Network Working Group   B. Braden, USC/ISI
Request for Comments: 2309   D. Clark, MIT LCS
Category: Informational   J. Crowcroft, UCL
                                                     B. Davie, Cisco Systems
                                                   S. Deering, Cisco Systems
                                                             D. Estrin, USC
                                                            S. Floyd, LBNL
                                                         V. Jacobson, LBNL
                                                     G. Minshall, Fiberlane
                                                        C. Partridge, BBN
                                            L. Peterson, University of Arizona
                                           K. Ramakrishnan, ATT Labs Research
                                                      S. Shenker, Xerox PARC
                                                    J. Wroclawski, MIT LCS
                                                           L. Zhang, UCLA
                                                             April 1998

          Recommendations on Queue Management and Congestion Avoidance
                              in the Internet

Status of Memo

Copyright Notice

Abstract

     This memo presents two recommendations to the Internet community
     concerning measures to improve and preserve Internet performance.
     It presents a strong recommendation for testing, standardization,
     and widespread deployment of active queue management in routers,
     to improve the performance of today's Internet. It also urges a
     concerted effort of research, measurement, and ultimate deployment
     of router mechanisms to protect the Internet from flows that are
     not sufficiently responsive to congestion notification.

1. INTRODUCTION

   The Internet protocol architecture is based on a connectionless end-
   to-end packet service using the IP protocol. The advantages of its
   connectionless design, flexibility and robustness, have been amply
   demonstrated. However, these advantages are not without cost:
   careful design is required to provide good service under heavy load.
   In fact, lack of attention to the dynamics of packet forwarding can
   result in severe service degradation or "Internet meltdown". This
   phenomenon was first observed during the early growth phase of the
   Internet of the mid 1980s [Nagle84], and is technically called
   "congestion collapse".

   The original fix for Internet meltdown was provided by Van Jacobson.
   Beginning in 1986, Jacobson developed the congestion avoidance
   mechanisms that are now required in TCP implementations [Jacobson88,
   HostReq89]. These mechanisms operate in the hosts to cause TCP
   connections to "back off" during congestion. We say that TCP flows
   are "responsive" to congestion signals (i.e., dropped packets) from
   the network. It is primarily these TCP congestion avoidance
   algorithms that prevent the congestion collapse of today's Internet.

   However, that is not the end of the story. Considerable research has
   been done on Internet dynamics since 1988, and the Internet has
   grown. It has become clear that the TCP congestion avoidance
   mechanisms [RFC2001], while necessary and powerful, are not
   sufficient to provide good service in all circumstances. Basically,
   there is a limit to how much control can be accomplished from the
   edges of the network. Some mechanisms are needed in the routers to
   complement the endpoint congestion avoidance mechanisms.

   It is useful to distinguish between two classes of router algorithms
   related to congestion control: "queue management" versus "scheduling"
   algorithms. To a rough approximation, queue management algorithms
   manage the length of packet queues by dropping packets when necessary
   or appropriate, while scheduling algorithms determine which packet to
   send next and are used primarily to manage the allocation of
   bandwidth among flows. While these two router mechanisms are closely
   related, they address rather different performance issues.

   This memo highlights two router performance issues. The first issue
   is the need for an advanced form of router queue management that we
   call "active queue management." Section 2 summarizes the benefits
   that active queue management can bring. Section 3 describes a
   recommended active queue management mechanism, called Random Early
   Detection or "RED". We expect that the RED algorithm can be used
   with a wide variety of scheduling algorithms, can be implemented
   relatively efficiently, and will provide significant Internet

performance improvement.

The second issue, discussed in Section 4 of this memo, is the
potential for future congestion collapse of the Internet due to flows
that are unresponsive, or not sufficiently responsive, to congestion
indications. Unfortunately, there is no consensus solution to
controlling congestion caused by such aggressive flows; significant
research and engineering will be required before any solution will be
available. It is imperative that this work be energetically pursued,
to ensure the future stability of the Internet.

Section 5 concludes the memo with a set of recommendations to the
IETF concerning these topics.

The discussion in this memo applies to "best-effort" traffic. The
Internet integrated services architecture, which provides a mechanism
for protecting individual flows from congestion, introduces its own
queue management and scheduling algorithms [Shenker96, Wroclawski96].
Similarly, the discussion of queue management and congestion control
requirements for differential services is a separate issue. However,
we do not expect the deployment of integrated services and
differential services to significantly diminish the importance of the
best-effort traffic issues discussed in this memo.

Preparation of this memo resulted from past discussions of end-to-end
performance, Internet congestion, and RED in the End-to-End Research
Group of the Internet Research Task Force (IRTF).

2. THE NEED FOR ACTIVE QUEUE MANAGEMENT

The traditional technique for managing router queue lengths is to set
a maximum length (in terms of packets) for each queue, accept packets
for the queue until the maximum length is reached, then reject (drop)
subsequent incoming packets until the queue decreases because a
packet from the queue has been transmitted. This technique is known
as "tail drop", since the packet that arrived most recently (i.e.,
the one on the tail of the queue) is dropped when the queue is full.
This method has served the Internet well for years, but it has two
important drawbacks.

1.   Lock-Out

     In some situations tail drop allows a single connection or a few
     flows to monopolize queue space, preventing other connections
     from getting room in the queue. This "lock-out" phenomenon is
     often the result of synchronization or other timing effects.

2.   Full Queues

     The tail drop discipline allows queues to maintain a full (or,
     almost full) status for long periods of time, since tail drop
     signals congestion (via a packet drop) only when the queue has
     become full. It is important to reduce the steady-state queue
     size, and this is perhaps queue management's most important
     goal.

     The naive assumption might be that there is a simple tradeoff
     between delay and throughput, and that the recommendation that
     queues be maintained in a "non-full" state essentially
     translates to a recommendation that low end-to-end delay is more
     important than high throughput. However, this does not take
     into account the critical role that packet bursts play in
     Internet performance. Even though TCP constrains a flow's
     window size, packets often arrive at routers in bursts
     [Leland94]. If the queue is full or almost full, an arriving
     burst will cause multiple packets to be dropped. This can
     result in a global synchronization of flows throttling back,
     followed by a sustained period of lowered link utilization,
     reducing overall throughput.

     The point of buffering in the network is to absorb data bursts
     and to transmit them during the (hopefully) ensuing bursts of
     silence. This is essential to permit the transmission of bursty
     data. It should be clear why we would like to have normally-
     small queues in routers: we want to have queue capacity to
     absorb the bursts. The counter-intuitive result is that
     maintaining normally-small queues can result in higher
     throughput as well as lower end-to-end delay. In short, queue
     limits should not reflect the steady state queues we want
     maintained in the network; instead, they should reflect the size
     of bursts we need to absorb.

  Besides tail drop, two alternative queue disciplines that can be
  applied when the queue becomes full are "random drop on full" or
  "drop front on full". Under the random drop on full discipline, a
  router drops a randomly selected packet from the queue (which can be
  an expensive operation, since it naively requires an O(N) walk
  through the packet queue) when the queue is full and a new packet
  arrives. Under the "drop front on full" discipline [Lakshman96], the
  router drops the packet at the front of the queue when the queue is
  full and a new packet arrives. Both of these solve the lock-out
  problem, but neither solves the full-queues problem described above.

We know in general how to solve the full-queues problem for
"responsive" flows, i.e., those flows that throttle back in response
to congestion notification. In the current Internet, dropped packets
serve as a critical mechanism of congestion notification to end
nodes. The solution to the full-queues problem is for routers to
drop packets before a queue becomes full, so that end nodes can
respond to congestion before buffers overflow. We call such a
proactive approach "active queue management". By dropping packets
before buffers overflow, active queue management allows routers to
control when and how many packets to drop. The next section
introduces RED, an active queue management mechanism that solves both
problems listed above (given responsive flows).

In summary, an active queue management mechanism can provide the
following advantages for responsive flows.

1.   Reduce number of packets dropped in routers

     Packet bursts are an unavoidable aspect of packet networks
     [Willinger95]. If all the queue space in a router is already
     committed to "steady state" traffic or if the buffer space is
     inadequate, then the router will have no ability to buffer
     bursts. By keeping the average queue size small, active queue
     management will provide greater capacity to absorb naturally-
     occurring bursts without dropping packets.

     Furthermore, without active queue management, more packets will
     be dropped when a queue does overflow. This is undesirable for
     several reasons. First, with a shared queue and the tail drop
     discipline, an unnecessary global synchronization of flows
     cutting back can result in lowered average link utilization, and
     hence lowered network throughput. Second, TCP recovers with
     more difficulty from a burst of packet drops than from a single
     packet drop. Third, unnecessary packet drops represent a
     possible waste of bandwidth on the way to the drop point.

     We note that while RED can manage queue lengths and reduce end-
     to-end latency even in the absence of end-to-end congestion
     control, RED will be able to reduce packet dropping only in an
     environment that continues to be dominated by end-to-end
     congestion control.

2.   Provide lower-delay interactive service

     By keeping the average queue size small, queue management will
     reduce the delays seen by flows. This is particularly important
     for interactive applications such as short Web transfers, Telnet
     traffic, or interactive audio-video sessions, whose subjective

(and objective) performance is better when the end-to-end delay
is low.

3.   Avoid lock-out behavior

Active queue management can prevent lock-out behavior by
ensuring that there will almost always be a buffer available for
an incoming packet. For the same reason, active queue
management can prevent a router bias against low bandwidth but
highly bursty flows.

It is clear that lock-out is undesirable because it constitutes
a gross unfairness among groups of flows. However, we stop
short of calling this benefit "increased fairness", because
general fairness among flows requires per-flow state, which is
not provided by queue management. For example, in a router
using queue management but only FIFO scheduling, two TCP flows
may receive very different bandwidths simply because they have
different round-trip times [Floyd91], and a flow that does not
use congestion control may receive more bandwidth than a flow
that does. Per-flow state to achieve general fairness might be
maintained by a per-flow scheduling algorithm such as Fair
Queueing (FQ) [Demers90], or a class-based scheduling algorithm
such as CBQ [Floyd95], for example.

On the other hand, active queue management is needed even for
routers that use per-flow scheduling algorithms such as FQ or
class-based scheduling algorithms such as CBQ. This is because
per-flow scheduling algorithms by themselves do nothing to
control the overall queue size or the size of individual queues.
Active queue management is needed to control the overall average
queue sizes, so that arriving bursts can be accommodated without
dropping packets. In addition, active queue management should
be used to control the queue size for each individual flow or
class, so that they do not experience unnecessarily high delays.
Therefore, active queue management should be applied across the
classes or flows as well as within each class or flow.

In short, scheduling algorithms and queue management should be
seen as complementary, not as replacements for each other. In
particular, there have been implementations of queue management
added to FQ, and work is in progress to add RED queue management
to CBQ.

3. THE QUEUE MANAGEMENT ALGORITHM "RED"

   Random Early Detection, or RED, is an active queue management
   algorithm for routers that will provide the Internet performance
   advantages cited in the previous section [RED93]. In contrast to
   traditional queue management algorithms, which drop packets only when
   the buffer is full, the RED algorithm drops arriving packets
   probabilistically. The probability of drop increases as the
   estimated average queue size grows. Note that RED responds to a
   time-averaged queue length, not an instantaneous one. Thus, if the
   queue has been mostly empty in the "recent past", RED won't tend to
   drop packets (unless the queue overflows, of course!). On the other
   hand, if the queue has recently been relatively full, indicating
   persistent congestion, newly arriving packets are more likely to be
   dropped.

   The RED algorithm itself consists of two main parts: estimation of
   the average queue size and the decision of whether or not to drop an
   incoming packet.

   (a) Estimation of Average Queue Size

       RED estimates the average queue size, either in the forwarding
       path using a simple exponentially weighted moving average (such
       as presented in Appendix A of [Jacobson88]), or in the
       background (i.e., not in the forwarding path) using a similar
       mechanism.

          Note: The queue size can be measured either in units of
          packets or of bytes. This issue is discussed briefly in
          [RED93] in the "Future Work" section.

          Note: when the average queue size is computed in the
          forwarding path, there is a special case when a packet
          arrives and the queue is empty. In this case, the
          computation of the average queue size must take into account
          how much time has passed since the queue went empty. This is
          discussed further in [RED93].

   (b) Packet Drop Decision

       In the second portion of the algorithm, RED decides whether or
       not to drop an incoming packet. It is RED's particular
       algorithm for dropping that results in performance improvement
       for responsive flows. Two RED parameters, minth (minimum
       threshold) and maxth (maximum threshold), figure prominently in

this decision process. Minth specifies the average queue size
below which no packets will be dropped, while maxth specifies
the average queue size *above which* all packets will be
dropped. As the average queue size varies from minth to maxth,
packets will be dropped with a probability that varies linearly
from 0 to maxp.

> Note: a simplistic method of implementing this would be to
> calculate a new random number at each packet arrival, then
> compare that number with the above probability which varies
> from 0 to maxp. A more efficient implementation, described
> in [RED93], computes a random number *once* for each dropped
> packet.

> Note: the decision whether or not to drop an incoming packet
> can be made in "packet mode", ignoring packet sizes, or in
> "byte mode", taking into account the size of the incoming
> packet. The performance implications of the choice between
> packet mode or byte mode is discussed further in [Floyd97].

RED effectively controls the average queue size while still
accommodating bursts of packets without loss. RED's use of
randomness breaks up synchronized processes that lead to lock-out
phenomena.

There have been several implementations of RED in routers, and papers
have been published reporting on experience with these
implementations ([Villamizar94], [Gaynor96]). Additional reports of
implementation experience would be welcome, and will be posted on the
RED web page [REDWWW].

All available empirical evidence shows that the deployment of active
queue management mechanisms in the Internet would have substantial
performance benefits. There are seemingly no disadvantages to using
the RED algorithm, and numerous advantages. Consequently, we believe
that the RED active queue management algorithm should be widely
deployed.

We should note that there are some extreme scenarios for which RED
will not be a cure, although it won't hurt and may still help. An
example of such a scenario would be a very large number of flows,
each so tiny that its fair share would be less than a single packet
per RTT.

4. MANAGING AGGRESSIVE FLOWS

   One of the keys to the success of the Internet has been the
   congestion avoidance mechanisms of TCP. Because TCP "backs off"
   during congestion, a large number of TCP connections can share a
   single, congested link in such a way that bandwidth is shared
   reasonably equitably among similarly situated flows. The equitable
   sharing of bandwidth among flows depends on the fact that all flows
   are running basically the same congestion avoidance algorithms,
   conformant with the current TCP specification [HostReq89].

   We introduce the term "TCP-compatible" for a flow that behaves under
   congestion like a flow produced by a conformant TCP. A TCP-
   compatible flow is responsive to congestion notification, and in
   steady-state it uses no more bandwidth than a conformant TCP running
   under comparable conditions (drop rate, RTT, MTU, etc.)

   It is convenient to divide flows into three classes: (1) TCP-
   compatible flows, (2) unresponsive flows, i.e., flows that do not
   slow down when congestion occurs, and (3) flows that are responsive
   but are not TCP-compatible. The last two classes contain more
   aggressive flows that pose significant threats to Internet
   performance, as we will now discuss.

   o   Non-Responsive Flows

        There is a growing set of UDP-based applications whose
        congestion avoidance algorithms are inadequate or nonexistent
        (i.e, the flow does not throttle back upon receipt of congestion
        notification). Such UDP applications include streaming
        applications like packet voice and video, and also multicast
        bulk data transport [SRM96]. If no action is taken, such
        unresponsive flows could lead to a new congestion collapse.

        In general, all UDP-based streaming applications should
        incorporate effective congestion avoidance mechanisms. For
        example, recent research has shown the possibility of
        incorporating congestion avoidance mechanisms such as Receiver-
        driven Layered Multicast (RLM) within UDP-based streaming
        applications such as packet video [McCanne96; Bolot94]. Further
        research and development on ways to accomplish congestion
        avoidance for streaming applications will be very important.

        However, it will also be important for the network to be able to
        protect itself against unresponsive flows, and mechanisms to
        accomplish this must be developed and deployed. Deployment of
        such mechanisms would provide incentive for every streaming
        application to become responsive by incorporating its own

congestion control.

o   Non-TCP-Compatible Transport Protocols

The second threat is posed by transport protocol implementations
that are responsive to congestion notification but, either
deliberately or through faulty implementations, are not TCP-
compatible. Such applications can grab an unfair share of the
network bandwidth.

For example, the popularity of the Internet has caused a
proliferation in the number of TCP implementations. Some of
these may fail to implement the TCP congestion avoidance
mechanisms correctly because of poor implementation. Others may
deliberately be implemented with congestion avoidance algorithms
that are more aggressive in their use of bandwidth than other
TCP implementations; this would allow a vendor to claim to have
a "faster TCP". The logical consequence of such implementations
would be a spiral of increasingly aggressive TCP
implementations, leading back to the point where there is
effectively no congestion avoidance and the Internet is
chronically congested.

Note that there is a well-known way to achieve more aggressive
TCP performance without even changing TCP: open multiple
connections to the same place, as has been done in some Web
browsers.

The projected increase in more aggressive flows of both these
classes, as a fraction of total Internet traffic, clearly poses a
threat to the future Internet. There is an urgent need for
measurements of current conditions and for further research into the
various ways of managing such flows. There are many difficult issues
in identifying and isolating unresponsive or non-TCP-compatible flows
at an acceptable router overhead cost. Finally, there is little
measurement or simulation evidence available about the rate at which
these threats are likely to be realized, or about the expected
benefit of router algorithms for managing such flows.

There is an issue about the appropriate granularity of a "flow".
There are a few "natural" answers: 1) a TCP or UDP connection (source
address/port, destination address/port); 2) a source/destination host
pair; 3) a given source host or a given destination host. We would
guess that the source/destination host pair gives the most
appropriate granularity in many circumstances. However, it is
possible that different vendors/providers could set different
granularities for defining a flow (as a way of "distinguishing"
themselves from one another), or that different granularities could

be chosen for different places in the network. It may be the case
that the granularity is less important than the fact that we are
dealing with more unresponsive flows at *some* granularity. The
granularity of flows for congestion management is, at least in part,
a policy question that needs to be addressed in the wider IETF
community.

5. CONCLUSIONS AND RECOMMENDATIONS

   This discussion leads us to make the following recommendations to the
   IETF and to the Internet community as a whole.

   o   RECOMMENDATION 1:

        Internet routers should implement some active queue management
        mechanism to manage queue lengths, reduce end-to-end latency,
        reduce packet dropping, and avoid lock-out phenomena within the
        Internet.

        The default mechanism for managing queue lengths to meet these
        goals in FIFO queues is Random Early Detection (RED) [RED93].
        Unless a developer has reasons to provide another equivalent
        mechanism, we recommend that RED be used.

   o   RECOMMENDATION 2:

        It is urgent to begin or continue research, engineering, and
        measurement efforts contributing to the design of mechanisms to
        deal with flows that are unresponsive to congestion notification
        or are responsive but more aggressive than TCP.

   Although there has already been some limited deployment of RED in the
   Internet, we may expect that widespread implementation and deployment
   of RED in accordance with Recommendation 1 will expose a number of
   engineering issues. For example, such issues may include:
   implementation questions for Gigabit routers, the use of RED in layer
   2 switches, and the possible use of additional considerations, such
   as priority, in deciding which packets to drop.

   We again emphasize that the widespread implementation and deployment
   of RED would not, in and of itself, achieve the goals of
   Recommendation 2.

   Widespread implementation and deployment of RED will also enable the
   introduction of other new functionality into the Internet. One
   example of an enabled functionality would be the addition of explicit
   congestion notification [Ramakrishnan97] to the Internet
   architecture, as a mechanism for congestion notification in addition

to packet drops. A second example of new functionality would be
implementation of queues with packets of different drop priorities;
packets would be transmitted in the order in which they arrived, but
during times of congestion packets of the lower drop priority would
be preferentially dropped.

6. References

[Bolot94] Bolot, J.-C., Turletti, T., and Wakeman, I., Scalable
Feedback Control for Multicast Video Distribution in the Internet,
ACM SIGCOMM '94, Sept. 1994.

[Demers90] Demers, A., Keshav, S., and Shenker, S., Analysis and
Simulation of a Fair Queueing Algorithm, Internetworking: Research
and Experience, Vol. 1, 1990, pp. 3-26.

[Floyd91] Floyd, S., Connections with Multiple Congested Gateways in
Packet-Switched Networks Part 1: One-way Traffic. Computer
Communications Review, Vol.21, No.5, October 1991, pp. 30-47. URL
http://ftp.ee.lbl.gov/floyd/.

[Floyd95] Floyd, S., and Jacobson, V., Link-sharing and Resource
Management Models for Packet Networks. IEEE/ACM Transactions on
Networking, Vol. 3 No. 4, pp. 365-386, August 1995.

[Floyd97] Floyd, S., RED: Discussions of Byte and Packet Modes, March
1997 email, http://www-nrg.ee.lbl.gov/floyd/REDaveraging.txt.

[Gaynor96] Gaynor, M., Proactive Packet Dropping Methods for TCP
Gateways, October 1996, URL http://www.eecs.harvard.edu/~gaynor/
final.ps.

[HostReq89] Braden, R., Ed., "Requirements for Internet Hosts --
Communication Layers", STD 3, RFC 1122, October 1989.

[Jacobson88] V. Jacobson, Congestion Avoidance and Control, ACM
SIGCOMM '88, August 1988.

[Lakshman96] T. V. Lakshman, Arnie Neidhardt, Teunis Ott, The Drop
From Front Strategy in TCP Over ATM and Its Interworking with Other
Control Features, Infocom 96, MA28.1.

[Leland94] W. Leland, M. Taqqu, W. Willinger, and D. Wilson, On the
Self-Similar Nature of Ethernet Traffic (Extended Version), IEEE/ACM
Transactions on Networking, 2(1), pp. 1-15, February 1994.

[McCanne96] McCanne, S., Jacobson, V., and M. Vetterli, Receiver-driven Layered Multicast, ACM SIGCOMM

[Nagle84] Nagle, J., "Congestion Control in IP/TCP", RFC 896, January 1984.

[Ramakrishnan97] Ramakrishnan, K. K., and S. Floyd, "A Proposal to add Explicit Congestion Notification (ECN) to IPv6 and to TCP", Work in Progress.

[RED93] Floyd, S., and Jacobson, V., Random Early Detection gateways for Congestion Avoidance, IEEE/ACM Transactions on Networking, V.1 N.4, August 1993, pp. 397-413. Also available from http://ftp.ee.lbl.gov/floyd/red.html.

[REDWWW] Floyd, S., The RED Web Page, 1997, URL http://ftp.ee.lbl.gov/floyd/red.html.

[RFC 2001] Stevens, W., "TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms", RFC 2001, January 1997.

[Shenker96] Shenker, S., Partridge, C., and R. Guerin, "Specification of Guaranteed Quality of Service", Work in Progress.

[SRM96] Floyd. S., Jacobson, V., McCanne, S., Liu, C., and L. Zhang, A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing. ACM SIGCOMM '96, pp 342-355.

[Villamizar94] Villamizar, C., and Song, C., High Performance TCP in ANSNET. Computer Communications Review, V. 24 N. 5, October 1994, pp. 45-60. URL http://ftp.ans.net/pub/papers/tcp-performance.ps.

[Willinger95] W. Willinger, M. S. Taqqu, R. Sherman, D. V. Wilson, Self-Similarity Through High-Variability: Statistical Analysis of Ethernet LAN Traffic at the Source Level, ACM SIGCOMM '95, pp. 100-113, August 1995.

[Wroclawski96] Wroclawski, J., "Specification of the Controlled-Load Network Element Service", Work in Progress.

Security Considerations

   While security is a very important issue, it is largely orthogonal to
   the performance issues discussed in this memo. We note, however,
   that denial-of-service attacks may create unresponsive traffic flows
   that are indistinguishable from flows from normal high-bandwidth
   isochronous applications, and the mechanism suggested in
   Recommendation 2 will be equally applicable to such attacks.

Authors' Addresses

   Bob Braden
   USC Information Sciences Institute
   4676 Admiralty Way
   Marina del Rey, CA 90292

   Phone: 310-822-1511
   EMail: Braden@ISI.EDU

   David D. Clark
   MIT Laboratory for Computer Science
   545 Technology Sq.
   Cambridge, MA 02139

   Phone: 617-253-6003
   EMail: DDC@lcs.mit.edu

   Jon Crowcroft
   University College London
   Department of Computer Science
   Gower Street
   London, WC1E 6BT
   ENGLAND

   Phone: +44 171 380 7296
   EMail: Jon.Crowcroft@cs.ucl.ac.uk

   Bruce Davie
   Cisco Systems, Inc.
   250 Apollo Drive
   Chelmsford, MA 01824

   Phone:
   EMail: bdavie@cisco.com

Steve Deering
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706

Phone: 408-527-8213
EMail: deering@cisco.com

Deborah Estrin
USC Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA 90292

Phone: 310-822-1511
EMail: Estrin@usc.edu

Sally Floyd
Lawrence Berkeley National Laboratory,
MS 50B-2239,
One Cyclotron Road,
Berkeley CA 94720

Phone: 510-486-7518
EMail: Floyd@ee.lbl.gov

Van Jacobson
Lawrence Berkeley National Laboratory,
MS 46A,
One Cyclotron Road,
Berkeley CA 94720

Phone: 510-486-7519
EMail: Van@ee.lbl.gov

Greg Minshall
Fiberlane Communications
1399 Charleston Road
Mountain View, CA 94043

Phone: +1 650 237 3164
EMail: Minshall@fiberlane.com

Craig Partridge
BBN Technologies
10 Moulton St.
Cambridge MA 02138

Phone: 510-558-8675
EMail: craig@bbn.com

Larry Peterson
Department of Computer Science
University of Arizona
Tucson, AZ 85721

Phone: 520-621-4231
EMail: LLP@cs.arizona.edu

K. K. Ramakrishnan
AT&T Labs. Research
Rm. A155
180 Park Avenue
Florham Park, N.J. 07932

Phone: 973-360-8766
EMail: KKRama@research.att.com

Scott Shenker
Xerox PARC
3333 Coyote Hill Road
Palo Alto, CA 94304

Phone: 415-812-4840
EMail: Shenker@parc.xerox.com

John Wroclawski
MIT Laboratory for Computer Science
545 Technology Sq.
Cambridge, MA 02139

Phone: 617-253-7885
EMail: JTW@lcs.mit.edu

Lixia Zhang
UCLA
4531G Boelter Hall
Los Angeles, CA 90024

Phone: 310-825-2695
EMail: Lixia@cs.ucla.edu

Full Copyright Statement