Real-time Transport for Assured Forwarding: An Architecture for both Unicast and Multicast Applications

Ashraf Matrawy Ioannis Lambadaris
Broadband Networks Laboratory
Department of Systems and Computer Engineering
Carleton University, Ottawa, Canada

Abstract—**This paper presents a summary of our work on developing an architecture for transporting real-time traffic (MPEG4 video in this paper) in IP networks that provide service differentiation. We target our architecture at Assured Forwarding (AF) style services. This architecture assumes loss differentiation in the network and the network's ability to provide ECN messages to the sender. We did not consider policing/shaping at the edge routers. Rather we considered a more general case where marking and flow control are provided at the senders. For this network model, we developed a rate adaptation algorithm that can operate in both unicast and multicast applications with a minor modification.**

The simulation results presented in this paper represent the multicast case. The results show how the rate adaptation algorithm accommodates different receivers with different networking capabilities and provides them with different qualities by taking advantage of the queue management capabilities of the AF service. We also show the results of testing this architecture with different AF queuing mechanisms, namely RIO and WRED.

I. Introduction

In [1], [2], [3], problems with the end-to-end layered multicast approach are reported and discussed. These problems motivated a number of researchers to adopt network support in their work on multicast congestion control [4], [5], [6], [7]. Our motivation for this work is to develop a multicast congestion control scheme that relies on the IETF proposed Assured Forwarding (AF) [8] architecture. We considered AF because it helps us build a simple end-to-end architecture. It is also expected to be deployed soon in Internet routers as opposed to other proposed support mechanisms in the above references that require major changes in current router's functionality. For the sake of simpler experiments, we did not completely implement the IETF proposed AF service. Namely, we did not consider marking/policing at the edge routers and instead marked the packets at the sender. The architecture we propose here along with the rate adaptation algorithm operates equally well in both unicast and multicast applications with a minor modification. In this paper, we report results only from the multicast case as it proved more challenging than unicast.

The rest of this paper is organized as follows. The network model and the end-to-end architecture are presented in Sections II and III respectively. We discuss the rate adaptation algorithm in detail in Section IV. Simulation setup and results are presented in Section V. In Section VI we discuss some of the design issues and limitations. Section VII concludes the paper.

This research was funded in part by grants from: Communications and Information Technology Ontario (CITO), Natural Sciences and Engineering Research of Canada (NSERC), Mathematics of Information Technology and Complex Systems (MITACS), and Nortel Networks

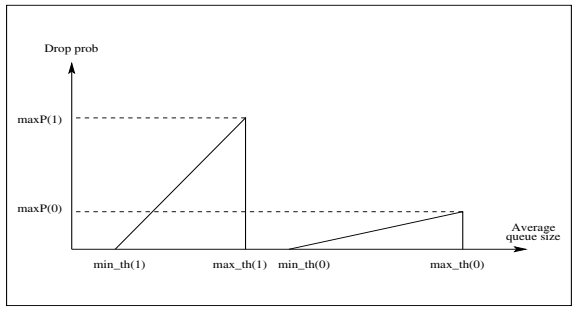


Fig. 1. Staggered parameter configuration of a two-priority RED queue

II. Network model

In this section, we briefly describe the network model we consider for our work. We consider IP networks that support priority-dropping as a means of providing different classes of services to its users. Priority-dropping is the process of packet dropping during congestion by the routers based on a priority level assigned to the packet at the sender or at an edge router. Routers achieve that by employing Active Queue Management (AQM) techniques that recognizes packet priorities and enqueue, dequeue, and drop packets based on this priority.

Random Early Detection (RED) is a widely used Active Queue Management technique [9]. Our network model assumes that routers support one of RED's extensions for service differentiation. We namely consider both RIO (RED with In/Out bits) [10] and WRED (Weighted RED) [11]. Our selection of RIO and WRED makes the architecture suitable for Assured Forwarding (AF) service [8]. Both RIO and WRED maintain a different set of parameters for each priority level and treat each of these levels as a different virtual queue. The difference is that WRED uses *one* average queue length to make dropping decisions while RIO uses *two*[1] average lengths to make dropping decisions. WRED calculates its average queue length based on the total number of packets in the queue. RIO maintains one average for the number of *in* (most important) packets and another average for the *out* packets. The number of RIO *out* packets can be calculated based on the total number of packets in the whole queue (coupled mode or RIO-C) or based on the number of *out* packets only (de-coupled mode or RIO-D).

In this paper, we present the results of using a two-priority queue model. However our architecture along with the simulation models we developed can support more priority levels

[1] two or more according to the number of priority levels

0-7803-7802-4/03/$17.00 © 2003 IEEE

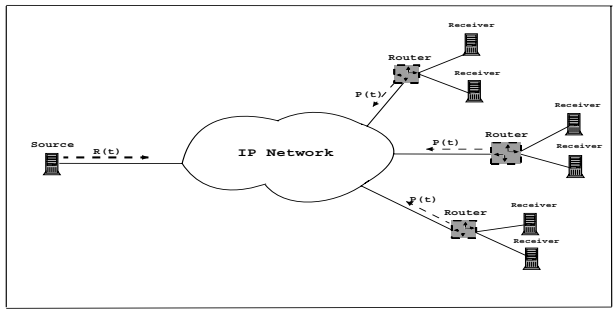


Fig. 2. End-to-end architecture

without any change. We use the staggered configuration of class parameters. This is setting $min_{th}(n-1) > max_{th}(n)$, where class $n-1$ is higher in priority than class n. Check Fig. 1 for the staggered parameters setting of a RED queue with two priorities. This configuration offers the maximum sheltering and isolation of classes with WRED. For RIO, in addition to staggered parameters, the choice of coupled versus de-coupled mode determines which class is more sheltered. The coupled mode (RIO-C) drops the lower priority class in favor of the higher priority class which is more protected. RIO-C also allows bandwidth borrowing among classes when the load is light. In the de-coupled mode (RIO-D), lower priority is more protected than RIO-C because in this case its average queue length is based solely on the number of lower priority packets in the queue.

We also assume that routers can send ECN (Explicit Congestion Notification) messages upstream to the sender with information about the router's congestion status. These messages are sent based on the specifications of BECN (Backward ECN) [12].

III. End-to-end architecture

We build an end-to-end architecture on top of the network model described in Section II. The results presented in this paper are based on testing the algorithm in the context of multicasting MPEG4 encoded real-time video. We send MPEG4 packets as *one* multicast group. These packets are marked with different priority levels by the rate adaptation algorithm at the sender. The algorithm decides how much is the total sending rate and the percentage of the packets marked with each priority level. These decisions are based on the congestion status reported to the sender by the different routers in the network. This congestion status is represented by the probability of the router sending a BECN message as described by $P_i(t)$ in Section IV. The algorithm always tries to set the rate for the high priority (most important) packets to accommodate the router with the worst congestion. That is the router with $Max\{P_1(t)\}$, where $i = 1$ refers to the highest priority layer. At lower priority levels, rates can be higher than receivers capacities as the packets will be dropped by the routers when they are not needed. An illustration of the end-to-end architecture is shown in Fig. 2.

For the unicast mode, the algorithm behaves differently at the lower priority by considering $Max\{P_i(t)\}$, as in the high priority level, because there is only one receiver to accommodate. In the rest of this paper, the multicast case is discussed as it is the more complex one. We will clarify the difference between unicast and multicast at any point where that difference exists and will summarize these differences in Section VI.

IV. The rate adaptation algorithm

A. The rate adaptation equation

Assume that MPEG4 traffic is generated at the source and divided into L layers marked with L different priorities.[2] Also we assume that this is the number of different priorities (and hence virtual queues) recognized at the routers. Let $R_i(t)$, $1 \leq i \leq L$, be the rate (in packets/sec) of layer i at the source at time t.

We also consider

$$P_i(t) = P_i^{Max}(t) + P_i^{Send}(t)\ P_i^{MinMax}(t)$$

where $P_i(t)$ is the probability that virtual queue i will generate a feedback message at time t. Also at time t we have

$$\begin{aligned} P_i^{Max}(t) &= \text{Prob}\{\text{QueueSize}(i) \geq max\} \\ P_i^{MinMax}(t) &= \text{Prob}\{min \leq \text{QueueSize}(i) \leq max\} \\ P_i^{Send}(t) &= \text{Prob}\{\text{Send feedback message} \\ &\quad \mid min \leq \text{QueueSize}(i) \leq max\} \end{aligned}$$

We derived $P_i(t)$ from the specification of BECN (Backward ECN) [12]. Considering the changes from *old* to *new* values of $R_i(t)$ and $P_i(t)$ in a small interval Δt, we use the following equation to update the rate $R_i(t)$:

$$R_i{}^{new} = R_i{}^{old}(1 - \alpha_i \Delta P_i), \quad 0 < \alpha_i < 1 \tag{1}$$

where

$$\Delta P_i = P_i{}^{new} - P_i{}^{old}$$

The rationale behind using this equation is to always change $R_i(t)$ in the opposite direction of change of $P_i(t)$ with a step α_i. We change α_i to control how much $R_i(t)$ changes in reaction to changing network conditions. $|\Delta P_i|$ can assume values between 0 and 1. At these extreme values, changes in R_i^{new} can be either no change at all (0%) or very high (100%). We select $\alpha_i = C_i\sqrt{|\Delta P_i|}$ where C_i is a constant for layer i and $0 < C_i < 1$. This sets the maximum rate change to $C_i R_i$ at layer i. The choice of square root function was motivated by our design goal of being able to react to very small changes of network conditions (when $|\Delta P_i| \leq 0.1$) as the square root of these small values is greater than the actual value ($\sqrt{x} > x;\ |x| \leq 1$). This helps to react to congestion while it is developing. Through simulations, we found that operating with values of C_i ranging from 0.05 to 0.25 keeps the system stable. A value of 0.1 at the high priority layer gave the best performance based on the criteria of (1) minimum packet loss ratio in the high priority layer and (2) matching of the source's rate to receivers' bandwidth capacities.

Equation (1) is subject to the constraints:

$$\begin{aligned} R_i^{min} \leq &\quad R_i &\quad \leq R_i^{max} \\ R^{min} \leq &\quad \textstyle\sum_{i=1}^{L} R_i &\quad \leq R^{max} \end{aligned}$$

[2] We use the term layers to describe the different priority levels of packets but we still send all of them in one multicast stream

where R_i^{min} and R_i^{max} are the value limits of the rate at layer i respectively and R^{min} and R^{max} are the limits for the total source rate. These values depend on the limitations imposed by the video encoder and on the outgoing link speed.

B. Round-Trip Time (RTT)

Routers send feedback messages to the sender with values of P_i^{new} that indicate the congestion status of the routers. The sender will evaluate the feedback from all routers every Δt and decide on a new rate R_i^{new}. The value of Δt will depend of the sender's estimation of the Round-Trip Time (RTT) from the routers that send the feedback information. We select the RTT value that corresponds to the router that has the worst situation at the high priority layer. That is the router with $Max(P_i^{new})$ in its feedback message.

C. Feedback suppression

The value of $P_i(t)$ will be estimated by routers and sent back to the sender. To reduce feedback, routers will send feedback messages with a probability instead of sending a feedback message for every packet that causes a problem. From simulations, sending 2% to 5% of the feedback messages kept feedback volume reasonable and in the same time kept the sender responsive to changes in network conditions.

D. Calculation of probabilities

The quantities $P_i^{Max}(t)$ and $P_i^{MinMax}(t)$ are calculated using real-time measurements from the network rather than being based on an analytical model. The reason for this is that in the general case where all kinds of traffic flows are coming into the routers queues, it is very hard to assume a certain model for the input traffic.

We bias the probability estimation by giving more weight to newer values to make the estimate a better representative of the current state of the network. Otherwise, after a long time of operation these values will converge to a constant value. We used the scheme used in [13] for measuring loss intervals. The probability is observed at each virtual queue i in k subsequent intervals and give these intervals different weights w_i, $1 \leq i \leq k$. To calculate $P_i(t)$ (whether $P_i^{Max}(t)$ or $P_i^{MinMax}(t)$) at the end of an interval m we use:

$$P_i(m) = \frac{\sum_{j=1}^{k} w_j \, P_{m-j}}{\sum_{j=1}^{k} w_j} \qquad (2)$$

Values of w_i are chosen so that recent probabilities have the same weight, while weights for older probabilities are smaller. The choice of a value for k is important. Large value of k will result in a smooth estimation of $P_i(m)$ while very large values of k will reduce the sender's responsiveness to changes in network conditions. For most or our simulations, we used $k = 10$, and $w = \{4, 4, 4, 4, 4, 2, 2, 2, 1, 1\}$. $P_i^{Send}(t)$ is calculated using the method in [9]. It depends on the average queue size and on the RED parameters.

E. Changing the equation parameters

The value of P_i^{new} will be changed every Δt. There are different criteria that we consider for changing these values:

- *At the highest priority layer, take the maximum:* In this case, we set the value of P_1^{new} to the maximum value received during Δt. This will result in accommodation of the router with the worst congestion situation. This is done subject to the constraint $R_i(t) \geq R_i^{min}$ to avoid the problem where a slow portion of the receivers drag the sending rate down dramatically[3].
- *At lower priority layers, take the minimum:* In this case, we set the value of P_i^{new} to the minimum value received during Δt. This results in maximizing $R_i(t)$ at layer i subject to $R_i(t) \leq R_i^{max}$. The reason behind this is to let receivers with extra available bandwidth utilize their links. Slow receivers will have these packets dropped by their routers.

F. The algorithm for changing R_i

REPEAT every RTT
  REPEAT for every layer i
    If Nofeedback
      increase R_i *by 1%*
    else If $(\Delta P_i > 0)$
      reduce R_i *using Eqn. (1)*
    else If $(\Delta P_i < 0)$
      If (i NOT highest priority layer)
        increase R_i *using Eqn. (1)*
    else If $(\Delta P_i = 0)$
      If $P_i^{new} > 0.9$
        reduce R_i *by 3%*
  END REPEAT for every layer i
END REPEAT every RTT

Note that the values of 1% and 3% are chosen to conservatively increase/decrease the rate as at this point we can not be sure exactly which direction ΔP_i is moving. Note also that the rate of the highest is increased only when there is no feedback to make sure that it is not increased beyond slow receivers capacities.

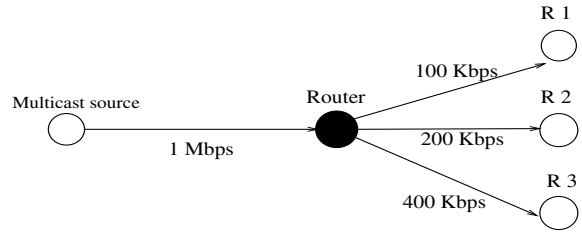
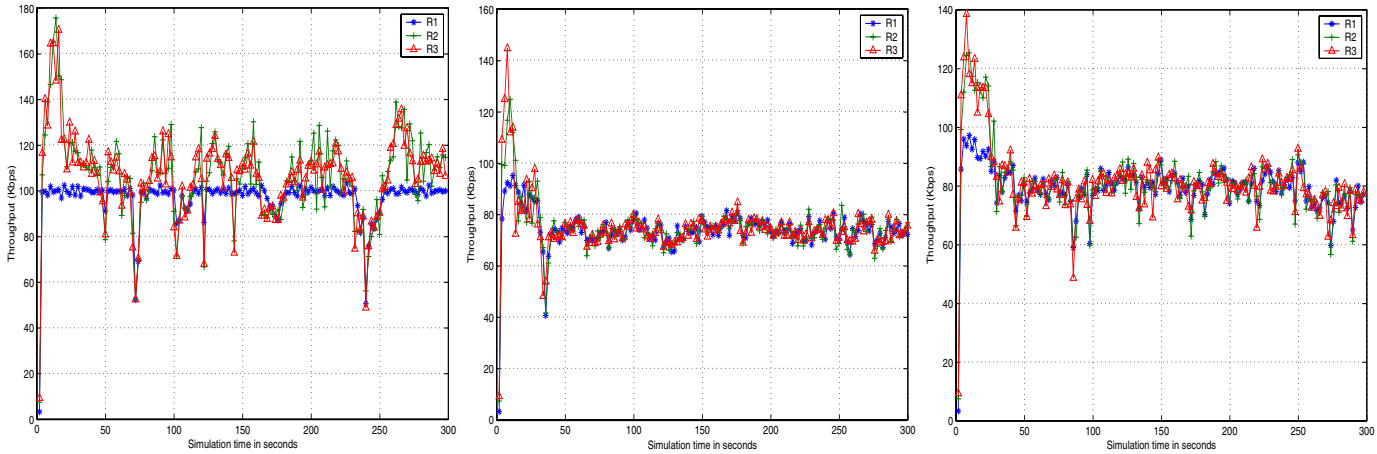


Fig. 3. Simulation setup

V. Simulation

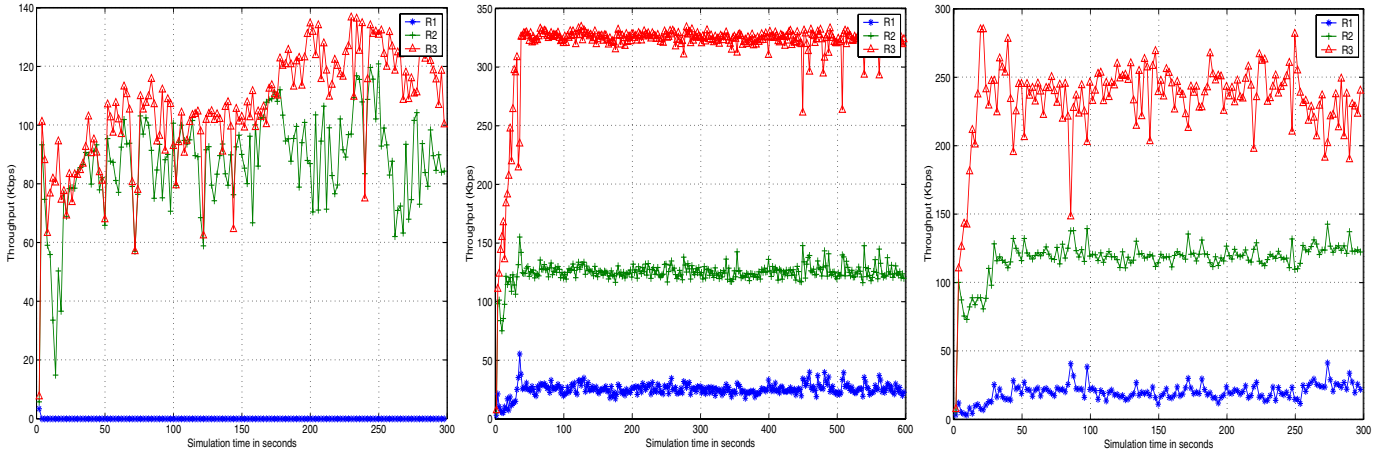
We carried out simulations using the network simulator *ns* [14] version 2.1b8a and simulated the topology in Fig. 3 with a two-priority queuing model. In the simulations, we compare the

[3]This is known as the "drop to zero" problem.



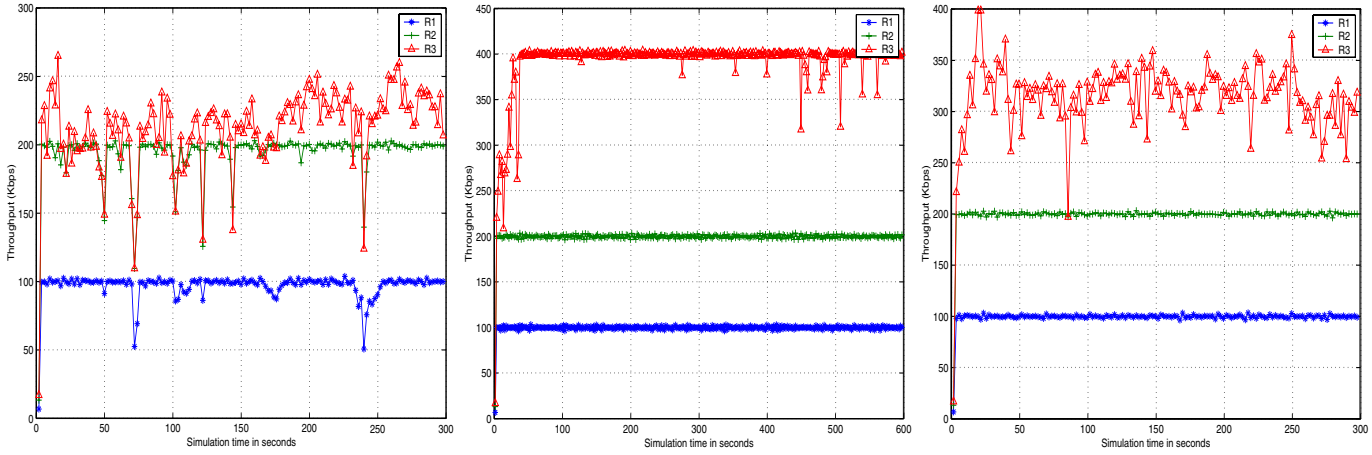
a) Using RIO-C b) Using RIO-D c) Using WRED

Fig. 4. High priority throughput



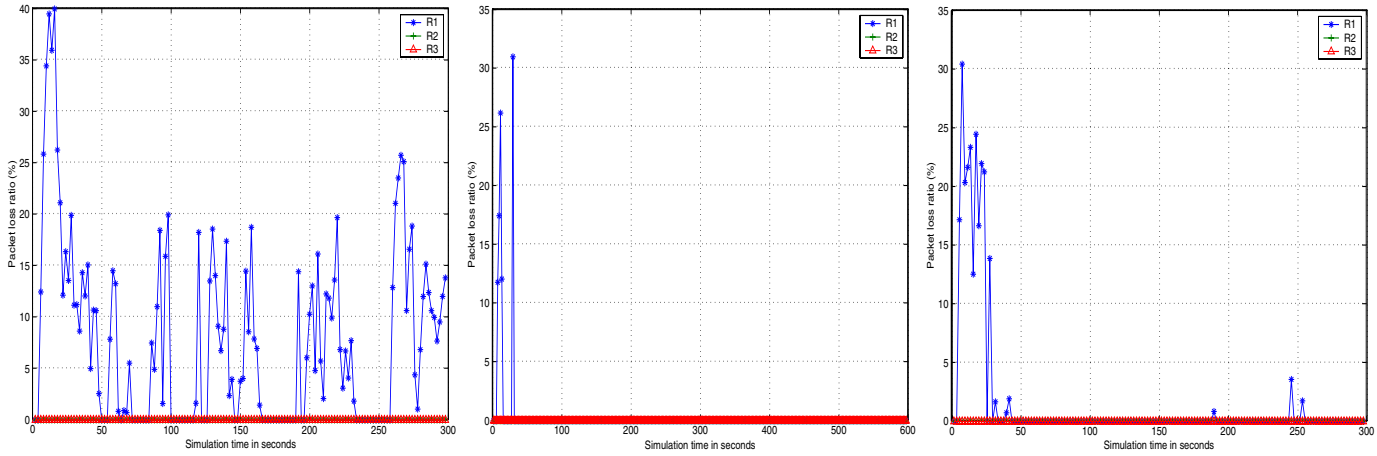
a) Using RIO-C b) Using RIO-D c) Using WRED

Fig. 5. Low priority throughput



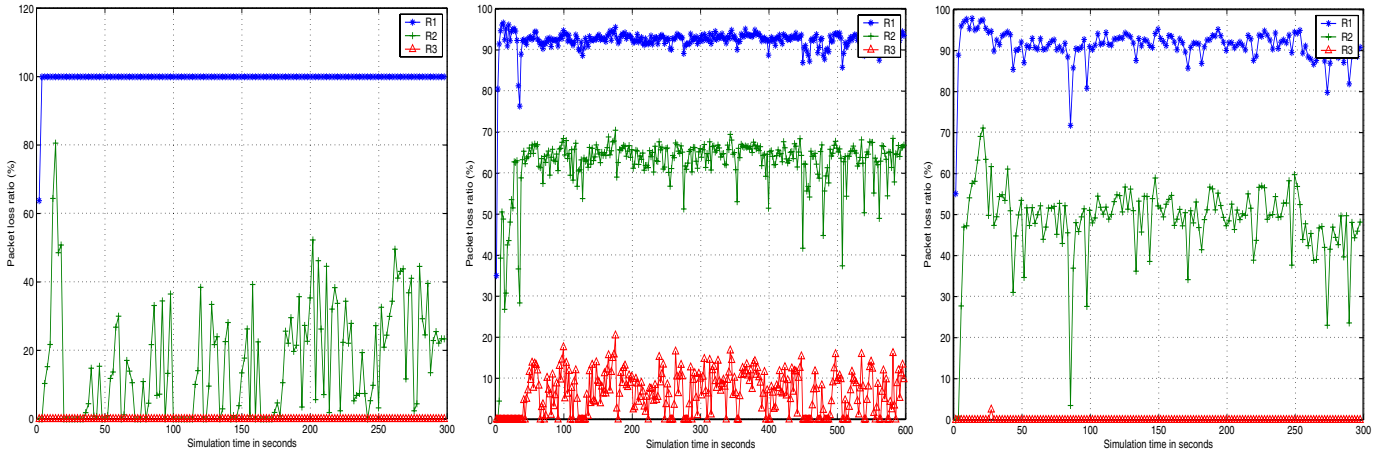
a) Using RIO-C b) Using RIO-D c) Using WRED

Fig. 6. Total throughput



a) Using RIO-C b) Using RIO-D c) Using WRED

Fig. 7. Packet loss in high priority



a) Using RIO-C b) Using RIO-D c) Using WRED

Fig. 8. Packet loss in low priority

performance of our proposal in the case of using RIO-C, RIO-D, or WRED (refer to Section II). Parameters for the two priority classes are the same in the three cases of RIO-C, RIO-D, and WRED. We used video traffic from an MPEG4 generator that we developed in [15].

The results we show are for simulations that are 300 seconds long. We use $C_i = 0.1$ for both priority levels $(i = 1, 2)$. We start the simulation with equal rates at the two priority levels $(R_1(t) = R_2(t))$. The goal of these simulation is to check whether the algorithm will match the rate of the highest priority layer with that of the slowest receiver and whether it will allow other receivers to get more traffic in the lower priority layer.

What we show here is the results of running our algorithm on the end-to-end architecture of Section III. We repeated the simulations with different random seeds. For each of the following metrics we compare the cases of RIO-C, RIO-D, and WRED.

- *Throughput of high priority layer:* This is shown in Fig. 4. We can see in the figure that the algorithm tries to bring down the rate for the high priority layer (for the three receivers) to a rate less than a 100Kbps which is the rate of the slowest receiver. The difference between parts a, b, and c of the figure is due to the way each queuing mechanism protects its highest priority level. Note that in all simulations we used the same set of parameters for the two priority levels. In RIO-C, this layer gets all of the 100Kbps, while in RIO-D and WRED it gets nearly 80Kbps because the lower priority layer is more protected with these two mechanisms. Also in this figure we can see that highest priority layer is more aggressive with RIO-C. Both RIO-D and WRED offer more isolation and protection and RIO-D offers best stability at this layer.
- *Throughput of lower priority layer:* This is shown in Fig. 5. In this layer, the algorithm allows receivers with higher capacities to better utilize their bandwidth. The best performance comes with RIO-D again because of its isolation of classes. It allowed each receiver its maximum share at this layer. Although this is not the best option of R1 as it does not make use of the low priority in this case with a high loss rate as seen in Fig. 8. WRED is better than RIO-C but still not as good as RIO-D. The reason

of the weak performance of RIO-C at this layer is RIO-C protects the high priority layer. This protection lets the high priority layer be aggressive and eventually brings down the rate down for the lower priority layer.

- *Total throughput:* Fig. 6 shows the summation of the throughput at both layers for each receiver. It is clear that maximum utilization is achieved with RIO-D. This however should not be taken as an absolute metric for judging the performance as we will comment later.
- *Packet loss at high priority layer:* In Fig. 7, we can see that the loss ratio at the high layer for R2 and R3 is zero all the time. However, for R1, while the loss ratio is high only in transient phase in the beginning of the simulations for RIO-D and WRED, it has high values at different times during the simulation for RIO-C. The reason is what we mentioned earlier about RIO-C protecting the high priority layer and allowing this layer to be aggressive. This results in overshoots of the rate at this layer causing these bursts high losses.
- *Packet loss at low priority layer:* This is shown in Fig. 8. It can be seen that RIO-C has the lowest loss rate at this layer for the three receivers because it does not allow the rate at this layer to go as high as the cases of RIO-D and WRED.

VI. Design issues and limitations

In this section, we present some notes on our design and point-out the limitations of our approach.

- The values of R_i^{min}, R_i^{max}, R^{min}, and R^{max} are not enforced in our simulations. So the ratio of R_0 to R_1 may not be practical in our results but we do that on purpose in our simulations to test the control mechanism we propose without imposing limitations on the rates. Applying these limits after the testing we present in this paper will result in better and more practical performance of our algorithm.
- The limitation of this approach is that multicast receivers should at least have their bandwidth grater than R_1^{min} (minimum rate at the high priority layer).
- To accommodate higher heterogeneity of receivers bandwidth, a higher number of layers should be used. We note here that neither the architecture nor the rate adaptation algorithm needs to be changed in order to support a higher number of layers.
- We drop low priority *multicast* packets at the input interface of the router if the virtual queue (of the output interface) it is supposed to be forwarded to is experiencing a high loss rate. This keeps these queues operating with smaller average lengths and allows non-multicast traffic to get a better share of the queues at the routers' output interfaces. Note that the packet loss ratio we show in these graphs are the total of those dropped at the input interface and in the virtual queues.
- Our work is also applicable to the *unicast* case although we do not present these results here. The difference in the unicast mode is the consideration of $max(P_i^{new})$ at both layers so that the rate at each layer meets what's available at the only receiver of the unicast flow. Also in unicast, we do not apply that dropping at the input interface described above.
- The architecture assumes that there will be loss at lower layer and hence we recommend the use of some Forward Error Correction (FEC) mechanism at the lower layer(s).

VII. Conclusion

We have presented an architecture and a rate adaptation algorithm for real-time video transport in Assured Forwarding networks. It can operate in either unicast or multicast modes and we presented the simulation results for the multicast case. We showed how it enables users with different bandwidth capabilities to receive the same video multicast in different qualities. This differentiation is based on encoding the video into two levels of priorities. One important basic layer and one (or more) enhancement layer(s). We can see how the algorithm is always trying to accommodate the slowest receiver at the high priority layer and how it allows increasing the rate at the lower priority layer.

The comparison between the three queuing mechanisms we considered reveals that selecting one of them is not a trivial task. While RIO-D and WRED result in better utilization of bandwidth, they also result in high loss rate in the lower layer to a level that is not desirable sometimes (check loss rate of R2 at lower layer). On the other hand, RIO-C offers different qualities with lower loss rates at the expense of less bandwidth utilization.

References

[1] A. Legout and E. Biersack, "Pathological Behaviors for RLM and RLC," *In Proc. of NOSSDAV*, 2000.

[2] R. Gopalakrishnan et al., "Stability and Fairness Issues in Layered Multicast," *In Proc. of NOSSDAV*, 1999.

[3] A. Matrawy, I. Lambadaris, and C. Huang, "On Layered Video Fairness on IP Networks," *In Proc. of IEEE GLOBECOM*, 2001.

[4] A. Clerget, "TUF: A Tag-based UDP Multicast Flow Control Protocol," *INRIA Technical-report 3728*, July 1999.

[5] M. Luby, L. Vicisano, and T. Speakman, "Heterogeneous Multicast Congestion Control based on Router Packet Filtering," *Presentation at RMRG meeting, Pisa, Italy*, June 1999.

[6] K. Nakauchi, H. Morikawa, and T. Aoyama, "A Network-Supported Approach to Layered Multicast," *In Proc. of IEEE ICC*, 2001.

[7] Z. Zhang and V. O. K. Li, "Router-Assisted Layered Multicast," *In Proc. of IEEE ICC*, 2002.

[8] J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski, "Assured Forwarding PHB Group," *RFC 2597*, June 1999.

[9] S. Floyd and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance," *IEEE/ACM Trans. on Networking*, pp. 397–413, August 1993.

[10] D. Clark and W. Fang, "Explicit Allocation of Best Effort Packet Delivery Service," *IEEE/ACM Trans. on Networking*, vol. 6, no. 4, pp. 362–373, August 1998.

[11] Technical Specification from Cisco, "Distributed Weighted Random Early Detection," *http://www.cisco.com/univercd/cc/td/doc/product/software/ios111/cc111/wred.pdf*.

[12] J. Hadi Salim, B. Nandy, and N. Seddigh, "A Proposal for Backward ECN for the Internet Protocol (IPv4/IPv6)," *Internet Draft, draft-salim-jhsbnns-ecn-00.txt, available via http://www.sce.carleton.ca/ nseddigh/publications.html*.

[13] J. Widmer and M. Handley, "Extending Equation-based Congestion Control to Multicast Applications," *In Proc. of SIGCOMM*, 2001.

[14] S. McCanne and S. Floyd, "ns Network Simulator," *Available via http://www.isi.edu/nsnam/ns/*.

[15] A. Matrawy, I. Lambadaris, and C. Huang, "MPEG4 Traffic Modeling using The Transform Expand Sample Methodology," *In Proc. of 4th IEEE International Workshop on Networked Appliances*, January 2002.