overlay service network that runs a QRON. The purpose of QRON is to find an overlay path satisfying a bandwidth constraint. QRON nodes use source routing and a number of backup paths to cope with bandwidth fluctuations. The authors were able to show that the QRON algorithms perform well under a variety of traffic loads while balancing the load among overlay brokers.

In a similar spirit, the QSON architecture[14] advocates a backbone overlay network for QoS routing. This architecture relies on well-established business relationships of two kinds. The first type of business relationships is defined by end-users who purchase QoS services from the QSON provider. The QSON provider is able to supply these services by engaging in SLAs with several Internet service providers (ISPs). This is the second kind of business relationships. The QSON overlay is spanned by QSON proxies located between ISP domains. Each proxy stores a list of paths to the other proxies. The proxies use probes to reserve bandwidth and to inform each other about changes in available bandwidth. Simulation results have shown that QSON is able to provide bandwidth reservation with low control overhead.

3 Principles of QoS Routing

In QoS networks every link and every node has a state described by specific QoS metrics. The *link state* can consist of available bandwidth, delay and cost whereas the *node state* can be a combination of available memory, central processing unit (CPU) utilization and harddisk storage. The link state and the node state may be considered separately or they may be combined. The focus here is on link state.

Routing is the process of finding a path between two hosts in a network. In QoS routing, the path must be selected such that QoS metrics of interest stay within specific bounds. The routing process relies on a *routing algorithm* for computing constrained paths and on a *routing protocol* for distributing state information. In general, the algorithm is independent from the protocol. The coupling between them is decided when the QoS routing architecture is specified. For example, the QoS-enabled OSPF (QOSPF) protocol suggests that a modified Bellman-Ford algorithm should be used for pre-computed paths and Dijkstra's algorithm for on-demand paths [15].

There are three basic forms of storing state information: *local state*, *global state* and *aggregated (partial) state* [16].

When a node keeps local state, it maintains information about its outgoing links only. No information about the rest of the network is available.

A global state is the combination of local states for all nodes in a graph. Global states are imprecise (*i. e.*, they are merely approximations of the global state) due to non-negligible delay in propagating information about local states[4]. When the network size grows, the imprecision grows as well. This makes it hard to maintain an accurate picture about resource availability in the network and in turn has a severe impact on QoS routing.

Aggregated state aims to solve scalability issues in large networks. The basic idea is to group together adjacent nodes into a single *logical node*. The local state of a logical node is the aggregation of local states for physical nodes that are part of the logical node. Similar to the case of global state, this leads to imprecision that grows with the amount of state information aggregated in the logical node.

Imprecision, also called uncertainty, is not generated by aggregation only. Other sources of uncertainty are network dynamics (churn), information kept secret by ISPs due to business reasons, as well as approximate state information due to systematic or random errors in the measurement process [17]. An interesting solution suggested for mitigating these problems is to replace the deterministic state information metrics with random variables. In this case, the routing algorithm must be changed such as to select feasible paths on a probabilistic basis, with the result that the selected paths are those most likely to satisfy the QoS constraints [18]. However, a non-trivial problem with this approach lies in the estimation of the probability distributions for state variables [19].

There are three different classes of routing strategies, each corresponding roughly to one form of maintaining state information: *source routing*, (*flat*) *distributed routing* and *hierarchical routing* [20].

In source routing the nodes are required to keep global state and the feasible path is computed at the source node. The main advantage in this case is that route computation is performed in a centralized fashion, avoiding some of the problems associated with distributed computation. The centralized computation can guarantee loop-free routes. One disadvantage of source routing is the requirement to maintain global state. In a network where the QoS metrics often change, this requires large communication overhead in order to keep the state information updated. Additionally, due to the propagation delay, the state information may become stale before reaching the destination. This leads to imprecise state information, as explained above. Furthermore, depending on the network size and the number of paths to compute, the source routing algorithm can result in very high computational overhead [20].

Distributed routing, typically, also relies on nodes maintaining global states, but the path computation is performed in a distributed fashion. This diminishes computational overhead and also allows concurrent computation of multiple routes in search for a feasible path. Distributed computation suffers from prob-

[4] Link state information is said to become stale when network latency prevents timely updates.

lems related to distributed state snapshot, deadlock and loop occurrence [16]. Additionally, when global state is maintained, distributed routing shares with source routing the problems related to imprecise state information.

Some suggestions on using flooding-based algorithms, require nodes to maintain local state only [21, 22]. This mitigates problems related to imprecise state information. However, flooding-based algorithms tend to generate large volumes of traffic compared to the other forms of routing.

In hierarchical routing, the network is divided into groups of nodes and the state information is aggregated for the nodes participating in a group. With this form of aggregation, a group appears as a logical node. One node in the group is designated *leader* or *border node* and acts as a gateway for the communication with other logical nodes. Each group can in turn be divided into smaller groups. Using this form of recursion, several hierarchical levels can be created. Nodes maintain global state information for peers within a group and aggregated state information about the other groups. The major advantage of hierarchical routing is scalability[20, 23]. In particular, since nodes maintain aggregated state information there is less state information to be transmitted to other nodes, hence less communication overhead. For the same reason, there is also less computational overhead. However, each level of hierarchy induces additional uncertainty in the state information. This problem becomes more difficult when several QoS metrics must be aggregated, since for some topologies there can be no meaningful way to combine the metrics [16]. Some solutions for topology aggregation are presented in [24, 25].

In large overlay networks such as Skype, Vuze[5] DHT and Gnutella, the nodes are computers controlled by regular users instead of a central authority. This type of decentralized network is appealing to service providers because the cost of bandwidth and computation required to deliver the service is shifted towards the end-users. Unfortunately, such an overlay network tends to be an unreliable infrastructure. By this, we mean that end-nodes are at owner's whim, and users can turned them off at any time. When a node is turned off a set of links is removed from the overlay network. If the links belong to established QoS paths, their removal will trigger path re-computations. Additionally, traffic flows may need to be reallocated to new paths. This type of node churn is similar to the topology dynamics occurring in mobile ad-hoc networks, when stations move out of radio range. Routing protocols that handle topology dynamics can be classified as *proactive* or *reactive* protocols.

Proactive protocols, such as destination sequence distance vector (DSDV) periodically update the routing tables [26]. In contrast, reactive protocols (*e. g.*, dynamic source routing (DSR) and ad-hoc on-demand distance vector (AODV)) update the routing tables only when routes need to be created or adjusted due to changes to topology [26]. Proactive protocols are in general better at providing QoS guarantees for real-time traffic such as multimedia. Their disadvantage lies in the traffic volume overhead generated by the protocol itself. Reactive protocols

[5] Formerly known as Azureus.

scale better than proactive protocols, but will experience higher latency when setting up a new route [26].

The focus of this section has been so far placed on single path routing. *Multipath routing* exploits path diversity in the network to find several paths that can satisfy a flow demand. In single path routing, a flow demand is dropped if no suitable path is found. With multipath routing, there is still a chance to satisfy the flow demand, particularly in the case of bandwidth, by spreading the flow over several paths. The ability to spread demands over multiple paths offers additional advantages for load balancing, congestion control, reliability and security [27]. However, multipath routing introduces additional overhead in the form of bandwidth usage and processing requirements. The bandwidth overhead is due to extra link state information that nodes must share. A multipath router must compute and store in memory more than one path for each known source-destination pair. This is the reason for higher processing requirements than in the case of single-path routing.

QoS routing in overlay networks faces additional challenges such as those related to estimation of link state parameters and selfish behavior. We describe them briefly below.

A typical overlay networks consist of nodes, which are computer processes executing at the application layer, and of virtual links[6] implemented on top of TCP or the User Datagram Protocol (UDP). In this scenario, the link state parameters are not readily available but must be estimated through active measurements.

Although active measurement methods exist for most metrics of interest [28, 29] there are still open questions to be answered before they can used in practice. One of these issues is that of finding an adequate measurement frequency. In other words, we are asking how long should we wait after obtaining data from one measurement until we start the next measurement. Ideally, one wants to capture all significant changes in the measured metric while keeping the volume of injected traffic at a minimum. Another issue is that of estimating the validity of the measurements when all nodes in the overlay execute them concurrently or when active measurements for different metric types are performed simultaneously.

In fact, the issues related to active measurements are related to a bigger problem: the likely mismatch in the interaction between overlay routing and traffic engineering [30, 31]. The goal of overlay routing is to satisfy in an optimal way the flow demands within the overlay network. On the other hand, the goal of traffic engineering is to optimize the performance of the *whole* network, including any existing overlays. Whereas traffic engineering addresses the needs of all nodes under a common administrative domain (*e. g.* , an AS), overlay routing caters for a subset of the Internet, with nodes from many different networks.

The *traffic matrix* concept is central to traffic engineering. A traffic matrix contains estimates of the traffic volumes exchanged by all source-destination pairs in the network. These estimates include both overlay and non-overlay traf-

[6] Also called logical links.

fic. The traffic matrix is used as input for the inter- and intra-domain routing algorithms. If the traffic matrix changes, flows within the network can be re-routed as a result. The overlay routing algorithm reacts to changes in the network layer routes by re-adjusting the overlay traffic to repair flow demands that are no longer satisfied. Thus, two closed-loop control mechanisms modify each other's input. The selfish behavior of the overlay network can lead to traffic oscillations. Results presented in [30, 31] indicate that the price for optimizing the traffic in the overlay can be severe performance degradation in the rest of the network. Gaining a better understanding of the interaction between traffic engineering and selfish routing is an important research topic.

4 Optimization Models

In Section 1 it was stated that QoS routing is a mechanism for optimizing network performance by constrained-path selection and traffic flow allocation. Given one flow demand and information about the network, the goal of constrained-path selection is to find a path such that the flow demand is satisfied. For multiple flow demands, the constrained-path selection algorithm must be run several times, once for each flow demand. In contrast, the goal of flow allocation is to satisfy multiple demands simultaneously. This is the same goal as that of traffic engineering. In fact, flow allocation is one of the components required by traffic engineering. Other required components are topology and state discovery, traffic demand estimation and configuration of network elements [7].

Flow allocation algorithms tend to utilize resources more efficiently since all routes are recomputed when the demands change. When used in combination with multipath routing the efficiency can be increased by exploiting path redundancy. However, a major disadvantage of flow allocation is that ongoing flows can be temporarily disrupted when routes change [7]. Constrained-path selection algorithms handle each changing flow demand by itself, without disrupting other flows. Although different, these two types of algorithms can be combined as it is shown in Section 5.

The algorithms assume that information about network topology is available in the form of a weighted digraph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes (vertices) in the network and the set $\mathcal{E}$ contains the links (edges). The weight of each link represents a set of metrics of interest, such as bandwidth, delay, jitter, packet loss and cost. In addition to the graph and link weights, information about the flow demands is available as well. A flow demand is expressed as a set of path constraints for the path $P(s, d)$, where $s \in \mathcal{V}$ is the source node and $d \in \mathcal{V}$ is the destination (sink) node. In its simplest form, the flow demand contains only the bandwidth required to transfer data from s to d. It is assumed here that flow demands are tied to the direction of the path.

In the case of a multi-constrained path (MCP) problem we attempt to find one constrained path at a time. This is a feasibility problem. Each link weight in $\mathcal{G}(\mathcal{V}, \mathcal{E})$ is a vector of QoS metrics, where each metric belongs to one of the following types:

additive: delay, jitter, cost
multiplicative: packet loss
min-max: bandwidth, policy flags

Multiplicative weights can be turned into additive weights by taking the logarithm of their product. The constraints on min-max metrics can be dealt with by pruning the links of the graph that do not satisfy the constraints [32]. Therefore, in the remainder of this section we focus on additive link weights only.

For $i = 1, \ldots, m$ we denote by $w_i(u, v)$ the ith additive metric for the link (u, v) between nodes u and v such that $(u, v) \in \mathcal{E}$. The MCP optimization problem for m constraint values L_i on the requested path is shown in Table 1.

$$\begin{array}{ll} \text{find} & \text{path } P \\ \text{subject to} & w_i(P) = \displaystyle\sum_{(u,v)\in P} w_i(u,v) \le L_i \text{ for } i = 1,\ldots,m \text{ and } (u,v) \in \mathcal{E} \end{array}$$

Table 1. Multi-constrained path selection problem (MCP).

The MCP selection problem problem can be converted to a multi-constrained optimal path (MCOP) selection problem by minimizing or maximizing over one of the metrics w_i. It is also possible to define a path-weight function f over all metrics and to optimize over the path-weight function itself, as shown in Table 2.

$$\begin{array}{ll} \text{minimize} & f(\boldsymbol{w}(P)) \\ \text{subject to} & w_i(P) = \displaystyle\sum_{(u,v)\in P} w_i(u,v) \le L_i \text{ for } i = 1,\ldots,m \text{ and } (u,v) \in \mathcal{E} \end{array}$$

Table 2. Multi-constrained optimal path selection problem (MCOP).

Wang and Crowcroft proved in [33] that MCP problems with two or more constraints are $\mathcal{NP}$-complete. By extension, MCOP problems with two or more constraints are $\mathcal{NP}$-complete as well. The apparent intractability of these problems suggests abandoning the search for exact solutions in the favor of heuristics that have a better chance of running in polynomial time. Chen and Nahrstedt suggest a $\mathrm{O}(2L)$ heuristic [21] for the MCP problem, where L is the length of the feasible path.

The results of a study [34] on the $\mathcal{NP}$-complexity of QoS routing found four conditions leading to its appearance:

- graphs with long paths (large hop-count),
- link weights with infinite granularity, or excessively large or small link weights,
- strong negative correlation among link weights,
- "critically constrained" problems, which are problems with constraint values close to the center of the feasible region.

The authors of the study consider that these conditions are unlikely to occur in typical networks. If they are right, the consequence is that the exponential run time behavior of exact algorithms occurs very seldom.

In the flow allocation problem, it is assumed that we know about one or more directed paths connecting a source node s and a destination node d. These paths can be discovered automatically, for example with a K shortest paths (KSP) algorithm [10, 35]. Due to space constraints we restrict our focus to bandwidth allocation problems only. We consider the following type of optimization problems: given a digraph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, a set $\mathcal{P}$ of directed paths and a set $\mathcal{D}$ of flow demands for bandwidth, we would like to allocate bandwidth on the paths in $\mathcal{P}$ such as to simultaneously satisfy all demands.

If the traffic volume pertaining to a specific flow is allowed to be distributed over several paths to the destination, this is said to be a feasibility problem for *bifurcated flows*. On the other hand, if the problem includes the requirement that the entire traffic flow between two nodes must be transmitted on a single path, we have a feasibility problem for *non-bifurcated flows*. This problem is known to be computationally intractable [36] for large networks (it is in fact $\mathcal{NP}$-complete). The remainder of this article considers only feasibility problems for bifurcated flows.

We adopt a notation called *link-path formulation* [36] to formalize our problem statement. Using this notation, we let the variable x_{dp} denote bandwidth allocated to demand d on path p. Recall that a demand is a request for a specific amount of bandwidth, h_d, from a source node to a destination node. The source node and the destination node can be connected by more than one path, which explains the use of the index variable p. We use the variables D and E to denote the number of demands in the demand set $\mathcal{D}$ and the number of edges (links) in the set $\mathcal{E}$, respectively. Further, the capacity of a link e is denoted by c_e. The indicator variable δ_{edp} is defined as

$$\delta_{edp} = \begin{cases} 1 & \text{if link } e \text{ is used by demand } d \text{ on path } p, \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

Our problem statement can now be written as shown in Table 3.

$$\begin{array}{lll} \text{find} & x_{dp} & \text{for all } d \in \mathcal{D}, p \in \mathcal{P} \\ \text{subject to} & \sum_p x_{dp} = h_d, & d = 1, 2, \ldots, D \\ & \sum_d \sum_p \delta_{edp} x_{dp} \leq c_e, & e = 1, 2, \ldots, E \end{array}$$

Table 3. Pure allocation problem (PAP).

In [36], it is suggested that the pure allocation problem (PAP) described in Table 3 can be reformulated in the form of the linear optimization problem shown in Table 4. The new problem, PAP with modified link-path formulation

(PAP-MLPF), has an additional variable z to be modified. Unlike the PAP, this problem *always* has a feasible solution in the sense that a minimum value for z can be found. If $z < 0$ in the solution, we have a successful bandwidth allocation. Otherwise the value of z indicates how much additional bandwidth is required to obtain feasibility.

$$\begin{array}{lll} \text{minimize} & z & \text{for all } d \in \mathcal{D}, p \in \mathcal{P} \\ \text{subject to} & \sum_p x_{dp} = h_d, & d = 1, 2, \ldots, D \\ & \sum_d \sum_p \delta_{edp} x_{dp} \le z + c_e, & e = 1, 2, \ldots, E \end{array}$$

Table 4. PAP with modified link-path formulation (PAP-MLPF).

5 A Framework for Overlay Routing Protocols

In this section we present performance results for a couple of routing protocols that combine theoretical concepts presented earlier in this paper.

The Overlay Routing Protocol (ORP) framework was developed at BTH in Karlskrona, Sweden as part of the Routing in Overlay Networks (ROVER) project. The framework consists of two protocols: the Route Discovery Protocol (RDP) and the Route Management Protocol (RMP).

RDP is used to find network paths subject to various QoS constraints [10, 37]. To achieve this goal, RDP uses a form of selective diffusion based on ideas presented in [21, 38].

The purpose of RMP is to alleviate changes in the path QoS metrics, due to node and traffic dynamics. This is done through a combination of path restoration and optimization algorithms for traffic flow allocation on bifurcated paths. The purpose of the flow allocation is to spread the demand on multiple paths towards the destination. The design of RMP is influenced by ideas presented in [23, 39].

5.1 Route Discovery Protocol

RDP is a distributed routing protocol relying on local state information. It's distributed path computation is achieved by selective diffusion. RDP uses two different kinds of packets: control packets (CPs), which are used to explore available paths from a source to a destination, and acknowledgement packets (APs) that transport data about available paths back to the source node.

When a node in the overlay wants to open a route to another overlay node it assembles a CP with the desired QoS constraints. The CP is sent to all adjacent nodes connected by links satisfying the QoS constraints. If at least one feasible link is found, the CP is added to the sender's list of active CPs and a timer is started accordingly. If no information is received before the timer expires, the CP is considered lost and it is removed from the active CP list.

If no feasible link exists, the CP is dropped and no further actions are taken. The receive and forward process is repeated at several nodes until one or more CPs reach the destination node, or all CPs are dropped by intermediate nodes. If all CPs are lost, the nodes on the feasible path eventually experience timeouts and thus are able to free any reserved resources.

The feasible path defined by the first CP that arrives at the destination is copied into an AP. The AP is sent back to the source node over the reverse feasible path[7]. All subsequent CPs that arrive to the destination node are dropped.

Chen and Nahrstedt [21, 16] provide worst-case complexity results for the time and communication overhead required to establish a constrained path with this algorithm. For a path length L, the time complexity is $\mathrm{O}(2L)$. In the case of RDP, the path length is bounded by the time-to-live (TTL) value carried by CPs and APs. Hence, RDP's time complexity for one QoS request is $\mathrm{O}(2\,\mathrm{TTL})$. The communication complexity for one QoS request is on the order $\mathrm{O}(E + \mathrm{TTL})$ according to [10].

5.2 RDP Simulation Results

The focus of the simulations is entirely on bandwidth reservations. In what follows, the term *QoS session* is used to denote a request for a directed path with a constraint on minimum available bandwidth. Each session has an associated session duration, which specifies the life length of the path. If a path is successfully established, the amount of bandwidth specified by the path constraint is reserved for the entire session duration. The links in these experiments are error-free and no churn occurs.

We present here results for the following metrics:

call blocking ratio: ratio between the number of infeasible QoS sessions and the total number of QoS sessions arrived at the network,

low-TTL blocking ratio: ratio between infeasible sessions due to low TTL value and the total number of infeasible sessions[8],

bandwidth overhead: ratio between the average number of RDP bytes per second and network capacity (*i. e.*, the aggregated volume of every link in the network).

Results involving additional metrics are available in [10].

[7] Traveling on the reverse feasible path between node v_1 and node v_N means traveling in the opposite direction on the feasible path (*i. e.*, over hops $v_N, v_{N-1}, \ldots, v_1$).

[8] Low-TTL blocking occurs when an AP is dropped because it has traveled the maximum number of hops allowed. The metric does not take into account the possibility that a feasible path may exist for higher TTL values.

The results shown in Figure 2 and Figure 3 are plotted against increasing values of network utilization. The network utilization, ρ, is defined as [13, 40]

$$\rho = \frac{\lambda T Q H}{\displaystyle\sum_{e \in \mathcal{E}} b_e} \tag{2}$$

where T is the average session duration, Q is the average amount of QoS (bandwidth) requested, H is the average path length across all node pairs, and b_e is the available bandwidth on link e. The simulation parameters are summarized in Table 5.

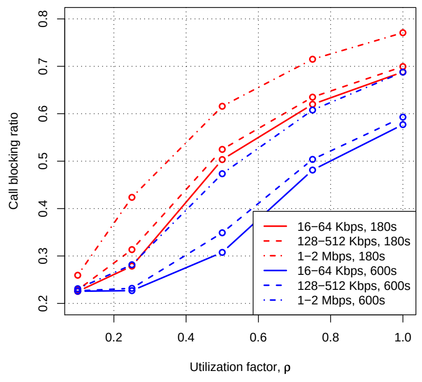
Parameter	**Assigned value**
Network utilization, ρ	$0.1, 0.25, 0.50, 0.75, 1.00$
Session duration	Generalized Pareto with mean 180 s and 600 s
Requested bandwidth ranges	Uniform, 16–64 Kbps, 128–512 Kbps and 1–2 Mbps
Network size	100 nodes
Topology	Barabási-Albert[41, 10]
Link bandwidth	Uniform, 10–10000 Kbps
TTL	8 hops

Table 5. Simulation parameters.

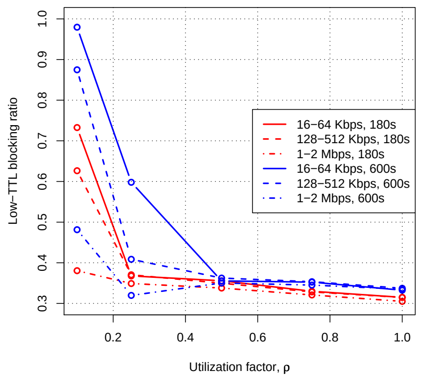
In the plots the blue color is used for sessions with mean duration of 180 seconds, while red color denotes sessions with mean duration of 600 seconds. Plots for 16-64 Kbps sessions are drawn with solid lines, those for 128–512 Kbps are drawn with dashed lines, and 1-2 Mbps session plots use alternating dots and dashes. Each hollow circle indicates the simulated utilization factor pertaining to the value on the y-axis.

The call blocking ratio and the low-TTL blocking ratio are shown side by side in Figure 2. It can be noticed by observing Figure 2(a) that 180 seconds sessions consistently experience higher call blocking ratio than 600 seconds sessions. The explanation is found in Equation 2. This equation is used to compute the utilization factor, ρ, by adjusting the arrival rate, λ, while the other parameters are kept fixed. For a given ρ value, the arrival rate must be higher for short sessions than for long sessions. A higher arrival rate implies that more feasible paths must be found. This leads to higher bandwidth overhead since more CPs are in the network, as it can be observed in Figure 3. Since less network capacity is available in this case than in the case of low arrival rate, the call blocking ratio is higher. Further support for this assertion is found in Figure 2(b). There it can be observed that, when the utilization factor exceeds 0.25, most call blocking is due to failure to satisfy the constraints of the QoS session, and not due to low-TTL blocking.

Two different graphs are used in Figure 3 to show the RDP bandwidth overhead. This is done because in the case of a single graph, the high bandwidth

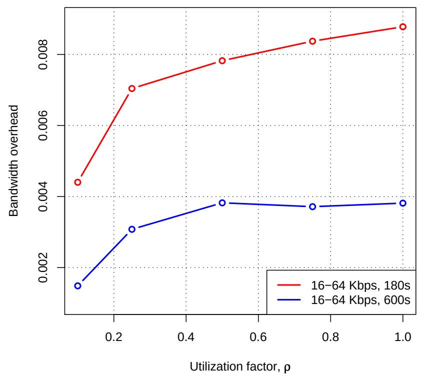


(a) Call blocking ratio.

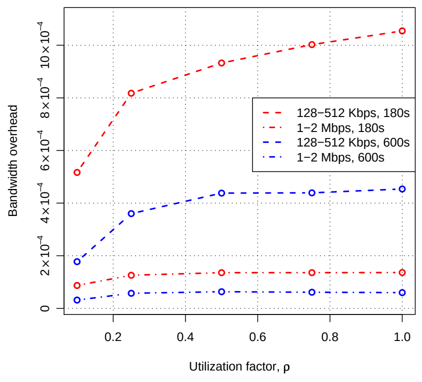


(b) Low-TTL blocking ratio.

Fig. 2. RDP call blocking.



(a) 16–64 Kbps.



(b) 128–512 Kbps and 1–2 Mbps.

Fig. 3. RDP bandwidth overhead.

overhead of 16-64 Kbps sessions would make it hard to distinguish the bandwidth overhead of the remaining sessions. Indeed, for 16-64 Kbps sessions the bandwidth overhead is 40–80 times higher than that of 1–2 Mbps sessions.

The results suggest that one needs to carefully consider the interaction between the TTL value and the bandwidth overhead. This is in fact a trade-off between success in finding feasible paths and efficiency in keeping the overhead low. In our simulations the worst-case cost in the case of RDP, namely 0.9 % bandwidth overhead, occurs for 16–64 Kbps flow demands with 180 seconds mean session duration when the utilization factor is one.

5.3 Route Maintenance Protocol

RMP combines path restoration and flow allocation to handle the type of resource fluctuations described in Section 3. In this paper the focus is on bandwidth constraints, but the protocol as such has support for different types of QoS constraints. RMP relies on two main components: an algorithm for distributing link-state information and an optimization algorithm for flow allocation.

Link-state information is distributed using the *link vector* algorithm proposed by Behrens and Garcia-Luna-Aceves [23]. The difference between a link-state algorithm and a link vector algorithm is that the link-state algorithm is require to broadcast complete topology information. When a link vector algorithm is used, a node uses selective diffusion to disseminate link-state information pertaining only to its preferred paths. This reduces the communication overhead associated with traditional link-state algorithms.

In the case of RMP, the preferred paths are setup using RDP. The QoS information provided by RDP ensures that a node has link-state information about each link in each of its preferred paths. The set of links belonging to a node's preferred paths is called the *source graph* of that node. Nodes exchange source graph information with their neighbors. Additionally, nodes have link-state information about their own outgoing links. The topology information known to a node consists of its own links, its own source graph and the source graphs reported by its neighbors [23].

Nodes report incremental source graph information to their neighbors. Obviously, when a node joins the overlay it receives complete source graphs from its neighbors. Beyond that, information is transmitted only if the link-state changes (*i. e.,* triggered updates). In other words, RMP is a reactive protocol.

A node enters *restoration mode* when a path is broken. A path is considered broken when one or several links are deleted from the source graph or when their updated state information makes it impossible to satisfy the path QoS constraints. In restoration mode the following actions are taken:

i) a broken path error message is sent to the source node of each affected flow,
ii) Yen's KSP algorithm [35] is executed to find the K backup paths to the destinations affected by the topology updates,
iii) the corresponding flow demands and the backup paths are used to construct a PAP-MLPF optimization problem as described in Section 4,

iv) the simplex method [42] is used to solve the PAP-MLPF problem,
v) if the simplex method is successful, the links on the new paths are added to the source graph; otherwise the affected flows are dropped (*i. e.*, packets belonging to them are not forwarded further).

The time complexity of the link vector algorithm after a single link change is $\mathrm{O}(n)$, where n is the number of nodes affected by the change. The upper bound for n is given by the length of the longest path in the network. The communication complexity $\mathrm{O}(E)$ is asymptotic in the number of links in the network [23].

5.4 RMP Simulation Results

The purpose of the experiments is to evaluate RMP's performance for different levels of churn. In the experiments we focus entirely on bandwidth reservations. A network topology with 100 nodes and 780 links is used for all experiments. The links in these experiments are error-free.

We denote by p_t the total number of preferred paths, by p_r the number of restored paths, and finally by p_f the number of path failures (*i. e.*, paths that could not be restored). The relation $0 \leq p_r + p_f \leq p_t$ always holds.

We focus on the following simulation metrics:

path failure ratio: ratio between the number of path failures and the total number of preferred paths in the network, p_f/p_t,
restored paths ratio: ratio between the number of restored paths and the number of broken paths, $p_r/(p_r + p_f)$ for $p_r + p_f > 0$[9],
bandwidth overhead: ratio between the average number of RDP bytes per second and network capacity (*i. e.*, the aggregated volume of every link in the network),

Additional metrics are available in [10].

We simulate 50 random flow demands (*i. e.*, $p_t = 50$). This value provides an acceptable trade-off between link utilization and the time required to run the simulations. The flow demand bandwidth is uniformly distributed over three different ranges: 16–64 Kbps, 128–512 Kbps and 1–2 Mbps, as in the case of RDP. The source and destination node of each flow demand is selected randomly.

Link bandwidth is interpreted in our simulations as residual capacity after bandwidth is reserved on preferred paths. The residual capacity determines the amount of path diversity within the network. Here, the residual capacity is exponentially distributed with mean value equal to the maximum bandwidth demand multiplied by an integer scaling factor. For example, for the bandwidth range 1–2 Mbps and a scaling factor of 2, the link bandwidth is exponentially distributed with mean value 4 Mbps. Intuitively, a scaling factor of 1 means that 63 % of the links have *less* capacity than the maximum value of the demand range. For a scaling factor of 5 only 18 % of the links have *less* bandwidth than the maximum value of the demand range.

[9] Instances where $p_r + p_f = 0$ are not used in computing the average.

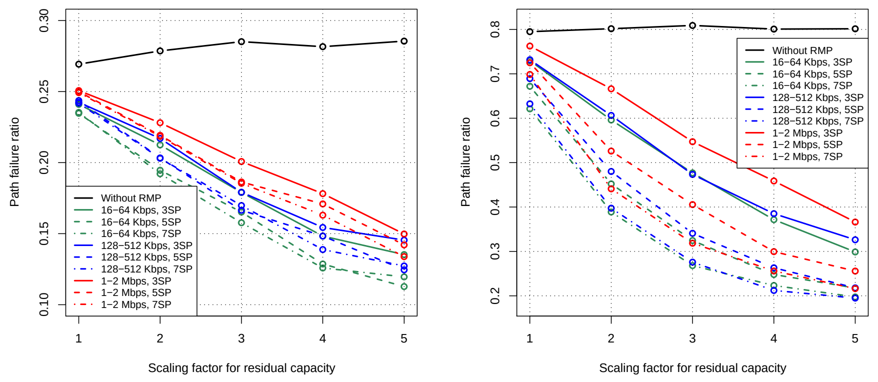
The following scaling factors are used: 1, 2, 3, 4, and 5. The use of exponential distribution with mean value based on the maximum bandwidth demand results in a good mix of links with very little bandwidth as well as links with lots of residual capacity. Using the upper bound of the bandwidth range is a matter of preference. In fact, any value within the bandwidth range can be selected and the mean link bandwidth is scaled accordingly. The residual network capacity increases proportionally with the integer multiple value.

We present simulation results for two different levels of churn: one based on the Gnutella session durations with mean duration of 130 seconds [10] and another based on exponential session durations with mean duration of 30 seconds. The two types of churn correspond roughly to the following scenarios:

Gnutella churn: general purpose peer-to-peer (P2P) overlay network,
Exp(30 s): wireless network with moving stations.

RMP is configured to use 3, 5, and 7 backup paths, respectively. They are abbreviated as 3SPs, 5SPs and 7SPs in the text and figures. A higher number of backup paths increases the chances for successful flow allocation in situations with low residual network capacity.

Figure 4 shows the performance of path restoration for each type of churn. The solid black line at the top of each sub-figure indicates the ratio of path failures to the total number of paths ($p_t = 50$) when no path restoration is in use. Lower path failure ratio values indicate higher RMP success in restoring paths. The path failure ratio is lower in the case of Gnutella churn because the session durations are longer on the average, which translates in fewer link failures per time unit.

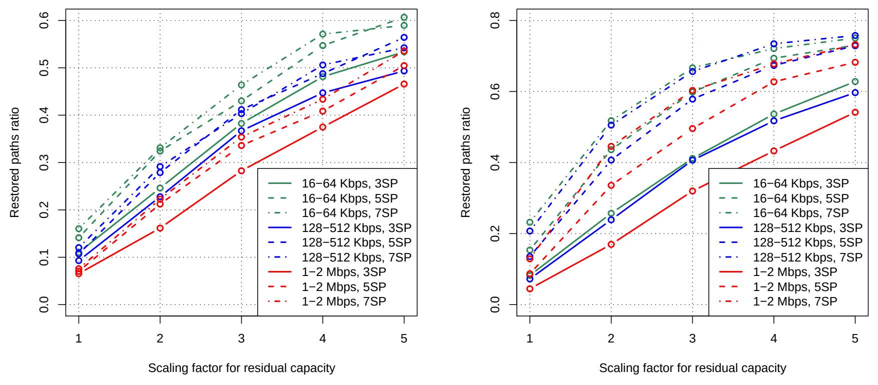


(a) Gnutella churn (b) Exp(30 s) churn

Fig. 4. RMP path restoration.

In both cases of churn, the path failure ratio decreases when RMP is used. Clearly, RMP's success is directly proportional to the amount of residual capacity. In terms of reduced path failure ratio, the largest gains are registered for Exp(30 s) scenarios. In this scenarios, the path failure ratio is very high in the absence of RMP, which means that there is a lot of room for improvement.

Using more backup paths (*i. e.*, 5SPs or 7SPs) shows most gain in the case of Exp(30 s) scenarios for bandwidth range 1–2 Mbps. With less aggressive churn, the usefulness of additional backup paths decreases.

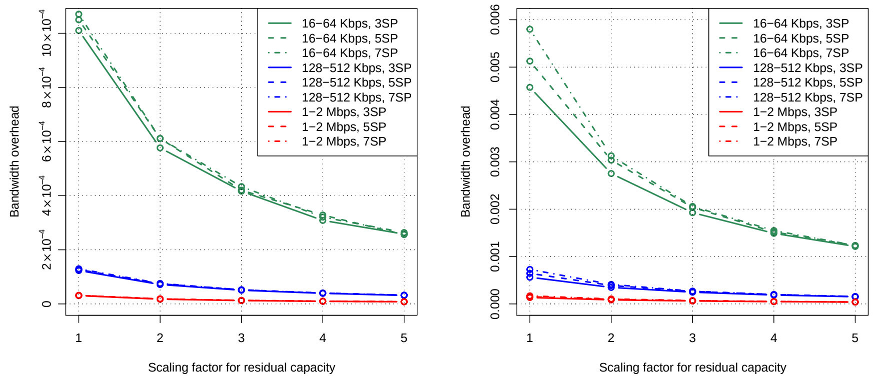


(a) Gnutella churn (b) Exp(30 s) churn

Fig. 5. RMP restored paths ratio.

A complementary view of RMP's path restoration success is shown in Figure 5 in terms of the ratio of number of restored paths to the number of broken paths. RMP's largest success in restoring broken paths is experienced in the case of 16-64 Kbps scenarios. In Figure 5(b) for scaling factor 4 and 5, the restored paths ratio for 128-512 Kbps scenarios with 7SPs is slightly higher than the restored path ratio for 16-64 Kbps scenarios with 7SPs. These minimal differences are due to the random selection of source and destination node for each flow demand.

The bandwidth overhead shown in Figure 6 is computed by dividing the bandwidth utilization with the network capacity (*i. e.*, the sum of residual capacity and used capacity). In contrast to RDP's bandwidth overhead, the RMP bandwidth overhead is biased. For example, the 16–64 Kbps scenarios dominate in each of the churn scenarios. This happens because the amount of average residual capacity per link is determined by the product between the scaling factor in use and the upper bound of the bandwidth range. Hence, the network capacity for 16–64 Kbps scenarios is always lower than for the other two bandwidth ranges. Comparison of bandwidth overhead is fair only for graphs within the same bandwidth range.



(a) Gnutella churn (b) Exp(30 s) churn

Fig. 6. RMP bandwidth overhead.

For each bandwidth range, the largest bandwidth overhead occurs in 7SPs scenarios with Exp(30 s) churn when the scaling factor is equal to one: 0.6 % in the case of 16–64 Kbps scenarios, 0.07 % in the case of 128–512 Kbps scenarios, and 0.02 % in the case of 1–2 Mbps scenarios.

6 Summary

This paper has introduced fundamental concepts related to QoS routing in overlay networks. Furthermore, the paper has reported on performance results for RDP and RMP, which are two protocols under the ORP framework. The focus of the performance results has been on the cost in the form of bandwidth overhead incurred from running these protocols.

The worst-case cost in the case of RDP, namely 0.9 % bandwidth overhead, occurs for 16–64 Kbps flow demands with 180 seconds mean session duration when the utilization factor is one. For RMP, the worst-case cost, 0.6 %, occurs in Exp(30 s) scenarios with 16-64 Kbps flow demands, 7SPs and scaling factor one for residual capacity. Assuming an additive cost when RDP and RMP are being used together, the worst-case cost is estimated to be as high as 1.5 % protocol overhead in terms of bandwidth.

RMP can be quite efficient in restoring broken paths provided enough residual capacity is available. For example in Exp(30 s) scenarios, in spite of aggressive churn, RMP is able to restore up to 40 % of broken paths used for transporting 1–2 Mbps flows, with approximately 0.02 % bandwidth overhead.

We plan to run similar tests in a more realistic environment, such as PlanetLab. However, a realistic PlanetLab implementation requires that ORP is extended to include two important elements: an overlay network that organizes nodes and transports ORP messages and the ability to measure link-state variables (*e. g.*, available bandwidth, delay and loss).

All experiments presented here focus on a single QoS metric: bandwidth. Additional experiments should be performed to evaluate ORP's performance when several QoS metrics are combined. For RMP, this requires that Yen's KSP algorithm is replaced by the Self-Adaptive Multiple Constraints Routing Algorithm (SAMCRA) [32] or another similar algorithm.

References

1. E. S. Crawley, R. Nair, B. Rajagopalan, and H. Sandick, *RFC 2386: A Framework for QoS-based Routing in the Internet*, IETF, Aug. 1998, category: Informational. [Online]. Available: http://www.ietf.org/ietf/rfc2386.txt
2. R. Braden, D. D. Clark, and S. Shenker, *RFC 1633: Integrated Services in the Internet Architecture: an Overview*, IETF, Jun. 1994, category: Informational. [Online]. Available: http://www.ietf.org/ietf/rfc1633.txt
3. J. Wroclawski, *RFC 2210: The Use of RSVP with IETF Integrated Services*, IETF, Sep. 1997, category: Standards Track. [Online]. Available: http://www.ietf.org/ietf/rfc2210.txt
4. K. Thompson, G. J. Miller, and R. Wilder, "Wide-area Internet traffic patterns and characteristics," *IEEE Network*, vol. 11, no. 6, pp. 10–23, Nov. 1997.
5. S. Blake, D. L. Black, M. A. Carlson, E. Davies, Z. Wang, and W. Weiss, *RFC 2475: An Architecture for Differentiated Services*, Dec. 1998. [Online]. Available: http://www.ietf.org/ietf/rfc2475.txt
6. C. Bouras and A. Sevasti, "Service level agreements for DiffServ-based services' provisioning," *Journal of Computer Networks*, vol. 28, no. 4, pp. 285–302, Nov. 2005.
7. Z. Wang, *Internet QoS: Architectures and Mechanisms for Quality of Service*. San Francisco, CA, USA: Morgan Kaufman Publishers, 2000, ISBN: 1-55860-608-4.
8. L. Burgsthaler, K. Dolzer, C. Hauser, J. Jähnert, S. Junghans, C. Macián, and W. Payer, "Beyond technology: The missing pieces for QoS success," in *Proceedings ot the ACM SIGCOMM Workshops*, Karlsruhe, Germany, Aug. 2003, pp. 121–130.
9. Y. Rekhter, T. Li, and S. Hares, *RFC 4271: A Border Gateway Protocol 4 (BGP-4)*, IETF, Jan. 2006. [Online]. Available: http://www.ietf.org/ietf/rfc4271.txt
10. D. Ilie, "On unicast QoS routing in overlay networks," Ph.D. dissertation, Blekinge Institute of Technology (BTH), Karlskrona, Sweden, Oct. 2008.
11. D. G. Andersen, "Resilient overlay networks," Master's thesis, Dept. of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, May 2001.
12. L. Subramanian, I. Stoica, H. Balakrishnan, and R. Katz, "OverQoS: An overlay based architecture for enhancing Internet QoS," in *Proceedings of NSDI*, San Francisco, CA, USA, Mar. 2004.
13. Z. Li and P. Mohapatra, "QRON: QoS-aware routing in overlay networks," *IEEE Journal on Selected Areas in Communications*, vol. 22, no. 1, pp. 29–40, Jan. 2004.
14. L. Lao, S. S. Gokhale, and J.-H. Cui, "Distributed QoS routing for backbone overlay networks," in *Proceedings of IFIP Networking*, Coimbra, Portugal, May 2006.
15. G. Apostolopoulos, D. Williams, S. Kamat, R. Guerin, A. Orda, and T. Przygienda, *RFC 2676: QoS Routing Mechanisms and OSPF Extensions*, IETF, Aug. 1999, category: Experimental. [Online]. Available: http://www.ietf.org/rfc2676.txt

16. S. Chen, "Routing support for providing guaranteed end-to-end quality-of-service," Ph.D. dissertation, Engineering College of the University of Illinois, Urbana, IL, USA, 1999.
17. D. H. Lorenz, "QoS routing and partitioning in networks with per-link performance-dependent costs," Ph.D. dissertation, Israel Institute of Technology, Haifa, Israel, 2004.
18. D. H. Lorenz and A. Orda, "QoS routing in networks with uncertain parameters," *IEEE/ACM Transactions on Networking*, vol. 6, no. 6, pp. 768–778, Dec. 1998.
19. S. Chen and K. Nahrstedt, "Distributed QoS routing with imprecise state information," in *Proceedings of ICCCN*, Lafayette, LA, USA, Oct. 1998.
20. ——, "An overview of quality of service routing for the next generation high-speed networks: Problems and solutions," *IEEE Network*, vol. 12, no. 6, pp. 64–79, Nov. 1998.
21. ——, "Distributed quality-of-service routing in high-speed networks based on selective probing," in *Proceedings of LCN*, Lowell, MA, USA, Oct. 1998, pp. 80–89.
22. E. Gelenbe, R. Lent, and A. Nunez, "Self-aware networks and QoS," in *Proceedings of the IEEE*, vol. 92, Sep. 2004, pp. 1478–1489.
23. J. Behrens and J. J. Garcia-Luna-Aceves, "Distributed, scalable routing based on link-state vectors," in *Proceedings of SIGCOMM*, London, UK, Aug. 1994, pp. 136–147.
24. W. C. Lee, "Topology aggregation for hierarchical routing in ATM networks," *ACM SIGCOMM Computer Communications Review*, vol. 25, no. 2, pp. 82–92, Apr. 1995.
25. K.-S. Lui, K. Nahrstedt, and S. Chen, "Routing with topology aggregation in delay-bandwith sensitive networks," *IEEE/ACM Transactions on Networking*, vol. 12, no. 1, pp. 17–29, Feb. 2004.
26. J. Schiller, *Mobile Communications*, 2nd ed. Boston, MA, USA: Addison Wesley, 2003, ISBN: 0-321-12381-6.
27. J. He and J. Rexford, "Towards internet-wide multipath routing," *IEEE Network*, vol. 22, no. 2, pp. 16–21, Apr. 2008.
28. K. P. Gummadi, S. Saroiu, and S. D. Gribble, "King: Estimating latency between arbitrary internet end hosts," in *Proceedings of IMW*, Marseille, France, Nov. 2002.
29. R. Prasad, C. Dovrolis, M. Murray, and K. C. Claffy, "Bandwidth estimation: Metrics, measurement techniques, and tools," *IEEE Network*, vol. 17, no. 6, pp. 27–35, Nov. 2003.
30. Y. Liu, H. Zhang, W. Gong, and D. Towsley, "On the interaction between overlay routing and traffic engineering," in *Proceedings of IEEE Infocom*, Miami, FL, USA, Mar. 2005.
31. L. Qiu, R. Yang, and S. Shenker, "On selfish routing in internet-like environments," *IEEE/ACM Transactions on Networking*, vol. 14, no. 4, pp. 725–738, Aug. 2006.
32. P. Van Mieghem and F. A. Kuipers, "Concepts of exact QoS routing algorithms," *IEEE/ACM Transactions on Networking*, vol. 12, no. 5, pp. 851–864, Oct. 2004.
33. Z. Wang and J. Crowfort, "Quality-of-service routing for supporting multimedia applications," *IEEE Journal on Selected Areas in Communications*, vol. 14, no. 7, pp. 1228–1234, Sep. 1996.
34. F. A. Kuipers and P. F. A. Van Mieghem, "Conditions that impact the complexity of QoS routing," *IEEE/ACM Transactions on Networking*, vol. 13, no. 4, pp. 717–730, Aug. 2005.
35. J. Y. Yen, "Finding the k shortest loopless paths in a network," *Management Science*, vol. 17, no. 11, pp. 712–716, Jul. 1971.

36. M. Pióro and D. Medhi, *Routing, Flow, and Capacity Design in Communication and Computer Networks.* San Francisco, CA, USA: Morgan Kaufman Publishers, 2004, ISBN: 0-12-557189-5.
37. K. De Vogeleer, D. Ilie, and A. Popescu, "Constrained-path discovery by selective diffusion," in *Proceedings of HET-NETs*, Karlskrona, Sweden, Feb. 2008.
38. E. Gelenbe, R. Lent, A. Montuori, and Z. Xu, "Cognitive packet networks: QoS and performance," in *Proceedings of IEEE MASCOTS*, Ft. Worth, TX, USA, Oct. 2002, pp. 3–12.
39. J. J. Garcia-Luna-Aceves, "Loop-free routing using diffusing computations," *IEEE/ACM Transactions on Networking*, vol. 1, no. 1, pp. 130–141, Feb. 1993.
40. A. A. Shaikh, "Efficient dynamic routing in wide-area networks," Ph.D. dissertation, University of Michigan, Ann Arbor, MI, USA, 1999.
41. A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, pp. 509–512, Oct. 1999.
42. D. G. Luenberger, *Linear and Nonlinear Programming.* Norwell, MA, USA: Kluwer Academic Publishers, 2004, ISBN: 1-4020-7593-6.