Proceedings of the 5th WSEAS International Conference on Applied Computer Science, Hangzhou, China, April 16-18, 2006 (pp153-158)

Postgate: QoS-aware Bandwidth Management for Last-mile ADSL Broadband Services

MING-I HSIEH and ERIC HSIAO-KUANG WU
Department of Computer Science and Information Engineering,
National Central University,
No.300, Jhongda Rd., Jhongli City, Taoyuan County 32001,
TAIWAN, R.O.C.

Abstract: ADSL has been one of major last-mile network solutions today. However, current ISPs only offer best-effort broadband serices. It would be a challenge to regulate those traffic by subscriber and ensure the quality of service for each flows to conform its requirements. This letter focuses on such a problem and proposes a novel bandwidth management scheme, Postgate.

Key–Words - ADSL, QoS, Fair Queueing

1 Introduction

Recent research[5] has shown that in most cases the bottleneck is at last-mile network. ADSL (Asymmetric Digital Subscriber Line), one of major last-mile broadband solutions today provides a much faster network access than the traditional dialup network. Nevertheless, the quality of service for current Internet is not growing rapidly as their bandwidth improvement. Current ISPs still only offer the best-effort services for ADSL subscribers. In such a case, while a ADSL subscriber is downloading some bulk files, their real-time applications, e.g. VoIP, would not get a pretty quality due to the vast queueing delay from the queued packets of those bulk files at ATU-C. Notice, the QoS guarantee for ADSL uplink is easily guaranteed by applying the traditional Diff-Serv schemes with some well-known fair queueing algorithms.[1][2][3][4] Thus, this paper only focuses on the QoS guaranteed for ADSL downstream.

First, let us consider a general ADSL last-mile network topology like Fig.1 , an ADSL link is shared by multiple flows from one or more users. (Here, for simplicity we assume that one user only offers a flow in the same time.) In such a case, if flow A is a bulk file transmission traffic, e.g. FTP, some packets of flow A would be queued at bottleneck, ATU-C, to ensure the bandwidth utilization. At the same time, let flow B be a real-time flow, e.g. VoIP, to share the downstream bandwidth of ADSL with flow A. When some packets of flows A is queued at ATU-C, if a packet of flow B arrives at ATU-C, such a packet would get a vast queueing delay incurred from the queued packets of flows A. In such a case, the quality for flow B, e.g. jitter and queueing delay, would be poor.

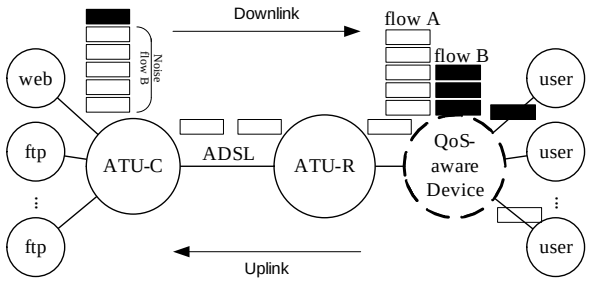

Figure 1: QoS-aware device and network topology

Because the ADSL subscriber never gets the permission to control the queue of ATU-C directly, the possible way to solve this problem is to control packets passed ATU-R and control the flow behaviors and the ATU-C queues indirectly. Let QoS-aware device bubble in Fig.1 be such a device to control the packets after ATU-R. If it could make the queue size of ATU-C smaller, it would be able to guarantee the lower queueing delay for real-time flows. But the only way to make the queue size of ATU-C lower is to decrease the throughput of such a device. This is a trade-off between the bandwidth utilization of ADSL downstream link and the low queueing delay. Before this paper, some coarse mechanisms for this problem

have been proposed in the practical firewall products or some documentation. Nevertheless, these mechanisms don't consider this trade-off. Here we call this trade-off as *bottleneck movement problem*.

Bottleneck movement problem considers how to *move* the bottleneck node from one node to another node. Considering Fig.1 , if the bottleneck node could be moved from the ATU-C to the QoS-aware device, the QoS-aware device can be easily applied the typical DiffServ or IntServ policies to ensure the quality for flows. While TCP flows and TCP-like flows are the dominating flows in the Internet, this problem focuses on the TCP flows and assumes that the CBR flows is well-behavior. But, in Internet the packet arriving rates from TCP flows at a router are instable, and the peak rate for such flows would be much higher than its average rate. In such a case, even if the throughput of the QoS-aware device is lower than the bandwidth of the ADSL downstream link, these flows still may incur that the packet are queued at ATU-C. It would broke the QoS guarantee for flows. Thus, the solution for this problem should consider how to control the TCP flows not to incur the vast queueing delay in ATU-C and use the ADSL downstream bandwidth effectively.

Before this paper, two mechanisms have been proposed: Partition solution and DiffServ solution[6]. Either in partition solution or in DiffServ solution they both uses the fixed ratio of ADSL downstream bandwidth.[1] Since they does not consider bottleneck movement problem, the service quality for flows would be coarse if the number of flows is quite larger. Thus, these existed solutions are infeasible to provide a QoS-aware network for users with best-effort Internet environment.

In this paper, we address the first adaptive bandwidth management mechanism, *Postgate*, to estimate the behavior of flows, regulate their packets to conform their available bandwidth, and ensure the quality of service and the bandwidth utilization. By applying our solution, the queueing delay of ATU-C is easily bounded into an acceptable value, and the bandwidth utilization is better than the existed solutions.

[1]The difference between partition solution and DiffServ solution is the bandwidth assignment. Partition solution uses a pre-assigned bandwidth for its flows, and DiffServ solution assigns a weight for each flow and schedules them using fair queueing algorithms.

2 Postgate

Since the goal of postgate is to solve the bottleneck movement problem, the first step is to make the queueing delay of ATU-C lower. Nevertheless, while the queueing delay at ATU-C increases, it implies that the peak rate of some flows or all flows over-uses the bandwidth and incurs the quite queueing delay. To solve this issue, an estimation mechanism must be provided to check the qeueing delay of ATU-C and control the throughput of the QoS-aware device. Notice if the throughput of the QoS-aware device is lower, the queued packets incurred by the peak bitrate of flows would be served faster. Thus, in the first half of this section the mechanism to estimate the throughput and the queueing delay in Postgate will be addressed. But since the queueing delay may only be incurred by some misbehaving flows, it would be unfair to keep their weights and reduce the rates for all flows. Thus, in the later half of this section, we will address the misbehaver detection mechanism in Postgate. Before the mechanism, let us list the notation used in Postgate.

Notations	
A_i	active queues at no.i ICMP packet
B_i	throughput of the QoS-aware device.
b_i^k	burst index for class k at no.i ICMP packet
N	maximum number of MTU packets per sec.
ϕ_i^k	weight for class k after no.i ICMP packet
ϕ_0^k	the initial weight for class k
p	number of referred ICMP packets
δ_i	queueing delay of ATU-C estimated by ICMP
δ	minimum delay requirement by active flows

2.1 Throughput estimation

To make the queueing delay lower, we must estimate the queueing delay first. Nevertheless, while the ADSL subscriber never get the permission to access the ATU-C directly, the queueing delay only could be obtained by ICMP or echo service, e.g. DNS, indirectly. Thus, let postgate periodically send an ICMP request to a router near ATU-C and use its echo packet to measure the minimum RTT.[2] Upon receiving an ICMP echo packet postgate can update the minimum RTT using the measured value and compute the queueing delay of ATU-C by the minimum RTT. While the measured queueing delay is available, let us define some notations. Let (1) denote the maximum queueing delay in the referred history.

[2]In most cases, this router is the gateway for the ADSL subscriber.

$$\overline{\delta_i} \leftarrow \max_{i-p \le j \le i} \delta_j \quad (1)$$

Since the peak bitrates of flows (aggregated flows) is related with its bitrates, the way to make the queued packets lower is to reduce the throughput of the QoS-aware device. Since $\overline{\delta_i}$ could be thought as the normalized queued size, the throughput should be corresponding to it. Nevertheless, to make the throughput more stable to wait the response from flows, we uses an AIMD policy for the function which maps the queueing delay to the throughput, i.e.

$$B_i \leftarrow \min \left(\begin{array}{c} 1 - \frac{\overline{\delta_i}}{25\delta}, \\ B_{i-1} + B_{i-1}\frac{\max\left(0.5\delta - \overline{\delta_i}, -0.1\right)}{p} \end{array} \right). \quad (2)$$

In this function, (2) , while the referred queueing delay is quite higher, the throughput would be decreased using the ratio of the queueing delay and the minimum delay requirements, where 25 is a coefficient value from our experiments. On the contrary, while the queueing delay is lower, the throughput will increase using a linear function. This policy matches the congestion control mechanism of TCP flows and TCP-like flows. It would be helpful to reduce the instable phenomenon while such a device adjusts the throughput.

2.2 Weight estimation

After throughput estimation bounds the queueing delay of ATU-C in an acceptable value and uses bandwidth effectively, it still may not be fair to share the pruned bandwidth for each flow by its weight since not all of flows are misbehavior. In this section, we address a weight estimation mechanism to guarantee that the service quality of those well-behavior flows is not effected by others. Let ϕ_i^j be the rate for a flow j at time i, and B_i be the throughput of the QoS-aware device. While the maximum value of the throughput is 1, full-rate, and the maximum value for a weights for a flow is also 1, full-throughput. To provide the enough bandwidth for well-behavior flows, the normalized aggregated weights must be not exceeding than the current throughput, i.e.

$$\frac{\sum_{j \in A_i} \phi_i^j}{\sum_j \phi_0^j} \le B_i. \quad (3)$$

Otherwise, the well-behaved flows may not obtain the enough bandwidth. Thus, the challenge of this mechanism is how to satisfy this equation and only adjust the weights of those misbehaving flows.

2.2.1 Burst index

To detect the misbehaving flows, we design *burst index* variable to indicate the behavior of flow. In an ideal condition, each flow should use their rates smoothly: this is its queue size in the QoS-aware device should be stable and lower. Thus, let (4) be the extra queued packets for class k in a short period, which s_i^k and d_i^k denote the queued size and the dropped size at time i, i.e.

$$\Delta_i^k \leftarrow s_i^k + d_i^k - s_{i-1}^k \quad (4)$$

, and burst index is defined as a filtered value of the extra queued packets, i.e.

$$b_i^k = \frac{\sum_{j=1}^{n} \sum_{k=i-jm+1}^{i-jm+m} \frac{1}{2^j}\Delta_j^k}{\text{MTU}\left(1 - \frac{1}{2^n}\right)} \quad (5)$$

, where m is the sample period of the extra queued packets, and n is the referred history size of burst index. While (5) is a filtered value for the extra queued packets, it would be able to react to the behavior of a flow. From our experiments in many cases, the values which the period of a sample Δ_i^k is 10ms, n is 5 and m is 100 are suitable values.

2.2.2 Proposed weights and weights adjustment

After the indices for the behavior of flows is available, we address the mapping function from their values to the weights for fair queueing algorithm. (6) is a function to propose a new weight for a class using the index.

$$\overline{s_i^k} \leftarrow \phi_{i-1}^k + \frac{\min\left(N\delta - 2b_i^k + 2, 0.5N\delta\right)}{2Np}\phi_0^k \quad (6)$$

Here, let $\overline{\phi_i^k}$ be a normalized value of the proposed weight, $\overline{s_i^k}$, since $\overline{s_i^k}$ may be out of range, i.e.

$$\overline{\phi_i^k} \leftarrow \left\{ \begin{array}{ll} w_0^k & \text{if } \overline{s_i^k} > w_0^k \\ \overline{s_i^k} & \\ 0 & \text{if } \overline{s_i^k} < 0 \end{array} \right. .$$

Notice, the weight, $\overline{\phi_i^k}$, is based on its last weight and its burst index. Thus, if a flow sent too many packets in recent, its burst index would be higher, and its weight would be adjusted to lower. But the aggregated proposed weights may not satisfy (3) . Thus,

we address (7) and (8) to adjust the proposed weights to suitable weights. Let $\overline{d_i^k} \leftarrow \overline{\phi_i^k} - \phi_{i-1}^k$ be the proposed adjustment value, $\overline{d_i^+}$ be the aggregated positive adjustment coefficient, and $\overline{d_i^-}$ be the aggregated negative adjustment coefficient, i.e.

$$\overline{d_i^+} \leftarrow \sum_{\overline{d_i^j} \geq 0 \ \forall j \in A_i} \overline{d_i^j} \text{ and } \overline{d_i^-} \leftarrow \sum_{\overline{d_i^j} < 0 \ \forall j \in A_i} \overline{d_i^j}. \quad (7)$$

Let $D_i \leftarrow \sum_{j \in A_i}^{j} \phi_i^j$ be the minimum throughput required by those proposed weights. Thus, if the newer throughput is higher than the last throughput, we only increase these stable flows. On the contrary, if the new throughput is lower than the last throughput, we only decrease these instable flows. Let $\overline{\gamma_i^+}$ and $\overline{\gamma_i^-}$ be the adjustment coefficient for the adjustment weights, i.e.

$$\overline{\gamma_i^+} \leftarrow 1 + \frac{\max(0, B_i - D_i)}{\overline{d_i^+}}, \text{ and} \quad (8)$$

$$\overline{\gamma_i^-} \leftarrow 1 + \frac{\max(0, D_i - B_i)}{\overline{d_i^-}}. \quad (9)$$

Finally, the weights for flows could be easily computed using the coefficient adjustment weights, i.e.

$$\phi_i^k \leftarrow \begin{cases} \phi_{i-1}^k + \overline{d_i^k \gamma_i^+} & \text{if } \overline{d_i^k} \geq 0 \\ \phi_{i-1}^k + \overline{d_i^k \gamma_i^-} & \text{else} \end{cases}. \quad (10)$$

Notice while ϕ_i^k proposed by this section might still be out of valid range, a normalized function should be used to limit ϕ_i^k into its value range, $\left(0, \phi_0^k\right]$.

3 Simulation

The simulation runs on ns 2.28 with network topology as Fig.2 . In this simulation, the queue length of ATU-C is 256 packets, and other queue lengths are quite large. Additionally, the unlabeled links are Fast Ethernet(100Mbps/1ms). The node pair, s_i and d_i, is a pair of a source node and a destination node. For each node pair, only one flow, TCP or UDP, will be over them.

For simulating a busy ADSL network with a real-time application, we set up 48 FTPs, one CBR and 4 classes with the same guaranteed rate. Notice the number of CBR flows does not affect the results in this simulation. Here we just set an 300kbps CBR to represent some CBR flows. The packet size of TCP is

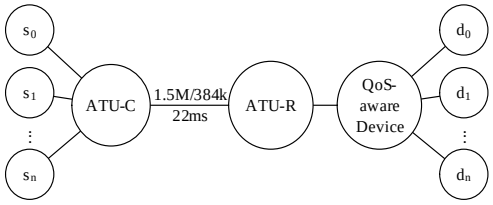

Figure 2: Topology of simulations

1460 bytes, and the packet size of UDP is 1000 bytes. Every 10sec, four FTPs will start, and after 350sec, every 10sec, four FTPs will stop. And a 300kbps CBR traffic is assigned class 3 which starts at 0s to simulate the real-time application.

For comparing with existing solutions, a SFQ(DiffServ) solution and a partition solution are also implemented at the QoS-aware device. In SFQ solution and partition solution, we set the throughput of the QoS-aware device as 90% to try to move the bottleneck from ATU-C to the QoS-aware device. For Postgate, p is set as 100 ICMP packets and δ is set as 100ms to make sure the low latency for the real-time application. Since the bitrate of CBR traffic is 300kbps and the guaranteed rate for CBR traffic is more than 300kbps, the CBR traffic should get a pretty quality of service if the QoS solution works fine.

Fig.3 is the results of queueing delay at ATU-C and the throughput of ADSL link from each solution. In this figure, Postgate is successful to make the queueing delay of ATU-C lower than 50ms and guarantee the utilization higher than 60% even if some bulk file transmissions start or stop. Although the utilization of SFQ solution(DiffServ) is high, its queueing delay is also instable and higher. Furthermore, the queueing delay of partition solution is also instable when all flows are active. In this figure, Postgate is successful to move the bottleneck node from ATU-C to QoS-aware device and still obtain an acceptable utilization even if the behavior of flows are vary. Notice, while the sending bitrate of CBR flows is corresponding to its assigned rate, the low queueing delay at ATU-C, bottleneck, means the low jitter, the low transmission delay and the high quality for CBR flows if its weight affords to provide its served rate.

Thus, let us turn back to consider the weights adjustment. Since Postgate would adjust the throughput to reduce the queued delay at ATU-C, if the weight adjustment mechanism does not work. The served rate for CBR flows would be still effected by the pruned throughput. Fig.4 represents the bandwidth

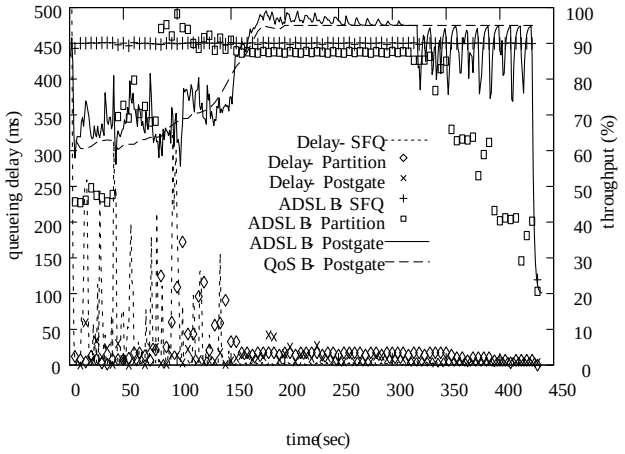

Figure 3: The queueing delay of ATU-C and the thoughputs

of each flows in the output link of the QoS-aware device. In this figure, CBR always get its served rate and does not effected by other flows, and other TCP flows could use other available bandwidth. The over-used flow, e.g. class 0 in duration [50,80], would get lower weights and resume their weights after it behavior is going to stable. The same situations also appeared in class 2 and class 3.

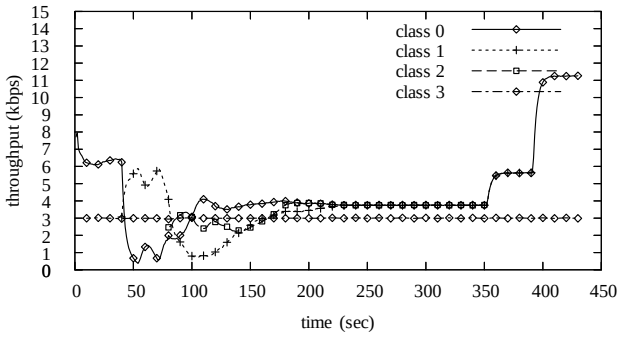

Figure 4: Bandwidth of Postgate

Then, we show the weights of Postgate in fig.5 . In this figure, the results is corresponding to the status of flows. While the flows of class 0 starts at 0 second, their weights are adjusted to lower after Postgate detects the high queueing delay. After the flows of class 2 starts, the weights of class 2 are also adjusted to lower due to the same reason. This phenomenon also appears on the weight of class 3, and their weight would be resumed after the flows are stable. Notice, in this figure, the weight for the CBR flow is never adjusted in Postgate. It conform to the goal of Postgate. While the CBR flows never over-use its bandwidth in this simulation, it should not be adjusted.

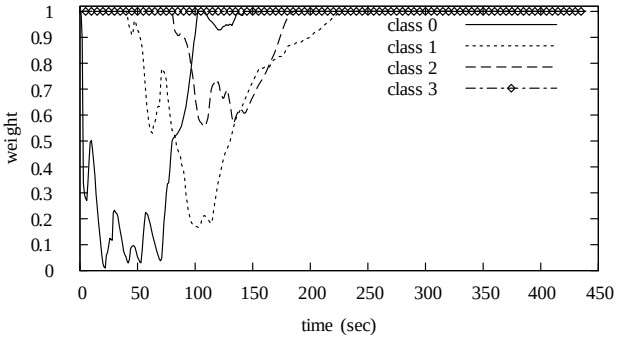

Figure 5: Weights of Postgate

While the queueing delay at ATU-C by applying Postgate is lower than 50ms, and the weight for CBR flow is always 1, it implies that the queueing delay for the packets of CBR flow either in ATU-C or in the QoS-aware device is lower. Nevertheless, except the quality of service for CBR flow, Postgate still be able to use the ADSL downstream bandwidth effectively.

4 Conclusion and Future Work

In this paper, we proposed a novel bandwidth management mechanism to provide QoS for ADSL network without ISP supports. Our simulation experiments have shown Postgate offer much pretty quality of service for real-time applications than others and still achieve a high bandwidth utilization. Nevertheless, this mechanism is still able to be improved in the continuous research efforts in the estimation policies with difference traffic types, e.g. HTTP.

References:

[1] S. Blake, D. Black, M.Carlson, E. Davies, Z. Wang and W. Weiss, "*An architecture for differentiated services*", RFC 2475, December 1998.

[2] K. Nichols, V. Jacobson and L. Zhang, "*Two-bit differentiated services architecture for the Internet*", IETF RFC 2638, July 1999.

[3] A. Demers, S.Keshav and S.Shenker, "*Design and Analysis of a fair queueing algorithm*", Proceeding of ACM SIGCOMM, September 1989.

[4] P. Goyal, H. M. Vin, and H. Cheng, "*Start-time fair queueing: A scheduling algorithm for integrated services packet switching networks*", TR-96-02, Dept. Comput. Sci., Univ. Texas at Austin, Jan 1996.

[5] Aditya Akella, Srinivasan Seshan and Anees Shaikh, "An Empirical Evaluation of WideArea Internet Bottlenecks", Proceeding of ACM SIG-METRICS, June 2003.

[6] Dan Singletary, *"ADSL Bandwidth Management HOWTO"*, The Linux Documentation Project, April 2003.