FOURTH QUARTER 2003, VOLUME 5, NO. 2

IEEE COMMUNICATIONS SURVEYS

The Electronic Magazine of Original Peer-Reviewed Survey Articles

www.comsoc.org/pubs/surveys

A SURVEY AND MEASUREMENT-BASED COMPARISON OF BANDWIDTH MANAGEMENT TECHNIQUES

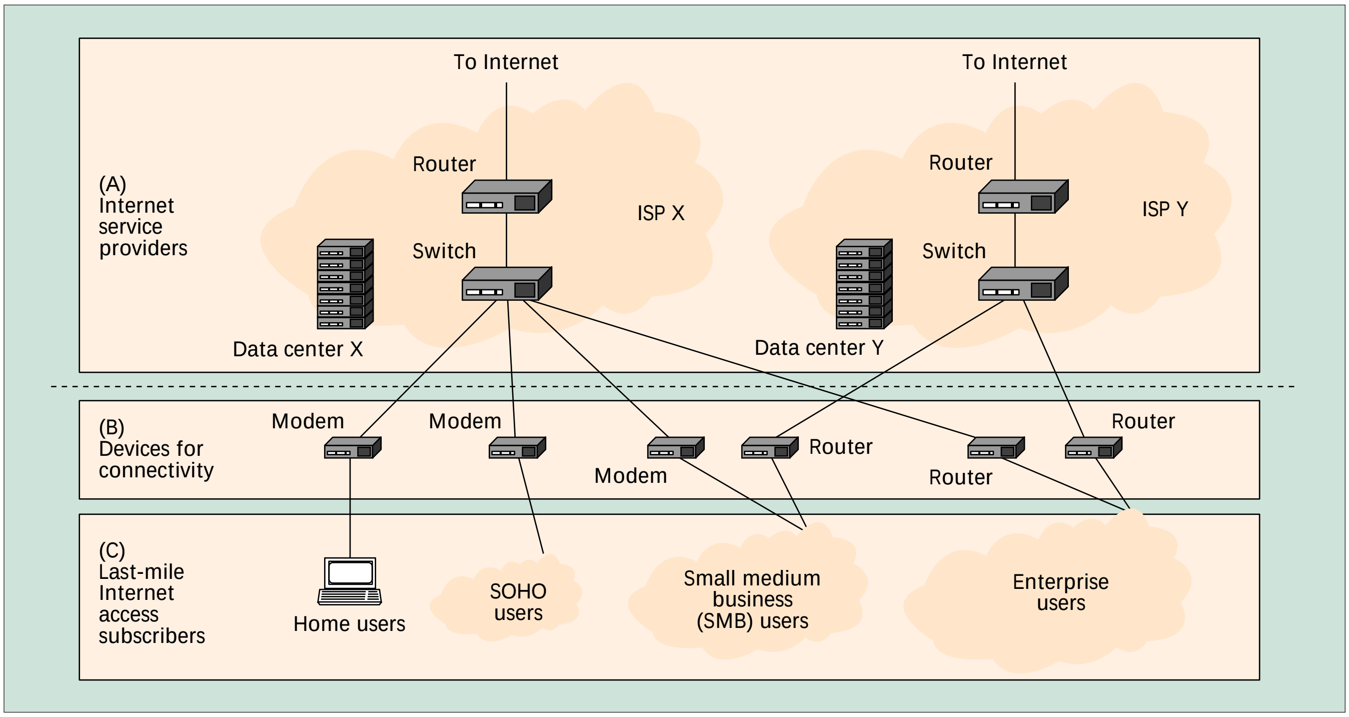
HUAN-YUN WEI, YING-DAR LIN, NATIONAL CHIAO TUNG UNIVERSITY

ABSTRACT

This article gives a brief tutorial for bandwidth management systems; a survey of the techniques used by eight real-world systems, such as Class-Based Queuing (CBQ), Per-Flow Queuing (PFQ), Random Early Detection (RED), and TCP Rate Control (TCR); a compact testbed with a set of methodologies to differentiate the employed techniques; and a detailed black-box evaluation. The tutorial describes the needs for the three types of policy rules: class-based bandwidth limitation, session-bandwidth guarantee, and inter/intra-class bandwidth borrowing. The survey portion investigates how the eight chosen commercial/open-source real systems enforce the three policy types. To evaluate the techniques, the designed testbed emulates real-life Internet conditions, such as many simultaneous sessions from different IPs/ports, controllable Wide Area Network (WAN) delay and packet loss rate for each session, and different TCP source implementations. The performance metrics include accuracy of bandwidth management, fairness among sessions, robustness under Internet packet losses and different operating systems, inter/intra-class bandwidth borrowing, and Voice over IP (VoIP) quality. The black-box test results demonstrate that (1) only the combination of CBQ+PFQ+TCR can solve the most difficult scenario (multiple sessions competing for the narrow 20kb/s class); (2) the TCR approach may degrade the goodput, fairness, and compatibility even under slight packet loss rates (0.5 percent); (3) without PFQ, TCR and RED have limited ability to isolate the sessions (especially for RED); (4) the G.729 VoIP quality over a 125kb/s access link becomes good only after exercising MSS-clamping to shrink the packet size of the background traffic down to 256 bytes.

In the current Internet, users access the Internet by subscribing to services from Internet Service Providers (ISPs). With dial-up/xDSL/cable modems or routers, subscribers can obtain Internet connectivity. However, the narrow access links become the bottlenecks and prevent subscribers from having good application performance. For a home subscriber, a background HTTP download transfer can cause an annoying delay to the interactive Telnet application. For small office/home office (SOHO), small/medium business (SMB), and enterprise subscribers, background FTP sessions may block mission-critical traffic, such as voice over IP (VoIP) and enterprise resource planning (ERP) traffic. Such conditions require a bandwidth manager deployed at each link between (B) and (C) in Fig. 1 to allocate bandwidth for interactive or delay-sensitive applications *within* the link. On the other hand, an ISP shares its backbone resources *among* its subscribers (including the last-mile or data-center subscribers in Fig. 1). Some subscribers may over-subscribe the access link and thus influence the performance of other subscribers, so an ISP may also need a bandwidth manager between the router and the switch at (A) in Fig. 1 to fairly distribute its resources among the subscribers. Given that network-wide, end-to-end QoS architectures such as DiffServ [1] are still being debated, edge-based managers are needed to at least control the traffic *within* each subscribed last-mile access link (customer-side edges) or *among* many subscribed access links (ISP-side edges).

Three Policy Types — Most bandwidth managers are policy-based gateways, in which a network administrator can define policy rules to solve the above mentioned problems. Each policy rule (Fig. 2) contains the *condition* and the *action* fields to define the specific actions for the specific conditions. The condition defines the packet-matching criteria to group a certain subnet/application/protocol into a bandwidth class. The action defines the parameters for a bandwidth class, such as 100 kb/s. The complexity of a policy rule ranges widely. For example, a 125 kb/s access link may be partitioned into a 90 kb/s VoIP class and a 35 kb/s FTP class (class-based bandwidth limita-



■ **FIGURE 1.** *Typical scenarios of current Internet subscribers.*

tion). If there is no voice call, FTP sessions can occupy the entire 125 kb/s link. Whenever a 30 kb/s VoIP session starts, the bandwidth manager allocates 30 kb/s for the VoIP class until the 90 kb/s is used up by the three 30 kb/s voice calls. This is defined as *inter-class bandwidth borrowing* where FTP can borrow the unused bandwidth from the VoIP class. Further, both classes can be set to guarantee the bandwidth of each session within the class. Thus, suppose three FTP sessions are mixed in the 35 kb/s class. Each session can be isolated to obtain a guaranteed 11.6 kb/s (*session bandwidth guarantee*), and each of the three voice calls mixed in the 90 kb/s queue gets a guaranteed 30 kb/s. If another FTP session attempts to start, the admission control can be set to fairly share the 35 kb/s among the four sessions, or to postpone the new session until a session in the 35 kb/s class stops. Of course, once a session leaves, the others can share the newly available bandwidth (*intra-class bandwidth borrowing*). The effectiveness of the three policy types (class-based bandwidth limitation, session bandwidth guarantee, and inter/intra-class borrowing) among the eight chosen real-world systems will be evaluated in this article based upon measurements made in a testbed developed in our lab. All results are reproducible through the tools in [2].

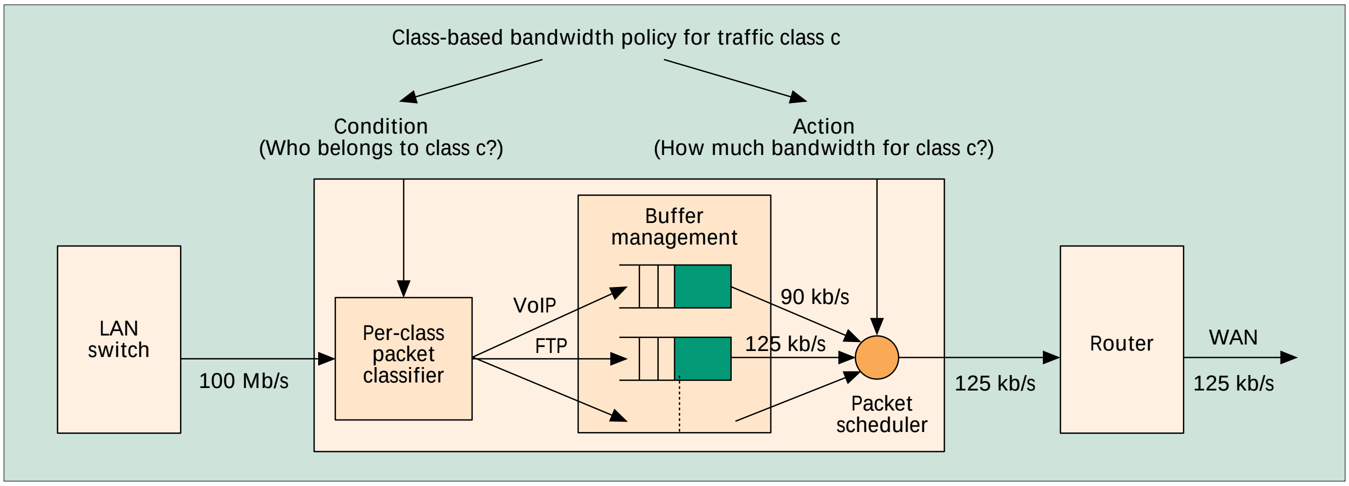
PERFORMANCE METRICS

Input traffic patterns may affect the effectiveness of the policy enforcement. Given the basic knowledge of TCP congestion control described in [3, 4], two sets of performance metrics are given below.

Response Delay: Long-Lived vs. Short-Lived Sessions — For example, the home subscribers (Fig. 1) care about how much delay is encountered when using the short-lived, interactive Telnet/Web applications but with long-lived FTP transfers in the background.

Bandwidth Isolation/Fairness/Efficiency/Flexibility — For the above FTP-vs-VoIP example, subscribers care about:

- How much the bursty FTPs can affect the voice quality of the constant-bit-rate VoIPs.


Class-based bandwidth policy for traffic class c
Condition (Who belongs to class c?)
Action (How much bandwidth for class c?)
LAN switch
100 Mb/s
Per-class packet classifier
VoIP
FTP
Buffer management
90 kb/s
125 kb/s
Packet scheduler
125 kb/s
Router
WAN
125 kb/s

■ **FIGURE 2.** *Building blocks of a bandwidth management system.*

Product		Bandwidth enforcement techniques	Grade (announced)	S/W Ver.	OS, HW/SW	Install at	Hardware			
Vendor	Model						Boot from	CPU	RAM	Interface
Open source [5]	ALTQ [6]	CBQ, CBQ+RED	100 Mb/s	2.2	FreeBSD, software	LAN-router link	Our P-III 700 MHz PC with 256M SDRAM, two Intel 100M NICs installed, booting from a hard disk.			
NetGuard	Guardian Pro	CBQ+TCR	10 Mb/s	5.02	NT 4.0, software					
Check Point	FloodGate	CBQ+PFQ+RED	45 Mb/s	4.1	NT 4.0, software					
Acute/ BroadWeb	*i*Policer 100-CR2202	TCR(WS+AP)	100 Mb/s	1.6.4	Embedded NT, hardware		Flash 32M	P-III 600	128M	10/100 Mb/s
Packeteer	PacketShaper 4500	CBQ+PFQ+TCR	45 Mb/s	4.1.2	Embedded Linux, hardware		Flash	P-III 600	128M	10/100 Mb/s
Sitara	QoSWorks QWX-10000	CBQ+TCR, MSS-clamping	100 Mb/s	1.8	Embedded FreeBSD, hardware		Hard Disk	P-III 600	192M	10/100 Mb/s
NetReality	WiseWan 500	CBQ+TCR	5 Mb/s	4.0	Proprietary, hardware	WAN link	Flash 32M	P 133	32M	V.35

Table 1. *Relevant information of devices under test.*

- How fairly the bursty FTPs compete for the 35 kb/s.
- How much bandwidth is wasted (how many packets are lost due to insufficient buffer in the bandwidth manager).
- To what degree can the FTPs utilize the unused VoIP bandwidth.

This study focuses on the second set of performance metrics above because most surveyed systems do not provide specific functions for the first set of performance metrics.

We first present a brief tutorial for bandwidth management systems. We then survey the techniques used in the eight chosen real-world systems. Next we show the designed testbed, methodologies, and the black-box evaluations to differentiate the techniques. The article ends with a summary of the findings and the conclusions.

Real-World Bandwidth Management Systems

Chosen Devices under Test (DUT)

As shown in Table 1, we chose seven products or open sources [4] to examine eight different combinations of techniques for managing bandwidth. They are chosen because of their strong reputations or market shares, and different combinations of employed techniques. The open-source ALTQ [5] package covers two systems (CBQ and CBQ+RED). These techniques are described in the following sections.

Internal Structures

The internal structure of a bandwidth manager is as follows. In Fig. 2, a class-based bandwidth policy matches packets of certain protocols/subnets into a per-class queue according to the specified criteria set in the packet classifier. The buffer manager then optimizes the queuing behaviors in the per-class queue. Finally, the traffic in the class is scheduled out at its corresponding specified bandwidth. To correctly set up a bandwidth management device involves the following steps:

Step 1. Defining the WAN-Link Bandwidth: In Fig. 2, the subscribed WAN link is 125 kb/s. If the traffic from the bandwidth manager to the router is more than 125 kb/s, queuing must occur at the router and may cause buffer overflows. Thus the first step to configure the bandwidth manager is to limit the traffic pumping into the router. In Fig. 2, a configured WAN-link bandwidth of 125 kb/s solves the problem. Table 2 compares the Step-1 capabilities among the chosen systems.

Step 2. Partitioning the WAN Link: In Fig. 2, after the configuration in Step 1, the 125 kb/s is partitioned into the 90 kb/s and the 35 kb/s classes. The next step then defines who belongs to which class. Table 2 compares the Step-2 abilities among the chosen systems. Hierarchical CBQ and flat CBQ will be explained in Step 5.

Step 3. Classification by the Packet Classifier: It is only when the passing traffic is correctly identified that the policy can be accurately enforced. In Fig. 2 packets are classified into their corresponding queues. Besides layer 3/4 packet headers such as the five tuples (src/dst IP/port, and protocol ID), layer-7 awareness can simplify the management. For example, active-mode FTP uses the FTP-Cmd port (port 21) to negotiate data transmission port X, and then sends the data using port X. If the bandwidth manager cannot recognize that port X was chosen to send the data, the bandwidth manager would never know what the session is transferring with port X. Table 2 compares the Step-3 abilities among the chosen systems.

Step 4. Optimization of Buffer Management: After packets are classified into the queues, the queuing behaviors dominate the effectiveness of the policy enforcement. In Fig. 2, buffer management techniques are employed to optimize the queuing behaviors. Based on the input traffic types, the techniques used can be categorized into

- TCP-unaware methods such as the *per-flow queuing* and the *active queue management* [7] (AQM).
- TCP-aware methods such as *TCP Rate Control* [8] and *MSS-clamping*, which refers to the control and adjustment of the MSS.

These methods can be combined to achieve the utmost benefits. The chosen DUTs have different combinations of these bandwidth enforcement techniques, as shown in Table 1. A high-level comparison of the combinations is described later. The session-bandwidth guarantee in Table 2 compares the Step-4 abilities among the chosen systems.

Step 5. Scheduling by the Packet Scheduler: The packets in the queues (Fig. 2) are waiting to be scheduled out onto the wire according to their specified bandwidth parameters. Many packet schedulers have been proposed and extensively analyzed in the literature [9–11]. The most popular one is the class-based queuing [12] (CBQ) approach. Only ALTQ and PacketShaper

Product		Step 1: Define WAN-link bandwidth	Step 2: Partition the WAN link	Step 3: Define the conditions for each action		Step 4: Fine-tune the parameters		
						Within a class		Among classes
Vendor	Model			Layer awareness	5 tuples	Session-BW guarantee	Borrowing among sessions	Borrowing among classes
Open source [5]	ALTQ [6]	Y	Y Hierarchical CBQ	4	Y	N	Y	Y
NetGuard	Guardian Pro	Y	Y Flat CBQ	4	Y	Y	Y	Y
Check Point	FloodGate	Y	Y Flat CBQ	7 (URL/MIME)	Y	Y	Y	Y
Acute	*i*Policer	N	N	4	Y (but only TCP)	Y	N	N
Packeteer	PacketShaper 4500	Y	Y Hierarchical CBQ	7 (URL) with auto traffic discovery	Y	Y	Y	Y
Sitara	QoSWorks QWX-10000	Y	Y Flat CBQ	4	Y	Y	Y	Y
NetReality	WiseWan 500	Y	Y Flat CBQ	7 (URL/MIME)	Y	Y	Y	Y

Table 2. *Functional comparison of the devices under test.*

support hierarchical CBQ that is more powerful in terms of bandwidth borrowing. Hierarchical CBQ can recursively divide the bandwidth into a tree-like hierarchy so that the bandwidth can be shared in a more flexible way than with a flat CBQ. Flat CBQ can only partition the WAN link into classes, but cannot further partition the classes into subclasses. The inter/intra-class bandwidth borrowing in Table 2 compares the Step-5 abilities among the chosen systems.

HIGH-LEVEL COMPARISON OF EMPLOYED TECHNIQUES

This section provides a high-level comparison of the bandwidth enforcement techniques employed in Table 1.

Class-Based Queuing — Aside from the basic function of a packet scheduler, namely scheduling packets onto the wire, CBQ [12] can provide inter-class bandwidth borrowing among classes, as described earlier. With CBQ, resource sharing on a link can be very flexible because the bottleneck link can be recursively divided into a tree hierarchy. If some class does not contain any traffic, its bandwidth can be borrowed to other busy classes. However, two problems arise due to multiple competing TCP sessions within each CBQ class:

- Large buffer requirement. A TCP session with a large TCP receiver window (RWND), if bounded by a very limited bandwidth (the bandwidth manager creates a bottleneck), may cause a huge number of packets to be queued at the bottleneck. Hence, the bandwidth manager should be equipped with a large memory to store the queued packets.
- Fairness among sessions within the same class. Given a small congestion window (either because of previous packet losses or because of starting a session), a TCP session with a large round trip time (RTT) will obtain a small bandwidth [4].

Class-Based Queuing with AQM — Active queue management [7] (AQM) can be employed to alleviate the above two problems. The most common approach is to apply Random Early Detection [13] (RED) to each CBQ queue. In Fig. 3, RED measures congestion by the exponential weighted average length of the CBQ queue. When a packet comes into the queue, RED uses a linearly increasing probabilistic function of the average queue length to decide whether to drop/mark the incoming packet or not. The RED-applied queue then expects the sender of the dropped/marked packet to slow down its transmission rate when detecting the congestion event (packet dropped/marked). RED interacts best with reactive traffic such as TCP. When source rates increase, the queue length grows, and consequently more packets are dropped or marked. This in turn causes the TCP sources to reduce their rates, and the cycle repeats. However, RED without per-flow information has limited ability to solve the second problem (fairness among sessions within the same class).

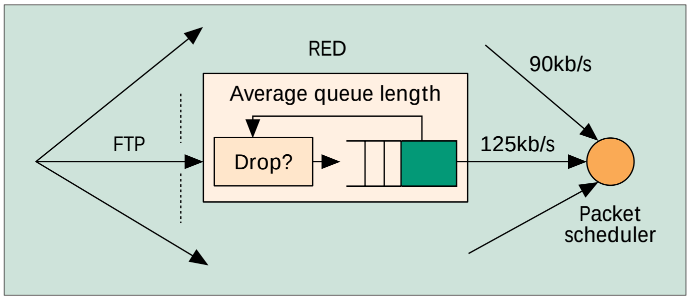


FIGURE 3. *CBQ with AQM.*

Class-Based Queuing with Per-Flow Queuing and AQM — A straightforward method to solve the unfairness problem is to apply a separate queue for each session (PFQ in Fig. 4). Sessions competing for the same CBQ queue are first classified by the per-flow classifier into the per-flow queues.

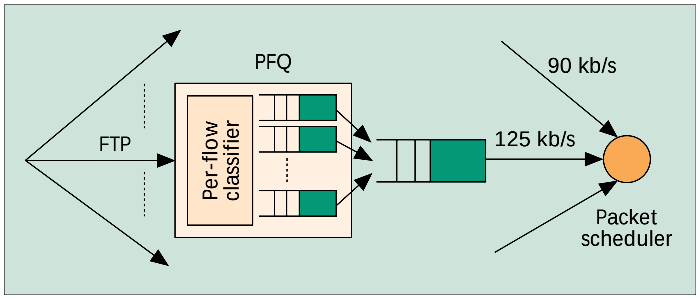


FIGURE 4. *CBQ with PFQ.*

Thus, each per-flow queue is isolated and is guaranteed to obtain a fixed bandwidth share of the class.

For the problem of the large buffer requirement, RED can be applied to each per-flow queue (CBQ + PFQ + AQM). Check Point's FloodGate belongs to this category.

Class-Based Queuing with Per-Flow Queuing and TCR — Given that the unfairness problem is solved by the CBQ with per-flow queuing (as in Fig. 4), the problem of the large buffer requirement can be solved in another way. Packeteer proposed TCP Rate Control [8] (TCR) to actively control the behaviors of the TCP sessions. Many followers employ TCR's concepts — window sizing (WS) and ACK pacing (AP) — in their products. To shape TCP flow *i* to bandwidth X, the window-sizing module measures the RTT of the flow *i* as Y and then accurately sizes the RWND of flow *i*'s feedback ACKs to X*Y. Hence, TCP sender *i* would limit only X*Y bytes outstanding in the WAN pipe. Consequently, there is almost no queuing of flow *i*'s packets at the bottleneck. The large buffer requirement is then resolved without dropping any packets. To further shape each session in a more fine-grained way, the queued feedback ACKs are released at even intervals. Thus, a TCP sender clocked by the stable ACKs will transmit its packets in a more accurate way. Products from Packeteer, Sitara, and WiseWAN belong to this category.

MSS-Clamping — Although CBQ + PFQ + TCR successfully solve the two problems of CBQ in an elegant way, mission-critical and delay-sensitive traffic such as Voice over IP (VoIP) may still perform poorly in the presence of a narrow access link. For example, the administrator may expect to allow the link to be fully allocated to FTP traffic when there is no VoIP traffic (Fig. 5). As soon as someone initiates a VoIP session, the packet scheduler should instantly allocate sufficient bandwidth to the voice traffic. Since a single packet should be completely put on to the wire before the scheduler handles the next packet, a large FTP packet can cause non-even spacing between the voice packets. Such a large packet requires a long transmission time to be completely put onto the wire. When the WAN link bandwidth is narrow, the long transmission time can block real-time VoIP packets, hence significantly increasing the end-to-end delay of the blocked voice packets, causing a large delay jitter and degrading the voice quality. A TCP sender always chooses the largest possible packet size (max segment size (MSS), typically ranging from 1000 to 1500 bytes) as its transfer unit. Fortunately, the MSS can be controlled by intercepting the TCP MSS negotiation during the TCP three-way handshaking. By spoofing both sides of the TCP session with a small MSS, the packet size of the following packets will never exceed that small MSS. Sitara incorporates this method.

Class-Based Queuing with TCR — Some implementations simplify the CBQ + PFQ + TCR to CBQ + TCR. The lack of PFQ may cause two problems: inability to control non-TCP traffic, and limited isolation of TCP sessions. The first problem arises because TCR can only optimize TCP traffic. The second problem occurs when the measurement of the RTT is inaccurate, causing inaccurate sizing of RWND in the TCP ACKs. The mixture of multiple inaccurately RWND-sizing TCP sessions in the CBQ queue may provide limited fairness among the sessions. The products from Acute and NetGuard belong to this category.

EVALUATION

BLACK-BOX VS. WHITE-BOX TESTING

In this section, we design a testbed to evaluate how effectively the commercially/publicly available bandwidth managers can enforce the policy. Performance testing can be divided into white-box tests and black-box tests. White-box tests are often used to evaluate some specific modules within a device to identify performance bottlenecks; hence, they often require the source code to insert hooks for recording timestamp information. In contrast, black-box tests view each device under test (DUT) as a black box and focus on the overall performance of the DUT. Since most of the chosen DUTs are commercial products, this study focuses on black-box testing to discover the general properties of a known algorithm. During the testing we also identify the problems of several algorithms to prevent developers from making the same mistakes.

BLACK-BOX BENCHMARKING TESTBED

Several assumptions are made: *the bottleneck of each test is at the DUT* so that when the results appear to be bad, the problem lies mostly in the DUT. On the other hand, managing bandwidth in a narrow access link is more difficult. So we assume *scarce access-link bandwidth* such as T1 (1.544 Mb/s) and 125 kb/s. Under such conditions, 20 simultaneous TCP sessions are adequate to differentiate the methods. We focus on *outbound traffic management* so that each component is tested through the model as in Fig. 2. Inbound traffic management, though important, is omitted.

After detailed observations, the contributions of the surveyed techniques are mostly for optimizing TCP sessions. TCP sessions are sensitive to the RTT, packet loss rates, buffer space availability at the bottlenecks, and other competing TCP sessions. So we use (1) *multiple TCP flows from different hosts* (different IP/port) to aggressively pump traffic through the DUT, and then through (2) *a controllable WAN environment* and then to the destinations. The WAN environment is emulated by the wan-emu, a self-developed Linux kernel device driver module that can have controllable delay, random loss rate, and jitter settings for the packets of each individual

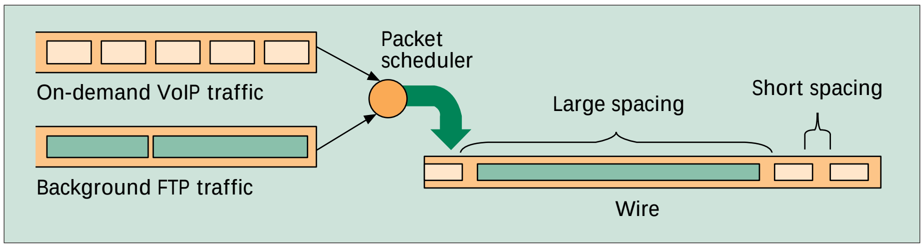
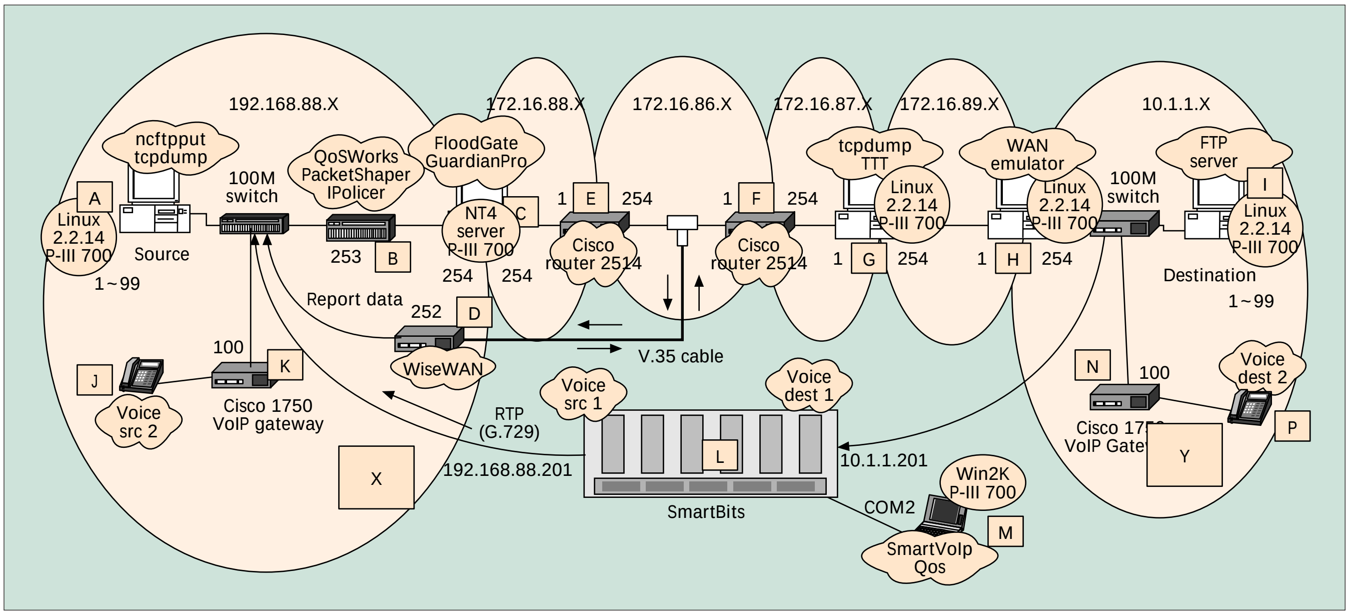


FIGURE 5. *Poor voice quality due to the large delay jitter caused by non-important, large-size background traffic.*



■ **FIGURE 6.** *Black-box testbed for evaluating bandwidth management systems.*

TCP session. Since the Internet is very dynamic, different sessions may have different routing paths and thus have different distances and link qualities. Through the wan-emu, a controllable testing environment can be used to quantitatively evaluate the interested performance metrics. Figure 6 and Table 3 provide complete information about our testbed and testing tools. Testing sessions are from X to Y, passing through the DUT, routers, monitoring point, and the wan-emu. The Cisco routers are installed specifically for the WiseWAN because the WiseWAN only supports the V.35 interface. The mechanisms of (1) and (2) are achieved through the below designs:

(1) IP-Aliasing: In Linux, each network interface card (NIC) can emulate 100 NICs, with each virtual NIC having a unique IP address. With the proper routing table setup at A in Fig. 5, we can direct certain sessions destined to a certain virtual NIC at I through a virtual NIC at A. Virtual NICs generate packets with their corresponding IP addresses so that the DUT will feel that the pass-through TCP sessions are from different LAN hosts, and incoming TCP ACKs are from different remote hosts. Packets are sent without link-layer collisions since only a single physical NIC is present at A and I. Hence, this design also emulates 100 hosts connected to a switch instead of a hub.

(2) wan-emu: The wan-emu is a Linux virtual interface driver that resides between the IP layer and the NIC driver. In this testbed, multiple wan-emu virtual devices are attached to the sink-side last-hop NIC driver (at H with IP 10.1.1.254) to have different impairments on different routes. With proper static routes, we can direct sessions destined to a virtual NIC at I through a specific wan-emu interface that has the desired link characteristics. Each pass-through packet is labeled with a timestamp indicating the time for the packet to be released. An interrupt is triggered every 1 ms to examine how many packets are due and should be released. The timer granularity can be easily tuned to 8192 Hz in Linux. Impairments such as the random/periodic loss rate and delay jitter are also implemented.

EVALUATION METRICS & RESULTS

To evaluate the three policy types described in Section 1.2, we design three test methodologies: Basic Test, Robustness Test, and Advanced Test.

Basic Test — In the Basic Test, *Accuracy* and *Fairness* statistics are to examine the effectiveness of the *Class-Based Band-*

Tool	Function	Description	Location in Fig. 5
ncftpput [14]	TCP traffic generator	**Traffic:** 20 ncftputs sessions from subnet X to subnet Y. **Packet size:** 1,500 bytes **TCP options:** SACK/timestamp/window-scaling disabled.	A
SmartVoIpQoS [15]	VoIP (UDP) traffic generator	**Traffic:** Single VoIP session with RTP format UDP packets. **Codec:** G.729 (50 frame/sec, frame size=74 byte, around 30kb/s)	M
VoIP Gateway	Same as above	Same as above	K and N
ttt [16]	Real-time traffic bandwidth monitor	Monitor the bandwidth of the traffic passing through it by protocols, source/destination IP, etc.	G
tcpdump [17]	Packet sniffer	Dump each packet's header to the RAM disk to avoid I/O overheads.	A and H
Self-written AWK scripts [2]	Data analyzer	Calculating statistics from the tcpdump result.	G
Self-written WAN emulator [2]	WAN emulator	To have different *delays*, *delay jitters*, and *random/periodic packet loss rates* impairments on different sessions.	H

■ **Table 3.** *Testing tools.*

Statistic	Quantify what?	Definition	Comparison
Accuracy	The differences between: 1. The class bandwidth settings. 2. The measured class bandwidth.	Averaged normalized goodput[1] $\left(\sum_{i=1}^{5} \frac{\text{measured class goodput for Run } i}{\text{given class goodput for Run } i} \right) \Big/ 5$	The closer to 1, the better
Fairness	Fairness of bandwidth obtained among the four sessions in each class.	Averaged CoV[2] among four sessions' goodputs $\left(\sum_{i=1}^{5} CoV \text{ of goodputs (among the four connections in Run } i) \right) \Big/ 5$	The closer to 0, the better
Retransmission ratio	Retransmission ratio in each class.	$\left(\sum_{i=1}^{5} \frac{\text{retransmitted packet count for Run } i}{\text{total packet send count for Run } i} \right) \Big/ 5$	The closer to 0, the better

[1] Goodput is the effective throughput (bytes/time) excluding the bandwidth consumed by retransmission.
[2] CoV denotes "coefficient of variation," which means "standard deviation over mean."

Table 4. *Basic test statistics.*

width Limitation and the *Session Bandwidth Guarantee* policies, respectively. The *Retransmission Ratio* statistic more or less implies how much buffer space is required at the bottleneck, i.e. the DUT. Some of the retransmissions are caused by the forced packet drops. As the queue grows larger, the forced packet drop increases. The WAN-link bandwidth is set to T1 (1.544 Mb/s) and is partitioned into five classes (20, 40, 128, 256, and 1100 kb/s), with each class matching four TCP sessions. Further, each class is set to guarantee that each session will obtain 1/4 bandwidth of the class. This test repeats in five consecutive runs, at 200-second intervals. Within each run, 20 simultaneous FTP sessions (Table 3) pump traffic from A to I for 250 seconds. Tcpdump-sniffed data from 30 to 230 seconds are analyzed. The statistics are explained in Table 4.

An intuitive example in Fig. 7: The 1.544 Mb/s pipe is divided into five small pipes (20kb/s, 40kb/s, 128kb/s, 256kb/s, and 1.1 Mb/s), with each small pipe being fed with four TCP sessions in five consecutive runs. Ideally, each session will obtain a quarter of the bandwidth of the small pipe to which it belongs. For example, each session in the 20kb/s pipe should obtain 5kb/s. However, in the five-run experiments we find that the flow aggregates (14, 7, 30, 5, and 39) are not stable. However, the average of them (19) approaches the ideal 20kb/s. If we easily conclude that the DUT is very accurate in dividing the big pipes into the small pipes, the instability among the five runs is hidden. After examining the Coefficient of Variation (CoV) of the five runs, the CoV (2.4) does not approach zero, so the DUT gives poor stability of accuracy. A DUT with good accuracy but with bad stability of accuracy means that the DUT actually provides poor accuracy. Regarding fairness, we use the CoV to examine the fairness of bandwidth shared by the four sessions. The average of the CoVs (0.32) does not approach zero, so the DUT does not provide good fairness. The standard deviation of the CoVs (0.063) approaches zero, so the DUT provides good stability of fairness. A DUT with poor fairness but with good stability means that the DUT always gives poor fairness among the sessions. For adaptive flows such as FTP, fairness may have little overall impact, whereas for VoIP traffic

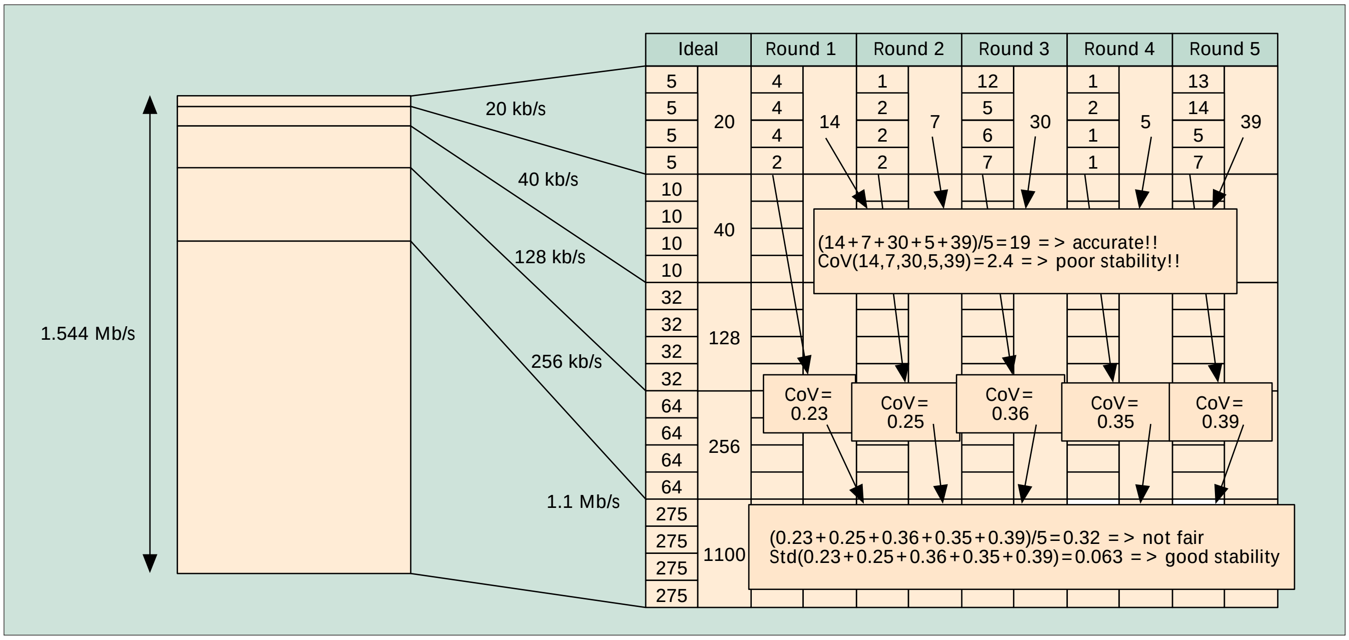


FIGURE 7. *An intuitive example for the Basic Test.*

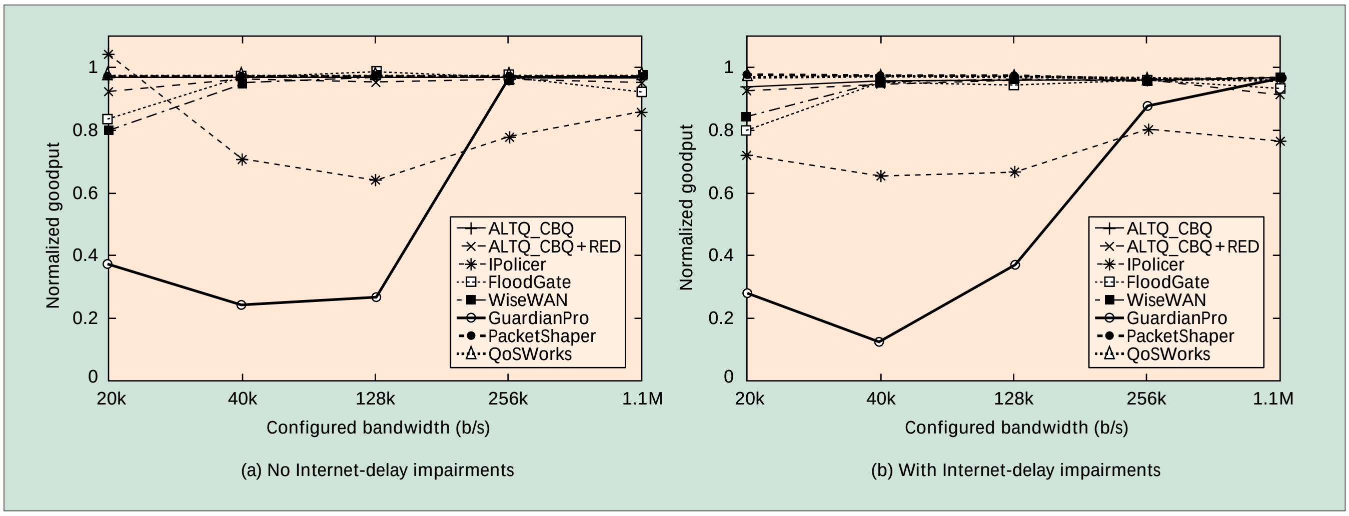


FIGURE 8. *Accuracy statistic.*

fairness could be a problem. We use FTP to test the DUTs because most of their techniques optimize the TCP traffic. VoIP testing is discussed in a later section.

Results of the Accuracy Statistic: Figure 8a reveals that the DUTs can be classified into three groups:

- ALTQ_CBQ, ALTQ_CBQ+RED, PacketShaper, and QoSWorks have the most accurate control for each class.
- WiseWAN and FloodGate are less effective in the narrowband class (20 kb/s) because of their large retransmission ratios, as shown in Fig. 10a.
- *i*Policer and Guardian Pro are the least effective.

Technically speaking, the first two groups have an accurate CBQ. However, the second group does not optimize the buffer requirement well so that overflowed packets diminish their goodputs in the 20 kb/s class. In the preliminary tests for the third group, a single pass-through session can be accurately controlled. However, when pumping more simultaneous sessions, the results become worse and worse.

Results of the Fairness Statistic: Figure 9a also distinguishes three groups:

- PacketShaper is the fairest.
- FloodGate and WiseWAN are less fair in the 20 kb/s class.
- *i*Policer, Guardian Pro, QoSWorks, ALTQ_CBQ, and ALTQ_CBQ+RED provide poor fairness.

In the narrow (20~40 kb/s) classes, pure CBQ has the poorest fairness. After applying a RED to each CBQ class, the unfair classes are alleviated. As an analogy, when the door is narrow (20-40 kb/s) and many people are competing to get out of the door, many people will queue at the door. RED punishes them (drop queued packets) and expects they will quench their aggressiveness. However, for a large door (128-1100 kb/s), RED is ineffective compared to the CBQ+PFQ+TCR (PacketShaper, QoSWorks, WiseWAN) and CBQ+PFQ+RED (FloodGate). Guardian Pro (CBQ+TCR) does not provide a queue for each session, so it cannot gracefully isolate the sessions. *i*Policer (pure TCR) lacks a packet scheduler to limit the total WAN-link bandwidth. Thus, sessions will influence each other at the next hop (Cisco router) and cause unfair goodputs.

Results of the Retransmission Statistic: Figure 10a shows a large retransmission ratio in the narrowband classes (20~40 kb/s), except for PacketShaper and QoSWorks, but especially in WiseWAN, *i*Policer, FloodGate, and ALTQ_CBQ+RED. FloodGate and ALTQ_CBQ+RED employ RED, so they have high retransmissions. WiseWAN does not stand at the LAN-WAN interconnected segment (Fig. 5), so enormous packet losses occur at the Cisco router. Results of *i*Policer and Guardian Pro cannot be interpreted.

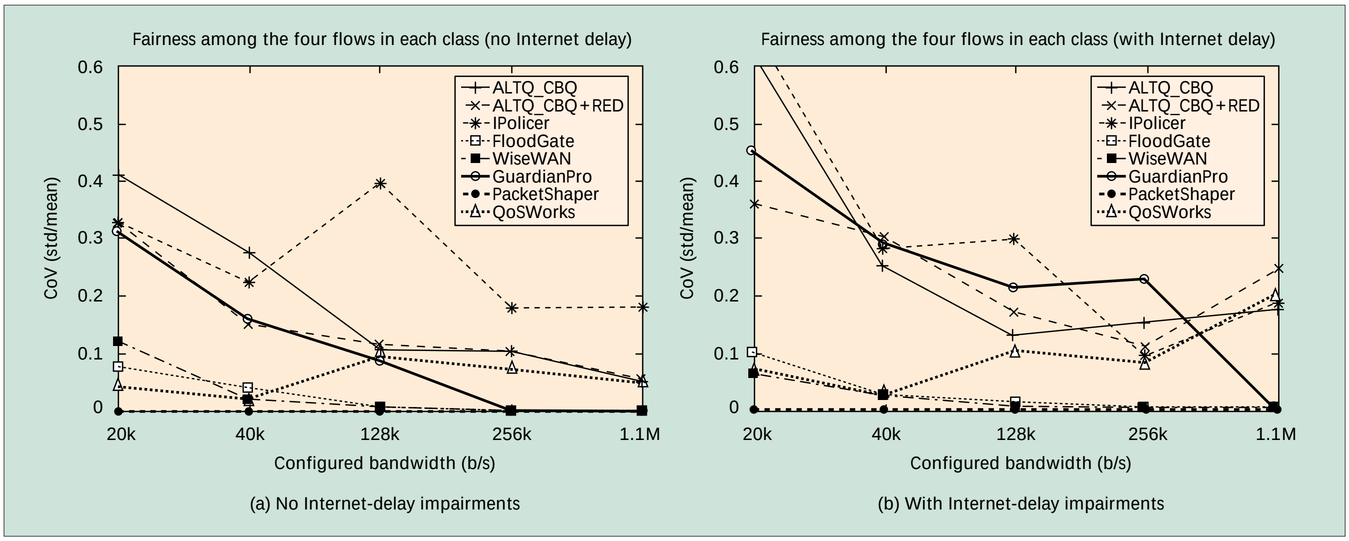


FIGURE 9. *Fairness statistic.*

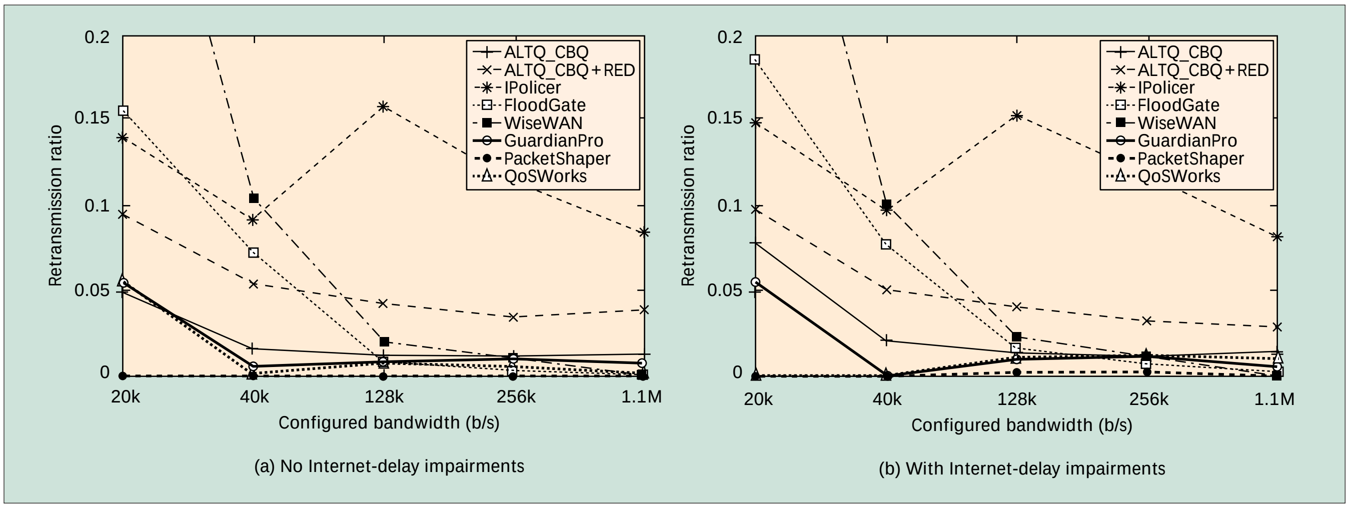


FIGURE 10. *Retransmission ratio statistic.*

Robustness Test — Since many DUTs have diverse buffer management techniques to provide session bandwidth guarantee for individual TCP sessions, this test focuses on the DUTs' robustness under different conditions. Packets may be generated by different operating systems with different TCP implementations and pass through paths with various delays and loss rates. Long-distance TCP sessions are expected to be vulnerable to Internet losses because they require more time to obtain ACKs to recover back to their target bandwidth. Besides, a TCP session will slow its sending rate when encountering packet losses. When the packets are dropped more frequently, the TCP sender will send more slowly. Consequently, three subtests were conducted:

- Heterogeneous Internet delays.
- Various Internet loss rates.
- Heterogeneous sending operating systems.

Table 5 describes our test methodology.

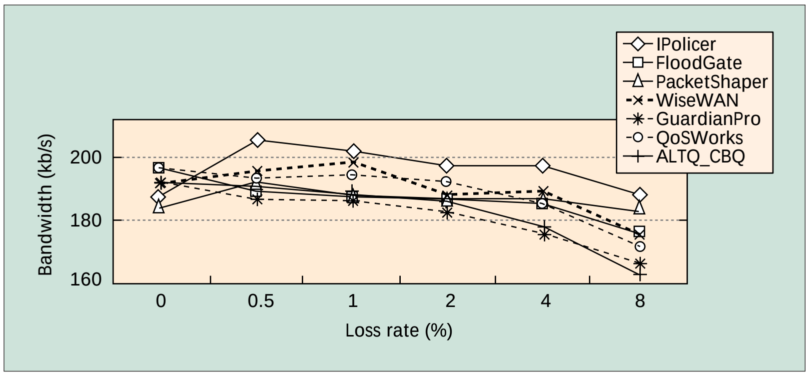
With Heterogeneous Internet Delays — Figure 8b, Fig. 9b, and Fig. 10b separately demonstrate the accuracy, fairness, and retransmission ratio compared to those in the basic test. Figure 8b is almost the same as Fig. 8a because their CBQ queuing behaviors are mostly the same (always backlogged). However, Fig. 9b reveals that packet loss at the DUT (either due to RED or buffer overflow) dominates the fairness statistic. Long-distance sessions passing through the *i*Policer and the ALTQ_CBQ suffer due to buffer overflows at the DUT because long-distance TCP sessions require more time to recover the degraded goodputs than short-distance TCP sessions. The ALTQ_CBQ+RED helps the ALTQ_CBQ at the 20 kb/s class, but becomes even worse in the 40, 128, and 1100 kb/s classes. CBQ+PFQ+RED in FloodGate (Fig. 9b) also gives worse goodput in the 20 kb/s class than that in Fig. 9a. QoSWorks (CBQ+TCR), though having the same retransmission ratio in Fig. 10a and 10b, provides even poorer fairness in the 1.1 Mb/s class. This may be caused by inaccurate RTT measurement. Guardian Pro (CBQ+TCR) also shows this phenomenon (no extra retransmission ratio, but it has extra unfairness in the 20-256 kb/s classes).

With Various Packet Loss Rates — A TCP session slows down its transmission rate when packet losses occur. Figure 11 shows the TCP goodput passing through each DUT with different Internet packet loss rates. Almost all the DUTs can smoothly lower their goodputs as packet loss rate increases, except for PacketShaper and *i*Policer. With detailed analysis using tcpdump, PacketShaper and *i*Policer stop sizing the TCP window (the TCR approach introduced earlier) when they have detected the TCP loss events (triple duplicate ACKs). Thus, the TCP sending window suddenly bumps up and causes a burst of packets pumping to the DUT, resulting in a higher goodput at the 0.5 percent loss rate.

With Different Sending Operating Systems — In this test, TCP sessions sending from different operating systems passing through PacketShaper have different results. In Fig. 12, the X axis is the elapsed time, the Y axis is the accumulated bytes sent, and thus the slope is the bandwidth. PacketShaper shrinks the TCP window to the condition that no more than four packets are in the WAN pipe. Thus, each packet loss resorts to a retransmission timeout instead of using fast retransmit [2].

Test Item	Description		Comparison standard
	DUT settings	Test methodology	
Internet delays	Same as basic test.	WAN delays of the four sessions in each class are 10ms, 50ms, 100ms, 150ms	Same as the basic test
Loss rates	200 kb/s for the test session.	A single TCP session is tested with 0.5%, 1%, 2%, 4%, and 8% periodic loss rates. The goodput is averaged over 200 seconds.	Whether the goodput can smoothly degrade.
Sending OS	80 kb/s for the test session.	1. WAN: delay=50ms, periodic loss rate = 1%. 2. TCP source OS = {Linux 2.2.14, Windows 2000, FreeBSD 4.0, Solaris8}, receiver = Linux 2.2.14. 3. Each time a single TCP session is tested.	How closely the byte-time lines of the operating systems can overlap with each other.

Table 5. *Robustness test methodology.*



■ **FIGURE 11.** *Goodput under various packet loss rates.*

Some OSes use a coarse-grained retransmission timer such that they slowly retransmit the lost packets. In contrast, Linux keeps a fined-grained retransmission timer and has the best performance when packet losses occur. *i*Policer has a serious bug when sending data from Windows 2000 to Linux 2.2.14. The tcpdump tool detected that the TCP ACK header length is miscalculated when passing through the *i*Policer, causing incorrect triggering of data packets from TCP senders. TCP has many options and various implementations, so explicitly modifying the ACK packet requires extensive tests. The other products can fairly treat TCP sessions from different operating systems.

Advanced Test — The advanced test aims to highlight the advanced features of the DUTs:

- The inter-class, intra-class bandwidth borrowing.
- The VoIP quality under a heavy background traffic.

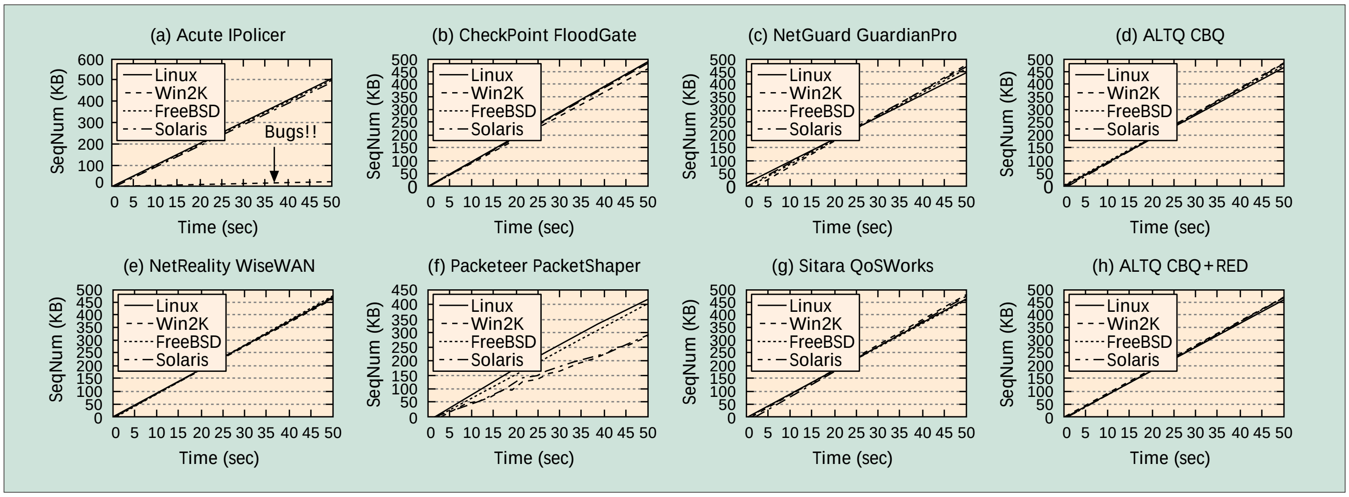
By starting and terminating the two TCP sessions in sequence, we can observe how effectively the two sessions can borrow the bandwidth from each other. The voice quality is separately tested through the SmartBits (quantitatively with PSQM) and the Cisco VoIP Gateway (subjectively by listening with ears). Table 6 describes our test methodology. The testbed for SmartBits and Cisco VoIP Gateway are connected as in Fig. 6. The SmartBits test can tell the PSQM ratings by generating, receiving, and analyzing the built-in voice traffic. With VoIP gateways, we dial the phone from X to Y in Fig. 6, speak to the phone at X, and listen to the phone at Y simultaneously.

Inter-Class Bandwidth Borrowing — In this test for ALTQ_CBQ and ALTQ_CBQ+RED, session 1 can only borrow approximately 51 percent of the newly available bandwidth released by session 2. FloodGate, PacketShaper, and QoSWorks can do 100 percent bandwidth borrowing. *i*Policer does not have this function, so it performs badly. Results are omitted due to space limitation.

Intra-Class Bandwidth Borrowing — In this test for *i*Policer, after session 2 terminates, session 1 cannot occupy any of the newly available bandwidth within the class. Guardian Pro and ALTQ_CBQ have fluctuating bandwidth sharing between the two sessions since they cannot effectively isolate the sessions. This phenomenon in ALTQ_CBQ is again slightly alleviated after applying RED. The other DUTs are quite similar in this test. Results are omitted due to space limitation

VoIP Quality — With the 125 kb/s WAN link (Fig. 13 and Table 7), even the baseline test shows zero jitter/latency/loss, and the PSQM rating equals 2.2 instead of 0. The G.729 codec compresses with distortion so the best PSQM is 2.2 instead of 0. After pumping the background FTP traffic, all DUTs except QoSWorks2 fail the test with large latencies, delay jitters, and loss rates. Transmitting a large packet (1500 bytes) onto the narrowband WAN link (125 kb/s) takes a long time such that its following small voice packet (74 bytes) has to wait until the previous large packet is completely scheduled out. After the QoSWorks exercises the MSS-clamping (results of the QoSWorks2) to limit all background FTP packets to 256 bytes, the voice quality becomes the best both in the Smartbits test and the VoIP Gateway tests. While it is promising, readers should be aware that a link full with small-size packets wastes bandwidth because of the overhead of headers.

In the VoIP Gateway test (Table 7), the results resemble those in the SmartBits test. After pumping the background FTP traffic, the dial can never be established. With the aid of bandwidth managers, the dial can be established except for the *i*Policer due to its inability to recognize UDP packets. The time to establish the call is dominated by the ability of inter-class bandwidth borrowing that can instantly allocate sufficient bandwidth from the background FTP traffic. PacketShaper provides the shortest time to establish the call, while ALTQ and QoSWorks take a long time (17~18 seconds). Since QoSWorks is essentially an enhanced ALTQ on a FreeBSD, some of their characteristics resemble each other. Regarding the voice quality, only the voice through the PacketShaper can be



■ **FIGURE 12.** *With different sending operating systems.*

Test item	Description		Comparison standard
	DUT settings	Test methodology	
Inter-class bandwidth borrowing	1. Link speed = T1 (1.544 Mb/s), divided into two classes, A, B. A=B = 777kb/s. 2. Class A matches session 1, Class B matches session 2. 3. A and B can borrow with each other.	Session 1 and 2 are started in sequence. After five seconds, session 2 stops.	1. Stability of each session. 2. How seamlessly the total bandwidth line can be when session 1 terminates.
Intra-class bandwidth borrowing	1. Link speed = T1 (1.544 Mb/s), divided into one class, A. A = 1.544 Mb/s. 2. The class matches session 1 and 2. 3. Per-session bandwidth: at least 777 kb/s, at most 1.544 Mb/s.		
VoIP test using SmartVoIpQoS	1. Link speed = {T1,125kb/s}, divided into two classes, A, B. 2. A = 30kb/s for voice traffic, B = {T1,125kb/s}-30kb/s for FTP traffic. 3. FTP traffic can occupy the voice class until voice traffic begins.	Background: 20 FTP sessions. Foreground: a 30kb/s G.729 VoIP session.	PSQM[1], jitter, delay, and loss.
VoIP test using VoIP Gateway (Cisco 1750)		Background: 20 FTP sessions. Foreground: Dial a phone (JP to NP, G.729 codec), hold X's and Y's phones, speak 1 to 10 at 2 word/sec, and judge the voice quality.	Measure with ears[2] by speaking to phone X and listening to phone Y.

[1] The PSQM (Perceptual Speech Quality Measurement) rating is calculated from the received (typically distorted) voice streams. PSQM rated as 6.5 has the poorest quality

[2] The VoIP Gateway is set to continuously sample the sound even when the primary tester keeps silent. Thus the session is always around 30 kb/s.

Table 6. *Advanced test methodology.*

barely recognized. With MSS-clamping, QoSWorks becomes the best though it takes six seconds to establish the call.

Conclusion

With the growth of many devices installed at the subscribed links, black-box testing is very important to evaluate each method. The testbed should be designed to be as compact as possible to easily reproduce the results. However, the testbed should be sophisticated enough to generate realistic conditions to uncover the unknown phenomenon. This study describes a compact but practical black-box testbed that emulates real-life Internet conditions, such as many simultaneous sessions from different hosts (physically only one host), heterogeneous WAN delays, delay jitters, packet loss rates, and different TCP source implementations, to evaluate commercial/open-source bandwidth enforcement techniques. The main contributions are to discover the unknown phenomenon and to quantify the known behaviors. Numerical results demonstrate that

- Only the CBQ + PFQ + TCR can solve the most difficult scenario (20 kb/s class).
- The TCR approach may degrade the goodput, fairness, or compatibility even under slight packet loss rates (0.5 percent).

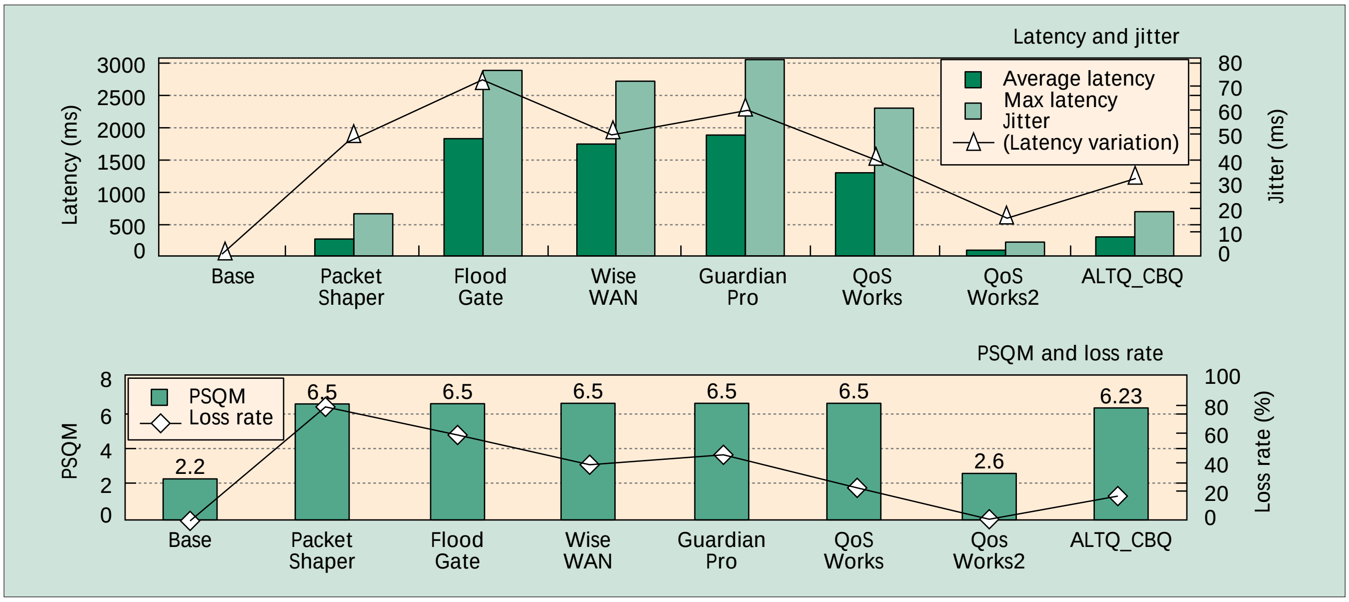


Figure 13. *VoIP quality test through SmartBits.*

- Without PFQ, the CBQ+TCR (due to inaccurate RTT measurement) and the CBQ+RED have limited ability to isolate the sessions (especially for RED).
- The G.729 VoIP quality over 125 kb/s access links becomes good only after exercising MSS-clamping to shrink the packet size of the background traffic down to 256 bytes.

	125 kb/s WAN link speed		
	Time to establish	End-to-end delay	Voice quality (legibility)
Baseline (only voice)	<1 sec	< 0.1 sec	Very good
Baseline (with background FTP)	Cannot establish the session		
*i*Policer	Cannot be tested (no UDP traffic control)		
FloodGate	7 sec	1 sec	Very poor (<10%)
Guardian Pro	3 sec	1.5 sec	Ultra poor (<1%)
WiseWAN	7 sec	1.5 sec	Ultra poor (<1%)
PacketShaper	1 sec	1 sec	Poor (60%)
ALTQ_CBQ	18 sec	1 sec	Very poor (<10%)
QoSWorks	17 sec	1 sec	Very poor (<10%)
QoSWorks with MSS-clamping	6 sec	< 0.2 sec	Very good

Table 7. *VoIP test results through VoIP gateway.*

A detailed comparison of functionality [2] among the DUTs gives further directions for enhancing the open source solutions, such as Packeteer's traffic discovery and QoSWorks' intuitive user interface. The ALTQ software lacks session-bandwidth guarantee and thus needs further refinements to satisfy the needs of enterprises.

Based on the designed testbed and methodologies, this study can differentiate the properties of different combinations of techniques used in the products. We believe that there are other methods to discover some other unknown behaviors of the black boxes. The importance of the performance metrics heavily depends on how the functions can solve the problem for the network administrators. For example, some would say TCP friendliness is important because a nonconformant TCP sender will affect other TCP traffic. Others would say that aggressively competing for WAN bandwidth is good for a company even though the packet loss rate increases. For another example, fairness among sessions does not make any sense for network administrators on the ISP side because they only focus on accurately providing bandwidth to each subscribed last-mile/data-center users. The sessions within each subscribed link may be encrypted by the IPsec protocol and cannot be recognized by the bandwidth manager on the ISP side. However, bandwidth managers that provide fairness among sessions on the customer side can be very useful to guarantee the bandwidth of each session. We have focused on discovering the phenomena instead of categorizing the results as good or bad. Several RFCs [18–20] define the terms for benchmarking layer-2 switches and firewall systems but do not define the testing methods. Given that bandwidth managers are becoming more popular, we look forward to seeing other state-of-the-art testing for such devices that can ensure the quality of the products.

REFERENCES

[1] S. Blake *et al.*, "An Architecture for Differentiated Services," RFC 2475, Dec. 1998.
[2] H. Y. Wei, WAN Emulator, http://speed.cis.nctu.edu.tw/wanemu/
[3] M. Allman, V. Paxson, and W. Stevens, "TCP Congestion Control," RFC 2581, Apr. 1999.
[4] W. R. Stevens, *TCP/IP Illustrated Volume 1 — The Protocols*, Addison-Wesley, 1994.
[5] M. W. Wu and Y. D. Lin, "Open Source Software Development: An Overview," *IEEE Computer*, June 2001, pp. 33–38.
[6] K. Cho, "Alternate Queuing for BSD UNIX (ALTQ)," http://www.csl.sony.co.jp/person/kjc/kjc/software.html
[7] B. Braden *et al.*, "Recommendations on Queue Management and Congestion Avoidance in the Internet," RFC 2309, Apr. 1998.
[8] S. Karandikar *et al.*, "TCP Rate Control," *ACM Comp. Commun. Review*, vol. 30, no. 1, Jan. 2000.
[9] L. L. Peterson and B. S. Davie, *Computer Networks: A System Approach, 2nd Ed.,* Morgan Kaufmann Publishers, 2000.
[10] A. Varma and D. Stiliadis, "Hardware Implementation of Fair Queuing Algorithms for Asynchronous Transfer Mode Networks," *IEEE Commun. Mag.*, vol. 35, no. 12, Dec. 1997, pp. 54–68.
[11] D. Stiliadis and A. Varma, "Latency-Rate Servers: A General Model for Analysis of Traffic Scheduling Algorithms," *IEEE/ACM Trans. Net.*, vol. 6, no. 5, Oct. 1998, pp. 611–24.
[12] S. Floyd and V. Jacobson, "Link-Sharing and Resource Management Models for Packet Networks," *IEEE/ACM Trans. Net.*, vol. 3, no. 4, 1995, pp. 365–86.
[13] S. Floyd and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance," *IEEE/ACM Trans. Net.*, vol. 1, no. 4, Aug. 1993, pp. 397–413.
[14] Ncftpput Software, http://www.ncftp.com
[15] Spirent Communications, http://www.spirent.com
[16] K. Cho, "TTT: Tele Traffic Tapper," http://www.csl.sony.co.jp/person/kjc/kjc/software.html
[17] Lawrence Berkeley National Laboratory, "tcpdump," http://www-nrg.ee.lbl.gov
[18] S. Bradner, "Benchmarking Terminology for Network Interconnection Devices," RFC1242, July 1991.
[19] S. Bradner and J. McQuaid, "Benchmarking Methodology for Network Interconnect Devices," RFC 2544, Mar. 1999.
[20] D. Newman, "Benchmarking Terminology for Firewall Performance," RFC 2647, Aug. 1999.

BIOGRAPHIES

HUAN-YUN WEI (hywei@cis.nctu.edu.tw) received the B.S. and the Ph.D. degrees in computer and information science from National Chiao Tung University, Hsinchu, Taiwan, in 1998 and 2003, respectively. His research interests include TCP rate shaping for enterprise edge devices, queuing analysis of TCP traffic shaping, high-speed packet classification, and performance analysis of network security gateways. He is especially interested in the design and implementation of FreeBSD/NetBSD/Linux kernels.

YING-DAR LIN (ydlin@cis.nctu.edu.tw) received the Bachelor's degree in computer science and information engineering from National Taiwan University in 1988, and the M.S. and Ph.D. degrees in computer science from the University of California, Los Angeles, in 1990 and 1993, respectively. His research interests include design, analysis, and implementation of network protocols and algorithms, wire-speed switching and routing, quality of services, network security, and content networking. Dr. Lin is a member of ACM and IEEE. He is the founder and director of Network Benchmarking Lab (NBL) which reviews the functionality, performance, conformance, and interoperability of networking products.