Fair Intelligent Feedback Mechanism On TCP Based Network*

Qing Yu
School of Information Technologies
University of Sydney
Australia

Ming Li, Doan B. Hoang
Department of Computer Systems,
Faculty of IT,
University of Technology, Sydney,
Australia

David Dagan Feng
Department of Electronic and
Information Engineering
HongKong Polytechnic University,
HongKong

Abstract - Network performance degradation due to congestion is a major problem in the Internet. This performance degradation is mainly due to the uncoordinated interaction of congestion control mechanism of the TCP and that of the underlying network. Another challenging problem is that TCP cannot achieve fair bandwidth allocation among competing TCP traffics. In this paper, we propose a Fair Intelligent Congestion Control Resource Discovery (FICCRD) mechanism that can improve end-to-end TCP performance by controlling the congestion and allocating fair share of bandwidth among competing TCP traffics. The key ideas of FICCRD are to integrate available network resources in estimating connections' fair share of network resource; to create feedback control loops between edge routers; to introduce a protocol whereby a special Resource Discovery (RD) packet is employed to collect and convey en route router state information; and to employ intelligent algorithms to match a connection's TCP sending rate to the rate at which the underlying network can support. Simulation results show that the mechanism can significantly improve in throughput, fairness, and packet loss rate for TCP connections. More importantly, the mechanism is transparent to TCP and requires no modifications to current TCP implementations.

Keywords: TCP, Congestion Control, Buffer Management, Explicit Window Adaptation, Feedback Control Loop

1. Introduction

The majority of data services in current Internet are based on TCP (Transport Control Protocol). As an end-to-end transport protocol, TCP makes no assumption on how the network processes its data. It performs its own data recovery and flow control. It relies on packet loss to signal congestion condition. When a TCP source implicitly learns of a packet loss from the retransmission timeout, it reduces its sending rate to alleviate the condition. By the time the source starts decreasing its rate, the network may have already been overly congested. Another challenging problem for TCP based network is that strict fairness among competing TCP flows cannot be achieved at the TCP level [1-8].

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 [1], ATM 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. Many studies have demonstrated that ATM ABR service can provide low-delay, fairness, and high throughput, and can handle congestion effectively inside the ATM network. 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 [2] proposed to modify TCP slightly to include an Explicit Congestion Notification (ECN) from routers to sources to trigger a window size reduction identical to that caused by the fast retransmit - fast recovery mechanism of TCP-Reno, yet without

* This research is partially supported by ARC Grant.

requiring the drop of a packet. 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 negatively affect the network performance. Other approaches such as Tri-S [3] and TCP-Vegas [4] attempt to estimate the bandwidth-delay product for each TCP connection and adjust the window size based on this estimate. But 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) [5] 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.

Harrisson et al. [6] proposed an edge-to-edge feedback-based control algorithm at network layer to regulate the aggregate traffic between each edge-pair. Each control loop creates a virtual link and the operation of the exchange of control packets is on a per-edge-to-edge virtual link basis. A new set of congestion control techniques are required to construct virtual links, which break up congestion at interior nodes and distribute the smaller congestion problems across the edge nodes. However, the scheme's requirement on routers is high. It may be unrealistic to expect routers to devote many resources to interpreting with the algorithm and handing admission control.

We believe that for effective congestion control and fair bandwidth allocation, the issues have to be tackled at different layer. At the network level, relevant information concerning the network operational conditions and availability of resources of the underlying network is vital to avoid network congestion. At the end-to-end transport level, relevant information concerning the end systems is crucial for fair bandwidth sharing among competing traffics. We also believe that feedback mechanism is essential for effective congestion control and fair bandwidth allocation. By coupling with relevant protocol for correctly elaborating, transporting necessary feedback information as well as control actions amongst participating entities, we can make congestion control decisions, resources allocation decisions more intelligent.

In this paper, we experiment with a scheme 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 proposed mechanism is called Fair Intelligent Congestion Control Resource Discovery (FICCRD) mechanism. We present simulation results to illustrate that our proposed mechanism can improve end-to-end TCP performance by controlling the congestion and allocating fair share of bandwidth to competing TCP traffics.

The paper is organized as follows. We begin in section 2 by describing the Fair Intelligent Congestion Control Resource Discovery (FICCRD) mechanism. The simulation methodology, simulation scenario, simulation results and evaluation are presented in section 3. Some concluding remarks and direction of future work are given in section 4.

2. Fair Intelligent Congestion Control Resource Discovery Mechanism

The aim of this paper is twofold. Firstly, it aims to develop a feedback loop between the edge routers to convey information concerning available network resources and network conditions from within the network. 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 have deployed a similar scheme, the Fair Intelligent Congestion Control (FICC) [8,9], 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. Our approach is experimental using simulation with the ns2 simulator. 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 [12,13].

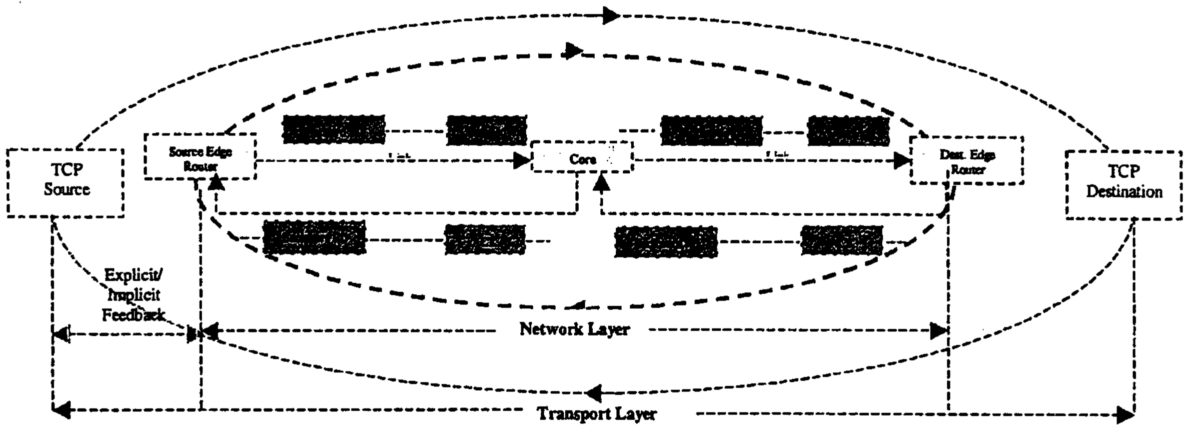

Figure 1. FICCRD Model

The FICCRD mechanism proposed in this paper has following features:

- Creating feedback loop mechanism between edge routers at network layer, a special packet – resource discovery (RD) packet is introduced to collect en route core router's state information, which is generated proportionally to the TCP traffic.
- Estimating fair share of bandwidth at each core router, on a per-output queue basis, for competing TCP connections and feedback relevant information concerning the network conditions to the edge router.
- Collecting core router resources, e.g. available bandwidth, available buffer to calculate en route network capacity and feedback relevant information to adjust TCP congestion window.
- Requiring no modification in the TCP protocol, i.e., transparent to the TCP layer.

2.1 FICCRD Model

The general model for Fair Intelligent Congestion Control Resource Discovery (FICCRD) mechanism is shown in [Figure 1]. Basically, the FICCRD works as follows. The incoming data packets are classified into flows and the arrival packet rate at the source edge router. The source edge router generates forward RD-packets for each flow proportionally to the arrival packet rate, which are turned around by the destination edge router and sent back to the source edge router as backward RD-packets. Each core router along the path directly updates feedback control information into RD-packets when they pass in the forward or backward direction. When the source edge router receives Backward RD-packet, it keeps the feedback information and consumes the RD-packet. A coherent approach [10, 11] is employed at source edge router to convey such feedback information concerning the network conditions to TCP source. In the following, we summarize the behaviors of both edge and core routers.

2.2. Edge Router behaviors

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 [7]. 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:

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

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.

A special resource discovery (RD) packet was generated 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 (Arrival Packet Rate) field.

The source edge router sends RD-packet to the destination edge router along the path. The destination edge router sends back the RD-packets. When the source edge router receives Backward RD-packet, it keeps the feedback information and consumes the RD-packet.

Two category methods (Explicit Window Adaptation, ACK Bucket Control) can be used at source edge router to convey such feedback information concerning the network conditions to TCP source. Basically Explicit Window Adaptation technique controls the maximum receiver window (RWND), which acts as an upper bound on the TCP congestion window (CWND) variable at the TCP source to control the TCP sender rate explicitly. ACK Bucket Control technique achieves the same goal by withholding the Acknowledgments at edge router to implicitly control the TCP sender rate to match with the explicit rate in underlying network. We already proposed Fair Intelligent Explicit Window Adaptation (FIEWA) scheme and Fair Intelligent ACK Bucket Control (FIABC) scheme for each category [10,11].

2.3. Core Router behaviors

Fair Intelligent Congestion Control (FICC) scheme [8,9] is used at each core router to estimate fair share of bandwidth for competing TCP connections, calculate acceptable network capacity and feedback relevant information to the edge router.

In detail, 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. To obtain an accurate MAPR, the router updates its MAPR according to the APR field of the arriving forward RD packets of each connection. MAPR is updated as

$$\text{MAPR} = \text{MAPR} + \beta * (\text{APR} - \text{MAPR}) \quad (2)$$

Where β is the exponential average factor. 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 scheme, 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. We believe that whether to increase or decrease the source rate should be determined primarily by the buffer utilization ratio (buffer in use/buffer capacity), not a fixed queue threshold regardless of the provided buffer capacity.

To calculate the expected rate (ER) based on the queue length at core router, linear queue control function DPF(Q) is employed in our scheme. The basic characteristics of the function is that it has values between 1 and 0 for queue lengths in the range of [Q0, buffer size] and between a and 1 for queue lengths in the range of [0, Q0]. The two lines intersect at Q0, where the value of DPF(Q) is 1. 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 at core router is shown in table 1.

Table 1.Description of FICCRD core router scheme

```
Parameters
 *β: The average ratio
 *BUR: Buffer Utilization Ratio
 *α: Queue control function parameter
Per Queue Variable
  MAPR: Mean Allowed Packet Rate
  DPF: Down Pressure Factor
  Q0: Target Queue Length
  Q: Current Queue Length
Initialization
  Q0 = BUR * BufferSize
At Core Router
  If (receive RD ($\hat{APR}_{i}(t)$, ER, DIR = forward))
    If (Q > Q0)
      If ($\hat{APR}_i(t)$< MAPR)
        MAPR = MAPR + β * ($\hat{APR}_{i}(t)$- MAPR)
    Else
      MAPR = MAPR+ β * ($\hat{APR}_{i}(t)$ - MAPR)
  If (receive RD ($\hat{APR}_{i}(t)$,ER, DIR = backward))
    If (Q > Q0)
      $DPF = \frac{BufferSize - Q}{BufferSize - Q_0}$
    Else
      $DPF = \frac{(a-1)*(Q_0-Q)}{Q_0}+1$
    ER=min (ER, DPF*MAPR)
```

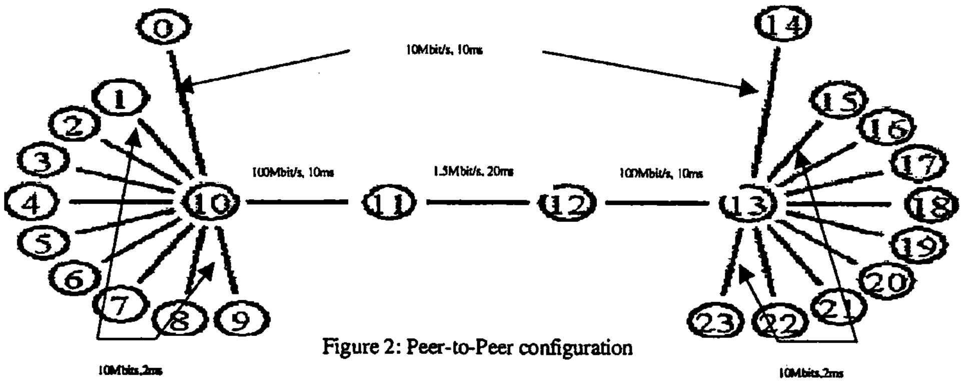

Figure 2: Peer-to-Peer configuration

3 Simulation Methodology

We use ns [14] network simulator to evaluate the proposed FICCRD mechanism. 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 the FICCRD mechanism were added and compiled into ns.

3.1 Simulation Scenario

The simulation performs with TCP applications running over IP network. Peer-to-Peer configuration [Figure 2] is employed in the simulation. There are 10 TCP 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. Each core router (11, 12) use the FICCRD mechanism with Bur of 0.5, α of 1.15 and β of 0.0625. The TCP version used in our simulations was TCP Reno, which includes fast retransmission and fast recovery.

3.2 Results and analysis

In this section, we present and discuss simulation results and compare the behavior of FICCRD with the case where simple droptail queueing is used. We evaluate the simulation results in terms of the fluctuation of packet queue length, the total packet dropping at bottleneck core router, the fluctuation of sender sequence number, and the goodput at TCP source.

General remarks

Simulation results are shown in Figure [3-6]. With "Droptail", buffer overflow at bottleneck core router is inevitable. The queue length of the bottleneck router would grow and repeatedly exceed the buffer size; this causes packet retransmission and greatly degrades the end-to-end throughput performance. It also can't achieve fairness among several TCP competing connections with different round-trip propagation delay. On the other end, with our mechanism (denote as "FICCRD"), there are no packets dropping at the bottleneck router. The data queue length at bottleneck router will still fluctuate at our target queue length with narrow range. These results in higher effective throughput than in "Droptail". Moreover, our mechanism can achieve fairness also among TCP competing connections with different round-trip propagation delay. Detailed analysis on the issues that contribute to the TCP performance is presented below.

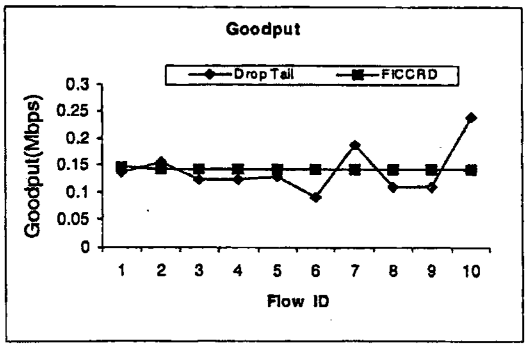

Figure 3. Goodput (Droptail Vs. FICCRD)

Goodput [Figure 3] Figure 3 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. Figure 3 shows an unfair division of bandwidth when Droptail is used. However, the throughput obtained by FICCRD is fairly distributed among connections.

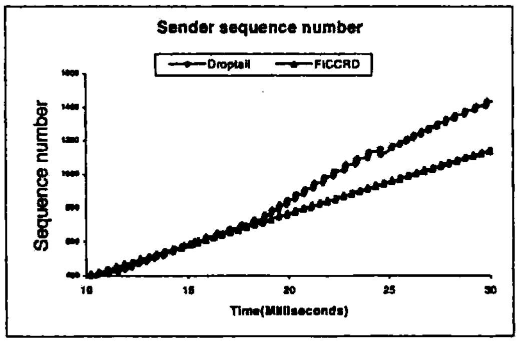

Figure 4. Sender Sequence Number (Droptail Vs. FICCRD)

Sender sequence Number [Figure 4] We plot the sender's sender sequence number as a function of time. We use TCP sender sequence number vs time. Simulation results demonstrate that that the TCP sender rate in our mechanism is more regulated than in "Droptail", which means that the burstiness of traffic is reduced based on explicit feedback rate in RD packet in "FICCRD".

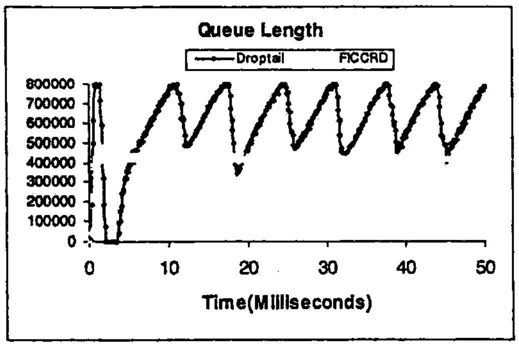

Figure 5. Bottleneck Queue Length (Droptail Vs. FICCRD)

Packet Queue Length [Figure 5] The main reason for TCP performance degradation is the overflow of buffers at bottleneck router. It is of interest to determine the buffer requirement for zero packet loss. With "Droptail", the packet queue length will grow above the maximum bottleneck router packet queue length, thus bottleneck router will drop the packet, the packet retransmission will happen due to the TCP source receive three duplicate ACKs. With "FICCRD", there are no packet dropping at bottleneck router, the packet queue length at bottleneck router still oscillates at our target queue length. We also show the droping total for each flow in Figure 6. It is clear to show that there are no packets droping in "FICCRD" comparing with "Droptail" that each flow has packets droping at bottleneck link.

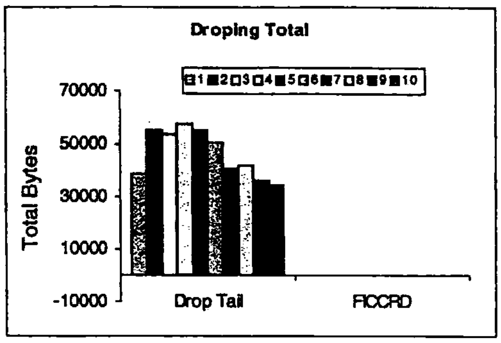

Figure 6. Droping Total (Droptail Vs. FICCRD)

4. Conclusion and Future Work

This paper proposes Fair Intelligent Congestion Control Resource Discovery (FICCRD) mechanism, which can improve end-to-end TCP performance by controlling the congestion and allocating fair share of bandwidth to competing TCP traffics. The significant contribution of our mechanism is that it integrates network resources, such as available bandwidth and available buffer at network layer to estimate fair share of network resources. The scheme presents the edge devices with necessary information for more effective QoS and congestion control.

Simulation results show that it can significantly improve in throughput, fairness, and packet loss rate for TCP connections. Importantly, it is transparent to TCP and requires no modifications to current TCP implementation.

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 QoS-aware technologies is a key factor for the continued success of the Internet. Our approach is comprehensive, flexible and can be easily extended for QoS-aware Internet. We are currently investigating the use of our Fair Intelligent

Congestion Control Resource Discovery (FICCRD) mechanism for congestion and QoS control in the Diffserv network region. The goal is to apply it not on a "per flow" basis but on a "per DSCP or class" basis so as to preserve the scalability of Diffserv.

References

[1] W. Stallings, "High-Speed Networks: TCP/IP and ATM Design Principles", Prentice-Hall, 1998.

[2] S. Floyd, "TCP and explicit congestion notification", Computer Communication Review, vol. 24, no. 5, pp. 8-23, October 1994.

[3] Z. Wang and J. Crowcroft, "A new congestion control scheme: Slow start and search (Tri-s)", Computer Communication Review, vol. 21, no. 1, pp: 32-43, January 1991.

[4] L. S. Brakmo and L.L.Peterson, "TCP Vegas: End to end congestion avoidance on a global Internet", IEEE Journal on Selected Areas in Communications, vol. 13, no. 8, pp. 1465-1480, October 1995.

[5] M. Gerla, W. Weng, R. Lo Cigno, "BA-TCP: A bandwidth aware TCP for satellite networks", Proceedings of IEEE ICCCN'99, Boston, Massachusetts, October 1999.

[6] David Harrison, Shivkumar Kalyanaraman, "Edge-to-Edge Traffic Control for the Internet: Concepts and Architecture," RPI ECSE Networks Laboratory Technical Report, ECSE-NET-2000-1, January 2000.

[7] I. Stoica, S. Shenker, and H. Zhang, "Core-Stateless Fair Queueing: Achieving Approximately Fair Bandwidth Allocations in High Speed Networks", In Proceedings of the ACM SIGCOMM'98, Vancouver, Canada, September 1998.

[8] Hoang, D. B., and Wang, Z., "Performance of TCP over ATM Networks with ABR and UBR Services", Computer Communications Journal, Vol. 23, Issue 9, pp. 802-815, April 2000.

[9] Hoang, D. B., and Wang, Z., "A Fair Intelligent Congestion Control for ATM", Technical Report 224, ISBN 1 86487326 4, Basser Computer Science Department, Sydney University, Australia, December 1999.

[10] Q. Yu, and Hoang, D. B., "Fair Intelligent Explicit Window Adaptation," IC'2001, Las Vegas, California, USA, pp. 486-492, June, 2001.

[11] Q. Yu, and Hoang, D. B., "An Intelligent Coherent Approach to Cooperation between TCP and ATM Congestion Control – Modelling and Simulation Analysis", accepted by Special Issue of the Simulation: The Transactions of the Society for Modelling and Simulation on Performance Modelling and Simulation of ATM Systems and Networks, April 2002.

[12] Clark, D. and W. Fang, Explicit allocation of best effort packet delivery service. IEEE/ACM Transactions on Networking, vol. 6, no 4. 1998

[13] S. Black et al., "An Architecture for Differentiated Services", RFC2475, Dec. 1998.

[14] ns-2, Network Simulator(version.2). LBL, URL: http://www.isi.edu/ns/nam.