Congestion Control Mechanism using Network Border Protocol

Ashakiran.G.N[1], M.V.Panduranga Rao[2], S.Basavaraj Patil[3]

[1]Dept of Computer Science and Engineering
BTL Institute of Technology
Bangalore, India
ashakirangn@yahoo.com

[2]Dept of Computer Science and Engineering
BTL Institute of Technology
Bangalore, India
raomvp@yahoo.com

[3]Dept of Computer Science and Engineering
BTL Institute of Technology
csehodbtlit@gmail.com

Abstract: ***Congestion in the network is increasing immensely due to the growth of usage of multimedia application. This lead to unresponsive and misbehaving traffic flows. There are few primitive scheduling schemes to control the congestion in the network, which worked only at end-to-end network, due which they were unable to prevent the congestion collapse and unfairness created by applications that are unresponsive to network congestion. So together we included those schemes and made improvements to make a more flexible solution for building a congestion control scheme in network using network border protocol framework. Using NBP framework we show the aggregation and connection admission control mechanisms into the NBP framework. The NBP framework provides a proper protocol designs which entails the exchange of feedback between routers at the borders of a network in order to detect and restrict unresponsive traffic flows before they enter the network, thereby preventing congestion within the network. The NBP framework is proposed with enhanced core-stateless fair queuing (ECSFQ) mechanism, which provides fair bandwidth allocations to competing flows.***

Keywords: Network Border Patrol; Leaky Bucket Algorithm; Feedback Control; Rate Control Algorithm;

1. Introduction

There has been a rapid increase in usage of networked multimedia parallels the growth of Internet. Despite novel techniques for data compression and multicast for data transmission, multimedia applications are bandwidth intensive, delay sensitive, and somewhat less tolerant. TCP being both a reliable and fair protocol (retransmits every lost or corrupted packet and slows down in case of congestion) is mostly suited for file transfers, terminal work and web browsing. This usually does not work in transporting interactive video and sound, where reliability is a weakness rather than strength, and consequently UDP is the protocol of choice. UDP has no mechanisms either to detect, or to control congestion, which classifies it as an unresponsive protocol. When low capacity link becomes a bottleneck and the network may enter into a state of congestion, UDP maintains its transmission rate. It may use almost all capacity of that link. While the self-clocking TCP, as congestion responsive, will slow down and thus decrease the good put that can eventually go to zero. The phenomenon is known as a congestion collapse since most of the network resources transmit undelivered packets [1]. When the number of TCP flows in the Internet is prevalent, the stability of the network is guaranteed by the congestion control mechanisms as an integral part of the transport protocol. In the presence of UDP, the situation radically changes, which makes any co-existence of different transport protocols a virtual impossibility and the appearance of congestion a reality. Recent research has focused on studying and resolving this problem, as in Network Border Patrol, [2], where all data flows are monitored and their sending rates are accordingly adjusted via traffic shapers placed at the edge routers.

Another solution is the Datagram Congestion Control Protocol (DCCP) [3], a sort of a blend between UDP and TCP, where the complexity of the latter is reduced just to its congestion control features. The suggested solutions are still being studied and experimented with, which makes the question of a suitable congestion control strategy when socially responsible and socially irresponsible protocols have to work together, as it is the case today on the Internet.

In this paper, we introduce and investigate a new Internet traffic control mechanism called Network Border Patrol (NBP). The basic principle of NBP is to compare, at the borders of the network, the rates at which each flow's packets are entering and leaving the network. If packets are entering the network faster than they are leaving it, then the network is very likely to be buffering or, worse yet, discarding the flow's packets. In other words, the network is receiving more packets than it can handle. NBP prevents this scenario by "patrolling" the network's borders, ensuring that packets do not enter the network at a rate greater than they are able to leave it. This has the beneficial effect of preventing congestion collapse from undelivered packets; because an unresponsive flow's otherwise undeliverable

packets never enter the network in the first place. NBP's prevention of congestion collapse comes at the expense of some additional network complexity, since routers at the borders of the network (i.e., edge routers) are expected to monitor and control the rates of individual flows. NBP also introduces an added communication overhead, since in order for an edge router to know the rate at which its packets are leaving the network, it must exchange feedback with other edge routers. However, unlike other existing approaches to the problem of congestion collapse, NBP's added complexity is isolated to edge routers; routers within the core of the network remain unchanged. Moreover, end systems operate in total ignorance of the fact that NBP is implemented in the network, so no changes to transport protocols are necessary. Note that the primary goal of NBP is to prevent congestion collapse from undelivered packets.

2. Background Work

2.1 Rate Control Algorithm

The NBP rate-control algorithm regulates the rate at which each flow is allowed to enter the network. Its primary goal is to converge on a set of per-flow transmission rates (hereinafter called ingress rates) that prevents congestion collapse due to undelivered packets. It also attempts to lead the network to a state of maximum link utilization and low router buffer occupancies, and it does this in a manner that is similar to TCP.

In the NBP rate-control algorithm, flows may be in one of two phases, slow start or congestion avoidance, similar to the phases of TCP congestion control.[4] The desirable stability characteristics of slow-start and congestion control algorithms have been proven in TCP congestion control, and NBP expects to benefit from their well-known stability features. In NBP, new flows entering the network start with the slow-start phase and proceed to the congestion-avoidance phase only after the flow has experienced incipient congestion.

The rate-control algorithm is invoked whenever a backward feedback packet arrives at an ingress router. Recall that backward feedback packets contain a timestamp and a list of flows arriving at the egress router from the ingress router as well as the monitored egress rates for each flow. Upon the arrival of a backward feedback packet, the algorithm calculates the current round-trip time between the edge routers and updates the base round-trip time (e.base RTT), if necessary.

The base round-trip time (e.baseRTT) reflects the best-observed round-trip time between the two edge routers. The algorithm then calculates deltaRTT, which is the difference between the current round-trip time (currentRTT) and the base round-trip time (e.baseRTT). A deltaRTT value greater than zero indicates that packets are requiring a longer time to traverse the network than they once did, and this can only be due to the buffering of packets within the network.

NBP's rate-control algorithm decides that a flow is experiencing incipient congestion whenever it estimates that the network has buffered the equivalent of more than one of the flow's packets at each router hop. To do this, the algorithm first computes the product of the flow's ingress rate (f.ingressRate) and deltaRTT (i.e., f.ingressRate deltaRTT). This value provides an estimate of the amount of the flow's data that is buffered somewhere in the network. If this amount (i.e., f.ingressRate deltaRTT) is greater than the number of router hops between the ingress and the egress routers (e.hopcount) multiplied by the size of the largest possible packet (MSS) (i.e., MSS e.hopcount), then the flow is considered to be experiencing incipient congestion.

The rationale for determining incipient congestion in this manner is to maintain both high link utilization and low queuing delay. Ensuring there is always at least one packet buffered for transmission on a network link is the simplest way to achieve full utilization of the link, and deciding that congestion exists when more than one packet is buffered at the link keeps queuing delays low.

The rate quantum is not allowed to exceed the flow's current egress rate divided by a constant quantum factor (QF). This guarantees that rate increments are not excessively large when the round-trip time is small. When the rate-control algorithm determines that a flow is experiencing incipient congestion, it reduces the flow's ingress rate.

If a flow is in the slow-start phase, it enters the congestion-avoidance phase. If a flow is already in the congestion-avoidance phase, its ingress rate is reduced to the flow's egress rate decremented by a constant value. In other words, an observation of incipient congestion forces the ingress router to send the flow's packets into the network at a rate slightly lower than the rate at which they are leaving the network.

NBP's rate-control algorithm is designed to have minimum impact on TCP flows. The rate at which NBP regulates each flow (f.ingressRate) is primarily a function of the round-trip time between the flow's ingress and egress routers (currentRTT). In NBP, the initial ingress rate for a new flow is set to be MSS/e.baseRTT, following TCP's initial rate of one segment per round-trip time.

NBP's currentRTT is always smaller than TCP's end-to-end round-trip time (as the distance between ingress and egress routers, i.e., the currentRTT in NBP, is shorter than the end-to-end distance, i.e., TCP's round-trip time). As a result, f.ingressRate is normally larger than TCP's transmission rate when the network is not congested, since the TCP transmission window increases at a rate slower than NBP's f.ingressRate increases. Therefore, NBP normally does not regulate TCP flows.

However, when congestion occurs, NBP reacts first by reducing f.ingressRate and, therefore, reducing the rate at which TCP packets are allowed to enter the network. TCP eventually detects the congestion (either by losing packets or due to longer round-trip times) and then promptly reduces its transmission rate. From this time point on, f.ingress Rate becomes greater than TCP's transmission rate, and therefore, NBP's congestion control does not regulate TCP sources until congestion happens again.

2.2 Leaky Bucket Algorithm:

The leaky bucket algorithm is used to regulate the traffic flow from the input port to the output port. We assume leaky bucket as a bucket with a small hole at the bottom. Hence

any packet that enters the bucket at any rate must go out of the bucket at a controlled rate from the hole at the bottom. Also we assume that the limit of the bucket is infinity. Hence there is no case of bucket getting filled and the packets getting lost due to the limit of the bucket.

2.3 Feedback Control Algorithm:

The feedback control algorithm in NBP determines how and when feedback packets are exchanged between edge routers. Feedback packets take the form of ICMP packets and are necessary in NBP for three reasons. First, forward feedback packets allow egress routers to discover which ingress routers are acting as sources for each of the flows they are monitoring.

Second, backward feedback packets allow egress routers to communicate per-flow bit rates to ingress routers. Third, forward and backward feedback packets allow ingress routers to detect incipient network congestion by monitoring edge-to-edge round-trip times.

2.4 Network Border Patrol:

Network Border Patrol is a core-stateless congestion avoidance mechanism. That is, it is aligned with the core-stateless approach [7], which allows routers on the borders (or edges) of a network to perform flow classification and maintain per-flow state but does not allow routers at the core of the network to do so.

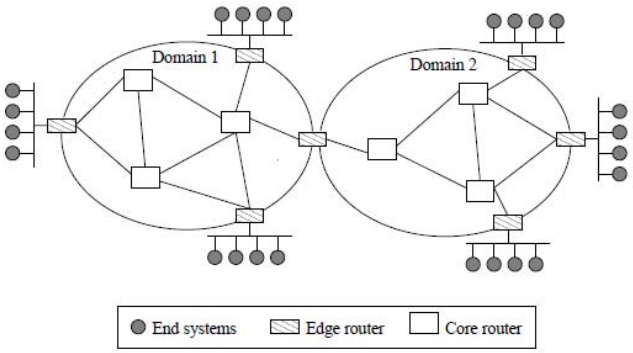

Figure. 1. The core-stateless Internet architecture assumed by NBP

Figure 1 illustrates this architecture. In this paper, we draw a further distinction between two types of edge routers. Depending on which flow it is operating on, an edge router may be viewed as ingress or an egress router. An edge router operating on a flow passing into a network is called an ingress router, whereas an edge router operating on a flow passing out of a network is called an egress router. Note that a flow may pass through more than one egress (or ingress) router if the end-to-end path crosses multiple networks.

NBP prevents congestion collapse through a combination of per-flow rate monitoring at egress routers and per-flow rate control at ingress routers. [8] Rate monitoring allows an egress router to determine how rapidly each flow's packets are leaving the network, whereas rate control allows an ingress router to police the rate at which each flow's packets enter the network. Linking these two functions together are the feedback packets exchanged between ingress and egress routers; ingress routers send egress routers forward feedback packets to inform them about the flows that are being rate controlled, and egress routers send ingress routers backward Feedback packets to inform them about the rates at which each flow's packets are leaving the network.

3. PROBLEM DEFINITION

3.1 Existing system:

As a result of its strict adherence to end-to-end congestion control, the current Internet suffers from two maladies: Congestion collapse from undelivered packets and unfair allocations of bandwidth between competing traffic flows.

The first malady congestion collapse from undelivered packets arises when packets that are dropped before reaching their ultimate continually consume bandwidth destinations. [9]

The second malady unfair bandwidth allocation to competing network flows arises in the Internet for a variety of reasons, one of which is the existence of applications that do not respond properly to congestion. Adaptive applications (e.g., TCP-based applications) that respond to congestion by rapidly reducing their transmission rates are likely to receive unfairly small bandwidth allocations when competing with unresponsive applications. The Internet protocols themselves can also introduce unfairness. The TCP algorithm, for instance, inherently causes each TCP flow to receive a bandwidth that is inversely proportional to its round-trip time [6]. Hence, TCP connections with short round-trip times may receive unfairly large allocations of network bandwidth when compared to connections with longer round-trip times.

The impact of emerging streaming media traffic on traditional data traffic is of growing concern in the Internet community. Streaming media traffic is unresponsive to the congestion in a network, and it can aggravate congestion collapse and unfair bandwidth allocation.

3.2 Proposed system:

To address the maladies of congestion collapse we introduce and investigate a novel Internet traffic control protocol called Congestion Free Router (CFR). The basic principle of CFR is to compare, at the borders of a network, the rates at which packets from each application flow are entering and leaving the network. If a flow's packets are entering the network faster than they are leaving it, then the network is likely buffering or, worse yet, discarding the flow's packets. In other words, the network is receiving more packets than it is capable of handling. CFR prevents this scenario by "patrolling" the network's borders, ensuring that each flow's packets do not enter the network at a rate greater than they are able to leave the network. This patrolling prevents congestion collapse from undelivered packets; because unresponsive flow's otherwise undeliverable packets never enter the network in the first place.

Although CFR is capable of preventing congestion collapse and improving the fairness of bandwidth allocations, these improvements do not come for free. CFR solves these problems at the expense of some additional network complexity, since routers at the border of the network are expected to monitor and control the rates of individual flows in CFR. CFR also introduces added communication overhead, since in order for an edge outer to know the rate at which its packets are leaving the network, it must exchange feedback with other edge routers. Moreover, end systems operate in total ignorance of the fact that CFR is implemented in the network, so no changes to transport protocols are necessary at end systems.

4. Architectural Components

The only components of the network that require modification by CFR are edge routers; the input ports of OutRouter routers must be modified to perform per-flow monitoring of bit rates, and the output ports of InRouter routers must be modified to perform per-flow rate control.[5] In addition, both the InRouter and the OutRouter routers must be modified to exchange and handle CFR feedback packets.

The input ports of OutRouter routers are enhanced in CFR. Fig. 2 illustrates the architecture of an OutRouter router's input port. Data packets sent by InRouter routers arrive at the input port of the OutRouter router and are first classified by flow. Flow classification is performed by InRouter routers on every arriving packet based upon a flow classification policy.

An example flow classification policy is to examine the packet's source and destination network addresses, and to aggregate all packets arriving on an InRouter router and destined to the same OutRouter router into the same CFR flow (i.e., a macro-flow).

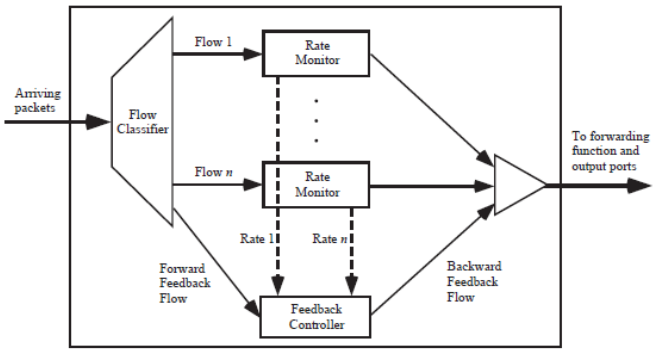

Figure. 2. An input port of an NBP egress router

Other flow classification policies can be used, for instance, in the case of IPv6, flows may be classified by examining the packet header's flow label, whereas in the case of IPv4, it could be done by examining the packet's source and destination addresses and port numbers.

After classifying packets into flows, each flow's bit rate is then rate monitored using a rate estimation algorithm such as the Time Sliding Window (TSW) algorithm. These rates are collected by a feedback controller, which returns them in backward feedback packets to an InRouter router whenever a forward feedback packet arrives from that InRouter router.

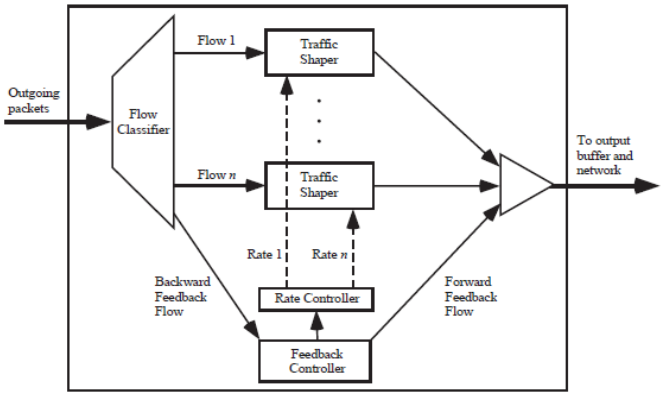

Figure. 3. An output port of an NBP ingress router

The output ports of InRouter routers are also enhanced in CFR. Each output port contains a flow classifier; per-flow traffic shapers (e.g., leaky buckets), a feedback controller, and a rate controller (see Fig. 3). The flow classifier classifies packets into flows, and the traffic shapers limit the rates at which packets from individual flows enter the network. The feedback controller receives backward feedback packets returning from OutRouter routers and passes their contents to the rate controller. It also generates forward feedback packets that are transmitted to the network's OutRouter routers.

5. Implementation

The various modules in the protocol are as follows:

A. Source Module:-

The task of this Module is to send the packet to the InRouter router.

B. InRouter Router Module:-

An edge router operating on a flow passing into a network is called an InRouter router. CFR prevents congestion collapse through a combination of per-flow rate monitoring at OutRouter routers and per-flow rate control at InRouter routers. Rate control allows an InRouter router to police the rate at which each flow's packets enter the network. InRouter Router contains a flow classifier, per-flow traffic shapers (e.g., leaky buckets), a feedback controller, and a rate controller

C. Router Module:-

The task of this Module is to accept the packet from the InRouter router and send it to the OutRouter router.

D. OutRouter Router Module:-

An edge router operating on a flow passing out of a network is called an OutRouter router. CFR prevents congestion collapse through a combination of per-flow rate monitoring at OutRouter routers and per-flow rate control at InRouter routers. Rate monitoring allows an OutRouter router to determine how rapidly each flow's packets are leaving the network. Rate monitored using a rate estimation algorithm such as

the Time Sliding Window (TSW) algorithm. OutRouter Router contains a flow classifier, Rate monitor, and a feedback controller.

E. Destination Module:-

The task of this Module is to accept the packet from the OutRouter router and stored in a file in the Destination machine.

6. Conclusion

Using network border protocol framework we have presented a novel congestion-avoidance mechanism for the Internet. Unlike the existing primitive Internet congestion control schemes, which were solely on end-to-end control, the NBP framework is able to prevent congestion collapse from undelivered packets. The enhanced core-stateless fair queuing complements NBP framework by providing fair bandwidth allocations in a core-stateless fashion.

NBP framework prevents congestion collapse through a combination of per-flow rate monitoring at OutRouter routers and per-flow rate control at InRouter routers. Rate monitoring allows an OutRouter router to determine how rapidly each flow's packets are leaving the network, whereas rate control allows an InRouter router to police the rate at which each flow's packets enter the network. Linking these two functions together are the feedback packets exchanged between InRouter and OutRouter routers; InRouter routers send OutRouter routers forward feedback packets to inform them about the flows that are being rate controlled, and OutRouter routers send InRouter routers backward feedback packets to inform them about the rates at which each flow's packets are leaving the network. By matching the InRouter rate and OutRouter rate of each flow, CFR prevents congestion collapse within the network.

References

[1] S. Floyd and K. Fall, "Promoting the Use of End-to-End Congestion Control in the Internet," IEEE/ACM Transactions on Networking, August 1999, To appear.

[2] J. Padhye,V. Firoiu, D. Towsley, and J. Kurose, "ModelingTCP Throughput: A SimpleModel and its Empirical Validation," in Proc. of ACM SIGCOMM, September 1998, pp. 303–314.

[3] B. Suter, T.V. Lakshman, D. Stiliadis, and A. Choudhury, "Design Considerations for Supporting TCP with Per-Flow Queueing," in Proc. of IEEE Infocom '98, March 1998, pp. 299–305.

[4] B. Braden et al., "Recommendations on Queue Management andCongestion Avoidance in the Internet," RFC 2309, IETF, April 1998.

[5] A. Demers, S. Keshav, and S. Shenker, "Analysis and Simulation of a Fair Queueing Algorithm," in Proc. of ACM SIGCOMM, September 1989, pp.1–12.

[6] A. Parekh and R. Gallager, "A Generalized Processor Sharing Approach to Flow Control – the Single Node Case," IEEE/ACM Transactions on Networking, vol. 1, no. 3, pp. 344–357, June 1993.

[7] I. Stoica, S. Shenker, and H. Zhang, "Core-Stateless Fair Queueing: Achieving Approximately Fair Bandwidth Allocations in High Speed Networks," in Proc. of ACM SIGCOMM, September 1998, pp. 118–130.

[8] D. Lin and R. Morris, "Dynamics of Random Early Detection," in Proc. Of ACM SIGCOMM, September 1997, pp. 127–137.

[9] D. Bertsekas and R. Gallager, Data Networks, second edition, Prentice Hall, 1987.

Ashakiran.G.N received the B.E. degree in Computer Science and Engineering from Sri Revana Siddeshwara Institute of Technology, Bangalore. At present persuing the Master of Technology in Computer Science and Engineering Department at BTL institute of Technology, Bangalore.

M.V.Panduranga Rao is a research scholar at National Institute of Technology Karnataka, Mangalore, India. His research interests are in the field of Real time and Embedded systems on Linux platform and Security. He has published various research papers across India and in IEEE international conference in Okinawa, Japan. He has also authored two reference books on Linux Internals. He is the Life member of Indian Society for Technical Education and IAENG.

S.Basavaraj. Patil Started career as Faculty Member in Vijayanagar Engineering College, Bellary (1993-1997).Then moved to Kuvempu University BDT College of Engineering as a Faculty member. During this period (1997-2004), carried out Ph.D. Research work on Neural Network based Techniques for Pattern Recognition in the area Computer Science & Engineering. Also consulted many software companies (ArisGlobal, Manthan Systems, etc.,) in the area of data mining and pattern recognition His areas of research interests are Neural Networks and Genetic Algorithms. He is presently mentoring two PhD and one MSc (Engg by Research) Students. He is presently working as professor at BTL institute of technology.