# Generic Congestion Control Through Network Coordination (Extended Abstract)

Xicheng Liu[1] Cynthia Cheung[1,2] John S. Baras[1]
[1]Institute for Systems Research, University of Maryland, College Park, MD 20742
[2]NASA Goddard Space Flight Center, Greenbelt, MD 20771
Emails: {oliu, cycheung, baras}@isr.umd.edu

## Abstract

*With the rapid emergence of new services and the quick change of traffic in Internet, the widely used end-to-end congestion control scheme becomes insufficient. This paper proposes a new congestion control architecture based on the coordination among the core router, the edge router, and the host. The core router notifies its immediately upstream nodes when it detects incipient congestion. Those nodes then constrain their sending rates to prevent congestion. No per-flow state information is maintained in routers. The application itself need not take care of the rate adaptation. The architecture provides a uniform solution accommodating both responsive and unresponsive traffic with trivial overhead. It reduces the packet loss to zero, and leads to higher network resource utilization. Response time to congestion is greatly reduced. The architecture has been implemented in the Linux kernel. Some preliminary experiments show its advantages.*

## 1. Introduction

The success of Internet owes much to the sound principles of the additive-increase-multiplicative-decrease (AIMD) congestion control of the TCP [6]. However, with rapid emergence of multiple services and quick evolution of Internet traffic, TCP control does not seem sufficient. Especially multimedia services produce many "unresponsive" traffic flows, which from the point of view of TCP are "unfriendly". Nevertheless, this kind of traffic accounts for larger and larger portion of the Internet usage. It will even dominate Internet traffic in some regions in the future. We need to prepare a long-term solution rather than relying on short-term measures.

In this paper we propose a novel congestion control architecture as a uniform solution for both responsive and unresponsive traffic, which we call Generic Congestion Control Architecture (GCCA). It is based on the coordination of the core router, the edge router, and the host. This is essentially different from the TCP end-to-end control policy, which depends mainly on the host ability. New service models including both the DiffServ [2] and the InterServ [3] would call forth closer cooperation between the network core and the edge. Our architecture fits well into these service models, especially the DiffServ. It completely prevents packet loss in the network and leads to higher utilization of network resources. Response time to congestion is dramatically reduced. The scheme is scalable to new services and to future traffic. Fairness is guaranteed without per-flow state information maintenance in the router. All these are achieved with trivial overhead.

The rest of this paper is organized as follows. Section 2 presents the design principles. Section 3 describes the new architecture and its components. Section 4 gives preliminary results from experiments based on an implementation in the Linux kernel. Conclusions are summarized in Section 5.

## 2. Design Principles

Both the host and the router mechanisms are important for efficient control of congestion [4], [6]. The router has the best view of the queue behavior, the flow dynamics, and the congestion status. The host has a direct control of the source. We provide them a way to coordinate directly and quickly by a node-to-node information feedback protocol called Generic Congestion Control Protocol (GCCP). When a router detects the liability of congestion, it notifies its immediately upstream nodes that contribute to the congestion through GCCP. Those nodes will control their sending rates to prevent the congestion. Packet loss would be avoided should the notification be made early enough. Therefore, an early congestion detector and a rate controller are needed in the router.

Upstream nodes buffer more packets when they slow down their sending rates. On one hand, this helps prevent the congestion in the downstream bottleneck link. On the other hand, it improves the utilization of the network memory resources. Transient bursty traffic can be tolerated

Part of this work was done when the first author was with Motorola-ICT Joint R&D Lab and University of Cambridge, UK

0-7803-7510-6/02/$17.00 ©2002 IEEE

within several steps of feedback of GCCP messages among routers. Some lasting congestion may be referred back to the host as a last resort. The host identifies the flows contributing to the congestion and slows down their rates. For the TCP connection, we keep original window-based control intact, and add a window adjuster. For the UDP flow, an adjustable traffic shaper is attached to each UDP port. In fact, the end-point rate controller is ready to accommodate any well-proven rate adaptation algorithm, such as those obtained in the study of TCP-friendly control [5], [8], [9]. Unlike the application adaptation [11] and the TCP friendly control, however, our approach is for the network and so is transparent to the application. The application need not take care of the rate adjustment itself.

The incipient congestion detection in the router is at the queue level. No per-flow status information is maintained. This makes the overhead as little as possible in the core network. To guarantee the fairness among flows, we use a revised version of the RED algorithm [4] for congestion detection and feedback. Statistically, the amount of flow reduction is proportional to the flow's contribution to the congestion. The edge router need feedback more information to the host to help identify a particular flow. However, it does not need to maintain the per-flow status.

Each router or host is independent in running the congestion detection algorithm and originating GCCP messages. Thus the architecture is completely distributed and autonomous. No global synchronization is needed. The architecture does not change any operation of existing protocols and modules in the network kernel. So an existing node can communicate with a GCCA-capable node without any change. The architecture can be deployed gradually in a node by node fashion.

## 3. Generic Congestion Control Scheme

GCCA modules in the router and the host are shown in Fig. 1. There are mainly three components, the GCCP, the early congestion detector, and the rate controller. In the router, the congestion detector monitors the packet queue and detects incipient congestion. When the incipient congestion is detected, it creates GCCP messages and sends them to involved upstream nodes. When the router receives a GCCP message from downstream nodes, it will adjust packet emission rate of the queue through the rate controller. In the host, there is no congestion detector, but separate rate controllers for the TCP and the UDP flows. The rate controller for the TCP connection is called a window adjuster. The left part of Fig. 1 shows possible packet flows in the source end, the right part shows those in the destination end. We can see that no GCCP message is ever sent from the destination host to the router.

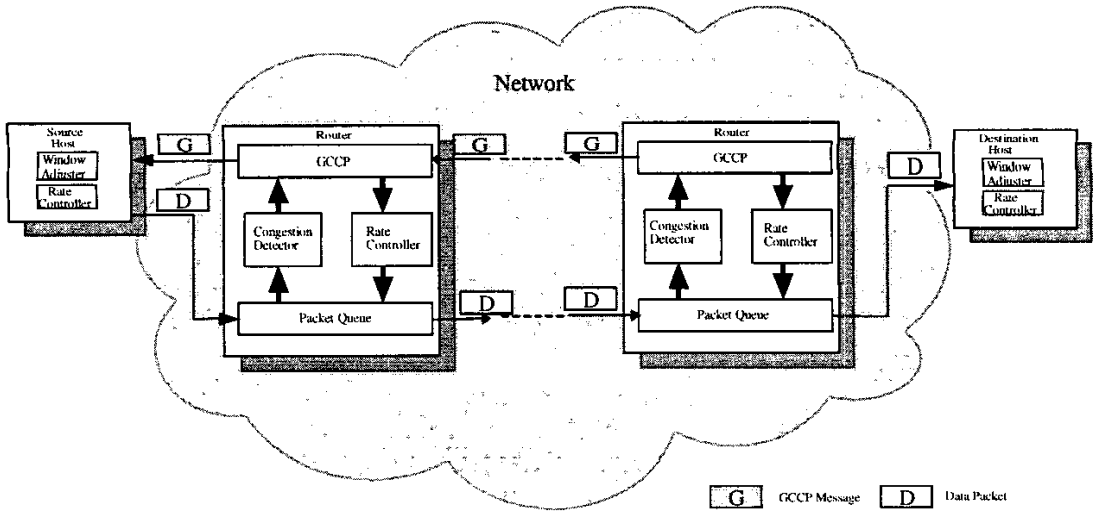


**Fig. 1. GCCA Components and Their Interactions**

### 3.1. Generic Congestion Control Protocol

GCCP is added into the TCP/IP protocol stack at the lower level of the network layer. This makes it efficient as well as general enough to be applicable to many types of physical networks. A GCCP message is immediately carried in a physical frame. To identify the frame as carrying a GCCP message, the sender assigns a special value to the type field in the frame header, and places the GCCP message in the frame's data field. The format of the GCCP message is given in Fig. 2.

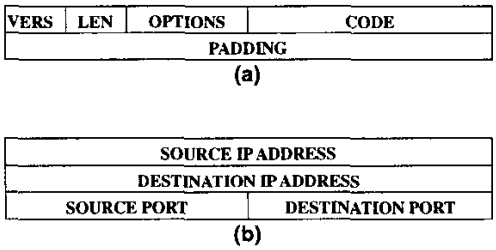


**Fig. 2. The GCCP Message Format. (a) The Overall Format; (b) the PADDING Field Format.**

Field VERS contains the version of GCCP that is used to verify if the sender and receiver agree on the format of the message; it contains 1 for current GCCP. Field LEN gives the message length measured in 32-bit words. The field OPTIONS indicates whether the sender of the GCCP message is an edge router, i.e. the receiver is a host. If yes, the field is set to 1 and the padding information is added; otherwise, set to 0 and no padding information exists. The field CODE specifies the congestion information. It contains 1 (CONGESTION) if the incipient congestion is detected; 0 (NORMAL) if the congestion disappears. The padding information includes four fields, which are used for the host to identify a particular flow. Fields SOURCE IP ADDRESS and DESTINATION IP ADDRESS contain the 32-bit IP addresses of the flow. Fields SOURCE PORT and DESTINATION PORT specifies the port numbers on both ends.

### 3.2. Congestion Detection and Feedback

The RED algorithm [4] can detect incipient congestion by calculating the average queue size (avg). The average queue size is compared with two thresholds, a minimum threshold ($min_{th}$) and a maximum threshold ($max_{th}$). When avg is between them, each arriving packet will be marked or dropped with probability $p_a$. From the point of view of traffic flows, the probability that a flow is marked or dropped is roughly proportional to that flow's share of the bandwidth through the router. Thus the fairness among flows is guaranteed.

We revise the RED algorithm to avoid any packet loss. When a packet is to be dropped or marked in RED, we emit a GCCP CONGESTION message instead. The packet is then queued as normal. This GCCP-capable RED algorithm is given is Fig. 3.

```
initialize status as NORMAL
for each packet arrival
   calculate the average queue size avg
   if min_th ≤ avg < max_th
      calculate probability p_a
      with probability p_a:
         send a GCCP CONGESTION message
         set status as CONGESTION
   else if max_th ≤ avg
      send a GCCP CONGESTION message
      set status as CONGESTION
   else if avg ≤ min_th
      if status is CONGESTION
         send a GCCP NORMAL message
         set status as NORMAL
```

**Fig. 3. The GCCP-Capable RED Algorithm**

### 3.3. Rate Control

In the router, we use an adjustable leaky bucket filter [12] to control the output rate of a queue. Upon receiving a GCCP CONGESTION message, we adjust the leaky bucket and decrease the rate linearly. When receiving a GCCP NORMAL message, the output rate is restored.

In the host, we use the same mechanism with that in the router for unresponsive flows (UDP flows). Every UDP port is attached with an adjustable leaky bucket filter. For the TCP flow, however, a much simpler algorithm exists. When a GCCP CONGESTION message is received for a connection, we decrease that connection's congestion window (cwnd) [1] and slow-start threshold (ssthresh) [1] by half. The reduced rate can increase later through the slow-start mechanism of TCP control. The GCCP NORMAL message is ignored. Existing TCP control is unchanged so that a TCP connection responds to receiver acknowledgements and packet loss (if any) as before.

## 4. Preliminary Results

We have implemented GCCA in the Linux kernel 2.2.10. The early congestion detector is mainly implemented in the enqueue function, and the rate controller in the dequeue function. The command tool tc [13] is used for changing and configuring queueing disciplines including the GCCP-capable RED algorithm.

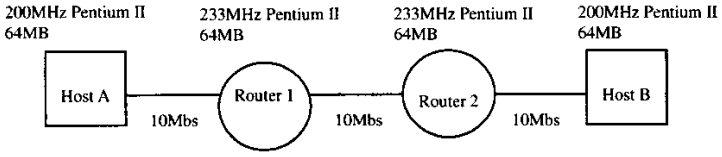


**Fig. 4. The Testbed for Experiments**

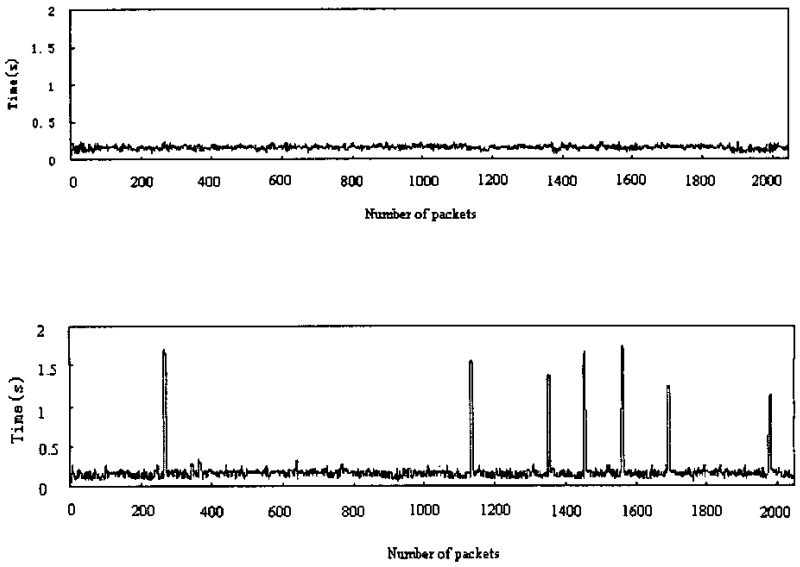


**Fig. 5. Comparison of the Jitter. The Upper is for GCCA, the Lower for E2E**

A testbed composed of two Linux routers and two Linux hosts is set up to evaluate the scheme. The testbed is illustrated in Fig. 4. In the experiments of this paper, we compare the GCCA with the widely used end-to-end control scheme (referred to as E2E below). For the E2E scheme, the RED algorithm is used for queue management in routers. A number of 8KB sized packets are transferred from Host A to Host B through Router 1 and Router 2 in the experiments. Router 2 is configured as a bottleneck. Traffic is generated and performance data are collected using the

packet level measurement tool DBS [7].

We do the experiments using three different groups of parameters. Table 1 shows the end-to-end delays in the experiments. The GCCA produces shorter delay than E2E even in such a small setting.

Table 2 shows the throughput in the experiments. Here the throughput represents the average rate at which data are transferred through the network. We can see that the GCCA increases the throughput by as many as 15% on average.

**Table 1. Comparison of the End-to-End Delay**

| Experiment | GCCA (s) | E2E (s) |
|---|---|---|
| 1 | 54 | 58 |
| 2 | 54 | 66 |
| 3 | 55 | 64 |

**Table 2. Comparison of the Throughput**

| Experiment | GCCA (Mbps) | E2E (Mbps) |
|---|---|---|
| 1 | 2.37 | 2.21 |
| 2 | 2.37 | 1.94 |
| 3 | 2.32 | 2.00 |

Fig. 5 shows jitters that the two schemes make in an experiment. The vertical axis (Y) represents the end-to-end delay of a packet. The horizontal axis (X) indicates the packet number. The jitter is calculated as the absolute value of the difference between delays of two successive packets. We can see that jitters from GCCA, i.e., variances of packet delay, are always within a small bound. By contrast, E2E produces unpredictable jitters from time to time.

In addition, no packet loss is observed for GCCA in the experiments.

## 5. Conclusions

In this paper, we have designed a novel congestion control architecture for Internet. It is based on the coordination among the core router, the edge router, and the host. By taking into account of both responsive and unresponsive traffic, it gives a uniform solution for them. We present its design principles. Main components of the architecture are described, namely, the GCCP, the early congestion detector, and the rate controller. The scheme is supported by a successful implementation in the Linux kernel. Preliminary experiments on a testbed composed Linux routers and hosts show that the architecture can produce better performance than the widely used end-to-end control scheme in terms of network delay, network throughput, and the jitter. Major advantages of the architecture also include zero packet loss, quicker response to congestion, and higher resource utilization. In addition, the architecture is scalable to new services and future traffic types. Further study is carrying out to refine details of the architecture.

## 6. Acknowledgements

The authors would like to acknowledge Mr. Fusheng Chen, Mr. Jun Li, Mr. Bin Pang, and Mr. Lihai Yang for their help in building the testbed, implementing the algorithms, and doing experiments.

## References

[1] Allman, M., Paxson, V., Stevens, W.: TCP Congestion Control. Internet RFC 2581. April 1999.

[2] Blake, S., Black, D., et al: An Architecture for Differentiated Services. Internet RFC 2475. December 1998.

[3] Braden, R., Clark, D., and Shenker, S.: Integrated Services in the Internet Architecture: An Overview. Internet RFC 1633. July 1994.

[4] Floyd, S., and Jacobson, V.: Random Early Detection Gateways for Congestion Avoidance. IEEE/ACM Transaction on Networking, 1(4), pp. 397-413, August 1993.

[5] Floyd, S., Handley, M., Padhye, J., and Widmer, J.: Equation-Based Congestion Control for Unicast Applications: the Extended Version. International Computer Science Institute tech report TR-00-003, March 2000.

[6] Jacobson, V.: Congestion Avoidance and Control. Proceedings of ACM SIGCOMM. (1988) 314-329.

[7] Murayama, Y.: DBS: A Powerful Tool for TCP Performance Evaluations. http://shika.aist-nara.ac.jp/member/yukio-m

[8] Padhye, J., Kurose, J., Towsley, D., Koodli, R.: A Model Based TCP-Friendly Rate Control Protocol. Network and Operating System Support for Digital Audio and Video (NOSSDAV), June 1999.

[9] Rejaie, R., Handley, M, and Estrin, D.: RAP: An End-to-end Rate-based Congestion Control Mechanism for Realtime Streams in the Internet. IEEE INFOCOMM 99.

[10] Rusling, D. A.: The Linux Kernel. http://www.kernelnotes.org/guides/TLK/tlk.html.

[11] Sisalem, D., Schulzrinne, H.: The Loss-Delay Adjustment Algorithm: A TCP-friendly Adaptation Scheme. Network and Operating System Support for Digital Audio and Video (NOSSDAV), July 1998.

[12] Turner, J.S.: New Directions in Communications (or Which Way to the Information Age). IEEE Communication Magazine, vol. 24, pp. 8-15, Oct. 1986.

[13] ftp://ftp.inr.ac.ru/ip-routing/