Packeteer Technical White Paper Series

TCP Rate Control and Alternatives

May 2002

Packeteer, Inc.
10495 N. De Anza Blvd.
Cupertino, CA 95014
408.873.4400
info@packeteer.com
www.packeteer.com

Table of Contents

TCP Rate Control and Alternatives

The de-facto network protocol standard of the TCP/IP protocol suite offers us many advantages and strengths, but the management and enforcement of QoS (Quality of Service) are not among them. Many standards have attempted to address this deficiency with little success — RSVP, expanded bandwidth, CoS/ToS, routers' queuing solutions, and more. Packeteer's PacketShaper product line with its patented TCP Rate Control delivers an effective QoS solution.

TCP Review

The Transmission Control Protocol (TCP) provides connection-oriented services for the protocol suite's application layer — that is, the client and the server must establish a connection to exchange data. TCP transmits data in segments encased in IP datagrams, along with checksums, used to detect data corruption, and sequence numbers to ensure an ordered byte stream. TCP is considered to be a reliable transport mechanism because it requires the receiving computer to acknowledge not only the receipt of data but also its completeness and sequence. If the sending computer doesn't receive notification from the receiving computer within an expected time frame, the sender times out and retransmits the segment.

TCP uses a sliding window flow-control mechanism to control the throughput over wide-area networks. As the receiver acknowledges initial receipt of data, it advertises how much data it can handle, called its *window size*. The sender can transmit multiple packets, up to the recipient's window size, before it stops and waits for an acknowledgment. The sender fills the pipe, waits for an acknowledgment, and fills the pipe again.

TCP's *slow-start* algorithm attempts to take full advantage of network capacity. While TCP flow control is typically handled by the receiver, the slow-start algorithm uses a congestion window, which is a flow-control mechanism managed by the sender. With TCP slow-start, when a connection opens, only one packet is sent until an ACK is received. For each received ACK, the sender can double the transmission size, within bounds of the recipient's window. Note that this algorithm introduces an exponential growth rate. But eventually, packets are dropped.

TCP's congestion-avoidance mechanisms attempt to alleviate the problem of abundant packets filling up router queues. TCP increases a connection's transmission rate using the slow-start algorithm until it senses a problem and then it backs off. It interprets dropped packets and/or timeouts as signs of congestion. The goal of TCP is for individual connections to burst on demand to use all available bandwidth, while at the same time reacting conservatively to inferred problems in order to alleviate congestion.

The Bandwidth Challenge

TCP/IP was designed primarily to support two traffic applications — FTP and Telnet. Today's networks must cope with increased application and traffic demands — high-speed users, interactive web traffic, web-based applications, peer-to-peer architectures, server-based applications, and more. These demands impact a user's quality of service by causing delays and bottlenecks at speed-conversion points in the network.

Many of TCP's control or reliability features contribute to performance problems:

- Retransmitting when the network cloud drops packets or delays acknowledgments

  When packets drop or acknowledgements are delayed due to congested conditions and overflowing router queues, retransmissions simply contribute more traffic and worsen the original problem.

- Steadily increasing bandwidth allocation

  TCP's slow-start algorithm rapidly turns into fast domination. Without regard for traffic's urgency, concurrent users, or competing applications, TCP simply expands each flow's usage until it causes problems. This turns each sizeable traffic flow into a bandwidth-hungry, potentially destructive consumer that could undermine equitable or appropriate allocation of network resources.

- Imposing overload

  TCP expands allocation until packets are dropped or responses are delayed. It floods routers by design. Once TCP infers congestion, it decreases bandwidth allocation rates.

As large amounts of data are forwarded to the routers at WAN access links, more congestion forms, bigger queues form, more delay is introduced, more packets are discarded, more timeouts occur (from both latency and discards), more retransmissions are sent, more congestion forms, and so on.

The large, bulk transfers aren't the only ones that suffer. When a large packet burst creates a bottleneck, small interactive packets (that don't burst and don't create problems) are delayed as well. Users notice most acutely when interactive response times turn sluggish.

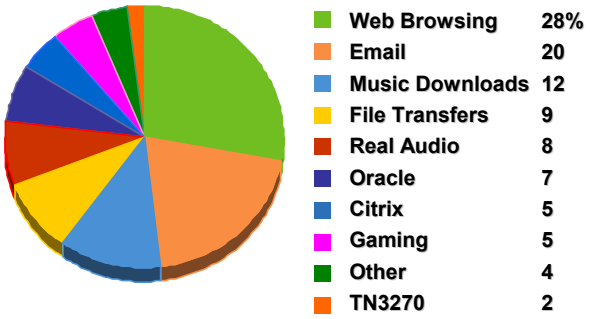
Bandwidth Management Approaches

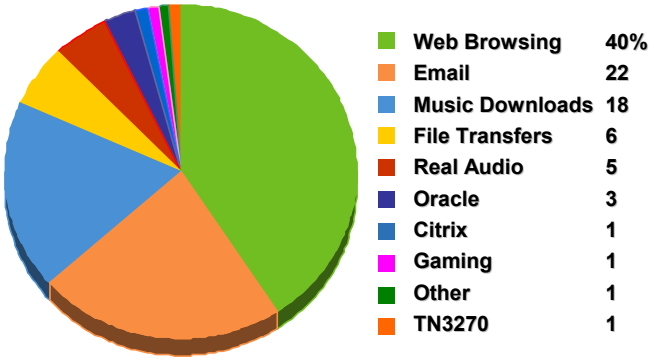
When faced with bandwidth constraints and performance that is unpredictable, inequitable, inconsistent, or just plain too slow, a number of solutions come to mind. This section addresses the following potential solutions, focusing on their advantages and limitations:

- Additional bandwidth
- Verbal or written decrees
- Queuing schemes on routers
- TCP Rate Control
- Multi-Protocol Label Switching (MPLS)

Adding Bandwidth

An obvious approach to overcoming bandwidth limitations is to add more bandwidth. At best, this is a short-term solution. As discussed, the nature of TCP is to expand traffic flows until bandwidth is consumed. Therefore, as soon as you increase bandwidth, it's consumed, and not necessarily by the traffic you'd choose. You end with more traffic passing through bigger and more expensive pipes, but performance for critical, urgent, and interactive applications still suffers.





What happens when you add bandwidth? Doubling the bandwidth allocates haphazard increased amounts to the most demanding, bandwidth-hungry applications — probably not the most urgent and critical applications.

You might actually need to expand your bandwidth, but you shouldn't proceed until you know that more network capacity will solve your issues. Do you actually need more bandwidth? Or do you need to be able to control the bandwidth you have? The following activities can help answer these questions:

- Detect and identify the types of traffic and applications traversing your network.
- Analyze those applications' behavior, usage, and performance over time.

  View your existing capacity divided into portions corresponding to the top bandwidth consumers, similar to the pie charts above. Evaluate whether considerable bandwidth goes to unimportant or non-urgent applications. Baseline response times. Determine how much of current capacity is wasted on retransmissions. Consider if individual high-capacity users undermine others' performance. Compare response times to performance needs. And so on.
- Allocate bandwidth to applications and users based on urgency and importance. Pace the large bursty transfers so as to not interfere with other applications' performance. Contain unneeded applications to bandwidth not otherwise used. Protect bandwidth for critical applications.

Re-examine and re-evaluate. Most likely, managed bandwidth translates to consistent and appropriate performance. If not, additional bandwidth can be applied judiciously and beneficially if these same techniques are sustained after an upgrade.

These procedures are not explained in this paper. PacketShaper can assist you in conducting these types of activities. For more information, see "*Four Steps to Application Performance*" on the Packeteer web site.

Verbal or Written Decrees

A university says, "Don't download MP3 music files." Or a corporation says, "Don't use Internet radio; put a radio on your desk instead." Managerial edicts are only as effective as an organization's ability to enforce them.

In addition, this approach only impacts the network load due to unsanctioned traffic. It does nothing to manage the concurrence of file transfers, web-based applications, large email attachments, ERP,

Citrix-based applications, and all the other traffic that is both necessary and important. Real-world traffic has an incredible variety of requirements that complicate the task of enforcing appropriate performance for all.

Using Queuing Schemes on Routers

For the most part, network devices have kept pace with evolving high-speed technology. Routers provide queuing (buffering and waiting) schemes, such as weighted fair queuing, priority output queuing, and custom queuing. These schemes attempt to prioritize and distribute bandwidth to individual data flows so that low-volume applications, such as interactive web applications, don't get overtaken by large data transfers, FTP for example.

The effectiveness of techniques that routers or any bandwidth solution use to handle congestion and performance issues are dependent on a very important capability — traffic classification. It's hard to control a certain type of traffic if you can't identify it. The ability to differentiate many types of traffic enables any solution to then apply its congestion-avoidance or performance-improvement techniques to carefully and precisely targeted traffic.

Router-based solutions have improved in the recent past. For example, they can now enforce per-traffic-type aggregate bandwidth rates for any traffic type they can differentiate. But a variety of router and queuing limitations remain:

- Routers manage bandwidth passively, discarding packets and providing no direct feedback to end systems. Routers use queuing (buffering and waiting) or packet tossing to try to control traffic sources and their rates.
- Queues, by their definition, oblige traffic to wait in lines and add delay to transaction time. Dropping packets is even worse for applications since it forces the application to wait for a timeout and then retransmit.
- Queues do not proactively control the rate at which traffic enters the wide-area network at the other edge.
- Queuing-based solutions are not bi-directional and do not control the rate at which traffic travels into a LAN from a WAN, where there is no queue.
- Routers can't enforce per-flow minimum or maximum bandwidth rates.
- Routers don't allow traffic to expand beyond its aggregate bandwidth limit when congestion and competing traffic are not issues.
- Queuing results in chunkier traffic and erratic performance because multiple, independent TCP sources are competing for bandwidth, ramping up and backing off, and queues accumulate at the access link. Queuing, especially weighted fair queuing, doesn't work well for chunky flows because packets arriving in chunks tend to be discarded.
- Routers don't enable distinct strategies for high-speed and low-speed connections.
- Routers don't allow specification of the maximum number of allowed flows for a given type of traffic or a given sender.

- Queuing lacks bits-per-second precision and attacks the problem only after congestion occurs. It's an after-the-fact approach to a real-time problem.
- Traffic classification is too coarse. Routers can't automatically detect and identify applications as they pass. They can't classify traffic by user name, Oracle database, MAC address, user speed, LDAP host list, and Citrix interactive versus print traffic. They can't identify non-IP traffic, much VoIP traffic, peer-to-peer traffic, games, and HTTP on non-standard ports.

Although routers don't identify large numbers of traffic types and enforce a variety of flexible allocation strategies, a strong case could be made that they shouldn't. The first and primary function of a router is to route. Similarly, although a router has some traffic-blocking features, it doesn't function as a complete firewall. And it shouldn't. It needs to focus its processing power on prompt, efficient routing responsibilities.

Precise Provision — TCP Rate Control

Imagine putting fine sand, rather than gravel, through a network pipe. Sand can pass through the pipe more evenly and quickly than chunks. Packeteer's patented TCP Rate Control conditions traffic so that it becomes more like sand than gravel. These smoothly controlled connections are much less likely to incur packet loss and impose latency.

TCP Rate Control overcomes TCP's shortcomings (those explained earlier in *The Bandwidth Challenge*), proactively preventing congestion on both inbound and outbound traffic. TCP Rate Control paces traffic, telling the end stations to slow down or speed up. It's no use sending packets any faster if they will be accepted only at a particular rate once they arrive. Rather than discarding packets from a congested queue, TCP Rate Control paces packets to prevent congestion. It forces a smooth even flow rate that maximizes throughput.

Packeteer's TCP Rate Control is embedded in its PacketShaper product line where TCP Rate Control is just one of many PacketShaper features that control bandwidth allocation and application performance.

Classification, an Essential Prerequisite

As stated in the previous section, precise traffic classification is a crucial part of bandwidth management. You can't control a given type of traffic if you can't identify it. The ability to differentiate hundreds and even thousands of different types of traffic gives PacketShaper a strategic advantage in that it can apply TCP Rate Control and PacketShaper's other bandwidth-allocation policies to precisely the right traffic. PacketShaper can differentiate traffic based on:

- Application
- Protocol
- Port number
- URL or wildcard
- Host name
- LDAP host lists
- Diffserv setting
- MPLS labels
- IP precedence bits
- IP or MAC address
- Subnet
- Travel direction (inbound/outbound)
- Source / destination
- Host speed range
- Mime type
- Web browser

- Oracle database
- Citrix published application
- Citrix ICA priority tagging
- VLAN varieties (ISL and 802.1p/q)

PacketShaper peers into packets and headers looking for characteristic markers of specific applications. As a result, it can distinguish multiple applications using the same port, follow an application as it port hops, spot traffic for specific databases, and recognize other traffic that proves illusory for routers and similar solutions. Each traffic category is called a *traffic class*.

For more about PacketShaper's traffic classification features, see the "Gaining Visibility into Application and Network Performance" paper on Packeteer's web site.

How TCP Rate Control Works

Traffic consists of chunks of data that accumulate at access links where speed conversion occurs. To eliminate the chunks, TCP Rate Control paces or smoothes the flow by detecting a remote user's access speed, factoring in network latency, and correlating this data with other traffic flow information. Rather than queuing data that passes through the box and metering it out at the appropriate rate, PacketShaper induces the sender to send just-in-time data. By changing the traffic chunks, or bursts, to optimally sized and timed packets, PacketShaper improves network efficiency, increases throughput, and delivers more consistent, predictable, and prompt response times.

TCP Rate Control uses three methods to control the rate of transmissions:

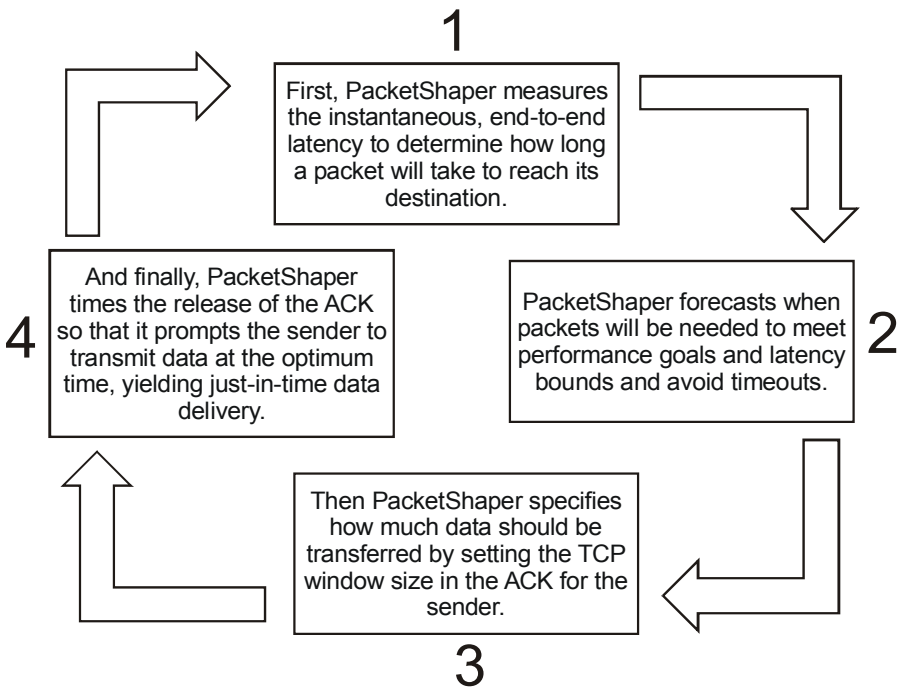
- Detects real-time flow speed
- Meters acknowledgments going back to the sender
- Modifies the advertised window sizes sent to the sender

Just as a router manipulates a packet's header information to influence the packet's direction, PacketShaper manipulates a packet's header information to influence the packet's rate.

TCP autobaud is Packeteer's technology that allows PacketShaper to automatically detect the connection speed of the client or server at the other end of the connection or on the other side of the Internet. This speed-detection mechanism allows PacketShaper to adapt bandwidth-management strategies even as conditions vary.

PacketShaper is a *predictive scheduler* that anticipates bandwidth needs and meters the ACKs and window sizes accordingly. It uses autobaud, known TCP behaviors, and bandwidth-allocation policies as predictive criteria.

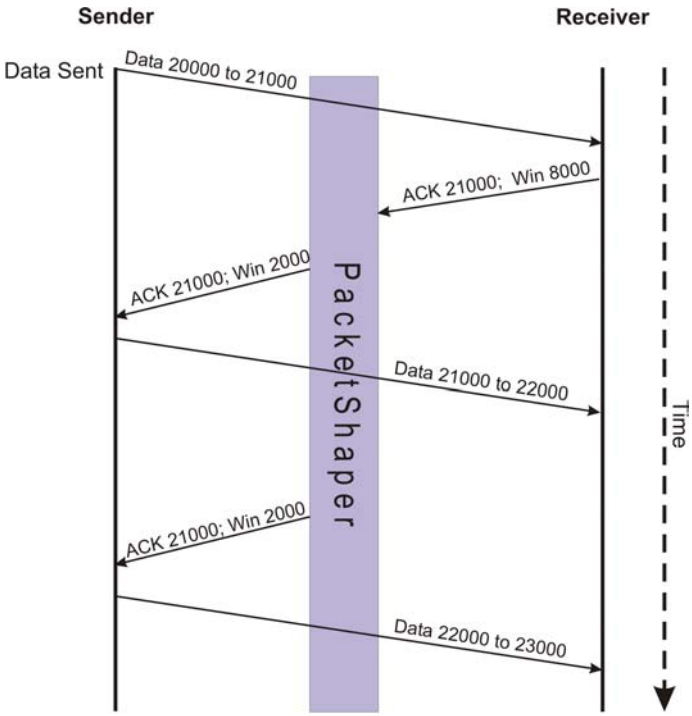
PacketShaper changes the end-to-end TCP semantics from its position in the middle of a connection. It calculates the packet round-trip time, intercepts the acknowledgment, and holds onto it for the amount of time that is required to smooth the traffic flow and increase throughput without incurring retransmission timeout. It also supplies a window size that helps the sender determine when to send the next packet and how much to send.



Evenly spaced packet transmissions yield significant multiplexing gains in the network. As packet bursts are eliminated, network utilization can increase up to 80 percent. Packet spacing also avoids the queuing bias imposed by weighted fair queuing schemes, which force packet bursts to the end of a queue, giving preference to low-volume traffic streams. Thus, sand-like packet transmissions yield increased network utilization and proceed cleanly through weighted fair queues.

In this packet diagram, PacketShaper intervenes and paces the data transmission to deliver predictable service. The sequence described by the packet diagram includes:

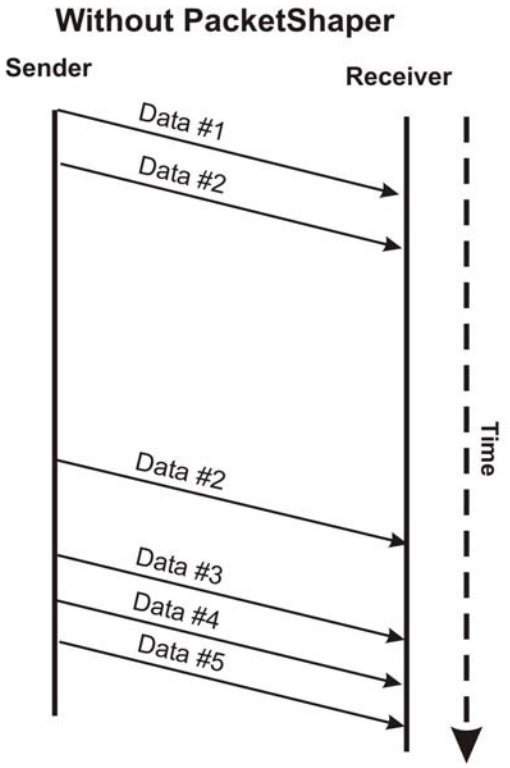
- A data segment is sent to the receiver.
- The receiver acknowledges receipt and advertises an 8000-byte window size.
- PacketShaper intercepts the ACK and determines that the data must be more evenly transmitted. Otherwise, subsequent data segments will queue up and packets will be tossed because insufficient bandwidth is available. In addition, more urgent and smaller packets from interactive applications would be held behind the flood of this more bulky data.
- PacketShaper revises the ACK to the sender, causing the sender to immediately emit data (ACK sequence number plus the window size allows the sender to transmit an additional packet).

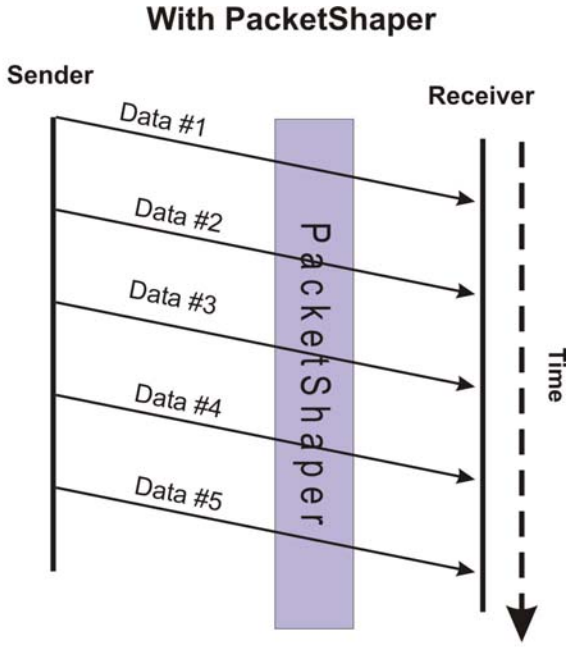


PacketShaper Takes Control of the Connection

The figure on the left in the following illustration with two packet diagrams provides an example of the traffic patterns when natural TCP algorithms are used. Near the bottom, observe the packet burst that occurs. This is quite typical of TCP's slow start (but huge later) growth and is what causes congestion and buffer overflow. Note

that the second packet must be transmitted twice, an unnecessary waste. The figure below on the right provides an example of the evenly spaced data transmissions that occur when TCP Rate Control is used. This even spacing not only reduces router queues but also helps increase the average bits per second since it uses more of the bandwidth more of the time. The pros and cons of the two mechanisms are listed below each diagram





Without PacketShaper: Chunky traffic flow (like gravel), less throughput, bursty sporadic transfer, more retransmissions.

With PacketShaper: Smooth traffic flow (like sand), more throughput, consistent transfer rate, fewer retransmissions.

Precise Bandwidth Allocation

PacketShaper's *policies* and *partitions* are rules governing how PacketShaper allocates bandwidth to each traffic class. They build on the power provided by TCP Rate Control and enable you to control bandwidth on a flow-by-flow or aggregate basis. Policies determine how an application's individual flows are treated in the context of competing applications as well as other flows of the same application. With policies, you can give each flow of critical traffic the bandwidth it needs for optimum performance, as well as protect it from bandwidth-hungry, less urgent traffic. Partitions manage allocation for the aggregate total of all flows, so that all of the flows for one traffic class are controlled together as one. With partitions, you can both protect and limit one type of traffic to a defined amount of bandwidth.

Types of Policies and Partitions

PacketShaper offers the following policy and partition types:

Type	Description
Priority Policy	Establishes a priority for traffic without specifying a particular rate. Use priority policies for non-IP traffic types, or traffic that does not burst (for example, Telnet).
Rate Policy	Smooths bursty traffic, such as HTTP, using TCP Rate Control. Keeps greedy traffic sessions in line or protects latency-sensitive sessions. Delivers an optional minimum rate for each individual session of traffic (such as 24 Kbps for an important streaming application), allows that session prioritized access to excess bandwidth, and sets an optional limit on the total bandwidth it can use.
Discard Policy	Discards all packets for a traffic class, thereby blocking the service. You might use this policy type for an unsanctioned application that you would prefer to not support on your network.
Ignore Policy	Exempts a traffic class from bandwidth management.
Never-Admit Policy	Restricts non-TCP traffic and intelligently rejects web and TCP traffic. Use this policy to redirect certain web users to alternate URLs.
Static Partition	Protects or caps all the traffic in one class. You specify the size of the reserved virtual link, choose if it can exceed that size, and optionally cap its growth. Partitions function like frame relay PVCs, but with the added important benefits that they cost less and they share unused bandwidth with other traffic.
Dynamic Partition	Creates per-user subpartitions dynamically, as needed, when users initiate traffic of a given class. Use dynamic partitions in situations where per-user bandwidth equity is important.

Bandwidth-Allocation Order

PacketShaper uses policies and partitions to determine how to allocate bandwidth. When determining bandwidth allocation, PacketShaper takes into account all bandwidth demands and uses the following basic allocation scheme:

- Traffic flows that have assigned guaranteed rates are satisfied first.
- All other traffic — traffic with and without assigned policies and unclassified traffic — competes for the remaining bandwidth (called *excess bandwidth*).
- Excess bandwidth is proportionately allocated based on the priorities in priority and rate policies.
- Flows from traffic classes with partitions are given more bandwidth (or less) to satisfy partitions' minimum (or maximum) rates.

PacketShaper's *policy test* command lets you see how rates are assigned when traffic from a user with a given connection speed prompts policy enforcement.

MPLS

Multi-protocol label switching is a relatively new technology that can improve network performance for select traffic. In the typical network without MPLS, packet paths are determined in real time as routers decide each packet's appropriate next hop. However, conventional IP routing requires time and eliminates opportunity to influence packets' paths. With MPLS, you predefine explicit paths for specific types of traffic, identified by path labels put in each packet.

Because MPLS custom routing doesn't extend to the local LAN and can't differentiate each application, QoS efforts won't yield end-to-end QoS for critical applications. PacketShaper complements MPLS installations for maximum benefits as it:

- Eases the bottlenecks that form at MPLS networks' entry points
- Identifies and separates distinct applications, assigning distinct MPLS labels if requested
- Enables MPLS to give its optimum paths to the most critical traffic
- Assesses performance before and after MPLS or other changes

For more on MPLS and PacketShaper, see "*MPLS, Only Better"* on the Packeteer web site.

The PacketShaper Advantage

PacketShaper provides patented and patent-pending technology that enables you to explicitly control TCP/IP bandwidth to keep your network under your control. This technology offers many advantages, including:

- Explicit bits-per-second rate control
- Minimum and maximum per-flow rates
- Relative priorities
- Minimum and maximum per-application, per-user, per-group rates
- Smoothed traffic flow
- Optimized throughput
- Evenly paced transmissions
- Consistent predictable response times
- Precise traffic classification
- Bi-directional traffic control

Comparison Table: Queues and TCP Rate Control		
	Queuing	**TCP Rate Control**
Efficiency	• Discards packets • Induces packet loss (tail-end drops) • Generates retransmissions (timeouts)	• Doesn't form queues • Transfers data more efficiently (more throughput, less time) • Reduces packet loss and retransmissions
Precision	• Limited traffic classification • No bits-per-second control • No per-session or per-user control	• Flexible, application-layer, extensive traffic classification • Explicit bits-per-second control • Rate-based QoS for individual application, sessions, users, and more
Full Duplex	• Outbound control, but no inbound control	• Inbound and outbound control
Proactive	• Reactive • Congestion already occurred if queues form	• Proactive • Prevents congestion *before* it occurs

For more information on individual features and examples, see "*PacketShaper Features,"* and for a more narrative PacketShaper explanation, see "*Four Steps to Application Performance."* Both papers and much more information are available on the Packeteer web site at www.packeteer.com.