Constrained-Path Discovery by Selective Diffusion

Karel de Vogeleer
Dept. of Telecommunication Systems
School of Engineering
Blekinge Institute of Technology
Karlskrona, Sweden
E-mail: karel.de.vogeleer@bth.se

Dragos Ilie
Dept. of Telecommunication Systems
School of Engineering
Blekinge Institute of Technology
Karlskrona, Sweden
E-mail: dragos.ilie@bth.se

Adrian Popescu
Dept. of Telecommunication Systems
School of Engineering
Blekinge Institute of Technology
Karlskrona, Sweden
E-mail: adrian.popescu@bth.se

August 6, 2008

Abstract

The demand for live and interactive multimedia services over the Internet raises questions on how well the Internet Protocol (IP) best-effort effort service can be adapted to provide adequate end-to-end quality of service (QoS) for the users. Although the Internet community has developed two different IP-based QoS architectures, neither has been widely deployed. Overlay networks are seen as a step to address the demand for end-to-end QoS until a better solution can be obtained.

As part of the telecommunication research at Blekinge Institute of Technology (BTH) in Karlskrona, Sweden, we are investigating new theories and algorithms concerning QoS routing. We are in the process of developing Overlay Routing Protocol (ORP), a framework for overlay QoS routing consisting of two protocols: Route Discovery Protocol (RDP) and

Route Management Protocol (RMP). In this paper we describe RDP and provide preliminary simulation results for it. The results indicate that the RDP scales almost linearly with the number of nodes in the network. The system's ability to find feasible paths is intimately related to the time-to-live (TTL) value used in RDP messages: a large TTL value increases the chance of finding a feasible path at the cost of a higher volume of RDP traffic.

1 Introduction

The predominant form of Internet routing is a combination of shortest-path routing for intradomain environments coupled with policy-based routing for interdomain communication. For the past ten years it has been argued that Internet must incorporate elements of QoS in order to be used as a platform for multimedia distribution. In particular, live or interactive multimedia communications place stringent constraints on the path between sender and receiver. Examples of such constraints are constraints on available bandwidth, packet delay, packet delay variation and packet loss rate.

Two major IP-based QoS architectures have been developed so far: Integrated Services (IntServ) [7] and Differentiated Services (DiffServ) [6]. Neither architecture has been widely deployed due to lack of a viable economical solution for network operators, poor backwards compatibility with existing technology and difficulties in the interaction between different network operators [2, 5, 8]. Additionally, we would like to mention the Asynchronous Transfer Mode (ATM) Private Network-to-Network Interface (PNNI) protocol, which has support for QoS routing [14, 18]. Although ATM failed to become the technology of choice for end-nodes due to the emergence of cheap Ethernet cards, the research into ATM technology has yielded valuable results for the Internet community.

There seems to be little hope for wide QoS deployment implemented at network layer, at least in the near future. To cope with this problem several researchers have investigated the possibility to deploy QoS in overlay networks on top of IP [1, 9, 11, 24, 26].

An overlay network utilizes the services of an existing network in an attempt to implement new or better services. An example of an overlay network is shown in Figure 1. The physical interconnections of three autonomous systems (ASs) are depicted at the bottom of the figure. The grey circles denote nodes that use the physical interconnections to construct virtual paths used by the overlay network at the top of the figure. Nodes participating in the overlay network perform active measurements of particular QoS metrics associated with the virtual paths. The results from the measurements can be used in rerouting of overlay traffic, in load balancing or for traffic shaping.

The work presented in this paper is part of the Routing in Overlay Networks (ROVER) project at BTH to implement a framework for overlay QoS routing called ORP [22]. ORP is part of a larger goal to research and develop a QoS layer on top of the transport layer. The main idea is to combine ORP together

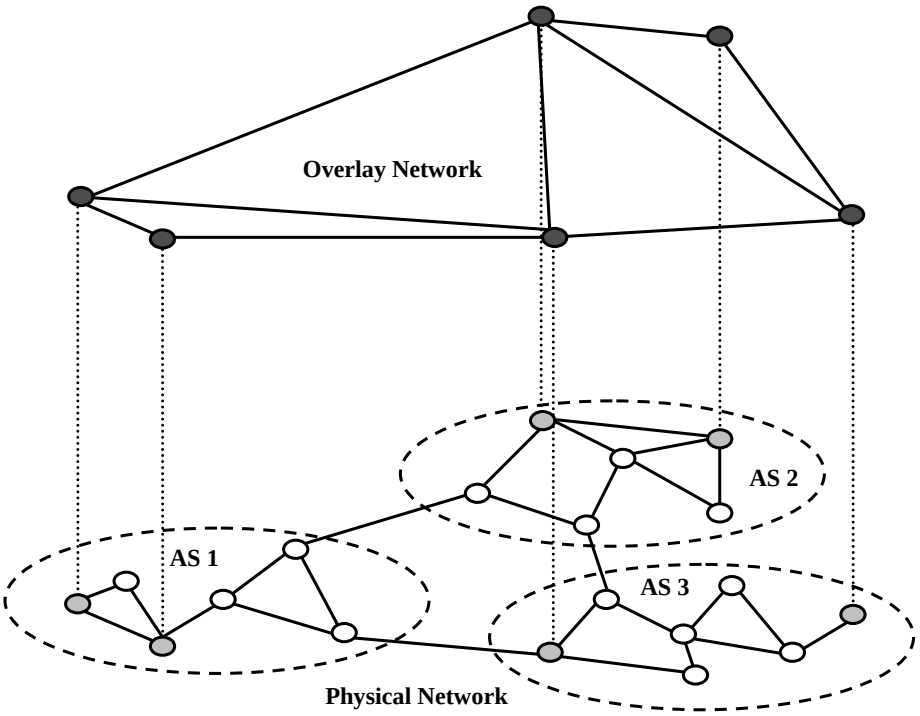

Figure 1: Overlay network

with additional QoS mechanisms, such as resource allocation and admission control, into a QoS layer. User applications that use the QoS layer can obtain soft QoS guarantees. These applications run on end-hosts without any specific privileges such as the ability to control the internals of TCP/IP stack, the operating system, or other applications that do not use the QoS layer. In terms of the OSI protocol stack, the QoS layer is a sub-layer of the application layer. Applications may choose to use it or to bypass it.

The QoS layer implements *per-flow* QoS resource management. In contrast to IntServ and DiffServ, we envision that it is mostly end-nodes in access networks that take part in the routing protocol. IP routers are not required to take part or be aware of the QoS routing protocol running on the end-nodes. In other words, we propose a QoS layer on top of the best-effort service provided by IP. Since a best-effort service leaves room for uncertainties regarding the resource allocation, we aim only for soft QoS guarantees.

The ORP framework consists of two protocols: Route Discovery Protocol (RDP) and Route Management Protocol (RMP).

RDP is used to find network paths subject to various QoS constraints [12,19]. To achieve this goal, RDP uses a form of selective diffusion in which a node that receives a path request forwards the request only on outgoing links that do not violate the QoS constraints. Eventually, the request may reach the destination

node if there is at least one path satisfying the constraints. At that point a reply message containing information about the complete path is sent back to the requesting node. The RDP is based on ideas presented in [10, 16, 17].

The purpose of the RMP is to alleviate changes in the path QoS metrics, due to node and traffic dynamics. This is done through a combination of route repair techniques 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 [20]. The design of RMP is influenced by ideas presented in [4, 15]. Since RMP is currently under development [21], only information pertaining to RDP is presented in this article.

The remaining of this paper is organized as follows. Section 2 introduces the format of the messages used by RDP. The diffusion process used for constrained-path discovery is presented in Section 3. The OMNeT++ simulation model for RDP is presented in Section 4. The simulation testbed is presented in Section 5. In Section 6 we analyze several simulation results from a performance perspective. This is followed by plans for future work in Section 7.

2 RDP Message Format

Since ORP messages can carry user data we switch freely between the terms *packet* and *message*. We assume that the transport layer below ORP allows a node to send packets to any known peer in the overlay.

All ORP messages start with the generic header shown in Figure 2. Field

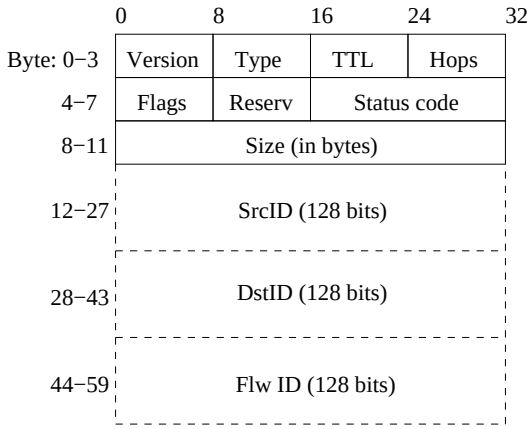

Figure 2: ORP generic packet header

values in the packet header are arranged in network byte order. The following elements are included in the ORP packet header:

Version ORP protocol version. At the moment of writing the protocol is at version 1.

Type ORP packet type.

Field value	Packet type
0	reserved
1	control packet (CP)
2	acknowledgement packet (AP)
3	data packet (DP)
4	used by the RMP

TTL Time-to-live, denoting how many overlay hops the packet is allowed to travel.

Hops Indicates the amount of links the packet already has passed. If the value in the Hops field equals the value in the TTL field the packet is dropped.

Flags Bitfield arranged as |0|0|E|0|D|C|B|R|, where 0 denotes unused bits.

- E indicates the node is leaving the overlay and all routes associated with *SrcID* should be rerouted or deleted.
- D indicates that the path associated with the *FlwID* should be deleted.
- C denotes a route change.
- B denotes a bidirectional route request.
- R indicates a redundant AP.

Reserv Reserved for future use.

Status Used to exchange status codes among nodes.

Size Packet size in bytes excluding the generic header.

SrcID Universally Unique Identifiers (UUID) denoting the source node of the packet[1].

DstID UUID denoting the destination node of the packet.

FlwID UUID of the flow to which this packet belongs.

RDP uses two different kinds of packets: control packets (CPs) and acknowledgement packets (APs).

A CP begins with the generic header followed by a data structure called *QoS map*, as shown in Figure 3. The QoS map starts with the *QoS constraints* for the requested path. ORP currently supports two type of QoS constraints: *minimum bandwidth* specified in kilobytes per second and *maximum path delay*, specified in milliseconds. We plan to integrate additional constraint types in

[1]The ORP UUIDs are defined as specified in [23].

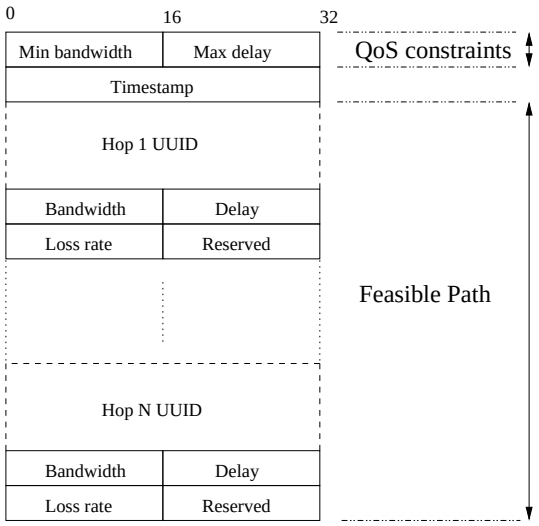

Figure 3: QoS map

future ORP versions. The *timestamp*, in Coordinated Universal Time (UTC) format, indicates the time when the QoS map was sent to the next hop.

Following the path QoS constraints comes the *feasible path* explored so far by the CP in question. Each node that forwards the CP appends an entry to the feasible path. The entry consists of the UUID of the downstream node and a set of QoS metrics associated with the link on which the packet is forwarded. Statistics currently supported by ORP are *bandwidth* (expressed in kilobytes per second), *delay* (expressed in milliseconds) and *packet loss rate*. The packet loss rate is a fraction with the accuracy $1/(2^{16}-1)$. A loss rate of 0 indicates that no packets are lost whereas a loss rate of $(2^{16}-1)$ denotes that all packets are lost. The use of the last field is not defined yet. The manner in which the the QoS metrics are computed is not within the scope of this paper.

When the destination node receives a CP it assembles an AP by copying from the CP the fields SrcID, DstID, FlwID and the feasible path. Then, the AP is sent back to the source node over the reverse feasible path[2]. The purpose of APs is to inform nodes on the feasible path that a complete route to the destination has been found.

When a route has been established between two nodes, the source node can start sending the data. Data is transported inside data packets (DPs). A DP consists of a generic header and the QoS map obtained from an AP, followed by application data. DPs using the same FlwID are said to form a *flow*.

Each node maintains a number of flow relays (FRs). A FR is an abstract data type associated with a single flow or a group of flows (*flow bundle*) sharing

[2]Traveling on the reverse feasible path means traveling in the opposite direction on the feasible path (*i. e.*, over hops $N, (N-1), \ldots, 1$).

common characteristics, *e. g.*, the same QoS constraints. The information in the FRs is updated by CPs and APs associated with the flows and by QoS measurements performed by the node in question.

At each node a list of active CPs is maintained. A CP is active from the time it is forwarded towards the destination until a corresponding AP is received or a timeout occurs. Each list entry contains information uniquely identifying a CP: a copy of the SrcID, the DstID and the FlwID. Further, a timer T_{out} is associated with every CP in the list. When the timer expires the corresponding CP is removed from the list.

3 Path Discovery

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 requesting node, also called *source node*, sends the CP 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 list of active CPs and a timer is started accordingly. If after T_{out} seconds no information is received the CP is considered lost and it is removed from the active CP list.

We compute the value of T_{out} by the following formula:

$$T_{out} = 0.2 \times (TTL - Hops).$$

Initially, $(TTL - Hops)$ was multiplied by 2 instead of 0.2 in order to obtain a conservative estimate of the round-trip time. However, it was observed that the T_{out} values were excessively large, keeping links blocked for unnecessary long durations of time. Based on empirical evidence, it was decided to scale down the T_{out} values by a factor of 1/10.

Each node receiving a CP checks whether its node UUID is matching an entry in the feasible path of the CP or not. A matching entry means that the CP has entered a loop and causes the CP to be disregarded. In this case however, the CP entry remains in the active CP list.

If no matching node UUID entry is found in the feasible path of the CP and at least one feasible link exists, then the received CP is added to the list of active CPs. For each feasible link found, the adjacent node UUID (denoted by Hop UUID in Figure 3) and the QoS statistics of the link are appended to a copy of the received CP. The modified CP is then forwarded over the link in question. This process is performed for each link, except for the one on which the packet arrived at the current node.

If no feasible link exists, the CP is dropped and no further actions are taken. The receiving and forwarding process is repeated at several nodes until one or more CPs reach the *destination node* or, alternatively, all CPs are dropped by intermediate nodes.

If all CPs are lost the nodes on the feasible path will eventually experience T_{out} timeouts and will thus be able to free any reserved resources.

The first CP that arrives to the destination node is used to obtain the feasible path between source and destination. The destination node will then create a FR for packets corresponding to the FlwID in the CP. The destination node sends an AP back to the source node over the reverse feasible path. If the received CP indicates that the source node wishes bidirectional communication, then the destination node begins immediately a route discovery process towards the source node, using the same QoS constraints specified in the CP.

All subsequent CPs that arrive to the destination node are used to construct corresponding APs. These APs are marked as redundant and then forwarded to the source on the reverse feasible path.

Each node receiving a AP checks whether the triple (SrcId, DstId, FlwID) is matching an entry in the list of active CPs or not. If a matching entry is found, the node either creates a FR or adds the flow to an existing flow bundle corresponding to a FR. Further, the CP entry is removed from the active CP list and the AP is forwarded to the next node on the reverse feasible path. If no matching entry is found, the AP is dropped silently. The manner in which the redundant APs are treated depends on the overlay policies. If the overlay policies favor backup paths or multipath routing, the redundant APs are treated just as regular APs. Otherwise, redundant APs are dropped.

The first AP to arrive at the source node signals that a feasible path has been set up and the application can begin sending DPs. A feasible path can be torn down by a CP with the delete (D) flag set.

4 Implementation

To evaluate the performance of RDP we use the public-source simulation environment *OMNeT++* [27]. OMNeT++ is an object-oriented, modular and open-architecture discrete event simulation environment with an embeddable simulation kernel.

An OMNeT++ simulation is build out of hierarchically nested modules, which is ideal for an object-oriented approach. Modules communicate with each other by means of messages and these messages may contain data of arbitrary length. Messages are transported through gates and over channels. A node maintains an arbitrary amount of gates and different gates are connected with channels. The topology of a network, in terms of gates, channels and modules, is defined in the Network Description (NED) language [27].

Our simulator includes two different modules: the `ORP` module and the `DATACENTER` module. The `ORP` module implements the RDP protocol and the `DATACENTER` module collects the simulation statistics. These statistics can easily be written to files with help of dedicated classes provided by the OMNeT++ framework.

As RDP is designed to run on top of the Internet we have attempted to use realistic Internet topologies in our simulations. There are several challenges in modeling the Internet topology, such as mapping the actual topology, characterising it, and developing models that capture fundamental properties [25].

Several topology generators are available [25, 28, 29], but the generated topologies differ significantly according to the characteristics of the network models used.

We have used the BRITE [25] software to generate network models according to the Barabási-Albert model. BRITE is a universal topology generator developed at Boston University. It is designed to be a flexible, extensible, interoperable, portable and user friendly topology generator. We have chosen BRITE because:

i) it has supports for realistic topology models based on power-law distributions,

ii) it can generate router level topologies,

iii) it is supported under OMNeT++, and

iv) the source code is freely available.

OMNeT++ allows arbitrary parameters to be defined in an external initialisation file that can be loaded in the simulation at any time. This allows the user to control the behaviour of the simulation without having to recompile the source code. The parameters available in our initialisation file are:

- TTL value of the packets,
- destination node to which a route will be opened,
- delay and bandwidth QoS constraints used for route requests,
- session arrival rate and session duration.

The destination node parameter can be a node identifier or a discrete probability distribution used to randomly select a node.

The RDP simulator currently supports Poisson arrivals and exponentially distributed session durations. Thus, the session arrival rate parameter describes the mean value λ of the Poisson distribution and the session duration parameter denotes the mean value $E[X] = 1/\lambda$ of the exponential distribution. More sophisticated distributions are planned to be used for future work.

In our simulations each node in the network attempts to establish a route at a time instant described by the arrival rate process. If RDP establishes a route, then that specific route will last for the duration of the session. The simulations are terminated when all established sessions end.

5 Simulation testbed

We used the Barabási-Albert model for generating the network topologies in our simulations. This model is based on the idea that self-organization in large networks leads to a state described by a scale-free power-law distribution [3].

Table 1: Parameter settings for the experiment

Parameter	Value
TTL	7
Receiver	intuniform(0, number of nodes)
Delay	1000 ms
Bandwidth	intuniform(64, Y)
Session arrival rate	10
Session duration	15

Power-laws are expressions of the form of

$$y \propto x^a$$

where $\propto$ means "proportional to", a is a constant and x and y are arbitrary measures. Besides characterising the Internet, power-laws also appear to describe natural networks such as human respiratory systems and automobile networks [13].

The scale-free distribution in the Barabási-Albert model can be explained by two mechanisms: *incremental growth*, which refers to the gradual increase in size of the network, and *preferential connectivity* referring to the tendency of new nodes joining a network to connect to nodes that are highly connected or popular.

The Barabási-Albert router model in BRITE interconnects the nodes following the incremental growth idea. The probability P that a node i wants to connect to another node j in the network is given by

$$P(i,j) = \frac{d_j}{\sum_{k \in V} d_k}$$

where d_j denotes the outdegree of node j, V is the set of nodes that joined the network and $\sum_{k \in V} d_k$ denotes the sum of outdegrees of all nodes that previously joined the network [25]. When we talk about the outdegree we refer to the amount of edges incident to a node.

The parameter settings for our simulations are provided in Table 1. The *intuniform* in the value field denotes a discrete uniform distribution.

The difference between each simulation run is determined by the amount of nodes and by the size of the interval from which the bandwidth value is chosen, *i. e.*, by the variable Y in Table 1. Initially, there are 50 simulated nodes and this number is incremented each simulation run by 50, until the number of nodes reaches 950. The value of the bandwidth constraint assigned to a route request is drawn from a discrete uniform distribution. We use three intervals for the uniform distribution: 64–1024 KB/s, 64–2048 KB/s and 64–5120 KB/s, respectively. These intervals were selected in order to study the effect of bandwidth reservation on the scalability of the algorithm. In each case,

the upper bound of the interval corresponds to the Y variable in Table 1. This results in three curves on each graph, each curve having a total of 19 simulation points. Each simulation point is simulated 10 times and the results are used to compute the average. Furthermore, each node runs in "idle"-mode, which means that if a source node does not receive an AP in time it will make no further attempt to try to open a new connection.

The TTL value was selected empirically to obtain a reasonable trade-off between success in finding a feasible path and the amount of flooding that occurs during the process.

We have instructed BRITE to generate flat "ROUTER (IP) ONLY" topologies with nodes randomly placed on a plane of size 1000×1000 points. We have increased the number of nodes by 50 for each generated topology, starting from a size of 50 to 950 nodes.

The router Barabási-Albert model does not handle delays, but BRITE still assigns propagation delay mapped to the distance between nodes in the plane. The delay constraint is set to the opportunistic value of 1 second. This exceeds by far the link delays assigned by BRITE, which are in the range of milliseconds. As a consequence, the QoS delay constraint will always be satisfied. Furthermore, we have assumed in our simulations loss-free links. Therefore our current experiments with the Barabási-Albert router model analyse only the bandwidth performance of the RDP.

We generate session arrival rates following the Poisson distribution with $\lambda = 10$ and session duration times following the exponential distribution with expected value $E[X] = 15$. These values were selected in order to generate a fair amount of session churn. Each time a node is scheduled to open a session, as decided by the session arrival distribution, it select the destination node (*i. e.*, the receiver) from a discrete uniform distribution

6 Performance Analysis

We evaluate the RDP performance in terms of protocol overhead, which is a function of the number of nodes in the overlay, and other parameters. We use the following metrics to determine the protocol overhead:

- *route establishment ratio* (%), computed as

$$\frac{\text{total number of established routes}}{\text{total number of route requests}}$$

- *average bandwidth utilization* (B/s), obtained by

$$\frac{\text{total number of bytes sent}}{T}$$

- *link load ratio*, defined as

$$\frac{1}{i} \sum_{j=1}^{i} \frac{n_j}{T \times b_j}$$

where i denotes the total number of links, n_j denotes the number of bytes sent over link j, T is the time duration of the simulation and b_j represents the bandwidth of link j.

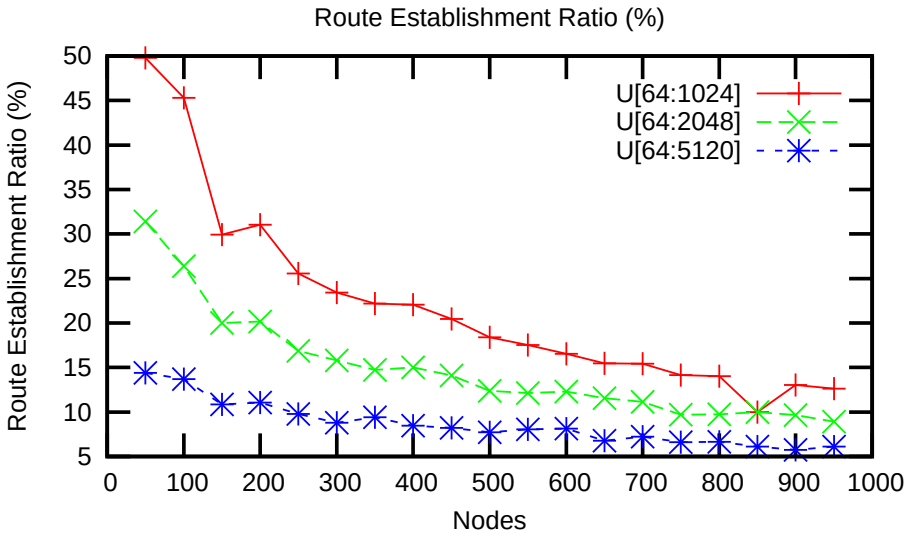

Figure 4: Route establishment ratio (%)

It is observed that the route establishment ratio, shown in Figure 4, has an overall decreasing slope. This indicates that finding feasible paths becomes more difficult with an increasing number of nodes. There are several reasons for why a route cannot be established:

i) no feasible path exists between source and destination,

ii) insufficient free bandwidth on feasible links due to previous RDP requests,

iii) too small TTL value in the CP (henceforth referred to as the *TTL problem*).

In the case when there is no feasible path between source and destination, there is not much that can be done. This case occurs when there is no path connecting the source node to the destination node or when there is no feasible path satisfying the combination of QoS constraints.

The second case for failing to establish a route is when previous RDP requests on a feasible link have allocated so much bandwidth that the amount of remaining free bandwidth is too small to satisfy the current constraint. In this case the application that uses RDP can use a lower bandwidth constraint than this, which may allow it to establish a feasible path.

The TTL problem is the particular reason for the decay of the curves in Figure 4. This problem can be solved by increasing the TTL value. However, an increase of the TTL may also increase the timeout T_{out}, which will result in longer durations for bandwidth reservation as well as additional flooding of the network by CPs.

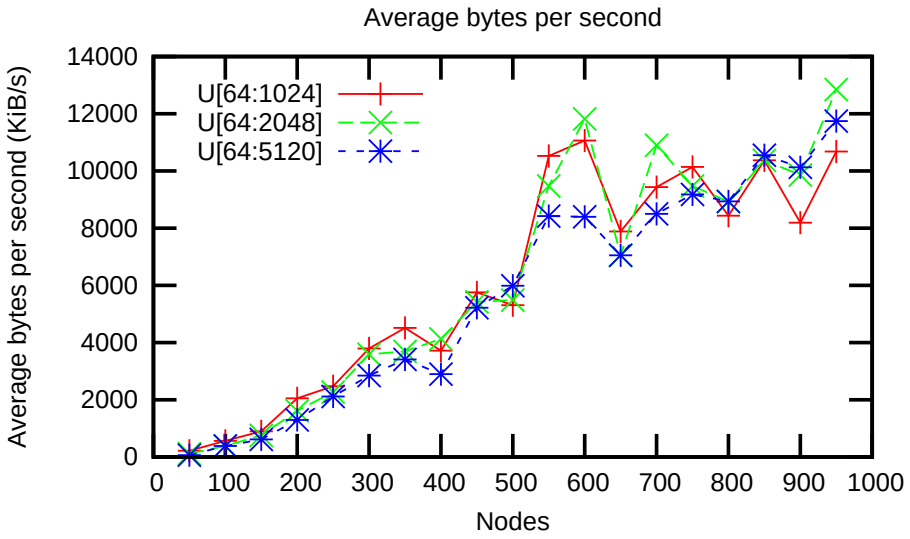

Figure 5: Average bandwidth utilization (KB/s)

Figure 5 shows the number of average number of bytes sent per second due to RDP messages. It can be observed that the curves follow roughly a linear growth. This is an indication that from the perspective of this metric the overlay network can scale to a large number of nodes. The fluctuations in the tail of the curve (*i. e.*, in the region of large number of nodes) can be explained by differences in the topology and simulation time between each simulation run.

The average link load ratio curves shown in Figure 6 appear to level out when the number of nodes increases. This is a manifestation of the TTL-problem. When the network radius grows source nodes will not be able to reach all destinations in the network due to small TTL value in the CPs. This means that CPs will traverse only a fraction of the links available in the network. Since the average link load ratio is computed over all links in the network, this value will decrease while the network radius increases, provided that the TTL is kept the same.

Furthermore, during the RDP simulations we observed that if, for a given request there is a high number of feasible links, this will create a large number of CPs that are duplicated and forwarded. Often, different CPs from the same flow arrive at the same intermediate node by travelling over different routes.

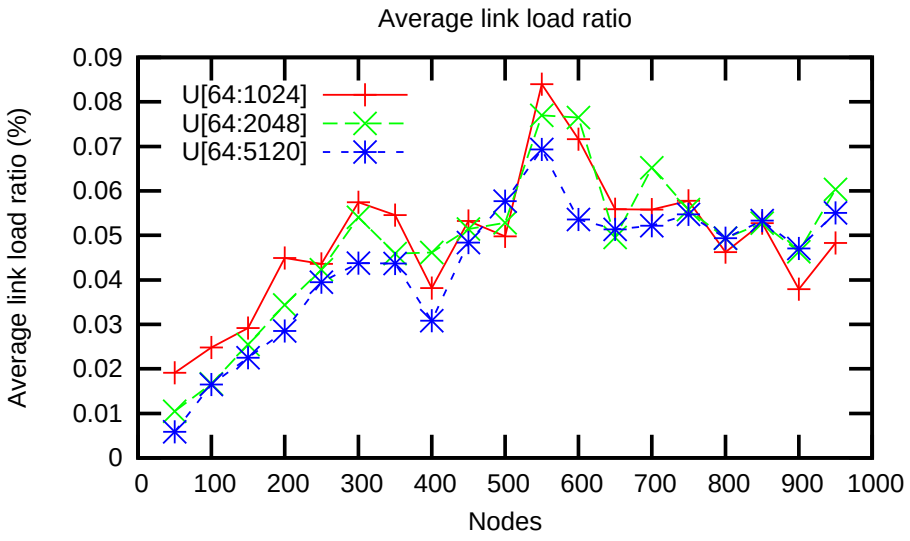

Figure 6: Link load ratio (%)

Therefore, it can be useful to introduce a mechanism that alleviates this behavior. Such a mechanism should block each CP that arrives at a node if a CP belonging to the same flow has already passed that node. This solution is expected to lower the protocol overhead at the cost of increased computational overhead per node. We are currently in the process of extending RDP to include this feature.

Throughout our simulation runs we used the following formula to calculate the timeout value of the timers:

$$T_{out} = 0.2 \cdot (TTL - Hops).$$

This is a simple formula but has the downside of yielding values that are larger than necessary. A better approach would be that the timer value should depend on the QoS delay constraint, which is available in every CP packet. Since the CP will be dropped if the overall path delay exceeds the QoS delay constraint, the delay constraint can act as an upper bound. We are therefore considering the following new solution to compute the timeout:

$$\begin{aligned} T_{out} &= 2\frac{D_{QoS}}{TTL} \times SSF \times (TTL - Hops) \\ &= 2D_{QoS} \times SSF \times (1 - \frac{Hops}{TTL}) \end{aligned}$$

D_{QoS} is the QoS delay constraint, the factor 2 is used to allow an AP to be sent in response to a CP, SSF is a safety scaling factor used to overcome problems

when for example the last links on a path to a destination has a significantly greater value then the first links, TTL is the time-to-live value and *Hops* the amount of hops that the CP already has passed. All these parameters are available in CPs. It is expected that the introduction of the D_{QoS} parameter will solve the problem with bandwidth being reserved for too long time by one flow when the TTL-value is increased.

7 Conclusions

The paper has reported simulation results for RDP, which is a protocol used for constrained-path selection. The results indicate that the RDP scales almost linearly with the number of nodes in the network. The system's ability to find feasible paths is intimately related to the TTL value used in RDP messages: a large TTL value increases the chance of finding a feasible path at the cost of a higher volume of RDP traffic.

Our future work will focus on issues regarding performance improvement of the current version of RDP. In particular, we plan to address the TTL-problem, extend the protocol to alleviate the issue with multiple CPs being routed over the same link and to test the new timeout formula. Another important issue is the behavior of RDP in the presence of churn.

Additionally, we intend to run the simulation on different types of topologies (*e. g.* , Waxman and hierarchical topologies). Also, we plan to observe the protocol behavior in the presence of session arrivals and session durations generated by long-range dependence processes. We expect that these results will provide additional clues on how to improve RDP's performance. When the protocol reaches maturity we plan to test it in a live environment such as PlanetLab.

Acknowledgments

We would like to thank the Swedish Internet Infrastructure Foundation (IIS) and Euro-NGI for granting and supporting the ROVER project during 2006 and 2007.

References

[1] David G. Andersen. Resilient overlay networks. Master's thesis, Dept. of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, May 2001.

[2] Grenville J. Armitage. Revisiting IP QOS. *ACM SIGCOMM Computer Communications Review*, 33(5):81–88, October 2003.

[3] Albert-Laszlo Barabasi and Reka Albert. Emergence of scaling in random networks. *Science*, 286:509–512, 1999.

[4] Jochen Behrens and J. J. Garcia-Luna-Aceves. Distributed, scalable routing based on link-state vectors. In *Proceedings of SIGCOMM*, pages 136–147, London, UK, August 1994.

[5] Gregory Bell. Failure to thrive: QoS and the culture of operational networking. In *Proceedings ot the ACM SIGCOMM Workshops*, pages 115–120, Karlsruhe, Germany, August 2003.

[6] Steven Blake, David L. Black, Mark A. Carlson, Elwyn Davies, Zheng Wang, and Walter Weiss. *RFC 2475: An Architecture for Differentiated Services*. IETF, December 1998. Category: Informational.

[7] Robert Braden, David D. Clark, and Scott Shenker. *RFC 1633: Integrated Services in the Internet Architecture: an Overview*. IETF, June 1994. Category: Informational.

[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*, pages 121–130, Karlsruhe, Germany, August 2003.

[9] Miguel Castro, Peter Druschel, Anne-Marie Kermarrec, Animesh Nandi, Anthony Rowstron, and Atul Singh. Splitstream: High-bandwidth multicast in a cooperative environment. In *Proceeding of IPTPS'03*, Berkeley, CA, USA, February 2003.

[10] Shigang Chen and Klara Nahrstedt. Distributed quality-of-service routing in high-speed networks based on selective probing. In *Proceedings of LCN*, pages 80–89, Lowell, MA, USA, October 1998.

[11] Yi Cui, Baochun Li, and Klara Nahrstedt. oStream: Asynchronous streaming multicast in application-layer overlay networks. *IEEE Journal on Selected Areas in Communications*, 22(1):91–106, January 2004.

[12] Karel De Vogeleer. QoS routing in overlay networks. Master's thesis, Blekinge Institute of Technology (BTH), Karlskrona, Sweden, June 2007. MEE07:24.

[13] Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos. On power-law relationships of the internet topology. In *Proceedings of SIGCOMM*, pages 251–262, Cambridge, MA, USA, August 1999.

[14] The ATM Forum. *Private Network-Network Interface Specification Version 1.0 (PNNI 1.0)*. The ATM Forum, March 1996. af-pnni-0055.000.

[15] J. J. Garcia-Luna-Aceves. Loop-free routing using diffusing computations. *IEEE/ACM Transactions on Networking*, 1(1):130–141, February 1993.

[16] Erol Gelenbe, Michael Gellman, Ricardo Lent, Preixiang Lei, and Pu Su. Autonoumous smart routing for network QoS. In *Proceedings of ICAC*, pages 232–239, New York, NY, USA, May 2004.

[17] Erol Gelenbe, Ricardo Lent, Alfonso Montuori, and Zhiguang Xu. Cognitive packet networks: QoS and performance. In *Proceedings of IEEE MASCOTS*, pages 3–12, Ft. Worth, TX, USA, October 2002.

[18] Oliver C. Ibe. *Essentials of ATM Networks and Services*. Addison Wesley, Boston, MA, USA, 1997. ISBN: 0-201-18461-3.

[19] Dragos Ilie. Overlay routing protocol (ORP). Unpublished architecture and design document, December 2004.

[20] Dragos Ilie. Optimization algorithms with applications to unicast QoS routing in overlay networks. Research Report 2007:09, Blekinge Institute of Technology, Karlskrona, Sweden, September 2007. ISSN: 1103-1581.

[21] Dragos Ilie. *On Unicast QoS Routing in Overlay Networks*. PhD thesis, Blekinge Institute of Technology (BTH), Karlskrona, Sweden, October 2008.

[22] Dragos Ilie and Adrian Popescu. A framework for overlay QoS routing. In *Proceedings of 4th Euro-FGI Workshop*, Ghent, Belgium, May 2007.

[23] P. Leach, M. Mealling, and R. Salz. *RFC 4122: A Universally Unique IDentifier (UUID) URN Namespace*, July 2005. Category: Standards Track.

[24] Zhi Li and Prashant Mohapatra. QRON: QoS-aware routing in overlay networks. *IEEE Journal on Selected Areas in Communications*, 22(1):29–40, January 2004.

[25] Alberto Medina, Anukool Lakhina, Ibrahim Matta, and John Byers. BRITE: Universal topology generation from a user's perspective. Technical Report BUCS-TR-20001-03, Boston University, Boston, MA, USA, April 2001.

[26] Lakshminarayanan Subramanian, Ion Stoica, Hari Balakrishnan, and Randy Katz. OverQoS: An overlay based architecture for enhancing Internet QoS. In *Proceedings of NSDI*, San Francisco, CA, USA, March 2004.

[27] András Varga. `OMNet++`, March 2006. `http://www.omnetpp.org`.

[28] Jared Winick and Sugih Jamin. Inet-3.0: Internet topology generator. Technical Report CSE-TR-456-02, University of Michigan, Ann Arbor, MI, USA, 2002.

[29] Ellen W. Zegura, Kenneth L. Calvert, and Samrat Bhattacharjee. How to model an internetwork. In *Proceedings of IEEE Infocom*, volume 2, pages 594–602, San Francisco, CA, USA, March 1996.