Network Tomography-based Unresponsive Flow Detection and Control*

Ahsan Habib, Bharat Bhargava
Center for Education and Research in Information Assurance and Security (CERIAS)
and Department of Computer Sciences, Purdue University, West Lafayette, IN 47907-1398
{habib, bb}@cs.purdue.edu

Abstract

To avoid a congestion collapse, network flows should adjust their sending rates. Adaptive flows adjust the rate, while unresponsive flows do not respond to congestion and keep sending packets. Unresponsive flows waste resources by taking their share of the upstream links of a domain and dropping packets later when the downstream links are congested. We use network tomography—an edge-to-edge mechanism to infer per-link internal characteristics of a domain—to identify unresponsive flows that cause packet drops in other flows. We have designed an algorithm to dynamically regulate unresponsive flows. The congestion control algorithm is evaluated using both adaptive and unresponsive flows, with sending rates as high as four times of the bottleneck bandwidth, and in presence of short and long-lived background traffic.

1 Introduction

An *unresponsive* flow does not control its sending rate in response to a congestion. *Adaptive* flow, such as TCP, reduces its sending rates during congestion. This behavior of TCP prevents a congestion collapse in a network. The unresponsive flow, such as UDP, sends at the same rate even if there is a congestion along the path. Because it does not use any feedback mechanism, and can not respond to the congestion. This behavior may cause the adaptive flows to starve, and introduces unfairness when both types of flows coexist at the same time in the Internet.

An adaptive flow might behave as unresponsive one if the implementation does not follow standard specifications. For example, selfish users can change the implementation of TCP so that it does not adjust the sending rate during a congestion. We can detect and control such selfish flows using the proposed scheme.

The research problem we address in this paper is to design a scalable congestion control scheme that involves only the edge routers. Involving core routers makes a scheme difficult to deploy. Congestion collapses can be mitigated using improved packet scheduling or an active queue management [2, 6]. However, the problem is associated with dynamic conditions such as network load, capacity, and the reaction of different transport protocols to congestion. Therefore, a dynamic control mechanism can solve this problem.

We use the network tomography, an edge-to-edge mechanism to infer per-link characteristics of a network domain, to detect congestion in a network domain. The tomography-based unresponsive flow detection scheme samples incoming flows at the ingress routers, and probes the network with sampled data. The edge-to-edge probing detects excessive packet loss inside of a network domain and the cause behind the loss. To alleviate the congestion, the unresponsive flows are regulated at the edge routers.

We design detection and adaptive control mechanism in this paper. During congestion, the control algorithm regulates the suspected flows such a way that the loss ratio of the congested links drops exponentially with time. In absence of the congestion, the flow rates are increased to the maximum value subscribed by the user. To achieve scalability, the detection and control processing is done without involvement of core routers. The scheme has been evaluated and the performance has been tested using the *ns-2* simulator.

The organization of this paper is as follows: Section 2 discusses the related work on congestion collapse from undelivered packets and mechanisms to address the problem. The loss inference using network tomography, detecting unresponsive flows, and congestion control algorithm is discussed in Section 3. The setup for the experiments, the simulation results, and overhead of the proposed scheme are provided in Section 4. Section 5 concludes the paper.

*This research is sponsored in part by the NSF grants ANI-0219110, CCR-001712, and CCR-001788, CERIAS grants, and IBM SUR grant.

2 Related Work

Floyd *et al.* [5] study congestion collapse from undelivered packets. This situation arises when bandwidth is continuously consumed by packets at the upstream that are dropped at the downstream. Several ways to detect unresponsive flows are presented. It is suggested that routers can monitor flows to detect whether flow is responsive to congestion or not. If a flow is not responsive to congestion, it can be penalized by discarding packets to a higher rate at the router. According to the authors there are some limitations of these tests to identify non-"TCP-friendly flow". It does not help to save bandwidth at the upstream if the flow sees the congestion at the downstream because this solution does not propagate the congestion information from downstream to upstream.

Seddigh *et al.* [10] suggest that if TCP and UDP are put into separate queues or Assured Forwarding classes, they may coexist fairly. This discrimination between TCP and UDP traffic may punish some well-behaved UDP flows. The core router does not know the profile of a flow and can not decide to allocate bandwidth to them fairly. The problem is associated with network load, capacity, and the reaction of different transport protocols to congestion. A dynamic control mechanism can solve this problem.

Albuquerque *et al.* [1] propose congestion avoidance mechanism named Network Border Patrol. To detect congestion, it measures entering rate of traffic to a domain and the leaving rate from the domain. It detects and restricts unresponsive traffic flows and eliminates congestion collapse. The border routers monitor all flows, measure rates, and exchange this information with all edge routers periodically and this can be expensive. Moreover, TCP is responsive so we do not need control mechanism for TCP at the edges.

Chow *et al.* [3] propose a framework where edge routers periodically obtain information from the core by probing and adjust the conditioner using the traffic dynamics. In this scheme, core needs to maintain all the state information. A simpler scheme can be employed where core sends packets to edge routers only at the time of congestion.

Wu *et al.* propose Direct Congestion Control Scheme (DCCS)in [11]. In this scheme, they detect congestion by observing packet drops with *lowest priority to drop* at the core router. We follow the same rule in our research to detect congestion. Our core is simpler in the sense that it detects drops of only unresponsive flows. The main difference between our work and [11] is that we design the shaper at the edge that controls the unresponsive flow. We explored the similar idea, and designed an adaptive shaper at the edge that controls the unresponsive flow in [7].

Mahajan *et al.* [8] use Aggregate-based Congestion Control (ACC) to detect and control high bandwidth aggregate flows. They use the history of packet drops over a time interval and then the ACC agent matches prefix of IP destination addresses to detect flows going to the same destination address for Denial of Service(DoS) attacks. The ACC agent controls the flows using a rate-limiter and pushes status messages reporting the arrival rate to the upstream routers. Our goal is to detect and control unresponsive flows, however, it can protect DoS attack by using their idea of prefix matching [8].

All of the above schemes involve core routers to detect congestion collapse due to unresponsive flows. Our tomography-based congestion control scheme does not involve core routers and, thus, achieves scalability. In next section, we discuss inference of packet loss using network tomography.

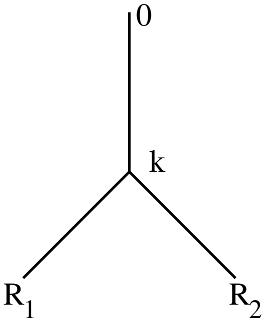

Figure 1. Binary tree to infer loss from source 0 to receivers R_1 and R_2.

3 Tomography-based Congestion Control (TCC)

3.1 Network Tomography and Loss Inference

Network tomography uses correlations among end-to-end measurements to infer per-link characteristics. For example, Duffield *et al.* [4] use unicast packet "stripes" (back-to-back probe packets) to infer a link loss by computing the correlation of a packet loss within a stripe at different destinations. This scheme sends a series of probe packets, called a stripe, with no delay among the transmissions of successive (usually three) packets. For a two-leaf binary tree (Figure 1) spanned by the nodes 0, k, R_1, R_2, stripes are sent from the root 0 to the leaves to estimate the characteristics of one link, say $k - R_1$ [4]. If a packet reaches the receiver, we can infer that the packet reached the branch point k. A complementary stripe is similarly sent to estimate the characteristics of the other link, $k - R_2$. The packet transmission probability from the root to node k is calculated as:

$$A_k = \frac{Z_{R_1} Z_{R_2}}{Z_{R_1 \cup R_2}}, \tag{1}$$

where Z_{R_1} represents the empirical mean of a binary variable, which takes the value of 1 when all packets sent to R_1 reach their destination and 0 otherwise. The mean is taken over n identical stripes. By combining estimates of stripes down each such tree, the characteristics of the common path from $0 - k$ are estimated. The loss ratio of a link is calculated by subtracting transmission probability from one.

This inference technique extends to general trees. Consider an arbitrary tree where for each node k, $R(k)$ denotes the subset of leaves descended from k. Let $Q(k)$ denotes the set of ordered pairs of nodes in $R(k)$ descended from k. For each $(R1, R2) \in Q(k)$, a stripe should be sent from the root to the receivers R_1 and R_2.

3.2 Congestion Detection

Congestion detection depends on delay and loss measurements. Delay is the end-to-end latency; packet loss ratio is defined as the ratio of number of dropped packets from a flow to the total number of packets of the same flow entered the domain. We first describe delay measurements and loss measurements before discussing the detection algorithm.

3.2.1 Delay Measurements

The unresponsive flows are sampled (using transport layer protocol information) at all ingress routers. The header of a sampled packet is used to probe the path of an unresponsive flow. The probe and user traffic follow the same path with a high probability, because the route does not get changed often inside a network domain. This measurement is a close approximation of the delay value that is experienced by the sampled flows in the network domain.

For delay probing, the ingress routers encode the current timestamp $t_{ingress}$ into the payload, and mark the protocol field of the IP header with a new value. An egress router recognizes such packets, and removes them from the network. Additionally, the egress router computes the edge-to-edge link delay for a packet from the difference between its own time and $t_{ingress}$. We ignore minor drifts of the clocks since all routers are in one administrative domain, and can be synchronized fairly accurately. The egress classifies the probe packet as belonging to flow i, and update the average packet delay, avg_delay^i, for delay sample $delay^i(t)$ at time t using an exponential weighted moving average (EWMA):

$$avg\_delay^i(t) = \alpha \times avg\_delay^i(t-1) + (1-\alpha) \times delay^i(t), \quad (2)$$

where α is a small fraction $0 \leq \alpha \leq 1$ to emphasize recent history rather than the current sample alone.

If the average packet delay of path k exceeds the delay guarantee of the path for flow i, it is an indication of congestion. If the network is properly provisioned and flows do not misbehave, there should not be any delay greater than the estimated path delay for any flow i. A flow may experience high delay due to some other flows that cause congestion in the network.

3.2.2 Loss Measurements

If the edge-to-edge link delay is higher than a predefined threshold, we use the loss inference mechanism described in previous subsection (using equation 1) to measure loss in links that experience high delay. The objective of loss measurements is to obtain loss ratio of each individual links. The links with high losses are identified, and the loss value is used to control the congestion.

The delay probing identifies the paths that need to be considered for loss measurements. Each loss probing needs one sender and two receiver nodes. The incidence of high delay and the edge-to-edge paths that have high delay are reported to a congestion controller that sits at any edge router. The controller collects a set of paths $\mathcal{P}$ for the loss probing. The set $\mathcal{P}$ and the topology are used to determine the root of the probing tree for stripe-based probing. Probes are sent from the root to all ordered pair of edge routers. The root is selected such a way that it can cover maximum number of links in the set $\mathcal{P}$. If some links in the path set $\mathcal{P}$ are not covered, we need to repeat the loss measurements from another edge router considering as a root of the tree.

3.2.3 Detection

The links with high losses and egress routers through which flows are exiting the domain are identified. At these egress routers, all flows that are consuming high bandwidth are isolated. These rates are sent to the ingress routers through which the flows enter into the domain. The ingress router compares the rate at which the suspected flows are entering and leaving the network domain. This identifies the flows that are not cooperating with the network to control their rates in response to congestion.

The rate of unresponsive flows can be reported per flow basis or in aggregate. If the number of flows to be reported exceeds a threshold, the feedback is done on an aggregate basis for each ingress router. This aggregation is done based on the traffic class. For each traffic class the unresponsive flows with high bandwidths are reported to the ingress routers. The identity of the ingress routers are obtained from the delay probes, where an identification code is used to relate a flow and its entry point. Otherwise, the egress does not know through which ingress routers the flows are entering into the domain. The detection algorithm runs as follows:

1. Each ingress router samples the user traffic for delay probing. The egress routers report the incidence of

high delay and the edge-to-edge path that has high delay to a congestion controller.

2. The controller generates a probing tree using the set of path $\mathcal{P}$ that has high delay. The root of the tree is the sender of the loss probing. These probes are sent to every order pair of the edge routers of the domain. The loss probing obtains the loss ratio of each individual link of the path in $\mathcal{P}$.

3. Using the links with high loss ratio and the topology tree, a set of egress routers $\mathcal{E}$ are obtained through which the unresponsive flows are leaving the domain. These flows need to be controlled because they are causing congestion in the domain.

4. Flows are analyzed at each egress router of the set $\mathcal{E}$. The flows that are having high bandwidth are reported to the ingress routers through which flows are entering into the domain.

3.3 Congestion Control

To control the unresponsive flows, the inferred loss ratio is used. As it is discussed in detection mechanism, the loss ratio is sent to the ingress routers to control the flows. For each flow that has high packet loss inside a domain, the router reduces the rate proportionally to the packet drop rate inside the network. Suppose that a flow has an original profile (target rate) of $targetrate$. In case of the packet drop ratio $lossratio$, the profile of the flow is updated temporarily (to yield the rate $newprofile$) using $newprofile = targetrate \times lossratio$.

The congestion control algorithm adjusts the rate of the flows (that are causing drops to other flows) such a way that the loss ratio inside a network domain converges to a low predefined threshold LLOSSTHR. If the loss ratio decreases with time due to the current control setup, the control parameters are not changed until the loss ratio converges to a value. If the converged loss value is higher than the LLOSSTHR, the rate is controlled based on the loss ratio. If the loss ratio stays below LLOSSTHR for a while, the control algorithm allows more traffic to enter into the domain. The rate is increased linearly until it crosses the LLOSSTHR. In this way, the loss oscillates towards the LLOSSTHR parameter.

The control algorithm runs as follows:

1. If the loss ratio jumps to a high value, the control algorithm decreases the incoming rate of the unresponsive flows. This control decreases the loss ratio exponentially with time. If the loss decreases with time, the control algorithm does not change the rate control (RC) parameter. We refer to this decreasing loss ratio direction as DOWNWARD direction.

2. When the loss ratio converges to a value higher than the LLOSSTHR, the algorithm decreases the rate again based on current loss ratio. If the converged loss ratio is below the LLOSSTHR for a specified time, the algorithm allows more traffic to enter into the domain.

3. If the loss ratio curve goes UPWARD (opposite of DOWNWARD) direction, the rate control is increased with the current loss ratio.

Therefore, the rate control algorithm conducts Additive Increase and Multiplicative Decrease (AIMD). In presence of loss, the rate is control aggressively, and in absence of loss the rate is increased linearly. Thus, the rate adjustment algorithm is similar to TCP congestion control algorithm. At the edge router, shaping is done based on the RC parameter. The value of the RC parameter varies from $0 \rightarrow 1$. To shape a flow, a random number is generated. If the random number is less than RC, a packet from this flow is dropped. Otherwise, the packet is admitted into the domain.

4 Experimental Study

4.1 Setup

Using the *ns-2* [9] simulator, we evaluate the performance of our unresponsive flow control scheme. To test our framework, we have used a topology shown in Figure 2. The same topology is used in [4] to infer loss. We generate several TCP and UDP type aggregate flows in the network. Each aggregate flow contains 10 to 100 micro flows. Cross traffic is used to vary the background traffic by setting the start and the finish times of these flows differently, in order to change the overall traffic situation over the paths of all flows. The sending rate and the round trip time (RTT) of different flows are changed over time to show the robustness of the control mechanism for a variety of flows.

We conduct a series of experiments to show congestion collapse in the absence of flow control mechanism, the effectiveness of the tomography-based inference of network parameters, and the unresponsive flow detection and control in a variety of scenarios.

4.2 Congestion Detection

The congestion is detected using edge-to-edge link delay and link loss ratio. If some links are congested, the edge-to-edge delay through these links become very high. This high delay is used as an indication of congestion. Figure 3 shows delay of the path $E1 \rightarrow E6$. The latency of this path is 100 ms when the links are idle. However, the delay goes as high as one second due to the congestion. The figure also shows that with proper congestion control the delay can be reduced to a desired level.

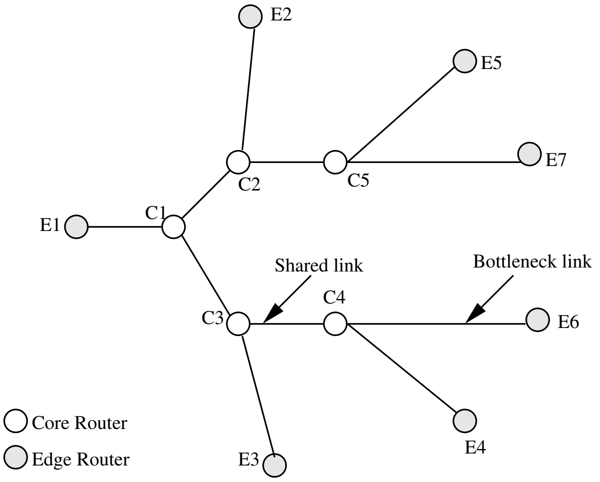

Figure 2. Simulation topology. Each edge router is connected with multiple domains. $C4 \rightarrow E6$ is the bottleneck link in the setup. Unresponsive flows take their share from the shared link $C3 \rightarrow C4$, and their packets are dropped in the bottleneck link.

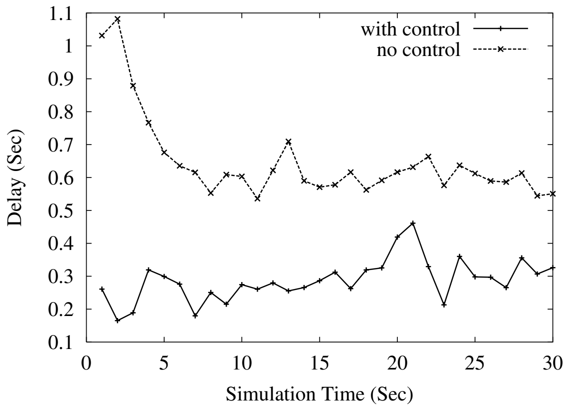

Figure 3. Delay pattern changes with excessive traffic. This high delay is an indication that the edge-to-edge path is congested. The flow control mechanism alleviates the congestion, and reduces the delay.

The congested links are identified using stripe based unicast probing. The probes are sent to obtain loss ratio of the links that lie on the high delay path. In our experiment, we send probes from $E1$ to all other edge routers to obtain the loss of the links on the path $E1 \rightarrow E6$. The inferred loss for the link $C4 \rightarrow E6$ is shown in Figure 4. It shows loss inference for the topology described above for 3-packet stripes. First experiment has fewer number of flows to cause packet drops inside the network domain. Second and third experiments have enough flows to cause huge packets drops in the network. The figure shows loss inference is close to the actual loss in most of the cases. In few cases, it overestimates or under-estimates the loss. We can reduce this effect by increasing the time interval to measure probe loss. 4-packet stripe has little advantage on 3-packet stripe in our experiment.

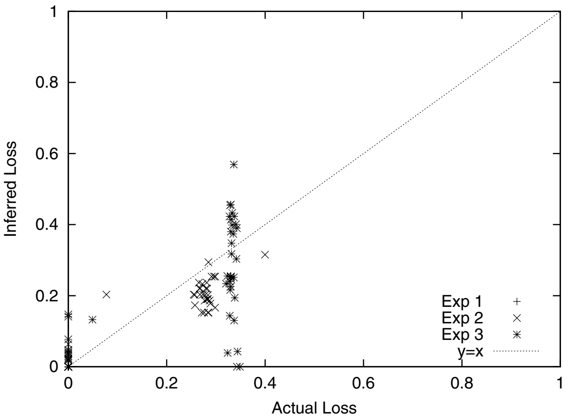

Figure 4. Inferring loss using unicast stripe-based probing. The actual loss is close to the inferred loss.

Upon detection of the congested links, we identify the unresponsive flows at the egress routers, and pass this information to the appropriate ingress routers to control them. The inference mechanisms converge to real measurements value in 15-20 second. The detection mechanism is effective if the congestion continues for a while. If the congestion lasts only a few seconds, we do not need to control that.

4.3 Congestion Control

The unresponsive flows are controlled using a shaper. The shaping algorithm drops packets based on the service level agreement (SLA) parameters of the flow, the drop rate, and the sending rate of the flow. First, we show that in absence of congestion control there might be a congestion collapse in a network domain. Second, we show the performance of an adaptive congestion control algorithm. Third, the robustness of the algorithm is shown with varying number of micro flows, where a micro flow is defined with five tuples (source addr, source port, dest addr, dest port, and protocol).

Congestion Collapse. Congestion collapse due to undelivered packets wastes resources in a network. In our experiments, TCP and UDP flows share the link $C3 \rightarrow C4$.

Then, the UDP flows experience congestion at the link $C4 \rightarrow E6$, which causes huge amount of packet drops. As, these UDP flows take the equal share with TCP flows of the link $C3 \rightarrow C4$, the resources are wasted in the next link. If we know the packets will be dropped any way at the link $C4 \rightarrow E6$, it is better to drop them earlier at the ingress router so that the TCP flows can get the wasted share of the link $C3 \rightarrow C4$, which increases the application level quality of the TCP flows. Figure 5 shows that without any flow control, the TCP flows obtain less than 1.5 Mbps, whereas, with flow control mechanism the bandwidth gain goes higher than 3 Mbps. The congestion window of a sampled TCP flow is shown with or without flow control in Figure 6. Flow control helps to increase the congestion window of the TCP flows.

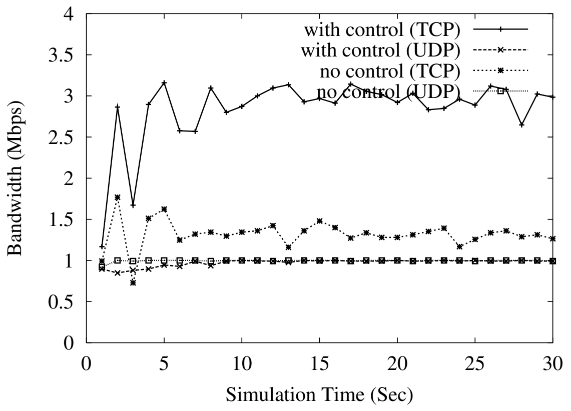

Figure 5. Congestion collapse if there is no flow control. TCP gets the wasted bandwidth by the UDP flows when flow control mechanism is used.

Adaptive Control. The adaptive control algorithm throttles the rate of unresponsive flows during congestion so that the loss ratio converges to a low and predefined value. The algorithm needs to infer loss periodically. An interval of 30 second is used to infer loss, and high loss is informed to the appropriate flow controller.. We run the experiment for 1200 second to evaluate the performance of the adaptive congestion control. Figure 7 shows that this mechanism helps the TCP flows to obtain 2.5 - 3 Mbps bandwidth. The UDP flows face more aggressive drops for a while, because the algorithm tries to determine the control rate at which the unresponsive flows should be shaped.

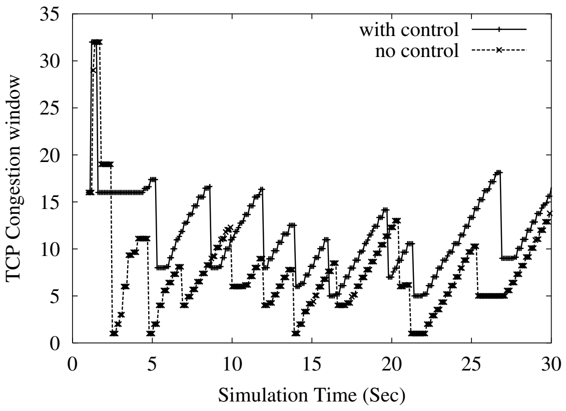

Figure 6. Congestion window of a TCP flow with or without flow control. The congestion window is reset to one several times if there is no flow control.

Figure 8 shows the loss pattern during this congestion control. Initially, the loss drops exponentially. When the loss ratio hits the lowest acceptable level at 600 second, the control mechanism allows more traffic into the domain. In this way, the rate control parameter is adjusted, and the loss ratio oscillates towards the LLOSSTHR value, which is 0.1 in our experiments.

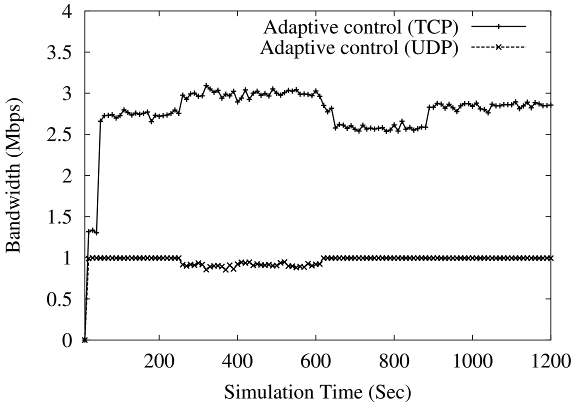

Figure 7. Bandwidth gain by TCP and UDP flow with adaptive flow control.

Robustness. To show the robustness of the control algorithm, we vary the number of micro flows per aggregate flows from 10 to 100. We sample the bandwidth of TCP and UDP flows from 50 second to 60 second, and take the average to plot in Figure 9. This figure shows that increasing number of flows does not hurt the performance of the congestion control algorithm. Experiments are conducted with different network dynamics that change the congestion dynamically to test the robustness of the control algorithm.

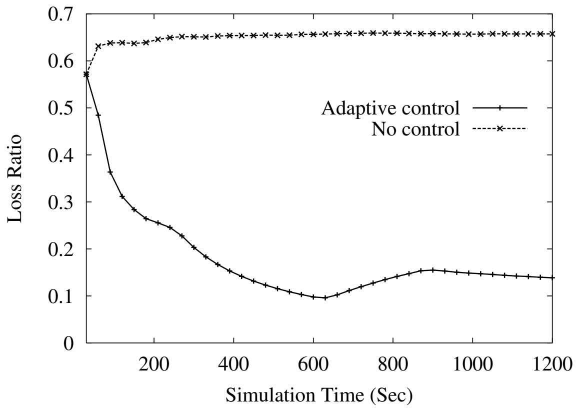

Figure 8. Loss ratio with adaptive flow control. Initially the loss decays exponentially, and the loss converges with time to a low value.

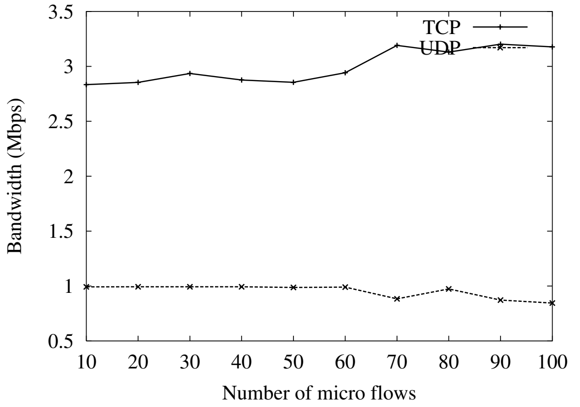

Figure 9. Bandwidth achieved by TCP and UDP flows with varying number of micro flows.

4.4 Overhead

The overhead of our scheme is low. If all links are OC3 type, on average each link experiences probe traffic less than 0.015% of the link capacity in the network domain shown in Figure 2. Our mechanism in non-intrusive, i.e., the injected traffic does not change the characteristics of the network domain.

5 Conclusion

We proposed and evaluated a new and scalable way to detect and regulate unresponsive flows to prevent congestion collapses due to undelivered packets. Our scheme does not require any help from the core routers and introduces a very low overhead. The changes require at the edges (ingress and egress) are minor. The implementation is simple and the deployment should be easy.

References

[1] C. Albuquerque, B. Vickers, and T. Suda. Network Border Patrol. In *Proc. IEEE INFOCOM*, Tel-Aviv, Israel, Mar. 2000.

[2] B. Braden and et al. Recommendations on queue management and congestion avoidance in the internet. RFC 2309, Apr. 1998.

[3] H. Chow and Leon-Garcia A. A feedback control extension to differentiated services. Internet Draft, draft-chow-diffserv-fbctrl-00.pdf, Mar. 1999.

[4] N. G. Duffield, F. Lo Presti, V. Paxson, and D. Towsley. Inferring link loss using striped unicast probes. In *Proc. IEEE INFOCOM*, Anchorage, AK, Apr. 2001.

[5] S. Floyd and K. Fall. Promoting the use of end-to-end congestion control in the Internet. *IEEE/ACM Transactions on Networking*, Aug. 1999.

[6] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, 1(4):397–413, Aug. 1993.

[7] A. Habib and B. Bhargava. Unresponsive flow detection and control in differentiated services networks. In *Proc. IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS)*, Anaheim, CA, Aug. 2001.

[8] M. Mahajan, S. M. Bellovin, S. Floyd, J. Ioannidis, V. Paxson, and S. Shenker. Controlling high bandwidth aggregates in the network. *ACM Computer Communication Review*, 32(3):62–73, July 2002.

[9] S. McCanne and S. Floyd. Network simulator ns-2. http://www.isi.edu/nsnam/ns/, 1997.

[10] N. Seddigh, B. Nandy, and P. Pieda. Study of TCP and UDP interaction for the AF PHB. Internet Draft, 1999.

[11] H. Wu, K. Long, S. Cheng, and J. Ma. A Direct Congestion Control Scheme for Non-responsive Flow Control in Diff-Serv IP Networks. Internet Draft, draft-wuht-diffserv-dccs-00.txt, Aug. 2000.