A Scalable Feedback-Based UPC-Parameters Renegotiation Scheme

Safiullah Faizullah
Hewlett-Packard, Piscataway, NJ, USA
Email: safi.research@gmail.com

Arshad M. Shaikh
Isra University, Hyderabad, Pakistan
Email: amshaikh@hotmail.com

***Abstract*—In QoS-enabled network, for example ATM, connections which exhibit rapidly changing traffic characteristics, renegotiating the bandwidth requirements becomes inevitable. The sender initiates renegotiation whenever bandwidth requirements change. If the newly requested bandwidth is not granted, the sender keeps on invoking the follow-up renegotiations with some frequency. Polling the network too often results in huge overhead traffic and a decrease in the throughput. While invoking follow-ups too infrequently results in the under-utilization of the network. Predicting an optimal follow-up rate is a hard problem. We have earlier proposed a Feedback Based UPC-Parameters Renegotiation Protocol for ATM networks which resolves this problem. In in this paper with simulation results we show that the proposed protocol is scalable and reliable.**

***Index Terms*—scalable, QoS, UPC, UPC-parameters, renegotiation, feedback-based, ATM, VBR, reliable, advertisement**

I. Introduction and Background

In today's connected world and round the clock usage of network and computer-based multimedia applications, the traffic generated by such applications require real bandwidth-on-demand from a QoS-enabled network, such as ATM. A multimedia connection frequently changes its traffic profile during the course of a session. Particularly, we observe this for applications using VBR video in which instantaneous bit-rate varies widely with seen content and encoder's state. Additionally, the user of a software based multimedia application may want to resize a video window and consequently request a larger or smaller image resolution; fast-forward, rewind, pause or jog can be requested from a video server; or the user may suddenly start to browse an image database, etc. QoS-enabled network such as ATM networks are intended to provide support for such video and multimedia applications via the so called VBR service class. VBR [1] service class supports connections with a static traffic profile pre-established at call setup. This traffic profile is defined through a set of traffic descriptors called UPC which includes *peak-rate*(λ_p), *burst-length*(BL) and *sustained-rate*(λ_s). The network admits a VBR connection based on its declared UPC. Once the connection is established, it is expected that the terminal device will comply with the declared UPC; the network may enforce the declared UPC using a leaky-bucket based network policing. A source complies with its UPC if it produces upto BL consecutive cells with inter-cell spacing of upto $1/\lambda_p$ and, until the cell-counter is cleared, all the remaining cells arrive at inter-cell spacing of upto $1/\lambda_s$. Each time a cell is admitted the leaky bucket counter is incremented. The leaky-bucket counter is constantly decremented at the rate of λ_s. Violating cells are usually marked by the policer and discarded at the switch node where it encounters congestion. To avoid cells being discarded, the source must regulate its traffic to fit to the initially declared UPC. This may be done at the source with a combination of a leaky-bucket traffic shaper, which mirrors the policing mechanism, and source rate control.

Dynamic bandwidth renegotiations are becoming popular alternatives to this static scheme. In these techniques applications/encoders initiate renegotiation every time bandwidth requirements change. If the newly requested bandwidth is not fully granted, the sender keeps on invoking the follow-up renegotiations with some frequency. While these techniques have obvious performance improvement [2]-[6] over their static counterparts, they have inherent disadvantages. Invoking follow-up renegotiations too infrequently results in under-utilization of the network. While polling the network too often induces a huge overhead traffic and results in a decrease in the network throughput if the network is congested. Predicting an optimal rate for follow-up renegotiations is a hard problem.

A number of solutions to the problem of dynamic resource allocation have recently been proposed in the literature [2]-[11]. We have proposed a renegotiation protocol based on network feedback mechanism. We have reviewed and highlighted the shortcomings of the prominent strategies in our earlier work—Thus, motivating our original work presented in [7] and we now

Manuscript received October 10, 2013; revised December 10, 2013

doi: 10.12720/ijsps.1.2.302-306

study our scheme for scalability and present the results of simulations conducted in this paper.

In our approach the sender invokes a follow-up renegotiation only when the network signals it for the availability of some bandwidth. Furthermore, the sender does not initiate a new renegotiation (for more bandwidth) if its previous request was not fully granted. However, any request to decrease the UPC requirements is renegotiated immediately. Simulation experiments were conducted to evaluate the strategy. Results show that the proposed method has significant performance benefits over its polling-based counterparts. In addition, in this paper utilizing simulation results we show that the proposed protocol is scalable and reliable.

The rest of the paper is organized as follows: Section II briefly describes our feedback-based approach. Implementation issues are discussed in Section III. Section IV presents experimental methodology and simulation results. Finally, Section V gives some concluding remarks.

II. Feedback Based UPC Renegotiation

In this section, we will outline our strategy which relies on feedback from the network instead of polling the network continuously. The sender needs not necessarily initiate renegotiation if the UPC requirements change. It invokes renegotiation for more bandwidth only if the previous demand was not unfulfilled. On the other hand, any request to decrease the UPC requirements are renegotiated immediately. Also, if the newly requested bandwidth is not granted, the sender does not disturb the network by invoking follow-up renegotiations. The network keeps track of the unfulfilled requests and informs the sender whenever some bandwidth is available.

The proposed algorithm is described with the help of Fig. 1. Whenever some bandwidth is released on a link (e.g., on DE), the switch immediately before the link (e.g., D) informs all the senders who may potentially use this link (e.g., L, V, W, X, H, T, A, U), that some bandwidth is now available in the network. Senders of active connections that needed more bandwidth but were denied earlier can invoke renegotiation at this point (after a random wait to avoid simultaneous initiation of requests). Note that when some bandwidth is released on a VP, each switch on that VP will send feedback to all the potential senders.

A disadvantage of this naive method is that the newly available bandwidth might not be on the VP on which the sender needs more bandwidth. So, this feedback results in huge rather useless traffic not only due to the delivery of feedback information to unconcerned senders but also due to the unsatisfiable renegotiation requests from these senders. An immediate improvement over the above approach is that the switch should advertise the availability of more bandwidth only to those senders which have active connections (VC's) via that link (for example A, L in Fig. 1). In this approach, the switch advertises only to those senders whose connections are bottlenecked at the next link. We maintain the bottleneck information by keeping a flag "NextLinkBottleneck" for every VC passing through a switch. Note that we flag a link as bottleneck for a connection only if the request for more bandwidth is not fully granted. Hence, this mechanism will give feedback only to those who are interested in getting more bandwidth; which is a plus point. During CAC as well as during re-negotiations, the flags are updated appropriately. On receiving feedback from the network, the senders wait for a random period of time before initiating follow-up renegotiations. This provides fairness and decreases the chance of simultaneous initiations.

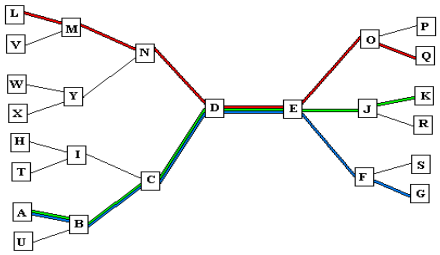

Figure 1. Sample network

III. Implementation Issues

In this section several issues are discussed that relate to the implementation details. These issues were faced during the simulation stage of this work.

Signaling: Whenever an application needs to renegotiate the bandwidth, it creates a Reneg-cell specifying its requested bandwidth. The cell travels through all the links on the VP and gathers information about the minimum available bandwidth in the forward direction. On the backward trip it reserves this amount on all links. Finally, it passes this information to the application. Also, the links which become the new bottleneck are flagged during this round-trip. The application then adjusts its bit-rate accordingly.

VBR UPC Parameters: The simulator we have used for simulations uses a different set of UPC parameters which comprises of Peak bit-rate (P), Mean burst length(BL) and Mean interval between bursts(MIB). The difference is insignificant because one set can be derived from the other.

We calculate the Average bit-rate as follows:
Average bit-rate = P * BL / MIB.

We use this Average bit-rate for renegotiations and bandwidth allocation.

Reneg-cell Format: For our Reneg-cell, we adopt a format similar to that of the ABR's RM-cell with the following modifications. Here PTI is taken to be "111" which was unused previously. The data portion of the Reneg-cell contains three fields: *Direction*, *RequestedBW* and *AllocatedBW*. If Direction has the value "1", the cell is a forward going Reneg-cell. Otherwise it is a backward going Reneg-cell. The application specifies the amount of required bandwidth in RequestedBW field which remains unchanged throughout the round trip. The AllocatedBW field keeps track of the minimum bandwidth (≥0) on the VP.

Feedback Reneg-cell: A Feedback Reneg-cell is distinguished from a regular Reneg-cell by the contents of its AllocatedBW field which in case of Feedback contains a negative value.

Bandwidth Locking: On the forward direction of Reneg-cell, the switching elements lock the min{requested, available}amount of bandwidth. On its way back, when the bandwidth to be allocated is determined, each switch allocates the AllocatedBW amount of bandwidth from the locked bandwidth to this VC and releases the lock.

Wait and Go: While the Bandwidth locking avoids the chance of allocating the same piece of bandwidth to different VC's. The adverse effect is that a request (forward Reneg-cell) normally locks more amount of bandwidth than what is actually allocated (because of the bottlenecks somewhere ahead), so the next request will see lesser amount of bandwidth which is not necessarily true. Therefore we introduce "wait and go" mechanism for the forward Reneg-cells as follows: if the amount of available bandwidth is less than the AllocatedBW and there is some bandwidth under lock (which could be released by a previous Reneg-cell on its backward journey) then the switch puts this cell on hold until either AllocatedBW amount of bandwidth becomes available or there is no more locked bandwidth. In case there is no more locked bandwidth, the Reneg-cell's AllocatedBW is reduced to whatever amount is available, and is locked under this cell and the cell proceeds.

Priority to Renegotiation Cells: To better utilize the network resources, feedback cells are sent on priority basis. To implement that we divide the output queue into two sub queues at the scheduler of each switching element - one for priority cells and the other for regular data cells. The size of the output queue is considered to be the sum of the sizes of these two queues. The switch serves the prioritized queue first, before the data queue. If the output queue is reached to its limit, the scheduler will drop cells from the data queue before the prioritized queue.

IV. Experimental Methodology and Results

We have incorporated both the traditional renegotiation algorithm and our feedback based renegotiation algorithm into an off-the-shelve simulator that we will simply refer to as NIST in this paper. We needed to add some missing functionalities to the simulator. Specially, we converted the static CAC module to handle dynamic call admission and termination. Further, we modified the VBR application to change its UPC requirements over time.

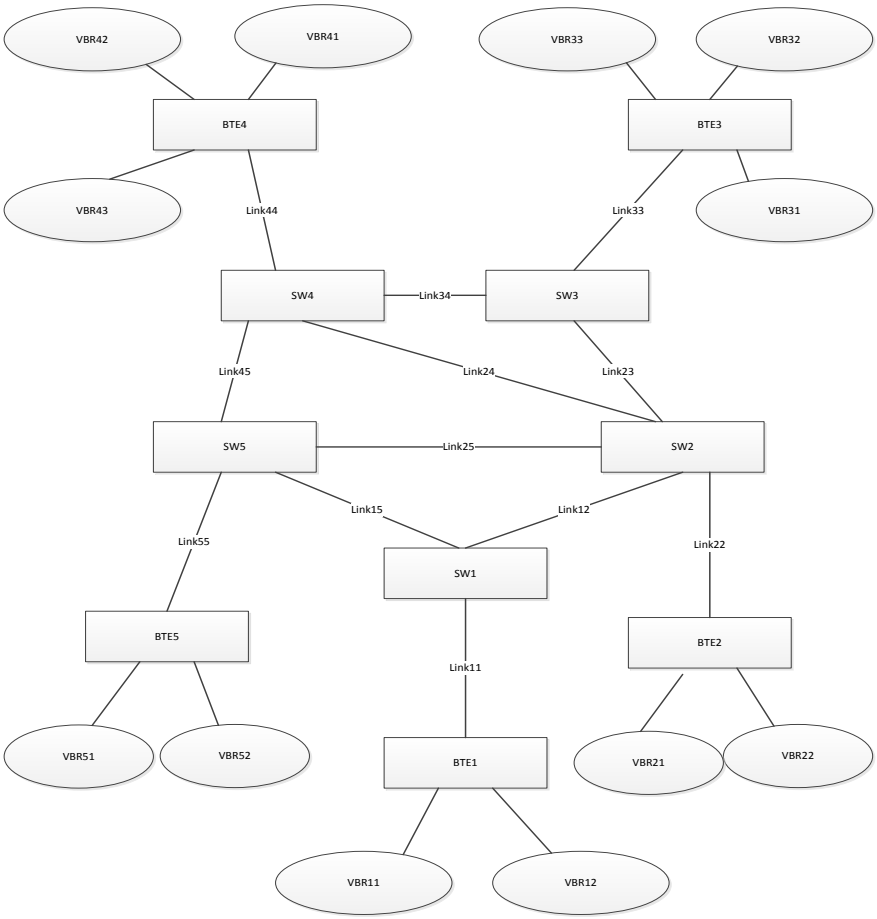

Figure 2. Results the proposed scheme vs. others.

In order to compare repeated renegotiation approach versus proposed network feedback mechanism, we conducted large number of experiments with varying follow-up rates, and at least two experiments with the

proposed technique and studied the utilization/throughput of the network. Fig. 2 shows one of the model networks used in the simulations. For scalability, we used a variation of network topologies. Table I shows a sample variation in UPC parameter requirements; these were changed for each run of the simulation. Note that the sample networks are kept congested since each source's average bit-rate is quite larger than the capacity of the links which is fixed as 50Mbps for all links in these experiments. The network configuration varied from simple network (Fig. 1 and Fig. 2) to more complex/campus like network (as shown in Fig. 4 where the internal network shown as cloud varied from simulation to simulation.) We can see in the simulation results (Fig. 3) that the network remained under- utilized for low follow-up rate. For higher follow-up rates the renegotiation overhead is so much that it reduces the throughput drastically. On the other hand, the network throughput using the proposed feedback based technique is clearly better than the repeated follow-up scheme. Fig. 5 and Fig. 6 show that the proposed scheme is highly scalable as the renegotiations overhead is relatively low and remain so under varying congestions for various network topologies.

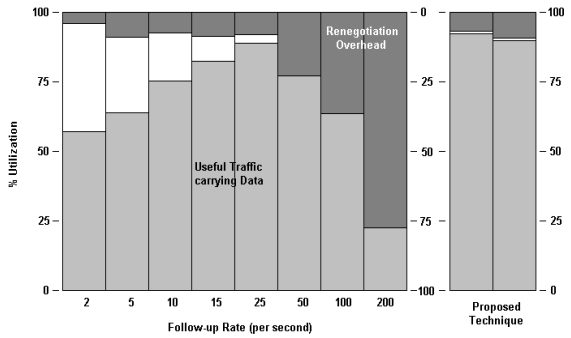

Figure 3. Results- the proposed scheme vs. others.

TABLE I. SAMPLE UPC VARIATION IN SIMULATED NETWORKS

Sender	Receiver	Peak Rate (Mbps)	Burst Length (uSecs)	Inter-burst Interval (mSecs)
vbr11	vbr32	80±30	400±10%	0.8±10%
vbr12	vbr22	65±15	800±10%	1.4±10%
vbr21	vbr12	140±40	500±10%	0.7±10%
vbr22	none	0	0	0
vbr31	vbr21	80±30	400±10%	0.8±10%
vbr32	none	85±20	500±10%	0.8±10%
vbr33	vbr43	90±25	500±10%	1.0±10%
vbr41	vbr33	100±30	600±10%	0.9±10%
vbr42	vbr22	70±20	900±10%	2.0±10%
vbr43	none	0	0	0
vbr51	vbr11	55±40	700±10%	1.5±10%
vbr52	vbr42	80±15	300±10%	0.8±10%

V. CONCLUDING REMARKS AND FUTURE WORK

A feedback driven scheme for UPC parameter renegotiation has previously been introduced. Simulation experiments were conducted to evaluate the strategy. Results show that the proposed method has significant performance benefits over its polling-based counterparts. Additional simulations conducted under various network topologies and various load support and highlight that the protocol is scalable and reliable. Our future research goal is to study the merit of the protocol further investigated when feedback is delivered by a multicast instead of multiple unicasts. It is also of interest to extend this work to wireless networks.

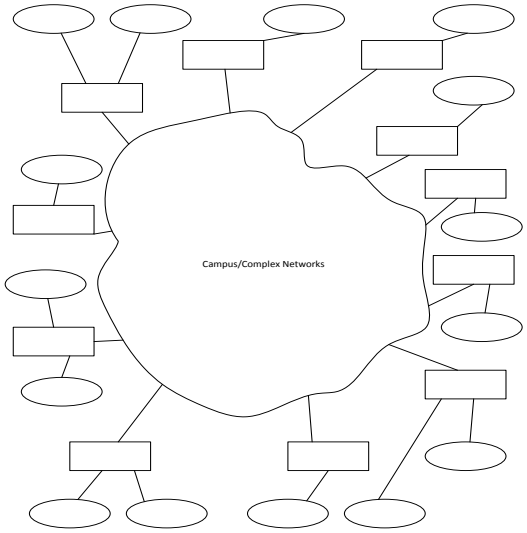

Figure 4. Results- the proposed scheme vs. others.

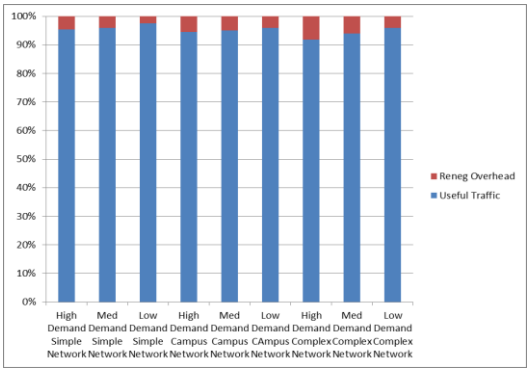

Figure 5. Results- the proposed scheme vs. others.

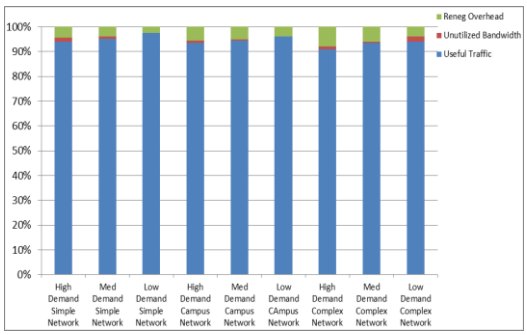

Figure 6. Results- scalability of the proposed scheme

APPENDIX A ALGORITHMS

/*Algorithm – VBR*/

VBRApp:- InitReneg

```
if ShortageInBW>0
    cell.PIT=111;
    cell.Direction=1;
    cell.RequestedBW=ShortageInBW;
    cell.AllocatedBW= cell.RequestedBW;
    send(cell);
endif
return;
```

VBRApp:- ReceiveBackwardRenegCell

```
AverageBitRate= AverageBitRate+cell.AllocatedBW;
ShortageInBW= ShortageInBW- cell.AllocatedBW;
if ShortageInBW<0 then
    ShortageInBW=0;
endif
return;
```

VBRApp:- ReceiveFeedbackRenegCell

```
wait(random);
InitReneg();
return;
```

/*Algorithm – Switch*/

Switch:-BackwardRenegCell_Inc_BW

```
if  cell.AllocatedBW= LockedBW[cell.VciVpi] then
    NextLinkBottleNeck[cell.VciVpi]=TRUE;
else
    NextLinkBottleNeck[cell.VciVpi]=FALSE;
endif
NextLink.LockedBW=NextLink.LockedBW -
                        LockedBW[cell.VciVpi];
LockedBW[cell.VciVpi]=0;
NextLink.AvailBW=NextLink.AvailBW -
                       cell.AllocatedBW;
process Reneg-cells on hold();
Forward_cell_to_next_hop(cell);
return;
```

Switch:-ForwardRenegCell_Inc_BW

```
If  cell.AllocatedBW>NextLink.AvailBW -
                        NextLink.LockedBW  then
    if  NextLink.LockedBW>0 then
        put cell on hold queue;
        return;
    else   /* if NextLink.LockedBW=0 */
        cell.AllocatedBW= NextLink.AvailBW;
    endif
endif
LockedBW[cell.VciVpi]=cell.AllocatedBW;
NextLink.LockedBW=NextLink.LockedBW+cell.Allocat
edBW;
Forward_cell_to_next_hop(cell);
return;
```

Switch:-ForwardRenegCell_Dec_BW

```
NextLink.AvailBW= NextLink.AvailBW+
                       (-cell.RequestedBW);
NextLinkBottleNeck[cell.VciVpi]=FALSE;
process_Reneg_cells_on_hold();
send_feedback_to_bottlenecked_connections();
Forward_cell_to_next_hop(cell);
return;
```

Switch:-BackwardFeedbackRenegCell

```
Forward_cell_to_next_hop(cell);
return;
```

REFERENCES

[1] MPEG-4 Standards ISO/IEC JTC1/SC29/WG11 N2201, '98

[2] R. Ulrich and P. Kritzinger, "Managing ABR capacity in reservation-based slotted networks," *Tech. Report, ICSI*, 1996.

[3] K. Nahrstedt and J. M. Smith, "Revision of QoS guarantees at the application network interface," *Technical Report*, University of Pennsylvania, Distributed Systems Lab, 1994.

[4] S. Chong, S. Li, and J. Ghosh, "Dynamic bandwidth allocation for efficient transport of real-time VBR video over ATM," in *Proc. IEEE INFOCOM Conf.*, 1994, pp. 81-90.

[5] V. Tyagi, V. Tyagi, and S. Sharma, "Traffic shaping by statistical characterization in ATM networks," *Intl. Journal of Essential Science*, vol. 5, no. 1, pp. 78-82, 2011.

[6] D. J. Reininger, D. Raychaudhuri, and J. Y. Hui, "Bandwidth renegotiation for VBR video over ATM networks," *IEEE Journal on Selected Areas in Communications*, vol. 14, no. 6, September 2006, pp. 1076-1086

[7] S. Faizullah and A. Shaikh, "A feedback-based UPC-parameters renegotiation strategy," in *Proc. Internl Conf. on Parallel and Distributed Processing Techniques and Application*, Las Vegas, NV, June 28-July 1, 1999, pp. 937-943.

[8] D. Reininger, G. Ramamurthy, and D. Raychaudhuri, "VBR MPEG video coding with dynamic bandwidth renegotiation," *IEEE International Conference on Communications, 95-R-008 (AP)*

[9] B. Mark and G. Ramamurthy, "Real-time estimation and dynamic renegotiation of UPC parameters for arbitrary traffic sources in ATM networks," *IEEE/ACM Transactions on Networking (TON)*, vol. 6, no. 6, pp. 811 – 827, Dec. 1998.

[10] M. T. Andrade and A. P. Alves, "Experiments with dynamic multiplexing and UPC renegotiation for video over ATM," in *Proc. Networking '00/Proc. IFIP-TC6 / EU Com. Intl Conf. on Broadband Com., High Perf. Networking, and Perf of Com. Networks*, pp. 895-907

[11] J. Alins, J. Pegueroles, J. Mata, and L. J. de la Cruz Llopis, "Two-level renegotiated constant bit-rate algorithm (R-CBR) for stored video services over QoS networks," in *Proc. IASTED International Conference on Communications Systems and Networks*, 2002.

Safi Faizullah received his Ph.D. in Computer Science from Rutgers University, New Brunswick, New Jersey, USA in 2002. He also received MS and M. Phil. Degrees in Computer Science from Rutgers University, New Brunswick, New Jersey, USA in 2000 and 2001, respectively. Dr. Faizullah also earned BS and MS degrees in Information and Computer Science from KFUPM, Dhahran, KSA in 1991 and 1994, respectively. His research interests are in computer networks, mobile computing, wireless networks, distributed and enterprise systems. He has authored over a 20 journals and conference papers. He works for Hewlett-Packard and is Visiting Scholar/Adjunct Professor of Computer Science at Rutgers University. Dr. Faizullah is a member of IEEE, PMI and ACM.

Arshad M. Shaikh received his MS degree in Computer Science from Rutgers University, New Brunswick, New Jersey, USA in 2002. He also received MS and BS from National University, Karachi, Pakistan. His research interests are in programing languages and computer networks. He has authored several conference papers. Mr. Shaikh worked for National University, Karachi, Virtual University, and Isra University, Hyderabad, Pakistan as lecturer.