A Scalable QoS Device for Broadband Access to Multimedia Services

Wenyu Zhu, Thomas Dreibholz, Erwin P. Rathgeb
University of Duisburg-Essen, Institute for Experimental Mathematics
Ellernstrasse 29, 45326 Essen, Germany

Xing Zhou
Hainan University, College of Information Science and Technology
Renmin Avenue 58, 570228 Haikou, Hainan, China

Abstract

This paper presents the performance evaluation of a novel network device being located in network edge nodes. It provides a solution for relaxed QoS guarantees to certain flows on a congested link by focussing packet discard on selected flows. However, unlike IntServ solutions – e.g. RSVP – our approach only requires minimal signalling and therefore provides both efficiency and scalability. In this paper, we first describe the ideas of our QoS device and then provide a simulative performance analysis for different multimedia flow scenarios.

Keywords: *Quality of Service, Broadband Internet, Flow Routing, Multimedia, Intelligent Packet Discard*

1 Introduction

With DSL technology becoming widespread, a rising amount of customers gets connected to high-speed Internet backbones. Such links do not only speed up existing applications but also make services like video and audio on demand possible. However, unlike for best-effort applications, such services have stricter QoS requirements. In particular, they need an assured bandwidth. We assume that core bandwidth is usually over-provided and only the link to the customer becomes the bottleneck. When multiple equal-priority flows exceed a DSL link's bandwidth, the quality of *all* flows suffers due to packet loss. Using RSVP to establish per-flow reservations would solve the problem – but also introduce complex signalling procedures and a lack of scalability [1]. In our paper, we present a novel, simple and scalable approach to ensure relaxed QoS guarantees which only requires a minimum effort on signalling. Our approach [3, 5, 10] is based on a QoS device located inside an edge node before the bottleneck. It is currently under consideration by ITU-T and ETSI [6, 7].

2 Our QoS Device

The key idea of our QoS device is that – in case of congestion – it is better to focus packet discard on selected flows than to discard arbitrary packets. Then, only the selected flows would suffer from quality loss instead of all flows. This "last straw"[1] principle, applied at the ATM cell level, was first suggested in [11] and its value has been shown in other publications since [8]. We apply it to our device by making the latest flow(s) the subject of discard (as the default policy; arbitrary other schemes may be applied as well). Therefore, our device has to know when a new flow starts, and has to maintain a record of this new flow.

[1]"It's the last straw which breaks the camel's back" – Proverb.

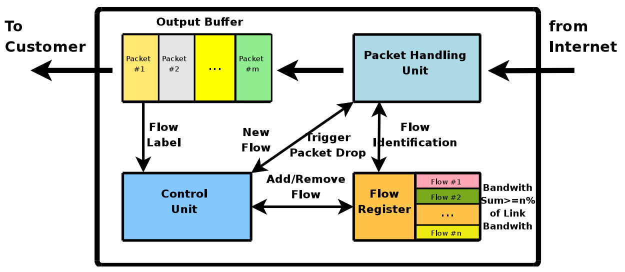

Figure 1. QoS Device Overview

For the device to recognise the start of a flow, its sender is only required to send a *start packet*; no further signalling is necessary. In particular, the sender is not required to wait for any acknowledgement – it may just start sending data. The start packet contains flow identity information (i.e. packet header fields) to identify the data flow and an estimation of the flow's bandwidth. This information is recorded by our device. Flow identification and record keeping can be realized easily as part of a flow router [9].

The device – which is illustrated in figure 1 – maintains a window of flows that are vulnerable to packet discard in case of congestion: the *drop window*. The identities of the drop window flows are stored in the flow register. As new flows start up, a flow moves through the drop window, until it is eventually removed from the window (by being overwritten by new entries), when one of the following conditions is satisfied: (1) The sum of the rates of the flows in the drop window, minus rate of the oldest flow, is greater than R, where R is a percentage of the link bandwidth. (2) The flow's entry is older than time t_{min}. (3) It has received at least p_{min} packets since startup.

When a flow identity is removed from the drop window, it becomes a *guaranteed area* flow, except under extreme traffic conditions (i.e. when dropping all drop window flows' packets is still not sufficient). Note, that the list of guaranteed area flow identities is not stored. All packets not belonging to flows of the drop window are implicitly assumed to be guaranteed flows. This ensures the scala-

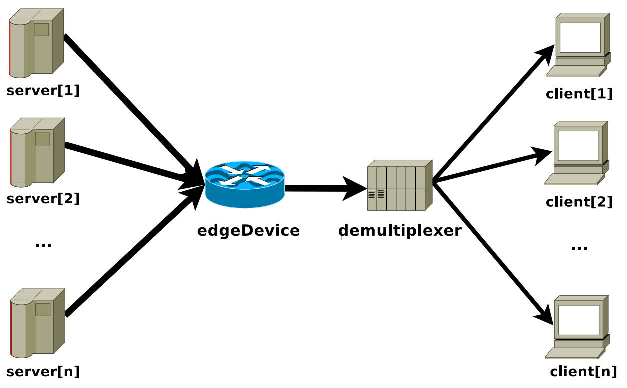

Figure 2. The Simulation Setup

bility of our device. Protection against denial of service attacks (e.g. by filling the drop window with non-existent flow identities) can be provided by a gateway or proxy [5].

By default, we assume that new flows go into the drop window first, in order to avoid disturbing already running flows. Consider a customer viewing a video flow for quite some time: clearly, he would be annoyed if his running flow would be considered for focussed packet loss whenever another user causes overload by new flows. Instead, it is recommended trying to guarantee older flows while discriminate new ones in case of congestion. Nevertheless, our QoS device approach allows the implementation of arbitrary policies, depending on the customers' requirements.

3 Model and Performance Metrics

For our performance analysis, we have modelled the QoS device using the OMNeT++ simulation environment and the SimProcTC [4] tool-chain. The basic simulation setup is illustrated in figure 2 and consists of the QoS edge device module as well as media sources and receivers. Each source can either generate CBR traffic using a configured frame size or by reading the frame sizes from a media trace file. For our analysis, we used MPEG traces as well as H.263 and MP3 traces from [1]. According to the frame size, frames are generated with a fixed frame rate (38fps for MP3, 30fps for video). The inter-frame time randomly varies by $\pm 25\%$ to avoid synchronization among the sources. Frames are segmented to packets, the packet MTU is 1,000 bytes (header overhead is neglected).

As performance metrics for our evaluation, we have chosen delay and packet loss rate. These metrics of the service user's perspective are independent of media codecs and easy to capture. As part of future work, however, more sophisticated methods of evaluating perceptual quality – like PEAQ for audio – are being considered. Delay and packet loss rate are recorded for each receiver.

Our QoS device model has been validated against a former, LISP-based fast-track simulation approach being described by us in [3]. But unlike the simple model, our new model contains support for a number of dropper strategies. The packet dropper is invoked each time the output buffer is too full for storing the incoming packet of a guaranteed area flow: *DropAll* traverses the output buffer and drops *all* packets belonging to drop window flows. *DropEnough* only drops enough packets to fit in the new guaranteed area packet. *SelectiveDropAll* first tries to drop all packets of the first drop window flow. If there is still not enough space, it continues with the second flow and so on (i.e. the drop window position becomes similar to a priority). *SelectiveDropEnough* stops dropping as soon as there is enough space. The number of total packet/flow identity comparison steps within the dropper function is denoted as the *dropper overhead*. It represents the additional effort of the QoS device in comparison to a regular router (i.e. the service provider's performance metric).

GNU R has been used for the statistical post-processing of the results. Each resulting plot shows the average of multiple simulation runs and the corresponding 95% confidence intervals.

4 Performance Analysis

In our performance analysis, we first illustrate the general behaviour of the device as proof of concept in a simple CBR flow setup in subsection 4.1, before we go to MP3 and heterogeneous multimedia scenarios in the following subsections.

4.1 A Proof of Concept

Our basic setup consists of 5 senders and their receivers (i.e. 5 flows). Each flow uses a frame rate of 50 fps and a fixed frame size of 5,000 bytes. Due to the MTU setting, a frame consists of a burst of 5 packets. The resulting total bandwidth is 10 Mbps, while we restrict the link bandwidth to only 8 Mbps. The size of the output buffer is crucial for the delay, so we vary it between 10^5 bits and 10^6 bits. The configured drop window bandwidth has been 2 Mbps, i.e. the flow register memorizes two flows for focussed packet discard; in our case: flow #4 and flow #5.

The simulation results are shown in figure 3: packet loss rate (left-hand side) and delay (right-hand side) for the strategies DropAll (solid lines) and DropEnough (dotted lines) at the clients #4 and #5 (drop window) and client #1 (guaranteed-area; other guaranteed-area flow curves have been omitted, since they are similar). As expected, the drop window flows suffer of focussed packet loss while the guaranteed-area flows' loss rate reduces to 0% when the output buffer size is large enough ($\geq 5 * 10^5$ bits). Furthermore, the loss rate for DropAll is higher, in particular when buffer space is scarce.

Clearly, the reduced loss rate of DropEnough leads to an increased delay. Comparing the delay results of both strategies, an interesting observation can be made: for DropAll, the delay of the drop window flows is lower than for the guaranteed-area flow. Drop window packets either get sent early, or the buffer fills up an they are dropped. However, for DropEnough, an inverse observation can be made: here, drop window may "survive" for some time, but get lost when a burst of guaranteed-area packets comes in.

Figure 4 presents the results for the SelectiveDropAll and SelectiveDropEnough strategies: while there is no difference for the guaranteed-area flow in comparison to DropAll/DropEnough, the selective strategies discriminate flow #5 for the benefit of flow #4. SelectiveDropEnough intensifies this effect by further reducing the loss rate of flow #4 at cost of flow #5. The delay result reflects the expectation from DropAll: SelectiveDropAll leads to the lowest delay for flow #5 (packets are either forwarded quickly or dropped) and the highest delay for the guaranteed-area flow. Again, the order is reversed for SelectiveDropEnough.

Clearly, SelectiveDropAll/SelectiveDropEnough are the better choices from the media user's perspective. But from

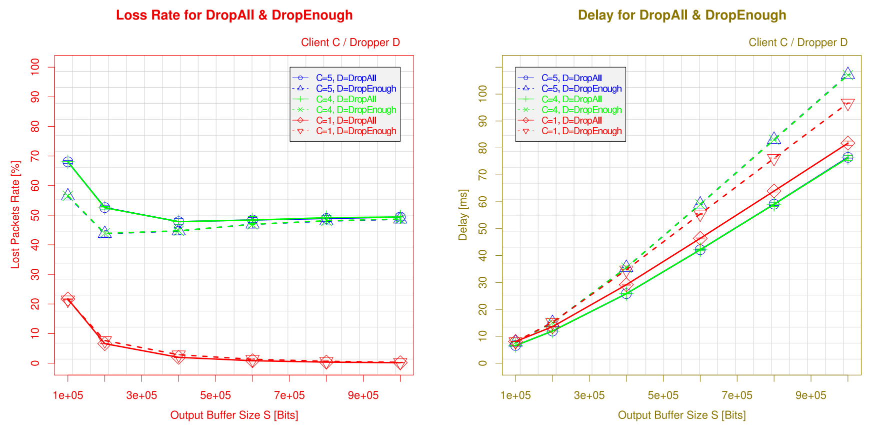

Figure 3. Proof of Concept: DropAll/DropEnough

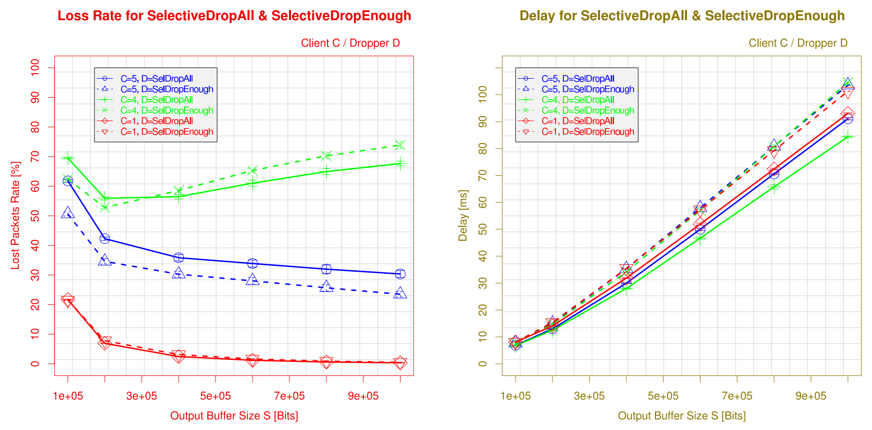

Figure 4. Proof of Concept: SelectiveDropAll/SelectiveDropEnough

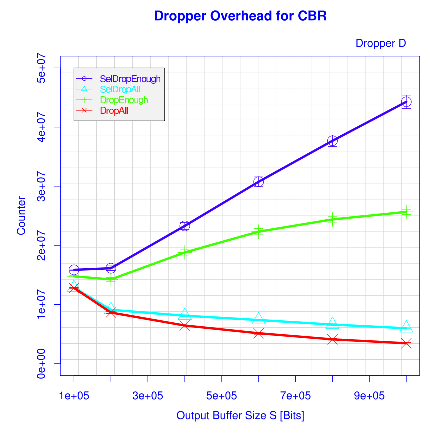

Figure 5. Dropper Overhead

the QoS device provider's perspective, it is clearly useful to keep the additional effort for the dropper small. Figure 5 therefore presents the overhead (as defined in section 3) for the simulations above. For DropAll and SelectiveDropAll, the effort decreases with the output buffer size – a large buffer leads to fewer dropper invocations. On the other hand, the effort increases for DropEnough and SelectiveDropEnough: these strategies just free enough space for storing the next incoming guaranteed-area packet. A burst of packets – caused by segmentation of frames larger than the MTU – leads to a series of dropper invocations. Furthermore, an increased buffer size leads to an increased effort to find packets to drop. Clearly, this effect is amplified by SelectiveDropEnough – which requires multiple iterations for different drop window flow identities.

In summary, our proof of concept has shown that the device using SelectiveDropAll works well from the user's perspective and is also efficient from the provider's perspective.

4.2 MP3 Flows Scenario

For the following simulations, we have set up realistic environments. Nowadays, the most popular multimedia application on the Internet is online radio. Therefore, our scenario contains 12 flows using the VBR MP3-traces from [1]. The total bandwidth estimation is about 2.5 Mbps, while the link bandwidth is only 2 Mbps. The drop window bandwidth is 500 kbps (i.e. 25% of the link bandwidth), leading to 2 flows (here: flow #11 and flow #12) in the flow register.

While the results for the delay are as expected from the proof-of-concept simulations in subsection 4.1 (and therefore a plot has been omitted), the packet loss rate is depicted

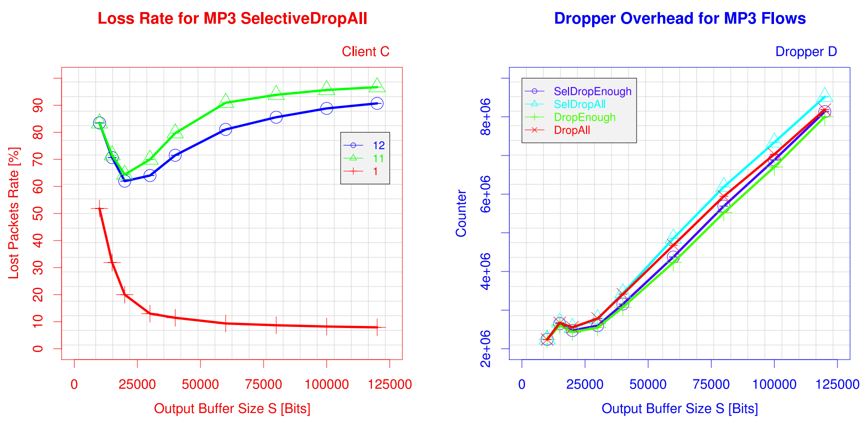

Figure 6. MP3 Flow Scenario

on the left-hand side of figure 6 for the SelectiveDropAll strategy. Again, we have chosen only one "representative" guaranteed-area flow (flow #1, the other flows behave in the same way) and the two drop window flows (flow #11 and flow #12). Flow #11 is the first flow in the drop window and therefore suffers – as expected – from the highest loss rate. Interesting is the loss rate hollow at an output buffer size of S=25,000 bits: for smaller settings of S, packets get dropped very often – regardless of their flow's status (guaranteed-area or drop window). On the other hand, for higher values of S, the QoS device's functionality comes into play: the buffer size if large enough to store sufficient drop window packets which can be dropped when the space gets scarce. That is, space is gained for guaranteed-area packets, leading to reduced guaranteed-area drops but increased loss for drop window flows.

While significant dropper overhead differences have been found in the proof-of-concept scenario in subsection 4.1, the results for the MP3 flow scenario (shown on the right-hand side of figure 6) appear strange: the differences between the strategies are quite small – and SelectiveDropAll even has a slightly higher overhead than SelectiveDropEnough. The reason for this behaviour is that the MP3 flows have significantly smaller frame sizes. The proof-of-concept simulation used a frame size of 5000 bytes, i.e. the each frame has been segmented into 5 packets. With regard to the frame rate, the network bandwidth is much higher, so the interval between two packets has been much smaller than the interval between two frames. Therefore, a large number of packets has arrived in a very short time – leading to a high congestion probability during this period. However, for the MP3 flows, the maximum frame size is 1044 bytes and most frames consist of much less than 1000 bytes. That is, the packet rate is almost equal to the frame rate, so the interval between two packets is approximately equal to the interval between two frames. Therefore, multi-packet congestion (see subsection 4.1) during this time is not likely. Therefore, the SelectiveDropAll and SelectiveDropEnough strategies have similar behaviour from the viewpoint of the dropper overhead. But since SelectiveDropEnough stops immediately after sufficient buffer space has been gained, SelectiveDropAll checks for further packets to be dropped (which is not necessary here to prevent congestion, but nevertheless increases the overhead).

4.3 Adapting the Drop Window Size

Another important parameter to be analysed is the number of flows in the drop window. We reuse the MP3 flow setup of subsection 4.2 and adjust the number of flow register entries by an appropriate setting of the flow register bandwidth (each MP3 flow has an average bandwidth of about 200 Kbps).

In the results analysis of the simulation, we stress on the packet loss rate – the delay results are as expected. Figure 7 shows the packet loss rates of the flows for the SelectiveDropAll strategy and output buffer sizes 20,000 bits (left-hand plot) and 120,000 bits (right-hand plot). Flow #1 is again the "representative" guaranteed-area flow, the other flows – beginning from flow #12 – are successively added to the drop window with increasing number of drop window entries m. That is, for a single entry, the drop window contains only flow #12, while it includes flow #12 to flow #7 for 6 entries.

For a too small output buffer (20,000 bits, left-hand plot), it becomes difficult for the dropper to find a sufficient number of drop window packets to gain space. Therefore, even for a flow register bandwidth of 60% of the link bandwidth (i.e. 6 flows here), the guaranteed-area packet losses do not reach 0%. Using a more appropriate buffer size (120,000 bits, right-hand plot), a flow register size of 40% (i.e. 4 flows) already achieves a lossless transport of the guaranteed-area packets.

In summary, it is crucial for the QoS device's functionality to have a certain output buffer space to find drop window packets to discard in case of overload. Clearly, the number of drop window flows also has to be sufficiently large. Therefore, the administrator of the device has to carefully provision these parameters to achieve the highest benefit for the user. However, unexpected things can always happen – and a configuration may be non-optimal, resulting in guaranteed-area losses.

4.4 Heterogeneous Multimedia Flows

In order to test our model in a more complex and heterogeneous environment, ten different media trace files (MP3, MPEG and H.263 from [1]) are used for the following simulation. The flows' total estimated bandwidth is about 12.5 Mbps, so the output rate is set to 10 Mbps to trigger congestion. The drop window is 2.5 Mbps, which consists of flow #10 (MPEG stream) and flow #9 (MP3 stream).

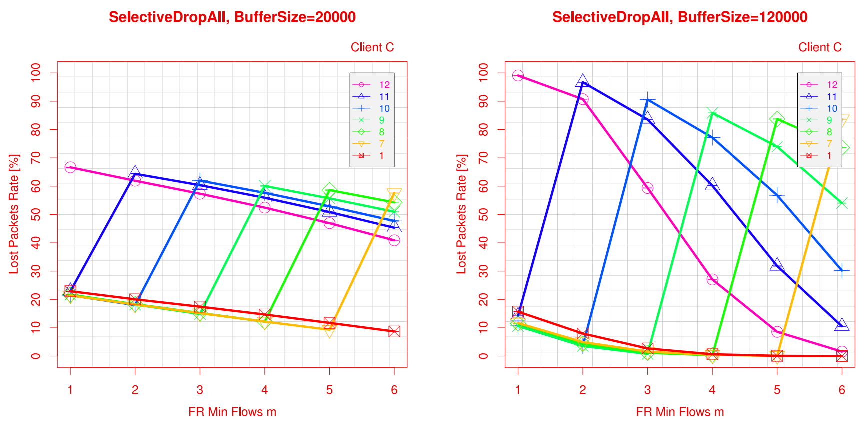

Figure 7. Variation of the Drop Window Size

In our analysis, we also present two selected guaranteed-area flows: flow #1 (a low-bandwidth MPEG-stream) and flow #3 (a high-bandwidth MPEG stream).

The left-hand side of figure 8 presents the packet loss rate using the SelectiveDropAll strategy. Although two flows (flows #10 and #9) are in the drop window, this is clearly insufficient: even for an output buffer size of as large as $1.6 * 10^6$ bits, flow #3 in the guaranteed area still suffers from a loss rate of more than 10%. The reason for this problem is the flow register policy: originally, the flows in the drop window follow the FIFO policy. That is, when a new flow identity is pushed into the tail of the flow register, the first flow identity in the flow register will be popped out. Finally, the latest flows are kept in the register.

To cope with this problem, we change the flow register policy: when a start packet comes in, the flows which have the highest bandwidth estimation are put into the drop window. In this case, the drop window will finally consist of high-bandwidth flows only. The right-hand side of figure 8 shows the results for using the new policy. Now, the drop window consists of the high-bandwidth flows #3 and #10, which are in the focus of packet discard. All the other flows in the guaranteed area benefit from lossless transmission.

However, while the new "Highest Bandwidth" policy achieves a significant benefit over "FIFO" in case of congestion, it has an obvious disadvantage: high-bandwidth flows are target of packet drops – without regard to how long they are already transmitting. Taking the example of the video user from section 2, a viewer already watching a movie for some time would clearly be disappointed when its stream gets the focus of packet discard in favour of another user's newly started soap-opera stream.

4.5 Avoiding Guaranteed-Area Losses

The solution for avoiding guaranteed-area losses in case of extreme overload is clearly to add another flow to the drop window, which – hopefully – leads to sufficient discard possibilities for the dropper to resolve the guaranteed-area congestion. Appending the new flow to the tail of the flow register, SelectiveDropAll will touch it only as last resort – after all other drop windows packets have been discarded from the output buffer. This should minimize the user's quality loss. Selecting a guaranteed-area flow for move into the drop window is challenging: as described in section 2, the identities of guaranteed-area flows are not stored. Our solution is simple: we derive identities from packet headers (addresses, ports, etc., see [2]) in the output buffer. That is, selecting a guaranteed-area packet obtains its flow identity.

Two policies for selecting a new drop window flow have been considered: "Add Random" simply selects a random guaranteed-area packet and adds its flow identity. "Add Max Packets" adds the flow currently having the largest number of packets in the output buffer (i.e. the largest contributor of the congestion).

For the evaluation of our flow selection policies, we reuse the multimedia flow setup from subsection 4.4 but add two more MP3 streams (flow #11 and flow #12). Initially, only the flows #12, #11 and #10 are in the drop window. Flow #3 and flow #10 are the highest bandwidth flows (MPEG streams). Obviously, this setup creates high overload, which cannot be solved by the two small-bandwidth MP3 entries. Therefore, the QoS device has to select a further drop window entry.

The left-hand side of figure 9 presents the packet loss rate for using the "Add Random" policy. As shown, this policy still does not entirely solve the problem: even for an output buffer size of $1.6 * 10^6$ bits, there are still guaranteed-area losses of up to 10%. The problem of this policy is that random selection may only add another small-bandwidth flow to the drop window. Then, the benefit remains small.

On the other hand, the "Add Max Packets" policy achieves the desired result (see the right-hand side of figure 9): it selects the high-bandwidth flow #3 for focussed discard. After that, the drop window size is large enough to avoid guaranteed-area losses.

5 Conclusions

This paper has described our QoS device approach to handle multimedia flow congestion on bottleneck links to broadband customers: instead of dropping packets of all streams indiscriminately (which leads to reduced QoS for all users), our device focuses packet discard on certain target flows. Since the device only has to store these selected flow identities and only requires minimal signalling overhead (i.e. the transmission of a start packet), it is – unlike classical IntServ approaches – very scalable. In our simulative performance evaluation, we have presented the general behaviour of the device for multimedia scenarios: using reasonable settings for output buffer size, flow register and policies, a significant performance benefit can be achieved for the users.

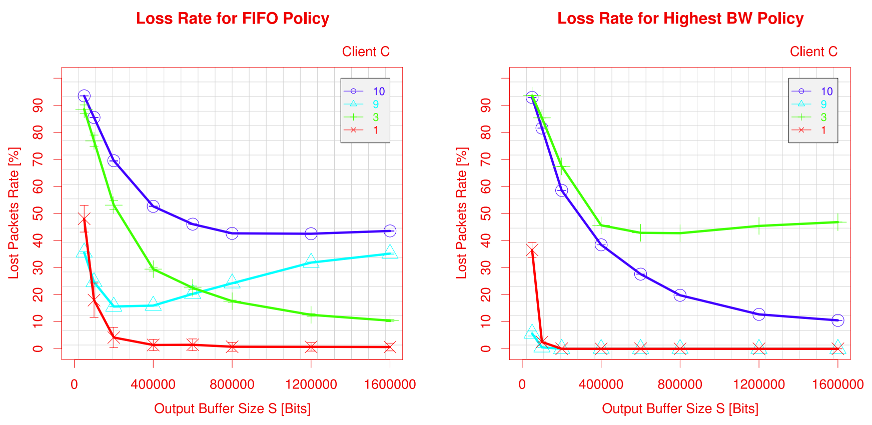

Figure 8. Multimedia Flow Scenario

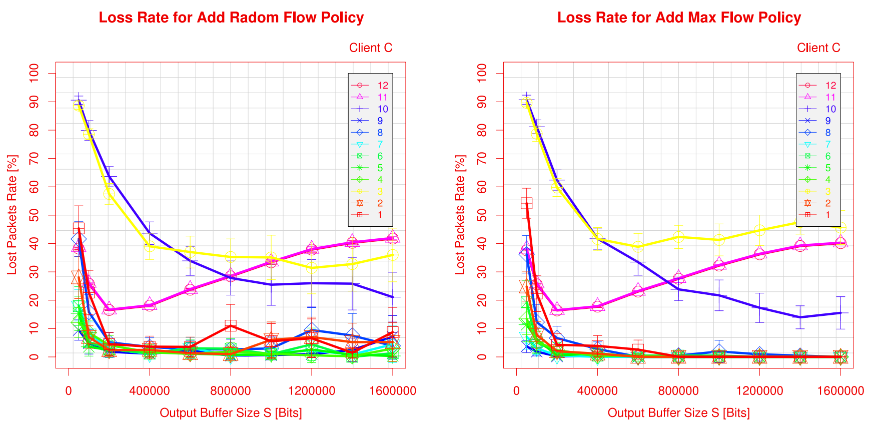

Figure 9. Flow Selection Policies

As part of future work, more analyses of the device parameters are necessary. In particular, it is useful to develop an algorithm to automatically adapt the parameters to the current flow scenario and changing needs of the customers. Furthermore, we are going to perform more codec-specific evaluations of the QoS by applying perceptual quality metrics like PEAQ and PEVQ. Also, we intend to realize the QoS device functionality as a queuing discipline (QDisc) for Linux, in order to perform lab experiments using real-world applications.

References

[1] T. Dreibholz. Management of Layered Variable Bitrate Multimedia Streams over DiffServ with Apriori Knowledge. Masters Thesis, University of Bonn, Institute for Computer Science, Feb. 2001.

[2] T. Dreibholz. An IPv4 Flowlabel Option. Internet-Draft Version 08, IETF, Individual Submission, July 2008. draft-dreibholz-ipv4-flowlabel-08.txt, work in progress.

[3] T. Dreibholz, A. IJsselmuiden, and J. L. Adams. An Advanced QoS Protocol for Mass Content. In *Proceedings of the IEEE Conference on Local Computer Networks (LCN) 30th Anniversary*, pages 517–518, Sydney/Australia, Nov. 2005. ISBN 0-7695-2421-4.

[4] T. Dreibholz and E. P. Rathgeb. A Powerful Tool-Chain for Setup, Distributed Processing, Analysis and Debugging of OMNeT++ Simulations. In *Proceedings of the 1st ACM/ICST OMNeT++ Workshop*, Marseille/France, Mar. 2008. ISBN 978-963-9799-20-2.

[5] T. Dreibholz, A. J. Smith, and J. L. Adams. Realizing a scalable edge device to meet QoS requirements for real-time content delivered to IP broadband customers. In *Proceedings of the 10th IEEE International Conference on Telecommunications (ICT)*, volume 2, pages 1133–1139, Papeete/French Polynesia, Feb. 2003. ISBN 0-7803-7661-7.

[6] A. IJsselmuiden and J. L. Adams. Delivering QoS from remote content providers. British Telecom Contribution TISPAN 01(03)TD132, ETSI, Sept. 2003.

[7] A. IJsselmuiden and J. L. Adams. Delivery of assured QoS content in NGNs. British Telecom Contribution D-,Q4,6,10,11,16,SG13, ITU-T, Feb. 2004.

[8] K. Kawahara, K. Kitajima, T. Takine, and Y. Oie. Performance evaluation of selective cell discarding schemes in ATM networks. In *Proceedings of the IEEE Infocom '96*, volume 3, pages 1054–1061, San Francisco, California/U.S.A., Mar. 1996. ISBN 0-8186-7292-7.

[9] L. G. Roberts. The Next Generation of IP – Flow Routing. In *Proceedings of the International Conference on Advances in Infrastructure for e-Business, e-Education, e-Science, e-Medicine on the Internet 2003S International Conference*, pages 25–, L'Aquila/Italy, July 2003.

[10] A. J. Smith and J. L. Adams. Packet discard control for broadband services. Technical Report EP 01 30 5209, British Telecom, June 2001.

[11] A. J. Smith, J. L. Adams, C. J. Adams, and A. G. Tagg. Use of the Cell Los Priority Tagging Mechanism in ATM switches. In *Proceedings of the ICIE 1991*, Singapore, Dec. 1991.