Fair Intelligent Congestion Control Resource Discovery Protocol on TCP Based Network

Doan B. Hoang[1], Qing Yu[2], Ming Li[1], and David Dagan Feng[2]
[1]*Department of Computer Systems, Faculty of Information Technology, University of Technology, Sydney, NSW 2007, Australia,*
e-mail:[dhoang, mingli]@it.uts.edu.au

[2]*School of Information Technologies, University of Sydney, NSW 2006, Australia,*
e-mail:[qing, feng]@it.usyd.edu.au

Abstract: Today's Internet only provides best-effort service for all traffics. The network is not able to guarantee the quality of service required by an application that demands more stringent response in terms of delay, jitters, bandwidth and etc. It is well accepted that, the deployment of QoS-aware technologies is a key factor for the continued success of the Internet. In this paper, we propose a Fair Intelligent Congestion Control Resource Discovery (FICCRD) protocol on TCP based network whereby a mechanism is employed at core routers to determine available network resources and convey this information to edge routers. At the edge routers, an intelligent control algorithm is employed to assist the TCP to maximize its traffic over the underlying network. The key ideas are to integrate available network resources in estimating connections' fair share; to create feedback control loops between edge routers; to employ a special Resource Discovery (RD) packet to collect and convey en route router state information; and to employ intelligent algorithms to match a TCP connection's sending rate to the rate at which the underlying network can support. We demonstrate that FICCRD protocol is effective, fair, flexible and can be easily extended for QoS control of the future Internet.

Key words: TCP, Congestion Control, Explicit Window Adaptation, Acknowledgment Bucket Control, Feedback Control Loop

---

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35673-0_28
C. McDonald (ed.), *Converged Networking*

1. INTRODUCTION

Today's Internet only provides best-effort service for all traffic. Traffic is processed as quickly as possible but there is no guarantee as to timeliness or actual delivery. The network makes no attempt to differentiate its service response between the traffic streams generated by concurrent users of the network. This means that the network is not able to guarantee the level of service required by an application that demands more stringent response in terms of delay, jitters, bandwidth, etc. A simple application such as videoconference over the Internet will quickly demonstrate the problems with real-time applications. A person taking part in a videoconference will immediately realize that the quality of the images is not as good as expected, the flow of images is not quite smooth and the synchronization between voice and images is far from perfect.

The evolution of the Internet is at a turning point. It is well accepted that, in its evolution in the 21st century, the deployment of Quality of Service (QoS) aware technologies is a key factor for the continued success of the Internet. Some advances in this direction have already been provided with the introduction of the Internet augmented architectures with QoS supports: the Internet Protocol Version 6 (IPV6), the Integrated Services [1] and Differentiated Services [2] by the IETF. Furthermore, the ReSerVation Protocol (RSVP) [3] and the Multi Protocol Label Switching (MPLS) technique [4] provide additional support in terms of signalling and traffic engineering. At the infrastructure level, the deployment of Wavelength Division Multiplexing (WDM) systems creates a huge transport capacity, which alleviates the bandwidth demand of multimedia applications over the Internet.

Despite these advances, many issues still remain to be solved. Among them, issues concerning admission control and resource allocation policies in the network, adequate user signalling protocols, network management systems for large numbers of users and security require closer attention. It is expected that, having in place the adequate infrastructure and protocols, high quality real-time applications can run on the Internet without significant interference from conventional data traffic.

Our approach is motivated by recent technological changes and the need for the adequate infrastructure and protocols towards the QoS-aware network. We believe that for effective QoS-aware technologies, QoS issues have to be tackled at several levels. At the network level, relevant information concerning the network operational conditions and availability of resources of the underlying network is essential to avoid network congestion. At the transport level, relevant information concerning the end systems is crucial for fair bandwidth sharing and end-to-end QoS per

connection. At the application level, relevant information concerning the end applications are necessary to provide appropriate QoS for the application.

In this paper we develop a feedback control loop at the network layer to provide the transport protocol with necessary information for optimising its operation. Several algorithms are employed to maximise the TCP sending rate of a connection over the available bandwidth of the underlying network. In particular, we propose Fair Intelligent Congestion Control Resource Discovery protocol (FICCRD) on TCP based network in which a mechanism is employed at core routers to determine available network resources and convey this information to edge routers. At the edge routers, an intelligent control algorithm is employed to assist the TCP to maximize its traffic over the underlying network. The key ideas are to integrate available network resources in estimating connections' fair share of network resource; to create feedback control loops between edge routers; to employ a special Resource Discovery (RD) packet to collect and convey en route router state information; and to employ intelligent algorithms to match a TCP connection's sending rate to the rate at which the underlying network can support. We demonstrate that the framework can significantly improve in throughput, fairness, and packet loss rate for end-to-end TCP connections. More importantly, our protocol is transparent to TCP, requires no modifications to current TCP implementations and can be easily extended for QoS control of the future Internet.

The paper is organized as follows. Section 2 describes some related work. Our proposal is presented in section 3. Section 4 describes simulation scenario, simulation environment and simulation parameters. The simulation results and evaluation are presented in section 5. Some concluding remarks and direction of future work are given in section 6.

2. RELATED WORK

A computer network typically uses store-and-forward routing to transfer data packets between users at geographically distributed nodes. Packets generated by a source node are delivered to their destination by routing them via a sequence of intermediate nodes. The traffic flowing through an intermediate node depends upon the number of source-destination pairs that are routed through that node and the rates at which these sources introduce packets into the network. If the source rates are increased without constraint, queues of packets waiting to be routed build up at bottleneck nodes. Eventually, the buffering capacity of these nodes is exceeded and packets are dropped, resulting in low throughput and high delay.

Congestion control mechanisms attempt to avoid such breakdown by imposing constraints on the sources. Two types of constraint are often used. In rate-based congestion control, a limit is placed on the rate at which a source can send packets. In window-based congestion control, at any instant there is a limit to the number of outstanding packets at the source, but there is no constraint on the rate at which packets can be sent.

In ATM network, the Available Bit Rate (ABR) service is adopted for transporting best-effort data traffic with no delay guarantees; ABR rate-based congestion control attempts to minimize the cell loss ratio, and provide minimum cell rate guarantees through the closed loop feedback control mechanism. The network provides feedback to the sources when network load changes, and the sources adjust their transmission rates accordingly. ABR congestion control handles congestion effectively. However, network congestion is not really eliminated but rather it is pushed out to the edge of the ATM network. Packets from TCP sources competing for the available ATM bandwidth are buffered in the routers or switches at the network edges, causing severe congestion, degraded throughput, and unfairness.

In IP networks, Floyd [5] proposed to modify TCP slightly to include an Explicit Congestion Notification (ECN) from routers to the sources to trigger a window size reduction identical to that caused by the fast retransmit – fast recovery mechanism of TCP-Reno, yet without dropping packets. By combining explicit notification with RED, the performance of both delay-sensitive (telnet-like) and delay-insensitive (ftp-like) traffic can be improved. However, because of its binary feedback, ECN cannot avoid window and network oscillations. These can negate the gain in network performance. Other approaches such as Tri-S [6] and TCP-Vegas [7] attempt to estimate the bandwidth-delay product for each TCP connection and adjust the window size based on this estimate. However, these schemes introduce complexity in the end-system and require extensive modifications to current TCP implementations.

Gerla et al. proposed a feedback-based algorithm (BA-TCP) [8] at network layer to convey the round-trip propagation delay and available bandwidth information for each TCP connection. The end hosts use this information to adjust their congestion window. However, knowledge of the RTT is usually not available at current router. The scheme also needs to modify current TCP implementation by adding one state variable to store the round-trip propagation delay and advertising the minimum of the receiver's buffer space and the available bandwidth-delay product.

Hjalmtysson [9] described a control-on-demand model for programmable networks. This model allows the installed control programs to exploit lower-level facilities, in particular hardware facilities. Through filtering and frame

peeking, control programs can inspect "flows" or "aggregate" or "control" stream of Internet and then extract the required information for intelligent control. Harission and Kalyanaraman [10] proposed edge-to-edge feedback-based control algorithm at network layer to regulate the aggregate traffic between each edge-pair. However, the scheme's requirement on routers is high. It may be unrealistic to expect routers to devote much of their resources to exercising the algorithm and handing the admission control.

3. FAIR INTELLIGENT CONGESTION CONTROL RESOURCE DISCOVERY PROTOCOL

In this section, we provide details of the proposed framework and show how the network operates under the proposed framework. The aim of FICCRD protocol is twofold. Firstly, it aims to develop a feedback loop in WAN region between the edge routers to convey information concerning available network resources and network conditions from within the network. We have deployed a similar scheme, the Fair Intelligent Congestion Control (FICC) [11, 12], successfully for ATM's ABR congestion control. In this paper the innovative aspect is in investigating how such a feedback scheme can be useful over the Internet. Secondly, it aims to maximize a TCP connection throughput by matching its sending rate to the rate at which the underlying network can support. We aim to establish a feasible framework over which explicit feedback information can be conveyed. Quality of Service (QoS) and congestion control mechanisms can then be deployed at the edge devices based on the feedback information. Our initial implementation is concerned with individual connections to see how TCP operates under this scheme. Our next step is to investigate the scalability of the scheme by applying it on a per-class basis as specified by the differentiated services code point (DSCP) in DiffServ.

Basically the control loop operates as follows. A special resource discovery (RD) packet is generated at source edge router for each flow proportionally based on the packet arrival rate. The RD-packet will carry the arrival packet rate (or its estimate) in its APR field. Each core router along the path directly updates feedback information concerning the network conditions in the RD packets when they pass in the forward or backward direction. The destination edge router sends back the RD-packets. When the source edge router receives Backward RD-packet, it passes the feedback information to an algorithm that is responsible for matching the TCP sending rate to the expected rate supported by the network.

3.1 Edge Router behaviours

As far as the FICCRD is concerned, an edge router is responsible for estimating packet arrival rate for its connection, initiating and maintaining the RD control loop, collecting the feedback information, and exercising control algorithm for coordinating control between TCP and the underlying network. Edge router model is described in Figure 1.

3.1.1 Forwarding behaviours

In the forward direction, the source edge router classifies the incoming packets into flows and the arrival packet rate of each flow denoted as $APR_i(t)$. Since an exact computation of the arrived rate of each flow is hardly feasible, a per-flow rate estimate $\hat{APR}_i(t)$ is updated upon the reception of every packet using exponential averaging formula as in CSFQ [13]. Using an exponential weight gives more reliable estimation for bursty traffic, even when the packet inter-arrival time has significant variance.

If we indicate the arrival time of the k-th packet of flow i as T_i^K and its length as $l_i^k(t)$, the new estimate of $\hat{APR}_i(t)$ can be computed as follows:

$$APR_i^{new}(t) = (1 - e^{-T_i^k / K}) \frac{l_i^k}{T_i^k} + e^{-T_i^k / K} APR_i^{old} \quad (1)$$

Where T_i^K represents the k-th sample of the interarrival time of flow i, i.e., $T_i^K = t_i^k - t_i^{(k-1)}$ and K is a constant. RD packets are generated at source edge router for each flow proportionally based on the packet arrival rate $\hat{APR}_i(t)$ in Eqn. (1). RD-packet will carry the arrival packet rate $\hat{APR}_i(t)$ in its APR field.

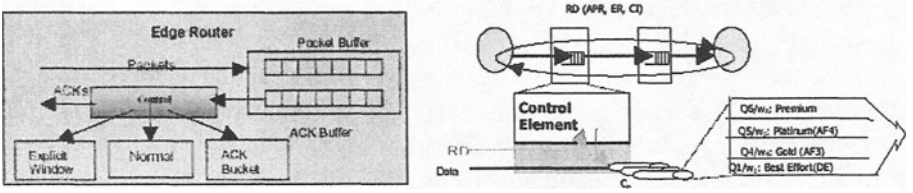

Figure 1. [Edge Router & Core Router Models]

3.1.2 Backwarding behaviours

In the backward direction, two approaches (Explicit Window Adaptation, ACK Bucket Control) can be employed at source edge router to convey such

feedback information concerning the network conditions to TCP source. The Explicit Window Adaptation approach controls the maximum receiver window (RWND) at edge router, which acts as an upper bound on the TCP congestion window (CWND) variable at the TCP source, thus effectively controls the TCP sender rate by matching it with the feedback information concerning the network conditions at the network layer. The ACK Bucket Control approach aims to match the TCP sender rate with the available feedback resources by withholding the Acknowledgments at the edge router. We have recently proposed a Fair Intelligent Explicit Window Adaptation (FIEWA) scheme and a Fair Intelligent ACK Bucket Control (FIABC) scheme for each of the two approaches [14, 15]. The essential ideas of these schemes are summarized below.

3.1.2.1 Fair Intelligent Explicit Window Adaptation

The objective of Fair Intelligent Explicit Window Adaptation (FIEWA) is to match the sum of the windows of all active TCP connections sharing the buffer at the edge router to the effective network delay-bandwidth product, thus avoiding packet losses whenever possible. Two key components are included in FIEWA: (i) A mechanism to signal window updates from the network to the source, and (ii) a scheme at the bottleneck point to estimate the available bandwidth based on the congestion state of the network. The former can be accomplished by allowing the network edge elements to modify the receiver's advertised window field carried by TCP acknowledgements from the destination to the source. The latter problem, however, is significantly more difficult. First, when there are multiple bottlenecks on the path of a TCP connection, the window estimation algorithms at these bottlenecks may interact in undesirable ways. Second, estimating the available bandwidth of an output link of a bottleneck element is rather difficult. Finally the advantages of the scheme must be compared against those of implicit (packet discard) schemes of comparable complexity.

For the specific network environment under consideration, the situation is much simpler. First, we assume that the congestion between source edge router to destination edge router is well control; the only bottleneck in the path of the TCP connections occurs at the source edger router. Second, both the bandwidth available in the edge router, and the delay through it, remain relatively steady over short timescales, independent of the number of TCP connections transported over it. This makes it easier for the edge router to estimate the available bandwidth-delay product for each TCP connection.

Basically, the FIEWA works as follows. FIEWA sends explicit feedback information in the form the receiver's advertised window field of the returning TCP acknowledgments to TCP sources to adjust their window

sizes. If the current value in the receiver's advertised window, which is set by the destination system, exceeds the feedback value computed in the edge router, the receiver's advertised window is marked down to the feedback value. The computed feedback value is a function of the free buffer space at the edge router.

Let $Q_e(t) = BufferSize - Q(t)$ denote the empty buffer space at time t when a returning ACK arrives at an edge device, where $BufferSize$ is the total buffer space and $Q(t)$ the total buffer occupancy at time t. Let $W_r(t)$ denote the value in the receiver's advertised window field seen in the ACK. The FIEWA algorithm computes a target window size for the TCP connection as a function of the available buffer, that is $f(Q_e(t))$. This computed value is then used to mark down the receiver's advertised window field in the acknowledgement. Since setting the window size smaller than the maximum segment size (MSS) negotiated during connection establishment can lead to starvation and deadlocks, a minimum window size of MSS is enforced. Thus, the feedback value, $W_r'(t)$, used to set the receiver's advertised window field, is computed at the edge device as

$$W_r'(t) = MAX\ (MIN\ (W_r(t), f(Q_e(t))), MSS\ ) \quad (2)$$

The difficult task in such an algorithm is to design the feedback function $f(Q_e(t))$. The dynamics of the system depend heavily on this feedback function. The goal of the function is to provide all TCP connections with similar feedback, and as a result they all operate with equal windows. Since a linear function is simple and still effective, it is employed in our feedback function. Explicitly, the window for each connection is set to the available buffer space multiplied by a fraction which determines the buffer occupancy in steady state. That is

$$f(Q_e(t)) = DPF\ (t) * Q_e(t) = DPF\ (t) * (BufferSize\ - Q(t)) \quad (3)$$

3.1.2.2 Fair Intelligent ACK Bucket Control

The objective of Fair Intelligent ACK Bucket Control scheme (FIABC) is to match the TCP sender rate with the available bandwidth in underlying network by withholding the acknowledgment packets at edge router, so that TCP source cannot send packets more than that can be handled by the underlying network. We choose to withhold ACK packets at edge router based on the following considerations. Since the essential information contained in an acknowledgment is only a sequence number, this "acknowledgment buffer" is really only a list of numbers. To differentiate this type of buffering from the traditional packet buffer, we will call it

acknowledgment bucket. By controlling the flow of acknowledgments from the destination to the source, we can practically eliminate the large packet buffer and still achieve the same output rate as if we have unlimited packet buffer at the network edges.

The acknowledgment bucket is analogous to the storage of permits in the well-known leaky bucket scheme. TCP acknowledgments serve as permits that allow the edge device to request packets from the source. When the acknowledgment bucket is empty, the edge cannot request any more packets. The acknowledgment bucket gets filled up according to TCP dynamics and is drained out according to the available bandwidth in the underlying network. The acknowledgment bucket serves as a translator, which transforms the available bandwidth command into a sequence of TCP acknowledgments whose effect is to have the TCP source sent data no faster than the available bandwidth.

The difficult task in such an algorithm is to determine the forwarding rate for the acknowledgments. Since the edge device serves as a virtual source for the underlying network, our approach determines the acknowledgment packet releasing by comparing the request from underlying network with current edge device buffer length. In detail, our approach is to estimate the amount of data requested by underlying network from last ACK release at edge device corresponding the available bandwidth denoted as "AB". That is, the amount of data for each connection required by underlying network at time t_i, denoted as $R_{Underlying}(t_i)$, is set to $AB(t_i)*(t_i - t_{lastACKrelease})$ multiplied by a queue control function DPF that determines edge router buffer occupancy in steady state.

$$R_{Underlying}(t_i) = DPF(t_i) * AB(t_i) * (t_i - t_{lastACKrel\ ease}) \quad (4)$$

Note that the amount of data requested by the underlying network $R_{Underlying}(t_i)$ from last ACK release must be smaller than the advertisement window size in the latest ACK packet received by edge router.

$$R_{underlying}(t_i) = Min(R_{underlying}(t_i), RWND) \quad (5)$$

We denote $Q(t_i)$ the packet queue length at time t_i at edge device. Since the packet queue at edge device services as a virtual source for the underlying network, FIABC schedule the ACK releasing from the ACK Bucket by comparing currently what the underlying network request the packets from the edge device $R_{Underlying}(t_i)$ with currently what edge device

can provide in its packet queue $Q(t_i)$. Whenever $R_{Underlying}(t_i) > Q(t_i)$, the edge device will release an acknowledgment from the ACK Bucket.

3.2 Core Router Behaviours

We employed the Fair Intelligent Congestion Control (FICC) scheme [12] at each core router to estimate a fair share of bandwidth for competing TCP connections, calculate available network capacity and feedback relevant information to the edge router. It has been demonstrated that FICC is simple, robust, and effective congestion control algorithm. Importantly, FICC is also able to allocate bandwidth fairly among its connections. Core router model is described in Figure 1.

Essentially, in order to estimate the current traffic generation rate of the network and allocate it among connections fairly, a Mean Allowed Packet Rate (MAPR) is kept at each core router.

$$MAPR = MAPR + \beta * (APR - MAPR) \tag{6}$$

Where APR is the value of the current arrival packet rate carried in the APR field of the arriving forward RD-packet. MAPR represents an estimate of the average load passing through the router at the current time. When the network operates at the acceptable level, the correspondent MAPR is regarded as the optimal packet rate for each flow.

In Fair Intelligent Congestion Control mechanism, the network is expected to work at the target operating point. The target operating point adopted in this scheme is a pre-set Buffer Utilization Ratio (BUR), which means that the optimal control is to keep the buffer utilization at an optimal level. The motivation behind this idea is to make efficient use of the buffer capacity.

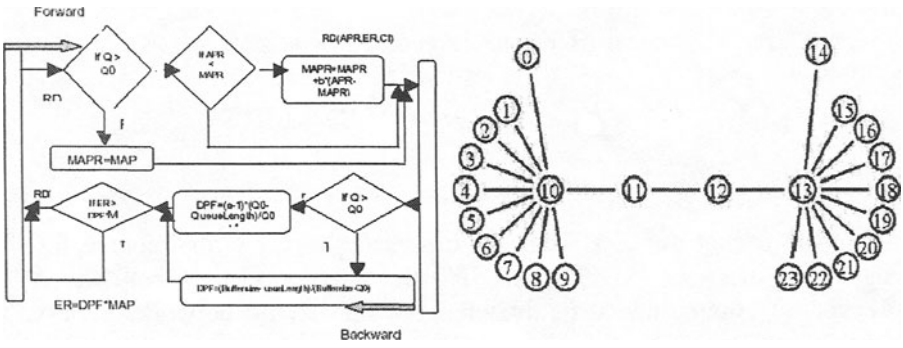

Figure 2. [Core router's Algorithm & Peer to Peer Configuration]

To calculate the expected rate (ER) based on the queue length at core router, a linear queue control function DPF is employed in our scheme. The basic characteristics of the function are that it has a value equal to 1 when the queue length is target queue length Q0, and a value less/larger than 1 when the queue length is larger/less than Q0. The larger/smaller the queue length, the smaller/larger the factor to push forward the network to the target operating point. Since queue is built up and drained out continuously, queue control function is desired to perform continuous control to produce proper effect on the queue fluctuation and smooth the computed ER values. The pseudocode of FICCRD is shown in Figure 2. Refer to [11, 16] for further description of the algorithm.

4. SIMULATION SETUP

We use ns (version 2) [17] network simulator to evaluate the proposed FICCRD protocol. Ns is a discrete event simulator for network research. It provides substantial support for TCP, router queuing mechanisms, and various topologies. New components – edge router, core router and new protocol – FICCRD were added and compiled into ns.

The simulation performs with TCP applications running over an IP network. Peer-to-Peer configuration [Figure 2] is employed in the simulation. There are 10 sources sending data to 10 distinct destinations through a single bottleneck link between 11 and 12 with buffersize of 200pkts, propagation delay of 20 ms and bandwidth of 10 Mbits/s. Source 0 sends data to the destination 14 (on path 10, 11, 12, 13) has propagation delay of 10 ms and bandwidth of 10 Mbits/s. All the other links are wired with propagation delay of 2 ms and bandwidth of 10Mbits/s. The point we want to look at is the fair share of bandwidth among multiple connections with different RTT sharing a bottleneck link. Each source has an infinite data to send. The size of data packets is 4Kbyte. The simulation is run for 200 seconds. TCP clock granularity is set to 0.3 seconds and the receiver's window size is set to128 Kbytes. The TCP version used in our simulations was TCP Reno, which includes fast retransmission and fast recovery.

5. SIMULATION RESULTS AND ANALYSIS

In this section we present the simulation results for FICCRD (denoted as "FICCRD (FIEWA)" and "FICCRD (FIABC)"), simple Droptail (denoted as "Droptail"), RED (denoted as "RED") and ECN with RED (denoted as "RED+ECN") for the purpose of comparison and discussion. We evaluate

the simulation results in terms of the goodput, the sender sequence number, the packet queue length, and the dropping total at the bottleneck core router.

5.1 Packet Queue Length

The main reason for TCP performance degradation is due to the overflow of buffers at bottleneck router. An important attribute of the TCP congestion control mechanism is that it does not assume any explicit signaling of congestion state from the underlying network for. It infers the congestion state of the network implicitly: the arrival of acknowledgements (ACKs), timeouts, and receipt of the duplicate ACKs. As a result, the window-based congestion control mechanisms of TCP may interact with the underlying network in causing severe congestion, degraded throughput, and unfairness.

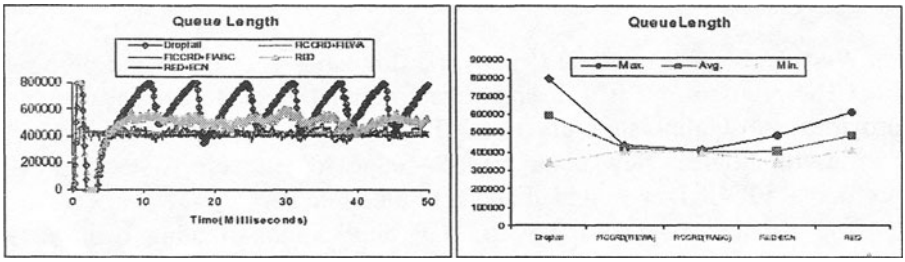

Figure 3. [Bottleneck Queue Length]

As shown in Figure 3, with "Droptail", the packet queue length grows beyond the maximum packet buffer size of the bottleneck router, and packets have to be dropped, packet retransmission occurs when the TCP source becomes aware of the loss of packet. With "RED", two fixed thresholds (high threshold and low threshold) are used to detect congestion, different dropping policies are employed for the congestion period when the queue length is greater than the high threshold and the congestion period when the queue length is between the two thresholds. Both "Droptail" and "RED" use dropping mechanism at router to indicate the congestion and control TCP sending rate implicitly. The queue length variation is between 300000 bytes to 800000 bytes for "Droptail" and between 400000 bytes t0 600000 bytes for "RED". By combining ECN with RED, the bottleneck router can set the congestion bit in the ACK return back to TCP, thus providing TCP an explicitly information to controlling its sending rate.

Rather than using a fixed queue threshold to arbitrarily divide a network into congestion and non-congestion states, our FICC algorithm aims for a target operating point where the router queue length is at an optimal level for good throughput and low delay, and where the allocation is optimal for each

connection. The queue control function encourages traffic sources if the target operating point is not reached, and discourages the sources if the switch operates beyond its target operating point. As shown in Figure 3, FICCRD maintains the queue level at the target level around 400000 bytes with narrow range ([390000 bytes, 410000 bytes]).

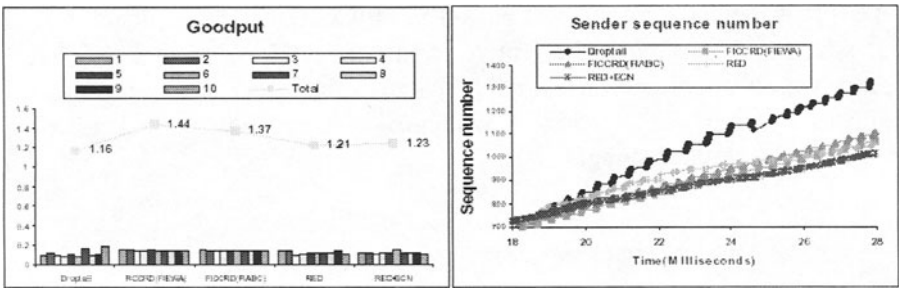

Figure 4. [Goodput & Sender Sequence Number]

5.2 Goodput

We use the current acknowledged bytes divided by the simulation duration time to calculate the goodput. Figure 4 shows the goodput for each flow during intervals 50 – 200 seconds. The goodput shown in this figure has been normalized by the bottleneck fair share (i.e., by 0.15 Mbits/s). Therefore, a goodput of 0.15 represents a data throughput of 0.15 Mbits/s. The figure shows an unfair allocation of bandwidth when other schemes are used. In contrast, the throughput obtained by FICCRD is fair in that all TCP connections are allocated roughly the same amount of bandwidth. We notice the average effective throughput shows a slightly lower than the estimated value of 0.15Mbits/s. This mainly comes from the inefficiency of the TCP slow-start, the overhead costs concerning packet headers, the paddings, and the bandwidth required for RD packets.

The fairness property of FICCRD comes from three main factors: Firstly, at each core router, FICCRD estimates accurately the fair share of its connections. Secondly, FICCRD builds in an oversell feature that allows unconstrained connections to take up the leftover bandwidth that cannot be taken up by constrained connections. This allows the bandwidth to be shared among unconstrained connections fairly. Thirdly, FICCRD does not allow the operating point to diverge far wary from its stable point. This results in fair share and small deviations in the router queue length, in packet delay and packet delay variation.

5.3 TCP Sender Sequence Number

We use TCP sender sequence number versus time to paint a picture of TCP sending rate as shown in Figure 4. Comparatively, it is clear that the TCP sending rate in FICCRD mechanisms is less than other cases before the packet dropping, which means that the burstiness of traffic is reduced based on explicit feedback rate in RD packet in "FICCRD". The feedback loop in our schemes is explicit and effective in that the TCP source rate is directly controlled to adapt to the available bandwidth, rather than progressively adjusted through TCP window flow control, which relies on packet drop as indication of network congestion.

6. CONCLUSION AND FUTURE WORK

This paper proposes the Fair Intelligent Congestion Control Resource Discovery (FICCRD) protocol for the purpose of improving end-to-end TCP performance by controlling the congestion and allocating fair share of bandwidth to competing TCP traffics. The essential idea of the Fair Intelligent Congestion Control Resource Discovery protocol is the introduction of a feedback loop and its associated protocol to collect relevant information about the underlying network between a source and a destination edge routers pair; and the use of the information to regulate the TCP congestion control and maximize its performance.

The significant contributions of FICCRD protocol include: a) integrating of network resources, such as available bandwidth, available buffer, queuing delay and jitter to estimate fair share of network resources among competing traffic, b) allowing possible integration of LAN and WAN together to provide end-to-end QoS for application by allowing the mapping of LAN request to appropriate QoS of WAN supported by feedback control mechanism, and c) permitting an admission control mechanism to make use of the feedback information to provide adequate level of control.

The performance evaluation is based on the goodput, fairness, buffer requirement, etc. The simulation results show that our scheme is effective in improving goodput, achieving fairness, and minimizing packet loss rate for end-to-end TCP connections. Importantly, our framework is transparent to TCP, requires no modifications to the current TCP implementation and can be easily extended for QoS control of the future Internet.

REFERENCES

[1] Clark, D., S. Shenker, and L. Zhang. *Supporting real-time applications in an integrated services packet networks: Architecture and mechanism.* in *Proceedings of ACM SIGCOMM.* 1992. Baltimore, USA.

[2] Stoica, I. and L. Zhang. *A model for service differentiation in the Internet.* in *NOSSDAV.* 1998.

[3] Zhang, L., *RSVP: A new resource reservation protocol.* IEEE Network, 1992. **7**(5): p. 8-18.

[4] Rosen, E., A. Viswanathan, and R. Callon, *Multiprotocol Label Switching Architecture*, in *IETF RFC 3031.* 2001.

[5] Floyd, S., *TCP and explicit congestion notification.* Computer Communication Review, 1994. **24**(5): p. 8-23.

[6] Wang, Z. and J. Crowcroft, *A new congestion control scheme: slow start and search (Tri-S).* Computer Communication Review, 1991. **21**(1): p. 32-43.

[7] Brakmo, L.S. and L. Peterson, *TCP Vegas: End to end congestion avoidance on global Internet.* IEEE Journal on Selected Areas in Communications, 1995. **13**(8): p. 1465-1480.

[8] Gerla, M., W. Weng, and R.L. Cigno. *Bandwidth feedback control of TCP and real time sources in the Internet,* in *Proceedings of IEEE GLOBECOM.* 2000. San Francisco, CA, USA.

[9] Hjalmtysson, G., *Control-on-Demand: An efficient approach to router programmability.* IEEE Journal on Selected Areas in Communications, 1999. **17**(9): p. 1549-1562.

[10] Harrison, D. and S. Kalyanaraman, *Edge-to-edge traffic control for the Internet: Concepts and architecture.* 2000, RPI ECSE Network Laboratory Technical Report.

[11] Hoang, D.B. and Z. Wang, *A Fair Intelligent Congestion Control for ATM.* 1999, Technical Report 224, ISBN 1864873264, School of Information Technologies, University of Sydney.

[12] Hoang, D.B. and Q. Yu. *Performance of the Fair Intelligent Congestion Control for TCP applications over ATM networks.* in *Proceedings of the Second International Conference on ATM (ICATM'99).* 1999. Colmar, France.

[13] Stoica, I., S. Shenker, and H. Zhang. *Core-stateless fair queueing: achieving approximately fair bandwidth allocations in high speed networks.* in *Procedings of the ACM SIGCOMM'98.* 1998. Vancouver, Canada.

[14] Yu, Q. and D.B. Hoang, *An intelligent coherent approach to cooperation between TCP and ATM congestion control.* Journal of the Society for Modelling and Simulation International, 2002. **78**(4): p. 258-267.

[15] Yu, Q. and D.B. Hoang. *Fair Intelligent Explicit Window Adaptation.* in *IC'2001.* 2001. Las Vegas, California, USA.

[16] Yu, Q., et al. *Fair intelligent feedback mechanism on TCP based network.* in *Proc. of the 2002 International Conference on Internet Computing (IC'2002),* June 2002. Las Vegas, California, USA, p. 1009-1015.

[17] LBL, *NS-2, Network simulation (version 2).* 2000, LBL.