STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LI**, Number 1, 2006

SUPPORTING MULTIMEDIA STREAMING APPLICATIONS INSIDE THE NETWORK

ADRIAN STERCA, FLORIAN BOIAN, DARIUS BUFNEA, CLAUDIU COBÂRZAN

Abstract. There are many reasons for quality degradation of multimedia streams inside the network. Two of the most important ones are related to the UDP protocol and the Random-Drop policy implemented inside most of the routers currently deployed on the Internet. We intend to discuss, in this paper, some of multimedia streaming applications' problems centered around the UDP protocol and Random-Drop policy. We also present some mechanisms to alleviate these problems. More specifically, we present a queue management algorithm PDQMAMS (Priority-Drop Queue Management Algorithm for Multimedia Streams) for supporting the quality of multimedia streams inside the network.

1. Introduction

As multimedia encoding standards like MPEG and H.264/AVC become more efficient and as new digital video processing tools are developed, multimedia streaming applications gain a more important percent of the data transferred over the Internet. However, the current Internet does not support the high bandwidth and low latency demands of multimedia applications. Hence, communicating partners need to adapt to rapidly changing connection parameters and also to provide the best use of the scarce bandwidth of nowadays networks. Because of the latter reason, most of multimedia streaming applications choose to use UDP instead of TCP as the transport protocol, trading speed over reliability.

UDP offers several advantages over TCP, the most important being its speed. But the speed of UDP comes with a cost: it is unreliable and not responsive to congestions. Because UDP is unresponsive to congestion, it can cause the occurrence of congestion and can lead to unfairness against TCP-friendly flows. Most of the routers in the Internet use a Random Drop policy for freeing up space when congestion occurs and the queue is full. This policy gives multimedia packets an

Received by the editors: April 2, 2006.

2000 *Mathematics Subject Classification.* 90B18, 68M20.

1998 *CR Categories and Descriptors.* 90B18 [**Operations research, mathematical programming**]: Operations research and management science – *Communication networks*; 68M20 [**Computer science**]: Computer system organization – *Performance evaluation; queueing; scheduling*;

uniform importance although multimedia encoding schemes regard them as having different levels of priorities. This way of things is very bad for multimedia streaming applications. Because most of the encoding standards use layered/predictive encoding to achieve a good compression rate (i.e. some video frames are encoded as the difference between the current frame and the previous and/or the next one), dropping an independent frame in a router makes the other frames dependent on this one useless on the end-user's system even if they survive the ride through the Internet.

This paper presents a queue management algorithm for routers that tries to alleviate the aforementioned problems. It maintains a state for each multimedia flow and uses a priority-driven dropping scheme when the buffer overflows. Also, it is fair to each flow when packet drops are imperative and smart enough to drop packets belonging to less important frames, if congestion occurs. The rest of the paper is structured as follows: section 2 presents the main problems faced by multimedia streams inside the network and the causes of these problems; section 3 presents the AMSP protocol (Adaptation-aware Multimedia Streaming Protocol) whom our solution is based on and our own Priority-Drop Queue Management Algorithm for Multimedia Streams (PDQMAMS); in section 4 we perform experiments for proving PDQMAMS's qualities; section 5 presents some improvements of the algorithm we have in mind; then, in section 6 we refer to related work and the paper ends with conclusions and future work in section 7.

2. Problems of multimedia streams inside the network

Multimedia streaming application have huge requirements related to the network. One of the most important and hard to satisfy requirements is the high bandwidth that multimedia streaming applications need. For example, the MPEG-1 compression standard [1] demands a bandwidth of up to 1.5 Mbit/s, while the MPEG-2 standard [1] supports data rates of up to 4 Mbit/s. Besides huge bandwidth amounts, there are other requirements related to the timeliness of the transmission of multimedia streams. For an optimal quality of the multimedia data, real-time multimedia streaming often demands that communication be isochronous. This implies very short delays and also small values for jitter, but also good continuous throughput. If the delay between packets is too high, the video will periodically freeze at the receiver, waiting for the following frames to arrive. Jitter is equally important: a jitter too great means high fluctuations of the delay and this can lead to buffer underrun or overrun at the receiver, causing degradations of the video. These requirements are hard to satisfy in a best-effort network like the Internet.

The major problem of a best-effort network with respect to multimedia applications is that no QoS guarantees can be given. In a network with variable delays and different levels of congestion, communicating partners need to adapt

to rapidly changing connection parameters (e.g. reducing the quality of the multimedia stream, *adapting* the video or audio material). Multimedia streaming applications often choose UDP as the transport layer protocol, instead of TCP, trading low delay and high throughput over reliable delivery. More specifically, multimedia streaming applications choose UDP over TCP to avoid TCP's start-up delay, to avoid the overhead of maintaining a state for each connection (like TCP does) and to favour timeliness characteristics (e.g., delay, jitter, etc.) at the cost of reliability (e.g., retransmission timeouts, in-order delivery, etc.). The speed advantage of UDP over TCP comes with a drawback: it is unresponsive to congestion. This is a major pitfall of multimedia streaming applications, as multimedia streams, which are basically UDP flows, are not fair to TCP-friendly flows (i.e. a flow whose sending rate does not surpass the sending rate of a TCP connection in the same circumstances [2]) and also they are not fair to each other. Depending on each connection's parameters, one UDP flow can eat up a lot more downstream bandwidth than the other flows that pass through the same router, especially if these are TCP-friendly flows. In steady-state, when a TCP flow notices congestion (a packet drop or ECN packet [3]) it backs-off, reducing its sending rate by half. In contrast, a UDP flow does not sense congestion because it is stateless and not connection-oriented like TCP. If packets are dropped, UDP flows continue to send packets at the same rate and, in the worse case, they can even increase their sending rate. This way, they can lead to starvation of other TCP-friendly flows [4] or even to congestion collapse [5]. A remedy for the congestion and unfairness problems of UDP flows can be considered from two perspectives. First, congestion avoidance and fairness with respect to other flows can be achieved if multimedia streams implement some form of end-to-end congestion control similar to the AIMD mechanism of TCP - this could be implemented either to the transport level (DCCP [6]) or to the application level -, so that the multimedia flow decreases its rate in response to congestion notification from the network. Second, the problems of congestion control and fairness between flows can be tackled inside the routers using active queue management like RED (Random Early Detection) [4, 8] or fair scheduling mechanisms like SFQ (Stochastic Fairness Queuing) [9] and WFQ (Weighted Fair Queuing) [10]. Active queue management algorithms try to avoid congestion and achieve fairness by managing the length of the queue and dropping packets from the queue when necessary or appropriate. On the other hand, scheduling algorithms decide which packet to send next and they mainly provide fair allocations rather than congestion avoidance.

The drop policy of most routers currently deployed on the Internet is the Random-Drop Policy. We refer with the term of "Random-Drop Policy" to all dropping policies that don't take into consideration the differences between packets when taking a dropping decision and the drop is made only when the queue overflows (e.g. Head-Drop, Tail-Drop, Random-Drop). The Random-Drop Policy

is disastrous for multimedia streaming applications. The majority of multimedia encoding standards (e.g., MPEG, H.264/AVC, etc.) use a layered/predictive encoding scheme in order to achieve a good compression ratio, i.e., some of the frames depend on other *base frames* in the encoding process. In this encoding process, a basic layer is first encoded normally and any other enhancement layers are encoded as the difference between the base layer and the desired quality layer. For example, the MPEG-family of standards [11] uses four types of frames in the encoding process: *I-frames* (Intra-coded frames), *P-frames* (Predictive-coded frames), *B-frames* (Bi-directionally predictive-coded frames) and *D-frames* (DC-coded frames).

The Random-Drop Policy gives multimedia packets an uniform importance although multimedia encoding schemes regard them as having different levels of priorities. Dropping a packet belonging to a base layer (e.g. I-frame) makes subsequent packets/frames that depend on this one useless at the receiver, because it can not decode the enhancement layer (B- or P-frames) without the base layer (depending I-frame). A possible solution for this problem assumes assigning some sort of priorities (the priorities should be assigned either by the router or by the streaming protocol) to packets and drop them accordingly. In other words, we would rather drop an insignificant packet - one whose missing is tolerable by the receiver - than throw away indispensable data.

3. Priority-Drop Queue Management Algorithm for Multimedia Streams (PDQMAMS)

We argue that providing a real support for multimedia streams inside the network, at router level, requires some form of help from multimedia streaming protocols (priority schemes, feedback, etc.). The router algorithm, whether queue management algorithm or packet scheduling algorithm, can not do this job on its own. Following this direction, we present a router-queue management algorithm that uses knowledge from the multimedia streaming protocol (knowledge inserted in protocol's header inside packets) for preferentially dropping packets and, thus, for providing support for multimedia streams. We chose to use the AMSP protocol [12] instead of RTP [7], which is widely used for multimedia communication, because it conveys scaling information together with multimedia data as opposed to RTP which offers limited support for scaling. This is exactly what we need if we want to alleviate the impact of congestion on multimedia streams inside the network. More specifically, by assigning priorities to multimedia packets, AMSP conveys "intelligence" to network routers which would help them in taking better dropping decisions. AMSP also offers better QoS feedback support than RTP.

3.1. **The AMSP protocol.**

The Adaptation-aware Multimedia Streaming Protocol is a streaming protocol similar to RTP. It conveys time sensitive information like multimedia data together with scaling information, so that multimedia streams

can be adapted inside the network to the rapid changing parameters of the network. The scaling information can be used by common core routers to perform packet-level adaptation of multimedia streams (i.e. drop less important packets) or it can be used by scaling proxies that can perform complex media transformations inside the network (e.g. color reduction, temporal reduction, transcoding, etc.). The central concept of AMSP is the channel concept. Each channel is identified by an 8-bit field in the AMSP header called the ChannelID field. This field encapsulates the channel number, the priority of the channel and the dropping capability (whether packets belonging to this channel can be dropped). There a several types of channels AMSP supports: *control channel*, *media channels*, *metadata channels*, *scaling control channels*, *retransmission channels*, *feedback channels* and *auxiliary channels*. A multimedia stream is mapped onto one or more media channels, and thus, its packets get the priority of the respective channel(s). The idea of using channels to convey data with different levels of importance comes from the fact that multimedia streams are based on related layers with different levels of importance. AMSP offers elaborate feedback support including acknowledgements, non-acknowledgements, number of received/discarded/duplicated packets, round-trip time, jitter, bandwidth, buffer size, etc.

By assigning different levels of importance to multimedia packets, an AMSP-aware router could break this uniform treatment of packets when a dropping decision must be taken, which could only be beneficial to multimedia streams. By dropping less important packets (e.g. packets belonging to a B-frame or a P-frame, instead of an I-frame), an AMSP-aware router can achieve greater performance for multimedia traffic in case of congestion.

3.2. **The PDQMAMS algorithm.** The Priority-Drop Queue Management Algorithm for Multimedia Streams is essentially a queue management algorithm for a stateful AMSP-aware router that achieves fairness between AMSP-flows. Our goal in this paper was to develop a router algorithm that supports multimedia traffic inside the network. We could achieve a good result in this direction by implementing a simple queue management AMSP-aware algorithm that preferentially drops packets according to AMSP channel priorities when congestion occurs. We don't want to use for this purpose a queue scheduling algorithm since this would induce higher delays for packets inside routers; we want to use plain FIFO discipline. However, a simple AMSP-aware algorithm for queue management would have a major drawback: it would lack fairness among flows. An AMSP session (flow) will be identified by a pair of (server IP, client IP) together with a pair of (source port, destination port). Because the number of channels and the priority levels a multimedia stream should use is left to the application's choice, multiple AMSP sessions use different numbers of priority levels. Hence, if we have multiple AMSP sessions competing for bandwidth and these sessions have different numbers of priority levels (depending on each one's number of elementary streams) it is very

likely that the router always selects the packet with the lowest priority from the same set of streams (belonging to a constant AMSP session). But this won't be fair to AMSP flows since the algorithm will show bias for a constant AMSP flow when a dropping decision has to be made (this flow is the AMSP flow that owns the channel with the lowest priority from all the channels of all AMSP flows). It is clear that a simple stateless AMSP-aware router will not be able to achieve fairness when drops have to be done. Consequently, our queue management algorithm must be stateful, i.e. it must maintain a state for each flow.

We consider two kinds of flows: (1) the AMSP flows and (2) one flow that contains all other non-AMSP packets (called the *other flow*), and although we have a single physical queue of packets, each flow will have its own virtual queue. As we said before, the granularity of AMSP flows will be a pair of (IP,port) binoms (one for source and one for destination). In order to keep the state maintained at the router small, we use a hash bucket with a limited number of slots for flows differentiation. When a packet arrives at the router it is first classified: if it is an AMSP packet, the packet is assigned to a slot from the hash bucket by applying the hash function on a value composed by an aggregate of source IP, destination IP, source port, destination port and transport protocol value. If the packet does not belong to an AMSP flow, it is assigned to the *other flow*. Once a packet gets assigned to an AMSP slot/flow, it is further added to the specific channel this packet belongs to in the respective AMSP flow. The number of channel is determined from the ChannelID field of the AMSP header of the packet. Each flow has a number of linked lists for every channel it has. A graphical representation of PDQMAMS's architecture is shown in Fig. 1.

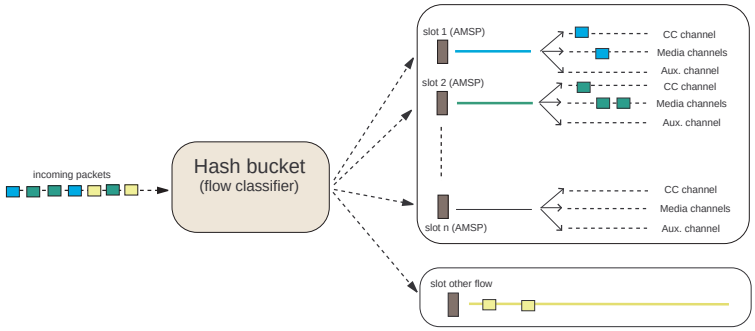


FIGURE 1. The PDQMAMS router architecture

An AMSP packet is placed on two linked lists: the global list with all packets from the queue and the list of the channel it belongs to, inside the AMSP flow. A non-AMSP packet is also placed on two linked lists: the global list and the list

of packets belonging to other flows. To be noted that due to the number of slots available for the hash function, it is possible that an AMSP packet gets placed on the same slot with other AMSP packets not belonging to the same AMSP flow. To minimize the number of collisions inside a slot, the hash function is perturbed every 10 seconds. We now detail each operation PDQMAMS performs.

I. Packet enqueuing. If the queue is not full, the packet is enqueued normally. The classifier first determines if this is an AMSP packet or not. If not, the packet is added to the *other flow*. If this is an AMSP packet, based on the hash value for this packet, it is attached to an AMSP slot/flow and inside the flow is assigned to the corresponding AMSP channel. The packet is also added to the main queue. If the queue is already full when the packet arrives, one or more packets are chosen for dropping (using a sort of weighted fair dropping algorithm, see subsection III, Packet dropping) and the new packet is added to the queue the same way as shown above. This way, newer packets are favoured, as for old enqueued packets, there is a good chance they will be useless at the receiver (because their presentation time has expired).

II. Packet dequeuing. Packets are dequeued for sending using a simple FIFO strategy. This strategy is very light, easy to implement and minimizes the maximum delay of a packet in the router's queue.

III. Packet dropping When the queue is full, a dropping decision has to be made in order to accommodate a new packet. The dropping decision has to subscribe to two guidelines: it must be as fair as possible to all flows and it must protect multimedia streams. For achieving fairness, the dropping choice must be influenced by the number of packets each flow has in the queue. For the second goal, the router must always select for dropping the lowest priority packet. In order to decide which flow to drop from, all the flows (their packets) are linearly mapped on a list with the length QLen, so that all the packets of each flow are placed consecutively on this list and QLen is the number of queued packets in the router (see Fig. 2). A random integer number between 1 and QLen is generated. Let this number be k. The flow chosen for dropping is the flow that the k-th packet from the list belongs to. After the flow for dropping is chosen, we must decide which packet(s) from that flow will be dropped. If the flow is an AMSP one, we always choose for dropping the packet(s) with the lowest priority(es). If multiple packets apply, we drop the oldest one. In the case of a non-AMSP flow, we apply a tail-drop discipline. This way, a flow that has a greater number of packets in the queue, has a higher probability to be selected for dropping. This is somewhat dangerous for the *other flow* because all non-AMSP packets are assigned to this flow, thus, the length of the *other flow* can significantly surpass the length of AMPS flows, making it very vulnerable to selection when drops have to be done. This is why, we do not map all the packets from the *other flow* on the linear list, but we map only a limited number of packets from this flow (the number of all non-AMSP packets divided by a certain weight).

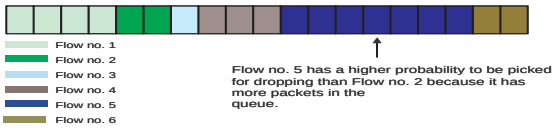


FIGURE 2. Packet dropping in PDQMAMS

Note that the degree of fairness depends on the number of slots in the hash bucket. If the number of slots is great, then the number of collisions will be small and there is a bigger chance that all the packets from a slot belong to the same flow, hence, dropping fairness is increased. Conversely, if the number of slots is small, there are more collisions in the slots and packets from different flows are mixed with a higher probability. This problem is somewhat alleviated by perturbing the hash function each 10 seconds.

4. Experimental results

To evaluate our algorithm, we have implemented PDQMAMS as a queue discipline under the Linux kernel version 2.4.24. We used for our tests the logical network topology depicted in Fig. 3. The connection between the router R1 and router R2 was limited using the Linux traffic control framework [13] and the code corresponding to the AMSP client and AMSP server was taken from the AMSPLibrary [12]. PDQMAMS was deployed on the router R1. All the links except the R1 - R2 link have capacities higher than the bandwidths requested by our AMSP sessions, so that the only congested link is R1 - R2.

FIGURE 3. The net topology used in experiments

In the first experiment, we wanted to see how PDQMAMS supports multimedia streams by dropping lower priority packets when the queue overflows. We started one AMSP server on S1 and one AMSP client on C1 and let them run for 300 seconds. The server was sending a synthetic 512kbit/s stream with 30 fps to the client C1. The size of the frames were chosen so that I-, P- and B-frames with priority 2 fit into an AMSP packet. All other frames are larger than the MTU and are fragmented by the application. The synthetic multimedia stream was encoded using a pattern of one I-frame followed by 5 P-B-B-B-B sequences, which is common for MPEG streams. The outgoing link from the PDQMAMS router to

the router R2 was limited to 64kbit/s. As you can see from Fig. 4 the percent of frames received is proportional to the frame priority. Hence, the frame type that has the highest percent of arrived frames is the I-frame which has the highest priority (zero). It is followed by P-frames which have priority 1 and then B-frames with priorities running from 2 to 6. Hence, by dropping less important packets (B-frames with priorities of 6, 5, 4 and 3), PDQMAMS increases the perceived quality of multimedia streams.

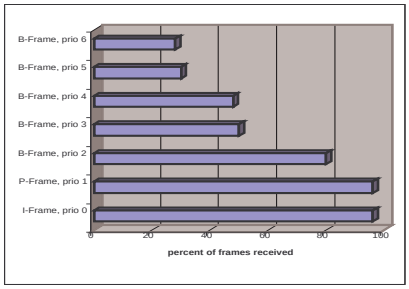
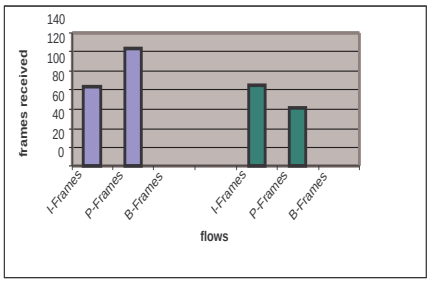
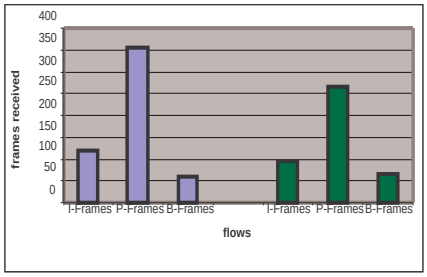


FIGURE 4. bandwidth=64kbit/s, one flow, duration=5 min.



(a) bandwidth=64kbit/s, time=2 min.



(b) bandwidth=128kbit/s, time=2 min.

FIGURE 5. Experiments with two flows (blue & green)

In the second and the third experiment we wanted to prove that PDQMAMS is fair to flows when drops are imperative. These experiments differ only in the bandwidth limitation applied to the link R1 - R2. In the experiment no. 2 this link had a capacity of 64kbit/s and in the experiment no. 3 the capacity is 128 kbit/s. Both experiments were run for 120 seconds. For both experiments two AMSP flows pass through the PDQMAMS router R1 and both flows have a bandwidth

demand greater than the one provided by the network (64kbit/s for experiment 2 and 128kbit/s for experiment 3). We started two AMSP-server process on machine S1 and S2 and two clients, one on machine C1 and the other one on machine C2. The first AMSP server sends a synthetic multimedia stream to client running on C1 and the second AMSP server sends the same synthetic multimedia stream to client running on C2. The synthetic multimedia stream was the same as the one used for the first experiment. As you can see from Fig. 5(a) and Fig. 5(b) the PDQMAMS-router does a satisfactory job in achieving fairness between the two AMSP flows, while complying to the lowest priority drop scheme for supporting the quality of multimedia streams. The reason we have more P-frames received than I-frames for some AMSP flows is because in the multimedia streams, the number of I-frames is much smaller than the number of P-frames (this is compliant with the MPEG standards for achieving a better compression ratio) since the encoding pattern of the stream was one I-frame and then 5 P-B-B-B-B sequences.

5. Improvements of the algorithm

The PDQMAMS algorithm presented above, as we have seen, does a good job in supporting/protecting multimedia streams (by employing a priority-based dropping scheme) and also achieving a good degree of fairness between flows. The quality of multimedia streams is surely increased at the receiver side. However, this algorithm does not directly address the problem of congestion or, more exactly, congestion avoidance. In order to pro-actively avoid congestion inside the router while still providing fairness between flows and support for multimedia streams, the PDQMAMS algorithm needs to incorporate probabilistic dropping schemes before the queue is full like the ones employed by RED [8]. This leads us to a modified version of PDQMAMS that is still AMSP-aware and stateful, but drops are made *early*, before the queue overflows and the dropping scheme is slightly different.

We could, try on a first approach to apply a RED [8] or an Adaptive RED [14] test on the global queue. This would improve congestion avoidance and prevent buffer overflows. However, RED was mainly designed for avoiding congestion inside routers by controlling the sending rate of TCP conformant flows. So it assumes the link is utilized mostly by TCP flows (Adaptive RED relaxes to some degree this restrictions). Our improved PDQMAMS can not impose such conditions, as one of its goals is to protect multimedia streams. But a RED/Adaptive RED test can still be applied to the *other flow* where the majority of packets will belong to TCP-friendly flows. On the other hand, an AMSP flow can be subject to a RED-like test based on the average length of the flow's virtual queue and a special threshold specific for multimedia streams. This threshold should specify an upper limit for an AMSP flow's queue length, above which the packets won't be of any good at the receiver side because the delay experienced by the packet is too high (and the presentation time at the receiver had already expired before this packet

would have left the queue). This threshold value can be considered as a kind of staleness threshold. If the average length of the virtual queue of an AMSP flow is greater than this staleness threshold, the lowest priority packet from this flow should be dropped.

The improved PDQMAMS algorithm outlined above will have a great chance in providing fairness for all flows, support for multimedia streams and pro-active congestion avoidance.

6. Related work

Our queue management algorithm drops lower priority packets when the router's queue becomes full. There is another approach to this prioritized treatment of packets inside a router's queue. Incoming lower priority packets can be dropped, before the queue is full, to make room for incoming higher priority packets. In this approach, an incoming lower priority packet is dropped if the number of lower priority packets from the queue is above a threshold value. This approach of managing prioritized packets inside a router's queue has the advantage of avoiding the complexity of searching the queue for a low priority packet to drop, which is done in our approach. This approach is the basis for the algorithm presented in [15]. The disadvantage of such algorithms is that they are less accurate (because they use thresholds for taking drop decisions) and they often lead to underutilisations of the queue. In [15], a version of such threshold-based algorithm which tries to alleviate this accuracy problem by using dynamic thresholds is presented. This algorithm uses not one threshold, but several different threshold values for each type of low priority frames from a multimedia stream. These thresholds are chosen in a dynamic way based on how the queue's fill level will be in the future (i.e., after several more frames are received, the *lookahead buffer*). In this algorithm, when a packet belonging to a low priority frame arrives at the router and the threshold for this kind of frame is surpassed (meaning that accepting this packet makes the lookahead buffer drop a future incoming high priority frame packet) this packet is dropped. This algorithm relies on the source sending in advance information about future incoming frames, to intermediate routers and on the fact that intermediate routers know in advance the arriving frame pattern from one second GOP.

Although our PDQMAMS does not use threshold values and drops a packet only when it's absolutely necessary (i.e., when the queue is full), thus being more accurate, complexity is added only when a packet is enqueued (like is depicted in figure 1) and, in our opinion, this complexity is comparable with the complexity of the algorithm presented in [15] when threshold values must be computed every time a packet arrives at the router. Also, the algorithm presented in [15] assumes it manages only a single multimedia flow and that available bandwidth changes on coarse-granularity time intervals, while our PDQMAMS handles several multimedia flows and achieves fairness between them.

7. Conclusions and future work

We have presented a queue management algorithm, namely Priority-Drop Queue Management Algorithm for Multimedia Streams (PDQMAMS) for helping multimedia streams inside the network. We showed that this queue management algorithm indeed provides support for multimedia streams while achieving a good degree of fairness among flows. We also sketched an improved version of PDQMAMS that tries to maintain the above qualities and also provide pro-active congestion avoidance mechanisms (mainly based on RED). As future plans we intend to implement and test this improved version of PDQMAMS under various conditions and prove its advantages.

References

[1] R. Steinmetz, K. Nahrstedt, *Multimedia: Computing, Communications and Applications*, Prentice Hall PTR, 1995.

[2] S.Floyd, K. Fall, *Promoting the Use of End-To-End congestion control in the Internet*, IEEE/ACM Transactions on Networking, August 1999.

[3] K. Ramakrishnan, S. Floyd, *A Proposal to add Explicit Congestion Notification (ECN) to IP*, RFC 2481, January 1999.

[4] B. Braden, D. Clark, J. Crowcroft, B. Davie, S. Deering, D. Estrin, S. Floyd, V. Jacobson, G. Minshall, C. Partridge, L. Peterson, K. Ramakrishnan, S. Shenker, J. Wroclawski, L. Zhang, *Recommendations on Queue Management and Congestion Avoidance in the Internet*, RFC 2309, April 1998.

[5] Nagle, J., *Congestion Control in IP/TCP*, RFC 896, January 1984.

[6] E. Kohler, M. Handley, S. Floyd, *Datagram Congestion Control Protocol (DCCP)*, Internet Draft, February 2004, http://www.icir.org/kohler/dcp/draft-ietf-dccp-spec-06.txt.

[7] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, *RTP: A Transport Protocol for Real-Time Applications*, RFC 3550, July 2003.

[8] S. Floyd, V. Jacobson, *Random Early Detection Gateways for Congestion Avoidance*, IEEE/ACM Transactions on Networking, Vol. 1, No. 4, pp. 397-413, August 1993.

[9] P. E. McKenney, *Stochastic Fairness Queuing*, Internetworking: Research and Experience, Vol. 2, pp. 113-131, 1991.

[10] A. Demers, S. Keshaw, S. Shenker, *Analysis and Simulation of a Fair Queuing Algorithm*, Journal of Internetworking: Research and Experience, 1, 1990, pp. 3-26.

[11] MPEG Standards, http://www.chiariglione.org/mpeg/index.htm.

[12] M. Ohlenroth, *Network-based Adaptation of Multimedia Content*, PhD thesis, Klagenfurt University, Austria, September 2003.

[13] B. Hubert, T. Graf, G. Maxwell, R. van Mook, M. van Oosterhout, P. B. Schroeder, J. Spaans and P. Larroy, *Linux Advanced Routing and Traffic Control*, August 2003, available at http://lartc.org/howto.

[14] S. Floyd, R. Gummadi, S. Shenker, *Adaptive RED: An Algorithm for Increasing the Robustness of RED's Active Queue Management*, august, 2001.

[15] A. Awad, M.W. McKinnon and R. Sivakumar, *MPFD: A Lookahead Based Buffer Management Scheme for MPEG-2 Video traffic*, Proceedings of the 8th IEEE International Symposium on Computers and Communication, 2003, (ISCC 2003).

Faculty of Mathematics and Computer Science, Babes-Bolyai University, Cluj-Napoca
E-mail address: {forest, florin, bufny, claudiu}@cs.ubbcluj.ro