Network Working Group
Request for Comments: 3819
BCP: 89
Category: Best Current Practice

P. Karn, Ed.
Qualcomm
C. Bormann
Universitaet Bremen TZI
G. Fairhurst
University of Aberdeen
D. Grossman
Motorola, Inc.
R. Ludwig
Ericsson Research
J. Mahdavi
Novell
G. Montenegro
Sun Microsystems Laboratories, Europe
J. Touch
USC/ISI
L. Wood
Cisco Systems

July 2004

Advice for Internet Subnetwork Designers

Status of this Memo

This document specifies an Internet Best Current Practices for the Internet Community, and requests discussion and suggestions for improvements. Distribution of this memo is unlimited.

Copyright Notice

Abstract

This document provides advice to the designers of digital communication equipment, link-layer protocols, and packet-switched local networks (collectively referred to as subnetworks), who wish to support the Internet protocols but may be unfamiliar with the Internet architecture and the implications of their design choices on the performance and efficiency of the Internet.

Table of Contents

1. Introduction and Overview. 2
2. Maximum Transmission Units (MTUs) and IP Fragmentation 4
2.1. Choosing the MTU in Slow Networks. 6
3. Framing on Connection-Oriented Subnetworks 7
4. Connection-Oriented Subnetworks. 9
5. Broadcasting and Discovery 10
6. Multicasting 11
7. Bandwidth on Demand (BoD) Subnets. 13
8. Reliability and Error Control. 14
8.1. TCP vs Link-Layer Retransmission 14
8.2. Recovery from Subnetwork Outages 17
8.3. CRCs, Checksums and Error Detection. 18
8.4. How TCP Works. 20
8.5. TCP Performance Characteristics. 22
8.5.1. The Formulae 22
8.5.2. Assumptions. 23
8.5.3. Analysis of Link-Layer Effects on TCP Performance. 24
9. Quality-of-Service (QoS) Considerations. 26
10. Fairness vs Performance. 29
11. Delay Characteristics. 30
12. Bandwidth Asymmetries. 31
13. Buffering, Flow and Congestion Control 31
14. Compression. 34
15. Packet Reordering. 36
16. Mobility 37
17. Routing. 39
18. Security Considerations. 41
19. Contributors 44
20. Informative References 45
21. Contributors' Addresses. 57
22. Authors' Addresses 58
23. Full Copyright Statement 60

1. Introduction and Overview

IP, the Internet Protocol [RFC791] [RFC2460], is the core protocol of the Internet. IP defines a simple "connectionless" packet-switched network. The success of the Internet is largely attributed to IP's simplicity, the "end-to-end principle" [SRC81] on which the Internet is based, and the resulting ease of carrying IP on a wide variety of subnetworks, not necessarily designed with IP in mind. A subnetwork refers to any network operating immediately below the IP layer to connect two or more systems using IP (i.e., end hosts or routers). In its simplest form, this may be a direct connection between the IP systems (e.g., using a length of cable or a wireless medium).

This document defines a subnetwork as a layer 2 network, which is a network that does not rely upon the services of IP routers to forward packets between parts of the subnetwork. However, IP routers may bridge frames at Layer 2 between parts of a subnetwork. Sometimes, it is convenient to aggregate a group of such subnetworks into a single logical subnetwork. IP routing protocols (e.g., OSPF, IS-IS, and PIM) can be configured to support this aggregation, but typically present a layer-3 subnetwork rather than a layer-2 subnetwork. This may also result in a specific packet passing several times over the same layer-2 subnetwork via an intermediate layer-3 gateway (router). Because that aggregation requires layer-3 components, issues thereof are beyond the scope of this document.

However, while many subnetworks carry IP, they do not necessarily do so with maximum efficiency, minimum complexity, or cost, nor do they implement certain features to efficiently support newer Internet features of increasing importance, such as multicasting or quality of service.

With the explosive growth of the Internet, IP packets comprise an increasingly large fraction of the traffic carried by the world's telecommunications networks. It therefore makes sense to optimize both existing and new subnetwork technologies for IP as much as possible.

Optimizing a subnetwork for IP involves three complementary considerations:

1. Providing functionality sufficient to carry IP.

2. Eliminating unnecessary functions that increase cost or complexity.

3. Choosing subnetwork parameters that maximize the performance of the Internet protocols.

Because IP is so simple, consideration 2 is more of an issue than consideration 1. That is to say, subnetwork designers make many more errors of commission than errors of omission. However, certain enhancements to Internet features, such as multicasting and quality-of-service, benefit significantly from support given by the underlying subnetworks beyond that necessary to carry "traditional" unicast, best-effort IP.

A major consideration in the efficient design of any layered communication network is the appropriate layer(s) in which to implement a given function. This issue was first addressed in the seminal paper, "End-to-End Arguments in System Design" [SRC81]. That paper argued that many functions can be implemented properly *only* on an end-to-end basis, i.e., at the highest protocol layers, outside the subnetwork. These functions include ensuring the reliable delivery of data and the use of cryptography to provide confidentiality and message integrity.

Such functions cannot be provided solely by the concatenation of hop-by-hop services; duplicating these functions at the lower protocol layers (i.e., within the subnetwork) can be needlessly redundant or even harmful to cost and performance.

However, partial duplication of functionality in a lower layer can *sometimes* be justified by performance, security, or availability considerations. Examples include link-layer retransmission to improve the performance of an unusually lossy channel, e.g., mobile radio, link-level encryption intended to thwart traffic analysis, and redundant transmission links to improve availability, increase throughput, or to guarantee performance for certain classes of traffic. Duplication of protocol functions should be done only with an understanding of system-level implications, including possible interactions with higher-layer mechanisms.

The original architecture of the Internet was influenced by the end-to-end principle [SRC81], and has been, in our view, part of the reason for the Internet's success.

The remainder of this document discusses the various subnetwork design issues that the authors consider relevant to efficient IP support.

2. Maximum Transmission Units (MTUs) and IP Fragmentation

IPv4 packets (datagrams) vary in size, from 20 bytes (the size of the IPv4 header alone) to a maximum of 65535 bytes. Subnetworks need not support maximum-sized (64KB) IP packets, as IP provides a scheme that breaks packets that are too large for a given subnetwork into fragments that travel as independent IP packets and are reassembled at the destination. The maximum packet size supported by a subnetwork is known as its Maximum Transmission Unit (MTU).

Subnetworks may, but are not required to, indicate the length of each packet they carry. One example is Ethernet with the widely used DIX [DIX82] (not IEEE 802.3 [IEEE8023]) header, which lacks a length

field to indicate the true data length when the packet is padded to a minimum of 60 bytes. This is not a problem for uncompressed IP because each IP packet carries its own length field.

If optional header compression [RFC1144] [RFC2507] [RFC2508] [RFC3095] is used, however, it is required that the link framing indicate frame length because that is needed for the reconstruction of the original header.

In IP version 4 (the version now in widespread use), fragmentation can occur at either the sending host or in an intermediate router, and fragments can be further fragmented at subsequent routers if necessary.

In IP version 6 [RFC2460], fragmentation can occur only at the sending host; it cannot occur in a router (called "router fragmentation" in this document).

Both IPv4 and IPv6 provide a "path MTU discovery" procedure [RFC1191] [RFC1435] [RFC1981] that allows the sending host to avoid fragmentation by discovering the minimum MTU along a given path and reduce its packet sizes accordingly. This procedure is optional in IPv4 and IPv6.

Path MTU discovery is widely deployed, but it sometimes encounters problems. Some routers fail to generate the ICMP messages that convey path MTU information to the sender, and sometimes the ICMP messages are blocked by overly restrictive firewalls. The result can be a "Path MTU Black Hole" [RFC2923] [RFC1435].

The Path MTU Discovery procedure, the persistence of path MTU black holes, and the deletion of router fragmentation in IPv6 reflect a consensus of the Internet technical community that router fragmentation is best avoided. This requires that subnetworks support MTUs that are "reasonably" large. All IPv4 end hosts are required to accept and reassemble IP packets of size 576 bytes [RFC791], but such a small value would clearly be inefficient. Because IPv6 omits fragmentation by routers, [RFC2460] specifies a larger minimum MTU of 1280 bytes. Any subnetwork with an internal packet payload smaller than 1280 bytes must implement a mechanism that performs fragmentation/reassembly of IP packets to/from subnetwork frames if it is to support IPv6.

If a subnetwork cannot directly support a "reasonable" MTU with native framing mechanisms, it should internally fragment. That is, it should transparently break IP packets into internal data elements and reassemble them at the other end of the subnetwork.

This leaves the question of what is a "reasonable" MTU. Ethernet (10 and 100 Mb/s) has an MTU of 1500 bytes, and because of the ubiquity of Ethernet few Internet paths currently have MTUs larger than this value. This severely limits the utility of larger MTUs provided by other subnetworks. Meanwhile, larger MTUs are increasingly desirable on high-speed subnetworks to reduce the per-packet processing overhead in host computers, and implementers are encouraged to provide them even though they may not be usable when Ethernet is also in the path.

Various "tunneling" schemes, such as GRE [RFC2784] or IP Security in tunnel mode [RFC2406], treat IP as a subnetwork for IP. Since tunneling adds header overhead, it can trigger fragmentation, even when the same physical subnetworks (e.g., Ethernet) are used on both sides of the host performing IPsec encapsulation. Tunneling has made it more difficult to avoid router fragmentation and has increased the incidence of path MTU black holes [RFC2401] [RFC2923]. Larger subnetwork MTUs may help to alleviate this problem.

2.1. Choosing the MTU in Slow Networks

In slow networks, the largest possible packet may take a considerable amount of time to send. This is known as channelisation or serialisation delay. Total end-to-end interactive response time should not exceed the well-known human factors limit of 100 to 200 ms. This includes all sources of delay: electromagnetic propagation delay, queuing delay, serialisation delay, and the store-and-forward time, i.e., the time to transmit a packet at link speed.

At low link speeds, store-and-forward delays can dominate total end-to-end delay; these are in turn directly influenced by the maximum transmission unit (MTU) size. Even when an interactive packet is given a higher queuing priority, it may have to wait for a large bulk transfer packet to finish transmission. This worst-case wait can be set by an appropriate choice of MTU.

For example, if the MTU is set to 1500 bytes, then an MTU-sized packet will take about 8 milliseconds to send on a T1 (1.536 Mb/s) link. But if the link speed is 19.2kb/s, then the transmission time becomes 625 ms -- well above our 100-200ms limit. A 256-byte MTU would lower this delay to a little over 100 ms. However, care should be taken not to lower the MTU excessively, as this will increase header overhead and trigger frequent router fragmentation (if Path MTU discovery is not in use). This is likely to be the case with multicast, where Path MTU discovery is ineffective.

One way to limit delay for interactive traffic without imposing a small MTU is to give priority to this traffic and to preempt (abort)

the transmission of a lower-priority packet when a higher priority packet arrives in the queue. However, the link resources used to send the aborted packet are lost, and overall throughput will decrease.

Another way to limit delay is to implement a link-level multiplexing scheme that allows several packets to be in progress simultaneously, with transmission priority given to segments of higher-priority IP packets. For links using the Point-To-Point Protocol (PPP) [RFC1661], multi-class multilink [RFC2686] [RFC2687] [RFC2689] provides such a facility.

ATM (asynchronous transfer mode), where SNDUs are fragmented and interleaved across smaller 53-byte ATM cells, is another example of this technique. However, ATM is generally used on high-speed links where the store-and-forward delays are already minimal, and it introduces significant (~9%) increases in overhead due to the addition of 5-byte cell overhead to each 48-byte ATM cell.

A third example is the Data-Over-Cable Service Interface Specification (DOCSIS) with typical upstream bandwidths of 2.56 Mb/s or 5.12 Mb/s. To reduce the impact of a 1500-byte MTU in DOCSIS 1.0 [DOCSIS1], a data link layer fragmentation mechanism is specified in DOCSIS 1.1 [DOCSIS2]. To accommodate the installed base, DOCSIS 1.1 must be backward compatible with DOCSIS 1.0 cable modems, which generally do not support fragmentation. Under the co-existence of DOCSIS 1.0 and DOCSIS 1.1, the unfragmented large data packets from DOCSIS 1.0 cable modems may affect the quality of service for voice packets from DOCSIS 1.1 cable modems. In this case, it has been shown in [DOCSIS3] that the use of bandwidth allocation algorithms can mitigate this effect.

To summarize, there is a fundamental tradeoff between efficiency and latency in the design of a subnetwork, and the designer should keep this tradeoff in mind.

3. Framing on Connection-Oriented Subnetworks

IP requires that subnetworks mark the beginning and end of each variable-length, asynchronous IP packet. Some examples of links and subnetworks that do not provide this as an intrinsic feature include:

1. leased lines carrying a synchronous bit stream;

2. ISDN B-channels carrying a synchronous octet stream;

3. dialup telephone modems carrying an asynchronous octet stream;

and

4. Asynchronous Transfer Mode (ATM) networks carrying an asynchronous stream of fixed-sized "cells".

The Internet community has defined packet framing methods for all these subnetworks. The Point-To-Point Protocol (PPP) [RFC1661], which uses a variant of HDLC, is applicable to bit synchronous, octet-synchronous, and octet asynchronous links (i.e., examples 1-3 above). PPP is one preferred framing method for IP, since a large number of systems interoperate with PPP. ATM has its own framing methods, described in [RFC2684] [RFC2364].

At high speeds, a subnetwork should provide a framed interface capable of carrying asynchronous, variable-length IP datagrams. The maximum packet size supported by this interface is discussed above in the MTU/Fragmentation section. The subnetwork may implement this facility in any convenient manner.

IP packet boundaries need not coincide with any framing or synchronization mechanisms internal to the subnetwork. When the subnetwork implements variable sized data units, the most straightforward approach is to place exactly one IP packet into each subnetwork data unit (SNDU), and to rely on the subnetwork's existing ability to delimit SNDUs to also delimit IP packets. A good example is Ethernet. However, some subnetworks have SNDUs of one or more fixed sizes, as dictated by switching, forward error correction and/or interleaving considerations. Examples of such subnetworks include ATM, with a single cell payload size of 48 octets plus a 5-octet header, and IS-95 digital cellular, with two "rate sets" of four fixed frame sizes each that may be selected on 20 millisecond boundaries.

Because IP packets are of variable length, they may not necessarily fit into an integer multiple of fixed-sized SNDUs. An "adaptation layer" is needed to convert IP packets into SNDUs while marking the boundary between each IP packet in some manner.

There are several approaches to this problem. The first is to encode each IP packet into one or more SNDUs with no SNDU containing pieces of more than one IP packet, and to pad out the last SNDU of the packet as needed. Bits in a control header added to each SNDU indicate where the data segment belongs in the IP packet. If the subnetwork provides in-order, at-most-once delivery, the header can be as simple as a pair of bits indicating whether the SNDU is the first and/or the last in the IP packet. Alternatively, for subnetworks that do not reorder the fragments of an SNDU, only the last SNDU of the packet could be marked, as this would implicitly

indicate the next SNDU as the first in a new IP packet. The AAL5 (ATM Adaptation Layer 5) scheme used with ATM is an example of this approach, though it adds other features, including a payload length field and a payload CRC.

In AAL5, the ATM User-User Indication, which is encoded in the Payload Type field of an ATM cell, indicates the last cell of a packet. The packet trailer is located at the end of the SNDU and contains the packet length and a CRC.

Another framing technique is to insert per-segment overhead to indicate the presence of a segment option. When present, the option carries a pointer to the end of the packet. This differs from AAL5 in that it permits another packet to follow within the same segment. MPEG-2 Transport Streams [EN301192] [ISO13818] support this style of fragmentation, and may either use padding (limiting each MPEG transport stream packet to carry only part of one IP packet), or allow a second IP packet to start in the same Transport Stream packet (no padding).

A third approach is to insert a special flag sequence into the data stream between each IP packet, and to pack the resulting data stream into SNDUs without regard to SNDU boundaries. This may have implications when frames are lost. The flag sequence can also pad unused space at the end of an SNDU. If the special flag appears in the user data, it is escaped to an alternate sequence (usually larger than a flag) to avoid being misinterpreted as a flag. The HDLC-based framing schemes used in PPP are all examples of this approach.

All three adaptation schemes introduce overhead; how much depends on the distribution of IP packet sizes, the size(s) of the SNDUs, and in the HDLC-like approaches, the content of the IP packet (since flag-like sequences occurring in the packet must be escaped, which expands them). The designer must also weigh implementation complexity and performance in the choice and design of an adaptation layer.

4. Connection-Oriented Subnetworks

IP has no notion of a "connection"; it is a purely connectionless protocol. When a connection is required by an application, it is usually provided by TCP [RFC793], the Transmission Control Protocol, running atop IP on an end-to-end basis.

Connection-oriented subnetworks can be (and are widely) used to carry IP, but often with considerable complexity. Subnetworks consisting of few nodes can simply open a permanent connection between each pair of nodes. This is frequently done with ATM. However, the number of connections increases as the square of the number of nodes, so this

is clearly impractical for large subnetworks. A "shim" layer between IP and the subnetwork is therefore required to manage connections. This is one of the most common functions of a Subnetwork Dependent Convergence Function (SNDCF) sublayer between IP and a subnetwork.

SNDCFs typically open subnetwork connections as needed when an IP packet is queued for transmission and close them after an idle timeout. There is no relation between subnetwork connections and any connections that may exist at higher layers (e.g., TCP).

Because Internet traffic is typically bursty and transaction-oriented, it is often difficult to pick an optimal idle timeout. If the timeout is too short, subnetwork connections are opened and closed rapidly, possibly over-stressing the subnetwork connection management system (especially if it was designed for voice traffic call holding times). If the timeout is too long, subnetwork connections are idle much of the time, wasting any resources dedicated to them by the subnetwork.

Purely connectionless subnets (such as Ethernet), which have no state and dynamically share resources, are optimal for supporting best-effort IP, which is stateless and dynamically shares resources. Connection-oriented packet networks (such as ATM and Frame Relay), which have state and dynamically share resources, are less optimal, since best-effort IP does not benefit from the overhead of creating and maintaining state. Connection-oriented circuit-switched networks (including the PSTN and ISDN) have state and statically allocate resources for a call, and thus require state creation and maintenance overhead, but do not benefit from the efficiencies of statistical multiplexing sharing of capacity inherent in IP.

In any event, if an SNDCF that opens and closes subnet connections is used to support IP, care should be taken to make sure that connection processing in the subnet can keep up with relatively short holding times.

5. Broadcasting and Discovery

Subnetworks fall into two categories: point-to-point and shared. A point-to-point subnet has exactly two endpoint components (hosts or routers); a shared link has more than two endpoint components, using either an inherently broadcast medium (e.g., Ethernet, radio) or a switching layer hidden from the network layer (e.g., switched Ethernet, Myrinet [MYR95], ATM). Switched subnetworks handle broadcast by copying broadcast packets, providing each interface that supports one, or more, systems (hosts or routers) with a copy of each packet.

Several Internet protocols for IPv4 make use of broadcast capabilities, including link-layer address lookup (ARP), auto-configuration (RARP, BOOTP, DHCP), and routing (RIP).

A lack of broadcast capability can impede the performance of these protocols, or render them inoperable (e.g., DHCP). ARP-like link address lookup can be provided by a centralized database, but at the expense of potentially higher response latency and the need for nodes to have explicit knowledge of the ARP server address. Shared links should support native, link-layer subnet broadcast.

A corresponding set of IPv6 protocols uses multicasting (see next section) instead of broadcasting to provide similar functions with improved scaling in large networks.

6. Multicasting

The Internet model includes "multicasting", where IP packets are sent to all the members of a multicast group [RFC1112] [RFC3376] [RFC2710]. Multicast is an option in IPv4, but a standard feature of IPv6. IPv4 multicast is currently used by multimedia, teleconferencing, gaming, and file distribution (web, peer-to-peer sharing) applications, as well as by some key network and host protocols (e.g., RIPv2, OSPF, NTP). IPv6 additionally relies on multicast for network configuration (DHCP-like autoconfiguration) and link-layer address discovery [RFC2461] (replacing ARP). In the case of IPv6, this can allow autoconfiguration and address discovery to span across routers, whereas the IPv4 broadcast-based services cannot without ad-hoc router support [RFC1812].

Multicast-enabled IP routers organize each multicast group into a spanning tree, and route multicast packets by making copies of each multicast packet and forwarding the copies to each output interface that includes at least one downstream member of the multicast group.

Multicasting is considerably more efficient when a subnetwork explicitly supports it. For example, a router relaying a multicast packet onto an Ethernet segment need send only one copy of the packet, no matter how many members of the multicast group are connected to the segment. Without native multicast support, routers and switches on shared links would need to use broadcast with software filters, such that every multicast packet sent incurs software overhead for every node on the subnetwork, even if a node is not a member of the multicast group. Alternately, the router would transmit a separate copy to every member of the multicast group on the segment, as is done on multicast-incapable switched subnets.

Subnetworks using shared channels (e.g., radio LANs, Ethernets) are especially suitable for native multicasting, and their designers should make every effort to support it. This involves designating a section of the subnetwork's own address space for multicasting. On these networks, multicast is basically broadcast on the medium, with Layer-2 receiver filters.

Subnet interfaces also need to be designed to accept packets addressed to some number of multicast addresses, in addition to the unicast packets specifically addressed to them. The number of multicast addresses that needs to be supported by a host depends on the requirements of the associated host; at least several dozen will meet most current needs.

On low-speed networks, the multicast address recognition function may be readily implemented in host software, but on high-speed networks, it should be implemented in subnetwork hardware. This hardware need not be complete; for example, many Ethernet interfaces implement a "hashing" function where the IP layer receives all of the multicast (and unicast) traffic to which the associated host subscribes, plus some small fraction of multicast traffic to which the host does not subscribe. Host/router software then has to discard the unwanted packets that pass the Layer-2 multicast address filter [RFC1112].

There does not need to be a one-to-one mapping between a Layer-2 multicast address and an IP multicast address. An address overlap may significantly degrade the filtering capability of a receiver's hardware multicast address filter. A subnetwork supporting only broadcast should use this service for multicast and must rely on software filtering.

Switched subnetworks must also provide a mechanism for copying multicast packets to ensure the packets reach at least all members of a multicast group. One option is to "flood" multicast packets in the same manner as broadcast. This can lead to unnecessary transmissions on some subnetwork links (notably non-multicast-aware Ethernet switches). Some subnetworks therefore allow multicast filter tables to control which links receive packets belonging to a specific group. To configure this automatically requires access to Layer-3 group membership information (e.g., IGMP [RFC3376], or MLD [RFC2710]). Various implementation options currently exist to provide a subnet node with a list of mappings of multicast addresses to ports/interfaces. These employ a range of approaches, including signaling from end hosts (e.g., IEEE 802 GARP/GMRP [802.1p]), signaling from switches (e.g., CGMP [CGMP] and RGMP [RFC3488]), interception and proxy of IP group membership packets (e.g., IGMP/MLD Proxy [MAGMA-PROXY]), and enabling Layer-2 devices to snoop/inspect/peek into forwarded Layer-3 protocol headers (e.g.,

IGMP, MLD, PIM) so that they may infer Layer-3 multicast group membership [MAGMA-SNOOP]. These approaches differ in their complexity, flexibility, and ability to support new protocols.

7. Bandwidth on Demand (BoD) Subnets

Some subnets allow a number of subnet nodes to share a channel efficiently by assigning transmission opportunities dynamically. Transmission opportunities are requested by a subnet node when it has packets to send. The subnet schedules and grants transmission opportunities sufficient to allow the transmitting subnet node to send one or more packets (or packet fragments). We call these subnets Bandwidth on Demand (BoD) subnets. Examples of BoD subnets include Demand Assignment Multiple Access (DAMA) satellite and terrestrial wireless networks, IEEE 802.11 point coordination function (PCF) mode, and DOCSIS. A connection-oriented network (such as the PSTN, ATM or Frame Relay) reserves resources on a much longer timescale, and is therefore not a BoD subnet in our taxonomy.

The design parameters for BoD are similar to those in connection-oriented subnetworks, although the implementations may vary significantly. In BoD, the user typically requests access to the shared channel for some duration. Access may be allocated for a period of time at a specific rate, for a certain number of packets, or until the user releases the channel. Access may be coordinated through a central management entity or with a distributed algorithm amongst the users. Examples of the resource that may be shared include a terrestrial wireless hop, an upstream channel in a cable television system, a satellite uplink, and an end-to-end satellite channel.

Long-delay BoD subnets pose problems similar to connection-oriented subnets in anticipating traffic. While connection-oriented subnets hold idle channels open expecting new data to arrive, BoD subnets request channel access based on buffer occupancy (or expected buffer occupancy) on the sending port. Poor performance will likely result if the sender does not anticipate additional traffic arriving at that port during the time it takes to grant a transmission request. It is recommended that the algorithm have the capability to extend a hold on the channel for data that has arrived after the original request was generated (this may be done by piggybacking new requests on user data).

There is a wide variety of BoD protocols available. However, there has been relatively little comprehensive research on the interactions between BoD mechanisms and Internet protocol performance. Research on some specific mechanisms is available (e.g., [AR02]). One item that has been studied is TCP's retransmission timer [KY02]. BoD

systems can cause spurious timeouts when adjusting from a relatively high data rate, to a relatively low data rate. In this case, TCP's transmitted data takes longer to get through the network than predicted by the TCP sender's computed retransmission timeout. Therefore, the TCP sender is prone to resending a segment prematurely.

8. Reliability and Error Control

In the Internet architecture, the ultimate responsibility for error recovery is at the end points [SRC81]. The Internet may occasionally drop, corrupt, duplicate, or reorder packets, and the transport protocol (e.g., TCP) or application (e.g., if UDP is used as the transport protocol) must recover from these errors on an end-to-end basis [RFC3155]. Error recovery in the subnetwork is therefore justifiable only to the extent that it can enhance overall performance. It is important to recognize that a subnetwork can go too far in attempting to provide error recovery services in the Internet environment. Subnet reliability should be "lightweight", i.e., it only has to be "good enough", *not* perfect.

In this section, we discuss how to analyze characteristics of a subnetwork to determine what is "good enough". The discussion below focuses on TCP, which is the most widely-used transport protocol in the Internet. It is widely believed (and is a stated goal within the IETF) that non-TCP transport protocols should attempt to be "TCP-friendly" and have many of the same performance characteristics. Thus, the discussion below should be applicable, even to portions of the Internet where TCP may not be the predominant protocol.

8.1. TCP vs Link-Layer Retransmission

Error recovery involves the generation and transmission of redundant information computed from user data. Depending on how much redundant information is sent and how it is generated, the receiver can use it to reliably detect transmission errors, correct up to some maximum number of transmission errors, or both. The general approach is known as Error Control Coding, or ECC.

The use of ECC to detect transmission errors so that retransmissions (hopefully without errors) can be requested is widely known as "ARQ" (Automatic Repeat Request).

When enough ECC information is available to permit the receiver to correct some transmission errors without a retransmission, the approach is known as Forward Error Correction (FEC). Due to the greater complexity of the required ECC and the need to tailor its design to the characteristics of a specific modem and channel, FEC

has traditionally been implemented in special-purpose hardware integral to a modem. This effectively makes it part of the physical layer.

Unlike ARQ, FEC was rarely used for telecommunications outside of space links prior to the 1990s. It is now nearly universal in telephone, cable and DSL modems, digital satellite links, and digital mobile telephones. FEC is also heavily used in optical and magnetic storage where "retransmissions" are not possible.

Some systems use hybrid combinations of ARQ layered atop FEC; V.90 dialup modems (in the upstream direction) with V.42 error control are one example. Most errors are corrected by the trellis (FEC) code within the V.90 modem, and most remaining errors are detected and corrected by the ARQ mechanisms in V.42.

Work is now underway to apply FEC above the physical layer, primarily in connection with reliable multicasting [RFC3048] [RFC3450-RFC3453] where conventional ARQ mechanisms are inefficient or difficult to implement. However, in this discussion, we will assume that if FEC is present, it is implemented within the physical layer.

Depending on the layer in which it is implemented, error control can operate on an end-to-end basis or over a shorter span, such as a single link. TCP is the most important example of an end-to-end protocol that uses an ARQ strategy.

Many link-layer protocols use ARQ, usually some flavor of HDLC [ISO3309]. Examples include the X.25 link layer, the AX.25 protocol used in amateur packet radio, 802.11 wireless LANs, and the reliable link layer specified in IEEE 802.2.

Only end-to-end error recovery can ensure reliable service to the application (see Section 8). However, some subnetworks (e.g., many wireless links) also have link-layer error recovery as a performance enhancement [RFC3366]. For example, many cellular links have small physical frame sizes (< 100 bytes) and relatively high frame loss rates. Relying solely on end-to-end error recovery can clearly yield a performance degradation, as retransmissions across the end-to-end path take much longer to be received than when link layer retransmissions are used. Thus, link-layer error recovery can often increase end-to-end performance. As a result, link-layer and end-to-end recovery often co-exist; this can lead to the possibility of inefficient interactions between the two layers of ARQ protocols.

This inter-layer "competition" might lead to the following wasteful situation. When the link layer retransmits (parts of) a packet, the link latency momentarily increases. Since TCP bases its

retransmission timeout on prior measurements of total end-to-end latency, including that of the link in question, this sudden increase in latency may trigger an unnecessary retransmission by TCP of a packet that the link layer is still retransmitting. Such spurious end-to-end retransmissions generate unnecessary load and reduce end-to-end throughput. As a result, the link layer may even have multiple copies of the same packet in the same link queue at the same time. In general, one could say the competing error recovery is caused by an inner control loop (link-layer error recovery) reacting to the same signal as an outer control loop (end-to-end error recovery) without any coordination between the loops. Note that this is solely an efficiency issue; TCP continues to provide reliable end-to-end delivery over such links.

This raises the question of how persistent a link-layer sender should be in performing retransmission [RFC3366]. We define the link-layer (LL) ARQ persistency as the maximum time that a particular link will spend trying to transfer a packet before it can be discarded. This deliberately simplified definition says nothing about the maximum number of retransmissions, retransmission strategies, queue sizes, queuing disciplines, transmission delays, or the like. The reason we use the term LL ARQ persistency, instead of a term such as "maximum link-layer packet holding time," is that the definition closely relates to link-layer error recovery. For example, on links that implement straightforward error recovery strategies, LL ARQ persistency will often correspond to a maximum number of retransmissions permitted per link-layer frame.

For link layers that do not or cannot differentiate between flows (e.g., due to network layer encryption), the LL ARQ persistency should be small. This avoids any harmful effects or performance degradation resulting from indiscriminate high persistence. A detailed discussion of these issues is provided in [RFC3366].

However, when a link layer can identify individual flows and apply ARQ selectively [LKJK02], then the link ARQ persistency should be high for a flow using reliable unicast transport protocols (e.g., TCP) and must be low for all other flows. Setting the link ARQ persistency larger than the largest link outage allows TCP to rapidly restore transmission without needing to wait for a retransmission time out. This generally improves TCP performance in the face of transient outages. However, excessively high persistence may be disadvantageous; a practical upper limit of 30-60 seconds may be desirable. Implementation of such schemes remains a research issue. (See also the following section "Recovery from Subnetwork Outages").

Many subnetwork designers have opportunities to reduce the probability of packet loss, e.g., with FEC, ARQ, and interleaving, at the cost of increased delay. TCP performance improves with decreasing loss but worsens with increasing end-to-end delay, so it is important to find the proper balance through analysis and simulation.

8.2. Recovery from Subnetwork Outages

Some types of subnetworks, particularly mobile radio, are subject to frequent temporary outages. For example, an active cellular data user may drive or walk into an area (such as a tunnel) that is out of range of any base station. No packets will be delivered successfully until the user returns to an area with coverage.

The Internet protocols currently provide no standard way for a subnetwork to explicitly notify an upper layer protocol (e.g., TCP) that it is experiencing an outage rather than severe congestion.

Under these circumstances TCP will, after each unsuccessful retransmission, wait even longer before trying again; this is its "exponential back-off" algorithm. Furthermore, TCP will not discover that the subnetwork outage has ended until its next retransmission attempt. If TCP has backed off, this may take some time. This can lead to extremely poor TCP performance over such subnetworks.

It is therefore highly desirable that a subnetwork subject to outages does not silently discard packets during an outage. Ideally, the subnetwork should define an interface to the next higher layer (i.e., IP) that allows it to refuse packets during an outage, and to automatically ask IP for new packets when it is again able to deliver them. If it cannot do this, then the subnetwork should hold onto at least some of the packets it accepts during an outage and attempt to deliver them when the outage ends. When packets are discarded, IP should be notified so that the appropriate ICMP messages can be sent.

Note that it is *not* necessary to completely avoid dropping packets during an outage. The purpose of holding onto a packet during an outage, either in the subnetwork or at the IP layer, is so that its eventual delivery will implicitly notify TCP that the subnetwork is again operational. This is to enhance performance, not to ensure reliability -- reliability, as discussed earlier, can only be ensured on an end-to-end basis.

Only a few packets per TCP connection, including ACKs, need be held in this way to cause the TCP sender to recover from the additional losses once the flow resumes [RFC3366].

Because it would be a layering violation (and possibly a performance hit) for IP or a subnetwork layer to look at TCP headers (which would in any event be impossible if IPsec encryption [RFC2401] is in use), it would be reasonable for the IP or subnetwork layers to choose, as a design parameter, some small number of packets that will be retained during an outage.

8.3. CRCs, Checksums and Error Detection

The TCP [RFC793], UDP [RFC768], ICMP, and IPv4 [RFC791] protocols all use the same simple 16-bit 1's complement checksum algorithm [RFC1071] to detect corrupted packets. The IPv4 header checksum protects only the IPv4 header, while the TCP, ICMP, and UDP checksums provide end-to-end error detection for both the transport pseudo header (including network and transport layer information) and the transport payload data. Protection of the data is optional for applications using UDP [RFC768] for IPv4, but is required for IPv6.

The Internet checksum is not very strong from a coding theory standpoint, but it is easy to compute in software, and various proposals to replace the Internet checksums with stronger checksums have failed. However, it is known that undetected errors can and do occur in packets received by end hosts [SP2000].

To reduce processing costs, IPv6 has no IP header checksum. The destination host detects "important" errors in the IP header, such as the delivery of the packet to the wrong destination. This is done by including the IP source and destination addresses (pseudo header) in the computation of the checksum in the TCP or UDP header, a practice already performed in IPv4. Errors in other IPv6 header fields may go undetected within the network; this was considered a reasonable price to pay for a considerable reduction in the processing required by each router, and it was assumed that subnetworks would use a strong link CRC.

One way to provide additional protection for an IPv4 or IPv6 header is by the authentication and packet integrity services of the IP Security (IPsec) protocol [RFC2401]. However, this may not be a choice available to the subnetwork designer.

Most subnetworks implement error detection just above the physical layer. Packets corrupted in transmission are detected and discarded before delivery to the IP layer. A 16-bit cyclic redundancy check (CRC) is usually the minimum for error detection. This is significantly more robust against most patterns of errors than the 16-bit Internet checksum. Note that the error detection properties of a specific CRC code diminish with increasing frame size. The Point-to-Point Protocol [RFC1662] requires support of a 16-bit CRC

for each link frame, with a 32-bit CRC as an option. (PPP is often used in conjunction with a dialup modem, which provides its own error control). Other subnetworks, including 802.3/Ethernet, AAL5/ATM, FDDI, Token Ring, and PPP over SONET/SDH all use a 32-bit CRC. Many subnetworks can also use other mechanisms to enhance the error detection capability of the link CRC (e.g., FEC in dialup modems, mobile radio and satellite channels).

Any new subnetwork designed to carry IP should therefore provide error detection for each IP packet that is at least as strong as the 32-bit CRC specified in [ISO3309]. While this will achieve a very low undetected packet error rate due to transmission errors, it will not (and need not) achieve a very low packet loss rate as the Internet protocols are better suited to dealing with lost packets than to dealing with corrupted packets [SRC81].

Packet corruption may be, and is, also caused by bugs in host and router hardware and software. Even if every subnetwork implemented strong error detection, it is still essential that end-to-end checksums are used at the receiving end host [SP2000].

Designers of complex subnetworks consisting of internal links and packet switches should consider implementing error detection on an edge-to-edge basis to cover an entire SNDU (or IP packet). A CRC would be generated at the entry point to the subnetwork and checked at the exit endpoint. This may be used instead of, or in combination with, error detection at the interface to each physical link. An edge-to-edge check has the significant advantage of protecting against errors introduced anywhere within the subnetwork, not just within its transmission links. Examples of this approach include the way in which the Ethernet CRC-32 is handled by LAN bridges [802.1D]. ATM AAL5 [ITU-I363] also uses an edge-to-edge CRC-32.

Some specific applications may be tolerant of residual errors in the data they exchange, but removal of the link CRC may expose the network to an undesirable increase in undetected errors in the IP and transport headers. Applications may also require a high level of error protection for control information exchanged by protocols acting above the transport layer. One example is a voice codec, which is robust against bit errors in the speech samples. For such mechanisms to work, the receiving application must be able to tolerate receiving corrupted data. This also requires that an application uses a mechanism to signal that payload corruption is permitted and to indicate the coverage (headers and data) required to be protected by the subnetwork CRC. The UDP-Lite protocol [RFC3828] is the first Internet standards track transport protocol supporting partial payload protection. Receipt of corrupt data by arbitrary

application protocols carries a serious danger that a subnet delivers data with errors that remain undetected by the application and hence corrupt the communicated data [SRC81].

8.4. How TCP Works

One of TCP's functions is end-host based congestion control for the Internet. This is a critical part of the overall stability of the Internet, so it is important that link-layer designers understand TCP's congestion control algorithms.

TCP assumes that, at the most abstract level, the network consists of links and queues. Queues provide output-buffering on links that are momentarily oversubscribed. They smooth instantaneous traffic bursts to fit the link bandwidth. When demand exceeds link capacity long enough to fill the queue, packets must be dropped. The traditional action of dropping the most recent packet ("tail dropping") is no longer recommended [RFC2309] [RFC2914], but it is still widely practiced.

TCP uses sequence numbering and acknowledgments (ACKs) on an end-to-end basis to provide reliable, sequenced delivery. TCP ACKs are cumulative, i.e., each implicitly ACKs every segment received so far. If a packet with an unexpected sequence number is received, the ACK field in the packets returned by the receiver will cease to advance. Using an optional enhancement, TCP can send selective acknowledgments (SACKs) [RFC2018] to indicate which segments have arrived at the receiver.

Since the most common cause of packet loss is congestion, TCP treats packet loss as an indication of potential Internet congestion along the path between TCP end hosts. This happens automatically, and the subnetwork need not know anything about IP or TCP. A subnetwork node simply drops packets whenever it must, though some packet-dropping strategies (e.g., RED) are more fair to competing flows than others.

TCP recovers from packet losses in two different ways. The most important mechanism is the retransmission timeout. If an ACK fails to arrive after a certain period of time, TCP retransmits the oldest unacked packet. Taking this as a hint that the network is congested, TCP waits for the retransmission to be ACKed before it continues, and it gradually increases the number of packets in flight as long as a timeout does not occur again.

A retransmission timeout can impose a significant performance penalty, as the sender is idle during the timeout interval and restarts with a congestion window of one TCP segment following the

timeout. To allow faster recovery from the occasional lost packet in a bulk transfer, an alternate scheme, known as "fast recovery", was introduced [RFC2581] [RFC2582] [RFC2914] [TCPF98].

Fast recovery relies on the fact that when a single packet is lost in a bulk transfer, the receiver continues to return ACKs to subsequent data packets that do not actually acknowledge any newly-received data. These are known as "duplicate acknowledgments" or "dupacks". The sending TCP can use dupacks as a hint that a packet has been lost and retransmit it without waiting for a timeout. Dupacks effectively constitute a negative acknowledgment (NAK) for the packet sequence number in the acknowledgment field. TCP waits until a certain number of dupacks (currently 3) are seen prior to assuming a loss has occurred; this helps avoid an unnecessary retransmission during out-of-sequence delivery.

A technique called "Explicit Congestion Notification" (ECN) [RFC3168] allows routers to directly signal congestion to hosts without dropping packets. This is done by setting a bit in the IP header. Since ECN support is likely to remain optional, the lack of an ECN bit must *never* be interpreted as a lack of congestion. Thus, for the foreseeable future, TCP must interpret a lost packet as a signal of congestion.

The TCP "congestion avoidance" [RFC2581] algorithm maintains a congestion window (cwnd) controlling the amount of data TCP may have in flight at any moment. Reducing cwnd reduces the overall bandwidth obtained by the connection; similarly, raising cwnd increases performance, up to the limit of the available capacity.

TCP probes for available network capacity by initially setting cwnd to one or two packets and then increasing cwnd by one packet for each ACK returned from the receiver. This is TCP's "slow start" mechanism. When a packet loss is detected (or congestion is signaled by other mechanisms), cwnd is reset to one and the slow start process is repeated until cwnd reaches one half of its previous setting before the reset. Cwnd continues to increase past this point, but at a much slower rate than before. If no further losses occur, cwnd will ultimately reach the window size advertised by the receiver.

This is an "Additive Increase, Multiplicative Decrease" (AIMD) algorithm. The steep decrease of cwnd in response to congestion provides for network stability; the AIMD algorithm also provides for fairness between long running TCP connections sharing the same path.

8.5. TCP Performance Characteristics

Caveat

Here we present a current "state-of-the-art" understanding of TCP performance. This analysis attempts to characterize the performance of TCP connections over links of varying characteristics.

Link designers may wish to use the techniques in this section to predict what performance TCP/IP may achieve over a new link-layer design. Such analysis is encouraged. Because this is a relatively new analysis, and the theory is based on single-stream TCP connections under "ideal" conditions, it should be recognized that the results of such analysis may differ from actual performance in the Internet. That being said, we have done our best to provide the designers with helpful information to get an accurate picture of the capabilities and limitations of TCP under various conditions.

8.5.1. The Formulae

The performance of TCP's AIMD Congestion Avoidance algorithm has been extensively analyzed. The current best formula for the performance of the specific algorithms used by Reno TCP (i.e., the TCP specified in [RFC2581]) is given by Padhye, et al. [PFTK98]. This formula is:

$$BW = \frac{MSS}{RTT*\sqrt{1.33*p} + RTO*p*[1+32*p^2]*\min[1,3*\sqrt{.75*p}]}$$

where

BW	is the maximum TCP throughout achievable by an individual TCP flow
MSS	is the TCP segment size being used by the connection
RTT	is the end-to-end round trip time of the TCP connection
RTO	is the packet timeout (based on RTT)
p	is the packet loss rate for the path (i.e., .01 if there is 1% packet loss)

Note that the speed of the links making up the Internet path does not explicitly appear in this formula. Attempting to send faster than the slowest link in the path causes the queue to grow at the transmitter driving the bottleneck. This increases the RTT, which in turn reduces the achievable throughput.

This is currently considered to be the best approximate formula for Reno TCP performance. A further simplification of this formula is generally made by assuming that RTO is approximately 5*RTT.

TCP is constantly being improved. A simpler formula, which gives an upper bound on the performance of any AIMD algorithm which is likely to be implemented in TCP in the future, was derived by Ott, et al. [MSMO97].

$$BW = C \frac{MSS}{RTT} \frac{1}{\sqrt{p}}$$

where C is 0.93.

8.5.2. Assumptions

Both formulae assume that the TCP Receiver Window is not limiting the performance of the connection. Because the receiver window is entirely determined by end-hosts, we assume that hosts will maximize the announced receiver window to maximize their network performance.

Both of these formulae allow BW to become infinite if there is no loss. However, an Internet path will drop packets at bottlenecked queues if the load is too high. Thus, a completely lossless TCP/IP network can never occur (unless the network is being underutilized).

The RTT used is the arithmetic average, including queuing delays.

The formulae are for a single TCP connection. If a path carries many TCP connections, each will follow the formulae above independently.

The formulae assume long-running TCP connections. For connections that are extremely short (<10 packets) and don't lose any packets, performance is driven by the TCP slow-start algorithm. For connections of medium length, where on average only a few segments are lost, single connection performance will actually be slightly better than given by the formulae above.

The difference between the simple and complex formulae above is that the complex formula includes the effects of TCP retransmission timeouts. For very low levels of packet loss (significantly less than 1%), timeouts are unlikely to occur, and the formulae lead to very similar results. At higher packet losses (1% and above), the complex formula gives a more accurate estimate of performance (which will always be significantly lower than the result from the simple formula).

Note that these formulae break down as p approaches 100%.

8.5.3. Analysis of Link-Layer Effects on TCP Performance

Consider the following example:

A designer invents a new wireless link layer which, on average, loses 1% of IP packets. The link layer supports packets of up to 1040 bytes, and has a one-way delay of 20 msec.

If this link were to be used on an Internet path with a round trip time greater than 80ms, the upper bound may be computed by:

For MSS, use 1000 bytes to exclude the 40 bytes of minimum IPv4 and TCP headers.

For RTT, use 120 msec (80 msec for the Internet part, plus 20 msec each way for the new wireless link).

For p, use .01. For C, assume 1.

The simple formula gives:

BW = (1000 * 8 bits) / (.120 sec * sqrt(.01)) = 666 kbit/sec

The more complex formula gives:

BW = 402.9 kbit/sec

If this were a 2 Mb/s wireless LAN, the designers might be somewhat disappointed.

Some observations on performance:

1. We have assumed that the packet losses on the link layer are interpreted as congestion by TCP. This is a "fact of life" that must be accepted.

2. The equations for TCP performance are all expressed in terms of packet loss, but many subnetwork designers think in terms of bit-error ratio. *If* channel bit errors are independent, then the probability of a packet being corrupted is:

   p = 1 - ([1 - BER]^[FRAME_SIZE*8])

   Here we assume FRAME_SIZE is in bytes and "^" represents exponentiation. It includes the user data and all headers (TCP,IP and subnetwork). (Note: this analysis assumes the

subnetwork does not perform ARQ or transparent fragmentation [RFC3366].) If the inequality

BER * [FRAME_SIZE*8] << 1

holds, the packet loss probability p can be approximated by:

p = BER * [FRAME_SIZE*8]

These equations can be used to apply BER to the performance equations above.

Note that FRAME_SIZE can vary from one packet to the next. Small packets (such as TCP acks) generally have a smaller probability of packet error than, say, a TCP packet carrying one MSS (maximum segment size) of user data. A flow of small TCP acks can be expected to be slightly more reliable than a stream of larger TCP data segments.

It bears repeating that the above analysis assumes that bit errors are statistically independent. Because this is not true for many real links, our computation of p is actually an upper bound, not the exact probability of packet loss.

There are many reasons why bit errors are not independent on real links. Many radio links are affected by propagation fading or by interference that lasts over many bit times. Also, links with Forward Error Correction (FEC) generally have very non-uniform bit error distributions that depend on the type of FEC, but in general the uncorrected errors tend to occur in bursts even when channel symbol errors are independent. In all such cases, our computation of p from BER can only place an upper limit on the packet loss rate.

If the distribution of errors under the FEC scheme is known, one could apply the same type of analysis as above, using the correct distribution function for the BER. It is more likely in these FEC cases, however, that empirical methods are needed to determine the actual packet loss rate.

3. Note that the packet size plays an important role. If the subnetwork loss characteristics are such that large packets have the same probability of loss as smaller packets, then larger packets will yield improved performance.

4. We have chosen a specific RTT that might occur on a wide-area Internet path within the USA. It is important to recognize that a variety of RTT values are experienced in the Internet.

   For example, RTTs are typically less than 10 msec in a wired LAN environment when communicating with a local host. International connections may have RTTs of 200 msec or more. Modems and other low-capacity links can add considerable delay due to their long packet transmission (serialisation) times.

   Links over geostationary repeater satellites have one-way speed-of-light delays of around 250ms, a minimum of 125ms propagation delay up to the satellite and 125ms down. The RTT of an end-to-end TCP connection that includes such a link can be expected to be greater than 250ms.

   Queues on heavily-congested links may back up, increasing RTTs. Finally, virtual private networks (VPNs) and other forms of encryption and tunneling can add significant end-to-end delay to network connections.

9. Quality-of-Service (QoS) considerations

It is generally recognized that specific service guarantees are needed to support real-time multimedia, toll-quality telephony, and other performance-critical applications. The provision of such Quality of Service guarantees in the Internet is an active area of research and standardization. The IETF has not converged on a single service model, set of services, or single mechanism that will offer useful guarantees to applications and be scalable to the Internet. Indeed, the IETF does not have a single definition of Quality of Service. [RFC2990] represents a current understanding of the challenges in architecting QoS for the Internet.

There are presently two architectural approaches to providing mechanisms for QoS support in the Internet.

IP Integrated Services (Intserv) [RFC1633] provides fine-grained service guarantees to individual flows. Flows are identified by a flow specification (flowspec), which creates a stateful association between individual packets by matching fields in the packet header. Capacity is reserved for the flow, and appropriate traffic conditioning and scheduling is installed in routers along the path. The ReSerVation Protocol (RSVP) [RFC2205] [RFC2210] is usually, but need not necessarily be, used to install the flow QoS state. Intserv defines two services, in addition to the Default (best effort) service.

1. Guaranteed Service (GS) [RFC2212] offers hard upper bounds on delay to flows that conform to a traffic specification (TSpec). It uses a fluid-flow model to relate the TSpec and reserved bandwidth (RSpec) to variable delay. Non-conforming packets are forwarded on a best-effort basis.

2. Controlled Load Service (CLS) [RFC2211] offers delay and packet loss equivalent to that of an unloaded network to flows that conform to a TSpec, but no hard bounds. Non-conforming packets are forwarded on a best-effort basis.

Intserv requires installation of state information in every participating router. Performance guarantees cannot be made unless this state is present in every router along the path. This, along with RSVP processing and the need for usage-based accounting, is believed to have scalability problems, particularly in the core of the Internet [RFC2208].

IP Differentiated Services (Diffserv) [RFC2475] provides a "toolkit" offering coarse-grained controls to aggregates of flows. Diffserv in itself does *not* provide QoS guarantees, but can be used to construct services with QoS guarantees across a Diffserv domain. Diffserv attempts to address the scaling issues associated with Intserv by requiring state awareness only at the edge of a Diffserv domain. At the edge, packets are classified into flows, and the flows are conditioned (marked, policed, or shaped) to a traffic conditioning specification (TCS). A Diffserv Codepoint (DSCP), identifying a per-hop behavior (PHB), is set in each packet header. The DSCP is carried in the DS-field, subsuming six bits of the former Type-of-Service (ToS) byte [RFC791] of the IP header [RFC2474]. The PHB denotes the forwarding behavior to be applied to the packet in each node in the Diffserv domain. Although there is a "recommended" DSCP associated with each PHB, the mappings from DSCPs to PHBs are defined by the DS-domain. In fact, there can be several DSCPs associated with the same PHB. Diffserv presently defines three PHBs.

1. The class selector PHB [RFC2474] replaces the IP precedence field of the former ToS byte. It offers relative forwarding priorities.

2. The Expedited Forwarding (EF) PHB [RFC3246] [RFC3248] guarantees that packets will have a well-defined minimum departure rate which, if not exceeded, ensures that the associated queues are short or empty. EF is intended to support services that offer tightly-bounded loss, delay, and delay jitter.

3. The Assured Forwarding (AF) PHB group [RFC2597] offers different levels of forwarding assurance for each aggregated flow of packets. Each AF group is independently allocated forwarding resources. Packets are marked with one of three drop precedences; those with the highest drop precedence are dropped with lower probability than those marked with the lowest drop precedence. DSCPs are recommended for four independent AF groups, although a DS domain can have more or fewer AF groups.

Ongoing work in the IETF is addressing ways to support Intserv with Diffserv. There is some belief (e.g., as expressed in [RFC2990]) that such an approach will allow individual flows to receive service guarantees and scale to the global Internet.

The QoS guarantees that can be offered by the IP layer are a product of two factors:

1. the concatenation of the QoS guarantees offered by the subnets along the path of a flow. This implies that a subnet may wish to offer multiple services (with different QoS guarantees) to the IP layer, which can then determine which flows use which subnet service. To put it another way, forwarding behavior in the subnet needs to be "clued" by the forwarding behavior (service or PHB) at the IP layer, and

2. the operation of a set of cooperating mechanisms, such as bandwidth reservation and admission control, policy management, traffic classification, traffic conditioning (marking, policing and/or shaping), selective discard, queuing, and scheduling. Note that support for QoS in subnets may require similar mechanisms, especially when these subnets are general topology subnets (e.g., ATM, frame relay, or MPLS) or shared media subnets.

Many subnetwork designers face inherent tradeoffs between delay, throughput, reliability, and cost. Other subnetworks have parameters that manage bandwidth, internal connection state, and the like. Therefore, the following subnetwork capabilities may be desirable, although some might be trivial or moot if the subnet is a dedicated point-to-point link.

1. The subnetwork should have the ability to reserve bandwidth for a connection or flow and schedule packets accordingly.

2. Bandwidth reservations should be based on a one- or two-token bucket model, depending on whether the service is intended to support constant-rate or bursty traffic.

3. If a connection or flow does not use its reserved bandwidth at a given time, the unused bandwidth should be available for other flows.

4. Packets in excess of a connection or flow's agreed rate should be forwarded as best-effort or discarded, depending on the service offered by the subnet to the IP layer.

5. If a subnet contains error control mechanisms (retransmission and/or FEC), it should be possible for the IP layer to influence the inherent tradeoffs between uncorrected errors, packet losses, and delay. These capabilities at the subnet/IP layer service boundary correspond to selection of more or less error control and/or to selection of particular error control mechanisms within the subnetwork.

6. The subnet layer should know, and be able to inform the IP layer, how much fixed delay and delay jitter it offers for a flow or connection. If the Intserv model is used, the delay jitter component may be best expressed in terms of the TSpec/RSpec model described in [RFC2212].

7. Support of the Diffserv class selectors [RFC2474] suggests that the subnet might consider mechanisms that support priorities.

10. Fairness vs Performance

Subnetwork designers should be aware of the tradeoffs between fairness and efficiency inherent in many transmission scheduling algorithms. For example, many local area networks use contention protocols to resolve access to a shared transmission channel. These protocols represent overhead. While limiting the amount of data that a subnet node may transmit per contention cycle helps assure timely access to the channel for each subnet node, it also increases contention overhead per unit of data sent.

In some mobile radio networks, capacity is limited by interference, which in turn depends on average transmitter power. Some receivers may require considerably more transmitter power (generating more interference and consuming more channel capacity) than others.

In each case, the scheduling algorithm designer must balance competing objectives: providing a fair share of capacity to each subnet node while maximizing the total capacity of the network. One approach for balancing performance and fairness is outlined in [ES00].

11. Delay Characteristics

The TCP sender bases its retransmission timeout (RTO) on measurements of the round trip delay experienced by previous packets. This allows TCP to adapt automatically to the very wide range of delays found on the Internet. The recommended algorithms are described in [RFC2988]. Evaluations of TCP's retransmission timer can be found in [AP99] and [LS00].

These algorithms model the delay along an Internet path as a normally-distributed random variable with a slowly-varying mean and standard deviation. TCP estimates these two parameters by exponentially smoothing individual delay measurements, and it sets the RTO to the estimated mean delay plus some fixed number of standard deviations. (The algorithm actually uses mean deviation as an approximation to standard deviation, because it is easier to compute.)

The goal is to compute an RTO that is small enough to detect and recover from packet losses while minimizing unnecessary ("spurious") retransmissions when packets are unexpectedly delayed but not lost. Although these goals conflict, the algorithm works well when the delay variance along the Internet path is low, or the packet loss rate is low.

If the path delay variance is high, TCP sets an RTO that is much larger than the mean of the measured delays. If the packet loss rate is low, the large RTO is of little consequence, as timeouts occur only rarely. Conversely, if the path delay variance is low, then TCP recovers quickly from lost packets; again, the algorithm works well. However, when delay variance and the packet loss rate are both high, these algorithms perform poorly, especially when the mean delay is also high.

Because TCP uses returning acknowledgments as a "clock" to time the transmission of additional data, excessively high delays (even if the delay variance is low) also affect TCP's ability to fully utilize a high-speed transmission pipe. It also slows the recovery of lost packets, even when delay variance is small.

Subnetwork designers should therefore minimize all three parameters (delay, delay variance, and packet loss) as much as possible.

In many subnetworks, these parameters are inherently in conflict. For example, on a mobile radio channel, the subnetwork designer can use retransmission (ARQ) and/or forward error correction (FEC) to trade off delay, delay variance, and packet loss in an effort to improve TCP performance. While ARQ increases delay variance, FEC

does not. However, FEC (especially when combined with interleaving) often increases mean delay, even on good channels where ARQ retransmissions are not needed and ARQ would not increase either the delay or the delay variance.

The tradeoffs among these error control mechanisms and their interactions with TCP can be quite complex, and are the subject of much ongoing research. We therefore recommend that subnetwork designers provide as much flexibility as possible in the implementation of these mechanisms, and provide access to them as discussed above in the section on Quality of Service.

12. Bandwidth Asymmetries

Some subnetworks may provide asymmetric bandwidth (or may cause TCP packet flows to experience asymmetry in the capacity) and the Internet protocol suite will generally still work fine. However, there is a case when such a scenario reduces TCP performance. Since TCP data segments are "clocked" out by returning acknowledgments, TCP senders are limited by the rate at which ACKs can be returned [BPK98]. Therefore, when the ratio of the available capacity of the Internet path carrying the data to the bandwidth of the return path of the acknowledgments is too large, the slow return of the ACKs directly impacts performance. Since ACKs are generally smaller than data segments, TCP can tolerate some asymmetry, but as a general rule, designers of subnetworks should be aware that subnetworks with significant asymmetry can result in reduced performance, unless issues are taken to mitigate this [RFC3449].

Several strategies have been identified for reducing the impact of asymmetry of the network path between two TCP end hosts, e.g., [RFC3449]. These techniques attempt to reduce the number of ACKs transmitted over the return path (low bandwidth channel) by changes at the end host(s), and/or by modification of subnetwork packet forwarding. While these solutions may mitigate the performance issues caused by asymmetric subnetworks, they do have associated cost and may have other implications. A fuller discussion of strategies and their implications is provided in [RFC3449].

13. Buffering, flow and congestion control

Many subnets include multiple links with varying traffic demands and possibly different transmission speeds. At each link there must be a queuing system, including buffering, scheduling, and a capability to discard excess subnet packets. These queues may also be part of a subnet flow control or congestion control scheme.

For the purpose of this discussion, we talk about packets without regard to whether they refer to a complete IP packet or a subnetwork frame. At each queue, a packet experiences a delay that depends on competing traffic and the scheduling discipline, and is subjected to a local discarding policy.

Some subnets may have flow or congestion control mechanisms in addition to packet dropping. Such mechanisms can operate on components in the subnet layer, such as schedulers, shapers, or discarders, and can affect the operation of IP forwarders at the edges of the subnet. However, with the exception of Explicit Congestion Notification [RFC3168] (discussed below), IP has no way to pass explicit congestion or flow control signals to TCP.

TCP traffic, especially aggregated TCP traffic, is bursty. As a result, instantaneous queue depths can vary dramatically, even in nominally stable networks. For optimal performance, packets should be dropped in a controlled fashion, not just when buffer space is unavailable. How much buffer space should be supplied is still a matter of debate, but as a rule of thumb, each node should have enough buffering to hold one link_bandwidth*link_delay product's worth of data for each TCP connection sharing the link.

This is often difficult to estimate, since it depends on parameters beyond the subnetwork's control or knowledge. Internet nodes generally do not implement admission control policies, and cannot limit the number of TCP connections that use them. In general, it is wise to err in favor of too much buffering rather than too little. It may also be useful for subnets to incorporate mechanisms that measure propagation delays to assist in buffer sizing calculations.

There is a rough consensus in the research community that active queue management is important to improving fairness, link utilization, and throughput [RFC2309]. Although there are questions and concerns about the effectiveness of active queue management (e.g., [MBDL99]), it is widely considered an improvement over tail-drop discard policies.

One form of active queue management is the Random Early Detection (RED) algorithm [RED93], a family of related algorithms. In one version of RED, an exponentially-weighted moving average of the queue depth is maintained:

> When this average queue depth is between a maximum threshold max_th and a minimum threshold min_th, the probability of packets that are dropped is proportional to the amount by which the average queue depth exceeds min_th.

When this average queue depth is equal to max_th, the drop probability is equal to a configurable parameter max_p.

When this average queue depth is greater than max_th, packets are always dropped.

Numerous variants on RED appear in the literature, and there are other active queue management algorithms which claim various advantages over RED [GM02].

With an active queue management algorithm, dropped packets become a feedback signal to trigger more appropriate congestion behavior by the TCPs in the end hosts. Randomization of dropping tends to break up the observed tendency of TCP windows belonging to different TCP connections to become synchronized by correlated drops, and it also imposes a degree of fairness on those connections that implement TCP congestion avoidance properly. Another important property of active queue management algorithms is that they attempt to keep average queue depths short while accommodating large short-term bursts.

Since TCP neither knows nor cares whether congestive packet loss occurs at the IP layer or in a subnet, it may be advisable for subnets that perform queuing and discarding to consider implementing some form of active queue management. This is especially true if large aggregates of TCP connections are likely to share the same queue. However, active queue management may be less effective in the case of many queues carrying smaller aggregates of TCP connections, e.g., in an ATM switch that implements per-VC queuing.

Note that the performance of active queue management algorithms is highly sensitive to settings of configurable parameters, and also to factors such as RTT [MBB00] [FB00].

Some subnets, most notably ATM, perform segmentation and reassembly at the subnetwork edges. Care should be taken here in designing discard policies. If the subnet discards a fragment of an IP packet, then the remaining fragments become an unproductive load on the subnet that can markedly degrade end-to-end performance [RF95]. Subnetworks should therefore attempt to discard these extra fragments whenever one of them must be discarded. If the IP packet has already been partially forwarded when discarding becomes necessary, then every remaining fragment except the one marking the end of the IP packet should also be discarded. For ATM subnets, this specifically means using Early Packet Discard and Partial Packet Discard [ATMFTM].

Some subnets include flow control mechanisms that effectively require that the rate of traffic flows be shaped upon entry to the subnet. One example of such a subnet mechanism is in the ATM Available Bit

rate (ABR) service category [ATMFTM]. Such flow control mechanisms have the effect of making the subnet nearly lossless by pushing congestion into the IP routers at the edges of the subnet. In such a case, adequate buffering and discard policies are needed in these routers to deal with a subnet that appears to have varying bandwidth. Whether there is a benefit in this kind of flow control is controversial; there are numerous simulation and analytical studies that go both ways. It appears that some of the issues leading to such different results include sensitivity to ABR parameters, use of binary rather than explicit rate feedback, use (or not) of per-VC queuing, and the specific ATM switch algorithms selected for the study. Anecdotally, some large networks that used IP over ABR to carry TCP traffic have claimed it to be successful, but have published no results.

Another possible approach to flow control in the subnet would be to work with TCP Explicit Congestion Notification (ECN) semantics [RFC3168] through utilizing explicit congestion indicators in subnet frames. Routers at the edges of the subnet, rather than shaping, would set the explicit congestion bit in those IP packets that are received in subnet frames that have an ECN indication. Nodes in the subnet would need to implement an active queue management protocol that marks subnet frames instead of dropping them.

ECN is currently a proposed standard, but it is not yet widely deployed.

14. Compression

Application data compression is a function that can usually be omitted in the subnetwork. The endpoints typically have more CPU and memory resources to run a compression algorithm and a better understanding of what is being compressed. End-to-end compression benefits every network element in the path, while subnetwork-layer compression, by definition, benefits only a single subnetwork.

Data presented to the subnetwork layer may already be in a compressed format (e.g., a JPEG file), compressed at the application layer (e.g., the optional "gzip", "compress", and "deflate" compression in HTTP/1.1 [RFC2616]), or compressed at the IP layer (the IP Payload Compression Protocol [RFC3173] supports DEFLATE [RFC2394] and LZS [RFC2395]). Compression at the subnetwork edges is of no benefit for any of these cases.

The subnetwork may also process data that has been encrypted by the application (OpenPGP [RFC2440] or S/MIME [RFC2633]), just above TCP (SSL, TLS [RFC2246]), or just above IP (IPsec ESP [RFC2406]).

Ciphers generate high-entropy bit streams lacking any patterns that can be exploited by a compression algorithm.

However, much data is still transmitted uncompressed over the Internet, so subnetwork compression may be beneficial. Any subnetwork compression algorithm must not expand uncompressible data, e.g., data that has already been compressed or encrypted.

We make a strong recommendation that subnetworks operating at low speed or with small MTUs compress IP and transport-level headers (TCP and UDP) using several header compression schemes developed within the IETF [RFC3150]. An uncompressed 40-byte TCP/IP header takes about 33 milliseconds to send at 9600 bps. "VJ" TCP/IP header compression [RFC1144] compresses most headers to 3-5 bytes, reducing transmission time to several milliseconds on dialup modem links. This is especially beneficial for small, latency-sensitive packets in interactive sessions.

Similarly, RTP compression schemes, such as CRTP [RFC2508] and ROHC [RFC3095], compress most IP/UDP/RTP headers to 1-4 bytes. The resulting savings are especially significant when audio packets are kept small to minimize store-and-forward latency.

Designers should consider the effect of the subnetwork error rate on the performance of header compression. TCP ordinarily recovers from lost packets by retransmitting only those packets that were actually lost; packets arriving correctly after a packet loss are kept on a resequencing queue and do not need to be retransmitted. In VJ TCP/IP [RFC1144] header compression, however, the receiver cannot explicitly notify a sender of data corruption and subsequent loss of synchronization between compressor and decompressor. It relies instead on TCP retransmission to re-synchronize the decompressor. After a packet is lost, the decompressor must discard every subsequent packet, even if the subnetwork makes no further errors, until the sending TCP retransmits to re-synchronize the decompressor. This effect can substantially magnify the effect of subnetwork packet losses if the sending TCP window is large, as it will often be on a path with a large bandwidth*delay product [LRKOJ99].

Alternate header compression schemes, such as those described in [RFC2507], include an explicit request for retransmission of an uncompressed packet to allow decompressor resynchronization without waiting for a TCP retransmission. However, these schemes are not yet in widespread use.

Both TCP header compression schemes do not compress widely-used TCP options such as selective acknowledgements (SACK). Both fail to compress TCP traffic that makes use of explicit congestion

notification (ECN). Work is under way in the IETF ROHC WG to address these shortcomings in a ROHC header compression scheme for TCP [RFC3095] [RFC3096].

The subnetwork error rate also is important for RTP header compression. CRTP uses delta encoding, so a packet loss on the link causes uncertainty about the subsequent packets, which often must be discarded until the decompressor has notified the compressor and the compressor has sent re-synchronizing information. This typically takes slightly more than the end-to-end path round-trip time. For links that combine significant error rates with latencies that require multiple packets to be in flight at a time, this leads to significant error propagation, i.e., subsequent losses caused by an initial loss.

For links that are both high-latency (multiple packets in flight from a typical RTP stream) and error-prone, RTP ROHC provides a more robust way of RTP header compression, at a cost of higher complexity at the compressor and decompressor. For example, within a talk spurt, only extended losses of (depending on the mode chosen) 12-64 packets typically cause error propagation.

15. Packet Reordering

The Internet architecture does not guarantee that packets will arrive in the same order in which they were originally transmitted; transport protocols like TCP must take this into account.

However, reordering does come at a cost with TCP as it is currently defined. Because TCP returns a cumulative acknowledgment (ACK) indicating the last in-order segment that has arrived, out-of-order segments cause a TCP receiver to transmit a duplicate acknowledgment. When the TCP sender notices three duplicate acknowledgments, it assumes that a segment was dropped by the network and uses the fast retransmit algorithm [Jac90] [RFC2581] to resend the segment. In addition, the congestion window is reduced by half, effectively halving TCP's sending rate. If a subnetwork reorders segments significantly such that three duplicate ACKs are generated, the TCP sender needlessly reduces the congestion window and performance suffers.

Packet reordering frequently occurs in parts of the Internet, and it seems to be difficult or impossible to eliminate [BPS99]. For this reason, research on improving TCP's behavior in the face of packet reordering [LK00] [BA02] has begun.

[BPS99] cites reasons why it may even be undesirable to eliminate reordering. There are situations where average packet latency can be reduced, link efficiency can be increased, and/or reliability can be improved if reordering is permitted. Examples include certain high speed switches within the Internet backbone and the parallel links used over many Internet paths for load splitting and redundancy.

This suggests that subnetwork implementers should try to avoid packet reordering whenever possible, but not if doing so compromises efficiency, impairs reliability, or increases average packet delay.

Note that every header compression scheme currently standardized for the Internet requires in-order packet delivery on the link between compressor and decompressor. PPP is frequently used to carry compressed TCP/IP packets; since it was originally designed for point-to-point and dialup links, it is assumed to provide in-order delivery. For this reason, subnetwork implementers who provide PPP interfaces to VPNs and other more complex subnetworks, must also maintain in-order delivery of PPP frames.

16. Mobility

Internet users are increasingly mobile. Not only are many Internet nodes laptop computers, but pocket organizers and mobile embedded systems are also becoming nodes on the Internet. These nodes may connect to many different access points on the Internet over time, and they expect this to be largely transparent to their activities. Except when they are not connected to the Internet at all, and for performance differences when they are connected, they expect that everything will "just work" regardless of their current Internet attachment point or local subnetwork technology.

Changing a host's Internet attachment point involves one or more of the following steps.

First, if use of the local subnetwork is restricted, the user's credentials must be verified and access granted. There are many ways to do this. A trivial example would be an "Internet cafe" that grants physical access to the subnetwork for a fee. Subnetworks may implement technical access controls of their own; one example is IEEE 802.11 Wireless Equivalent Privacy [IEEE80211]. It is common practice for both cellular telephone and Internet service providers (ISPs) to agree to serve one anothers' users; RADIUS [RFC2865] is the standard method for ISPs to exchange authorization information.

Second, the host may have to be reconfigured with IP parameters appropriate for the local subnetwork. This usually includes setting an IP address, default router, and domain name system (DNS) servers.

On multiple-access networks, the Dynamic Host Configuration Protocol (DHCP) [RFC2131] is almost universally used for this purpose. On PPP links, these functions are performed by the IP Control Protocol (IPCP) [RFC1332].

Third, traffic destined for the mobile host must be routed to its current location. This roaming function is the most common meaning of the term "Internet mobility".

Internet mobility can be provided at any of several layers in the Internet protocol stack, and there is ongoing debate as to which is the most appropriate and efficient. Mobility is already a feature of certain application layer protocols; the Post Office Protocol (POP) [RFC1939] and the Internet Message Access Protocol (IMAP) [RFC3501] were created specifically to provide mobility in the receipt of electronic mail.

Mobility can also be provided at the IP layer [RFC3344]. This mechanism provides greater transparency, viz., IP addresses that remain fixed as the nodes move, but at the cost of potentially significant network overhead and increased delay because of the sub-optimal network routing and tunneling involved.

Some subnetworks may provide internal mobility, transparent to IP, as a feature of their own internal routing mechanisms. To the extent that these simplify routing at the IP layer, reduce the need for mechanisms like Mobile IP, or exploit mechanisms unique to the subnetwork, this is generally desirable. This is especially true when the subnetwork covers a relatively small geographic area and the users move rapidly between the attachment points within that area. Examples of internal mobility schemes include Ethernet switching and intra-system handoff in cellular telephony.

However, if the subnetwork is physically large and connects to other parts of the Internet at multiple geographic points, care should be taken to optimize the wide-area routing of packets between nodes on the external Internet and nodes on the subnet. This is generally done with "nearest exit" routing strategies. Because a given subnetwork may be unaware of the actual physical location of a destination on another subnetwork, it simply routes packets bound for the other subnetwork to the nearest router between the two. This implies some awareness of IP addressing and routing within the subnetwork. The subnetwork may wish to use IP routing internally for wide area routing and restrict subnetwork-specific routing to constrained geographic areas where the effects of suboptimal routing are minimized.

17. Routing

Subnetworks connecting more than two systems must provide their own internal Layer-2 forwarding mechanisms, either implicitly (e.g., broadcast) or explicitly (e.g., switched). Since routing is the major function of the Internet layer, the question naturally arises as to the interaction between routing at the Internet layer and routing in the subnet, and proper division of function between the two.

Layer-2 subnetworks can be point-to-point, connecting two systems, or multipoint. Multipoint subnetworks can be broadcast (e.g., shared media or emulated) or non-broadcast. Generally, IP considers multipoint subnetworks as broadcast, with shared-medium Ethernet as the canonical (and historical) example, and point-to-point subnetworks as a degenerate case. Non-broadcast subnetworks may require additional mechanisms, e.g., above IP at the routing layer [RFC2328].

IP is ignorant of the topology of the subnetwork layer. In particular, reconfiguration of subnetwork paths is not tracked by the IP layer. IP is only affected by whether it can send/receive packets sent to the remotely connected systems via the subnetwork interface (i.e., the reachability from one router to another). IP further considers that subnetworks are largely static -- that both their membership and existence are stable at routing timescales (tens of seconds); changes to these are considered re-provisioning, rather than routing.

Routing functionality in a subnetwork is related to addressing in that subnetwork. Resolution of addresses on subnetwork links is required for forwarding IP packets across links (e.g., ARP for IPv4, or ND for IPv6). There is unlikely to be direct interaction between subnetwork routing and IP routing. Where broadcast is provided or explicitly emulated, address resolution can be used directly; where not provided, the link layer routing may interface to a protocol for resolution, e.g., to the Next-Hop Resolution Protocol [RFC2322] to provide context-dependent address resolution capabilities.

Subnetwork routing can either complement or compete with IP routing. It complements IP when a subnetwork encapsulates its internal routing, and where the effects of that routing are not visible at the IP layer. However, if different paths in the subnetwork have characteristics that affect IP routing, it can affect or even inhibit the convergence of IP routing.

Routing protocols generally consider Layer-2 subnetworks, i.e., with subnet masks and no intermediate IP hops, to have uniform routing metrics to all members. Routing can break when a link's characteristics do not match the routing metric, in this case, e.g., when some member pairs have different path characteristics. Consider a virtual Ethernet subnetwork that includes both nearby (sub-millisecond latency) and remote (100's of milliseconds away) systems. Presenting that group as a single subnetwork means that some routing protocols will assume that all pairs have the same delay, and that that delay is small. Because this is not the case, the routing tables constructed may be suboptimal or may even fail to converge.

When a subnetwork is used for transit between a set of routers, it conventionally provides the equivalent of a full mesh of point-to-point links. Simplicity of the internal subnet structure can be used (e.g., via NHRP [RFC2332]) to reduce the size of address resolution tables, but routing exchanges will continue to reflect the full mesh they emulate. In general, subnetworks should not be used as a transit among a set of routers where routing protocols would break if a full mesh of equivalent point-to-point links were used.

Some subnetworks have special features that allow the use of more effective or responsive routing mechanisms that cannot be implemented in IP because of its need for generality. One example is the self-learning bridge algorithm widely used in Ethernet networks. Learning bridges perform Layer-2 subnetwork forwarding, avoiding the need for dynamic routing at each subnetwork hop. Another is the "handoff" mechanism in cellular telephone networks, particularly the "soft handoff" scheme in IS-95 CDMA.

Subnetworks that cover large geographic areas or include links of widely-varying capabilities should be avoided. IP routing generally considers all multipoint subnets equivalent to a local, shared-medium link with uniform metrics between any pair of systems, and ignores internal subnetwork topology. Where a subnetwork diverges from that assumption, it is the obligation of subnetwork designers to provide compensating mechanisms. Not doing so can affect the scalability and convergence of IP routing, as noted above.

The subnetwork designer who decides to implement internal routing should consider whether a custom routing algorithm is warranted, or if an existing Internet routing algorithm or protocol may suffice. The designer should consider whether this decision is to reduce the address resolution table size (possible, but with additional protocol support required), or is trying to reduce routing table complexity. The latter may be better achieved by partitioning the subnetwork, either physically or logically, and using network-layer protocols to support partitioning (e.g., AS's in BGP). Protocols and routing

algorithms can be notoriously subtle, complex, and difficult to implement correctly. Much work can be avoided if existing protocols or implementations can be readily reused.

18. Security Considerations

Security has become a high priority in the design and operation of the Internet. The Internet is vast, and countless organizations and individuals own and operate its various components. A consensus has emerged for what might be called a "security placement principle": a security mechanism is most effective when it is placed as close as possible to, and under the direct control of the owner of the asset that it protects.

A corollary of this principle is that end-to-end security (e.g., confidentiality, authentication, integrity, and access control) cannot be ensured with subnetwork security mechanisms. Not only are end-to-end security mechanisms much more closely associated with the end-user assets they protect, they are also much more comprehensive. For example, end-to-end security mechanisms cover gaps that can appear when otherwise good subnetwork mechanisms are concatenated. This is an important application of the end-to-end principle [SRC81].

Several security mechanisms that can be used end-to-end have already been deployed in the Internet and are enjoying increasing use. The most important are the Secure Sockets Layer (SSL) [SSL2] [SSL3] and TLS [RFC2246] primarily used to protect web commerce, Pretty Good Privacy (PGP) [RFC1991] and S/MIME [RFCs-2630-2634], primarily used to protect and authenticate email and software distributions, the Secure Shell (SSH), used for secure remote access and file transfer, and IPsec [RFC2401], a general purpose encryption and authentication mechanism that sits just above IP and can be used by any IP application. (IPsec can actually be used either on an end-to-end basis or between security gateways that do not include either or both end systems.)

Nonetheless, end-to-end security mechanisms are not used as widely as might be desired. However, the group could not reach consensus on whether subnetwork designers should be actively encouraged to implement mechanisms to protect user data.

The clear consensus of the working group held that subnetwork security mechanisms, especially when weak or incorrectly implemented [BGW01], may actually be counterproductive. The argument is that subnetwork security mechanisms can lull end users into a false sense of security, diminish the incentive to deploy effective end-to-end

mechanisms, and encourage "risky" uses of the Internet that would not be made if users understood the inherent limits of subnetwork security mechanisms.

The other point of view encourages subnetwork security on the principle that it is better than the default situation, which all too often is no security at all. Users of especially vulnerable subnets (such as consumers who have wireless home networks and/or shared media Internet access) often have control over at most one endpoint -- usually a client -- and therefore cannot enforce the use of end-to-end mechanisms. However, subnet security can be entirely adequate for protecting low-valued assets against the most likely threats. In any event, subnet mechanisms do not preclude the use of end-to-end mechanisms, which are typically used to protect highly-valued assets. This viewpoint recognizes that many security policies implicitly assume that the entire end-to-end path is composed of a series of concatenated links that are nominally physically secured. That is, these policies assume that all endpoints of all links are trusted, and that access to the physical medium by attackers is difficult. To meet the assumptions of such policies, explicit mechanisms are needed for links (especially shared medium links) that lack physical protection. This, for example, is the rationale that underlies Wired Equivalent Privacy (WEP) in the IEEE 802.11 [IEEE80211] wireless LAN standard, and the Baseline Privacy Interface in the DOCSIS [DOCSIS1] [DOCSIS2] data over cable television networks standards.

We therefore recommend that subnetwork designers who choose to implement security mechanisms to protect user data be as candid as possible with the details of such security mechanisms and the inherent limits of even the most secure mechanisms when implemented in a subnetwork rather than on an end-to-end basis.

In keeping with the "placement principle", a clear consensus exists for another subnetwork security role: the protection of the subnetwork itself. Possible threats to subnetwork assets include theft of service and denial of service; shared media subnets tend to be especially vulnerable to such attacks. In some cases, mechanisms that protect subnet assets can also improve (but cannot ensure) end-to-end security.

One security service can be provided by the subnetwork that will aid in the solution of an overall Internet problem: subnetwork security should provide a mechanism to authenticate the source of a subnetwork frame. This function is missing in some current protocols, e.g., the use of ARP [RFC826] to associate an IPv4 address with a MAC address. The IPv6 Neighbor Discovery (ND) [RFC2461] performs a similar function.

There are well-known security flaws with this address resolution mechanism [Wilbur89]. However, the inclusion of subnetwork frame source authentication will permit a secure subnetwork address.

Another potential role for subnetwork security is to protect users against traffic analysis, i.e., identifying the communicating parties and determining their communication patterns and volumes even when their actual contents are protected by strong end-to-end security mechanisms. Lower-layer security can be more effective against traffic analysis due to its inherent ability to aggregate the communications of multiple parties sharing the same physical facilities while obscuring higher-layer protocol information that indicates specific end points, such as IP addresses and TCP/UDP port numbers.

However, traffic analysis is a notoriously subtle and difficult threat to understand and defeat, far more so than threats to confidentiality and integrity. We therefore urge extreme care in the design of subnetwork security mechanisms specifically intended to thwart traffic analysis.

Subnetwork designers must keep in mind that design and implementation for security is difficult [Schneier00]. [Schneier95] describes protocols and algorithms which are considered well-understood and believed to be sound.

Poor design process, subtle design errors and flawed implementation can result in gaping vulnerabilities. In recent years, a number of subnet standards have had problems exposed. The following are examples of mistakes that have been made:

1. Use of weak and untested algorithms [Crypto9912] [BGW01]. For a variety of reasons, algorithms were chosen which had subtle flaws, making them vulnerable to a variety of attacks.

2. Use of "security by obscurity" [Schneier00] [Crypto9912]. One common mistake is to assume that keeping cryptographic algorithms secret makes them more secure. This is intuitive, but wrong. Full public disclosure early in the design process attracts peer review by knowledgeable cryptographers. Exposure of flaws by this review far outweighs any imagined benefit from forcing attackers to reverse engineer security algorithms.

3. Inclusion of trapdoors [Schneier00] [Crypto9912]. Trapdoors are flaws surreptitiously left in an algorithm to allow it to be broken. This might be done to recover lost keys or to permit surreptitious access by governmental agencies. Trapdoors can be discovered and exploited by malicious attackers.

4. Sending passwords or other identifying information as clear text. For many years, analog cellular telephones could be cloned and used to steal service. The cloners merely eavesdropped on the registration protocols that exchanged everything in clear text.

5. Keys which are common to all systems on a subnet [BGW01].

6. Incorrect use of a sound mechanism. For example [BGW01], one subnet standard includes an initialization vector which is poorly designed and poorly specified. A determined attacker can easily recover multiple ciphertexts encrypted with the same key stream and perform statistical attacks to decipher them.

7. Identifying information sent in clear text that can be resolved to an individual, identifiable device. This creates a vulnerability to attacks targeted to that device (or its owner).

8. Inability to renew and revoke shared secret information.

9. Insufficient key length.

10. Failure to address "man-in-the-middle" attacks, e.g., with mutual authentication.

11. Failure to provide a form of replay detection, e.g., to prevent a receiver from accepting packets from an attacker that simply resends previously captured network traffic.

12. Failure to provide integrity mechanisms when providing confidentiality schemes [Bel98].

This list is by no means comprehensive. Design problems are difficult to avoid, but expert review is generally invaluable in avoiding problems.

In addition, well-designed security protocols can be compromised by implementation defects. Examples of such defects include use of predictable pseudo-random numbers [RFC1750], vulnerability to buffer overflow attacks due to unsafe use of certain I/O system calls [WFBA2000], and inadvertent exposure of secret data.

19. Contributors

This document represents a consensus of the members of the IETF Performance Implications of Link Characteristics (PILC) working group.

This document would not have been possible without the contributions of a great number of people in the Performance Implications of Link Characteristics Working Group. In particular, the following people provided major contributions of text, editing, and advice on this document: Mark Allman provided the final editing to complete this document. Carsten Bormann provided text on robust header compression. Gorry Fairhurst provided text on broadcast and multicast issues, routing, and many valuable comments on the entire document. Aaron Falk provided text on bandwidth on demand. Dan Grossman provided text on many facets of the document. Reiner Ludwig provided thorough document review and text on TCP vs. Link-Layer Retransmission. Jamshid Mahdavi provided text on TCP performance calculations. Saverio Mascolo provided feedback on the document. Gabriel Montenegro provided feedback on the document. Marie-Jose Montpetit provided text on bandwidth on demand. Joe Touch provided text on multicast, broadcast, and routing, and Lloyd Wood provided many valuable comments on versions of the document.

20. Informative References

References of the form RFCnnnn are Internet Request for Comments (RFC) documents available online at www.rfc-editor.org.

[802.1D] Information Technology Telecommunications and information exchange between systems Local and metropolitan area networks, Common specifications Media access control (MAC) bridges, IEEE 802.1D, 1998. ISO 15802-3.

[802.1p] IEEE, 802.1p, Standard for Local and Metropolitan Area Networks - Supplement to Media Access Control (MAC) Bridges: Traffic Class Expediting and Multicast.

[AP99] Allman, M. and V. Paxson, On Estimating End-to-End Network Path Properties, In Proceedings of ACM SIGCOMM 99.

[AR02] Acar, G. and C. Rosenberg, Weighted Fair Bandwidth-on-Demand (WFBoD) for Geo-Stationary Satellite Networks with On-Board Processing, Computer Networks, 39(1), 2002.

[ATMFTM] The ATM Forum, "Traffic Management Specification, Version 4.0", April 1996, document af-tm-0056.000. http://www.atmforum.com/

[BA02] Blanton, E. and M. Allman, On Making TCP More Robust to Packet Reordering. ACM Computer Communication Review, 32(1), January 2002.

[Bel98] Bellovin, S., "Cryptography and the Internet", in Proceedings of CRYPTO '98, August 1998. http://www.research.att.com/~smb/papers/inet-crypto.pdf

[BGW01] Borisov, N., Goldberg, I. and D. Wagner, "Intercepting Mobile Communications: The Insecurity of 802.11," In Proceedings of ACM MobiCom, July 2001.

[BPK98] Balakrishnan, H., Padmanabhan, V. and R. Katz. "The Effects of Asymmetry on TCP Performance." ACM Mobile Networks and Applications (MONET), 1998.

[BPS99] Bennet,, J.C.R., Partridge, C. and N. Shectman, "Packet Reordering is Not Pathological Network Behavior", IEEE/ACM Transactions on Networking, Vol. 7, No. 6, December 1999.

[CGMP] Farinacci D., Tweedly A. and T. Speakman, "Cisco Group Management Protocol (CGMP)", 1996/1997. ftp://ftpeng.cisco.com/ipmulticast/specs/cgmp.txt

[Crypto9912] Schneier, B., "European Cellular Encryption Algorithms" Crypto-Gram, December 15, 1999. http://www.counterpane.com

[DIX82] Digital Equipment Corp, Intel Corp, Xerox Corp, Ethernet Local Area Network Specification Version 2.0, November 1982.

[DOCSIS1] Data-Over-Cable Service Interface Specifications, Radio Frequency Interface Specification 1.0, SP-RFI-I05-991105, November 1999, Cable Television Laboratories, Inc.

[DOCSIS2] Data-Over-Cable Service Interface Specifications, Radio Frequency Interface Specification 1.1, SP-RFIv1.1-I05-000714, July 2000, Cable Television Laboratories, Inc.

[DOCSIS3] Lai, W.S., "DOCSIS-Based Cable Networks: Impact of Large Data Packets on Upstream Capacity", 14th ITC Specialists Seminar on Access Networks and Systems, Barcelona, Spain, April 25-27, 2001.

[EN301192] ETSI, European Broadcasting Union, Digital Video Broadcasting (DVB); DVB Specification for Data Broadcasting, European Standard (Telecommunications Series) EN 301 192 v1.2.1(1999-06).

[ES00] Eckhardt, D. and P. Steenkiste, "Effort-limited Fair (ELF) Scheduling for Wireless Networks, Proceedings of IEEE Infocom 2000.

[FB00] Firoiu V. and M. Borden, "A Study of Active Queue Management for Congestion Control" to appear in Infocom 2000.

[GM02] Grieco1, L. and S. Mascolo, "TCP Westwood and Easy RED to Improve Fairness in High-Speed Networks", Proceedings of the 7th International Workshop on Protocols for High-Speed Networks, April 2002.

[IEEE8023] IEEE 802.3 CSMA/CD Access Method. http://standards.ieee.org/

[IEEE80211] IEEE 802.11 Wireless LAN standard. http://standards.ieee.org/

[ISO3309] ISO/IEC 3309:1991(E), "Information Technology - Telecommunications and information exchange between systems - High-level data link control (HDLC) procedures - Frame structure", International Organization For Standardization, Fourth edition 1991-06-01.

[ISO13818] ISO/IEC, ISO/IEC 13818-1:2000(E) Information Technology - Generic coding of moving pictures and associated audio information: Systems, Second edition, 2000-12-01 International Organization for Standardization and International Electrotechnical Commission.

[ITU-I363] ITU-T I.363.5 B-ISDN ATM Adaptation Layer Specification Type AAL5, International Standards Organisation (ISO), 1996.

[Jac90] Jacobson, V., Modified TCP Congestion Avoidance Algorithm. Email to the end2end-interest mailing list, April 1990. ftp://ftp.ee.lbl.gov/email/vanj.90apr30.txt

[KY02] Khafizov, F. and M. Yavuz, Running TCP Over IS-2000, Proceedings of IEEE ICC, 2002.

[LK00] Ludwig, R. and R. H. Katz, "The Eifel Algorithm: Making TCP Robust Against Spurious Retransmissions", ACM Computer Communication Review, Vol. 30, No. 1, January 2000.

[LKJK02] Ludwig, R., Konrad, A., Joseph, A. D. and R. H. Katz, "Optimizing the End-to-End Performance of Reliable Flows over Wireless Links", Kluwer/ACM Wireless Networks Journal, Vol. 8, Nos. 2/3, pp. 289-299, March-May 2002.

[LRKOJ99] Ludwig, R., Rathonyi, B., Konrad, A., Oden, K. and A. Joseph, Multi-Layer Tracing of TCP over a Reliable Wireless Link, pp. 144-154, In Proceedings of ACM SIGMETRICS 99.

[LS00] Ludwig, R. and K. Sklower, The Eifel Retransmission Timer, ACM Computer Communication Review, Vol. 30, No. 3, July 2000.

[MAGMA-PROXY] Fenner, B., He, H., Haberman, B. and H. Sandick, "IGMP/MLD-based Multicast Forwarding ("IGMP/MLD Proxying")", Work in Progress.

[MAGMA-SNOOP] Christensen, M., Kimball, K. and F. Solensky, "Considerations for IGMP and MLD Snooping Switches", Work in Progress.

[MBB00] May, M., Bonald, T. and J-C. Bolot, "Analytic Evaluation of RED Performance", INFOCOM 2000.

[MBDL99] May, M., Bolot, J., Diot, C. and B. Lyles, "Reasons not to deploy RED", Proc. of 7th. International Workshop on Quality of Service (IWQoS'99), June 1999.

[MSMO97] Mathis, M., Semke, J., Mahdavi, J. and T. Ott, "The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm", Computer Communication Review, Vol. 27, number 3, July 1997.

[MYR95] Boden, N., Cohen, D., Felderman, R., Kulawik, A., Seitz, C., et al. MYRINET: A Gigabit per Second Local Area Network, IEEE-Micro, Vol. 15, No.1, February 1995, pp. 29-36.

[PFTK98] Padhye, J., Firoiu, V., Towsley, D. and J. Kurose, "Modeling TCP Throughput: a Simple Model and its Empirical Validation", UMASS CMPSCI Tech Report TR98-008, Feb. 1998.

[RED93] Floyd, S. and V. Jacobson, "Random Early Detection gateways for Congestion Avoidance", IEEE/ACM Transactions in Networking, Vol. 1 No. 4, August 1993. http://www.aciri.org/floyd/papers/red/red.html

[RF95] Romanow, A. and S. Floyd, "Dynamics of TCP Traffic over ATM Networks". IEEE Journal of Selected Areas in Communication, Vol.13 No. 4, May 1995, p. 633-641.

[RFC791] Postel, J., "Internet Protocol", STD 5, RFC 791, September 1981.

[RFC793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.

[RFC768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, August 1980.

[RFC826] Plummer, D.C., "Ethernet Address Resolution Protocol: Or converting network protocol addresses to 48-bit Ethernet address for transmission on Ethernet hardware", STD 37, RFC 826, November 1982.

[RFC1071] Braden, R., Borman, D. and C. Partridge, "Computing the Internet checksum", RFC 1071, September 1988.

[RFC1112] Deering, S., "Host Extensions for IP Multicasting", STD 5, RFC 1112, August 1989.

[RFC1144] Jacobson, V., "Compressing TCP/IP Headers for Low-Speed Serial Links", RFC 1144, February 1990.

[RFC1191] Mogul, J. and S. Deering, "Path MTU Discovery", RFC 1191, November 1990.

[RFC1332] McGregor, C., "The PPP Internet Protocol Control Protocol (IPCP)", RFC 1332, May 1992.

[RFC1435] Knowles, S., "IESG Advice from Experience with Path MTU Discovery", RFC 1435, March 1993.

[RFC1633] Braden, R., Clark, D. and S. Shenker, "Integrated Services in the Internet Architecture: an Overview", RFC 1633, June 1994.

[RFC1661] Simpson, W., "The Point-to-Point Protocol (PPP)", STD 51, RFC 1661, July 1994.

[RFC1662] Simpson, W., Ed., "PPP in HDLC-like Framing", STD 51, RFC 1662, July 1994.

[RFC1750] Eastlake 3rd, D., Crocker, S. and J. Schiller, "Randomness Recommendations for Security", RFC 1750, December 1994.

[RFC1812] Baker, F., Ed., "Requirements for IP Version 4 Routers", RFC 1812, June 1995.

[RFC1939] Myers, J. and M. Rose, "Post Office Protocol - Version 3", STD 53, RFC 1939, May 1996.

[RFC1981] McCann, J., Deering, S. and J. Mogul, "Path MTU Discovery for IP version 6", RFC 1981, August 1996.

[RFC1991] Atkins, D., Stallings, W. and P. Zimmermann, "PGP Message Exchange Formats", RFC 1991, August 1996.

[RFC2018] Mathis, M., Mahdavi, J., Floyd, S. and A. Romanow, "TCP Selective Acknowledgement Options", RFC 2018, October 1996.

[RFC2131] Droms, R., "Dynamic Host Configuration Protocol", RFC 2131, March 1997.

[RFC2205] Braden, R., Ed., Zhang, L., Berson, S., Herzog, S. and S. Jamin, "Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification", RFC 2205, September 1997.

[RFC2208] Mankin, A., Baker, F., Braden, B., Bradner, S., O`Dell, M., Romanow, A., Weinrib, A. and L. Zhang, "Resource ReSerVation Protocol (RSVP) -- Version 1 Applicability Statement Some Guidelines on Deployment", RFC 2208, September 1997.

[RFC2210] Wroclawski, J., "The Use of RSVP with IETF Integrated Services", RFC 2210, September 1997.

[RFC2211] Wroclawski, J., "Specification of the Controlled-Load Network Element Service", RFC 2211, September 1997.

[RFC2212] Shenker, S., Partridge, C. and R. Guerin, "Specification of Guaranteed Quality of Service", RFC 2212, September 1997.

[RFC2246] Dierks, T. and C. Allen, "The TLS Protocol Version 1.0", RFC 2246, January 1999.

[RFC2309] Braden, B., Clark, D., Crowcroft, J., Davie, B., Deering, S., Estrin, D., Floyd, S., Jacobson, V., Minshall, G., Partridge, C., Peterson, L., Ramakrishnan, K., Shenker, S., Wroclawski, J. and L. Zhang, "Recommendations on Queue Management and Congestion Avoidance in the Internet", RFC 2309, April 1998.

[RFC2322] van den Hout, K., Koopal, A. and R. van Mook, "Management of IP numbers by peg-dhcp", RFC 2322, 1 April 1998.

[RFC2328] Moy, J., "OSPF Version 2", STD 54, RFC 2328, April 1998.

[RFC2332] Luciani, J., Katz, D., Piscitello, D., Cole, B. and N. Doraswamy, "NBMA Next Hop Resolution Protocol (NHRP)", RFC 2332, April 1998.

[RFC2364] Gross, G., Kaycee, M., Li, A., Malis, A. and J. Stephens, "PPP Over AAL5", RFC 2364, July 1998.

[RFC2394] Pereira, R., "IP Payload Compression Using DEFLATE", RFC 2394, December 1998.

[RFC2395] Friend, R. and R. Monsour, "IP Payload Compression Using LZS", RFC 2395, December 1998.

[RFC2401] Kent, S. and R. Atkinson, "Security Architecture for the Internet Protocol", RFC 2401, November 1998.

[RFC2406] Kent, S. and R. Atkinson, "IP Encapsulating Security Payload (ESP)", RFC 2406, November 1998.

[RFC2440] Callas, J., Donnerhacke, L., Finney, H. and R. Thayer, "OpenPGP Message Format", RFC 2440, November 1998.

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

[RFC2461] Narten, T., Nordmark, E. and W. Simpson, "Neighbor Discovery for IP Version 6 (IPv6)", RFC 2461, December 1998.

[RFC2474] Nichols, K., Blake, S., Baker, F. and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, December 1998.

[RFC2475] Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z. and W. Weiss, "An Architecture for Differentiated Services", RFC 2475, December 1998.

[RFC2507] Degermark, M., Nordgren, B. and S. Pink, "IP Header Compression", RFC 2507, February 1999.

[RFC2508] Casner, S. and V. Jacobson, "Compressing IP/UDP/RTP Headers for Low-Speed Serial Links", RFC 2508, February 1999.

[RFC2581] Allman, M., Paxson, V. and W. Stevens, "TCP Congestion Control", RFC 2581, April 1999.

[RFC2582] Floyd, S. and T. Henderson, "The NewReno Modification to TCP's Fast Recovery Algorithm", RFC 2582, April 1999.

[RFC2597] Heinanen, J., Baker, F., Weiss, W. and J. Wroclawski, "Assured Forwarding PHB Group", RFC 2597, June 1999.

[RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P. and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

[RFC2630] Housley, R., "Cryptographic Message Syntax", RFC 2630, June 1999.

[RFC2631] Rescorla, E., "Diffie-Hellman Key Agreement Method", RFC 2631, June 1999.

[RFC2632] Ramsdell, B., Ed., "S/MIME Version 3 Certificate Handling", RFC 2632, June 1999.

[RFC2633] Ramsdell, B., "S/MIME Version 3 Message Specification", RFC 2633, June 1999.

[RFC2634] Hoffman, P., "Enhanced Security Services for S/MIME", RFC 2634, June 1999.

[RFC2684] Grossman, D. and J. Heinanen, "Multiprotocol Encapsulation over ATM Adaptation Layer 5", RFC 2684, September 1999.

[RFC2686] Bormann, C., "The Multi-Class Extension to Multi-Link PPP", RFC 2686, September 1999.

[RFC2687] Bormann, C., "PPP in a Real-time Oriented HDLC-like Framing", RFC 2687, September 1999.

[RFC2689] Bormann, C., "Providing Integrated Services over Low-bitrate Links", RFC 2689, September 1999.

[RFC2710] Deering, S., Fenner, W. and B. Haberman, "Multicast Listener Discovery (MLD) for IPv6", RFC 2710, October 1999.

[RFC2784] Farinacci, D., Li, T., Hanks, S., Meyer, D. and P. Traina, "Generic Routing Encapsulation (GRE)", RFC 2784, March 2000.

[RFC2865] Rigney, C., Willens, S., Rubens, A. and W. Simpson, "Remote Authentication Dial In User Service (RADIUS)", RFC 2865, June 2000.

[RFC2914] Floyd, S., "Congestion Control Principles", BCP 41, RFC 2914, September 2000.

[RFC2923] Lahey, K., "TCP Problems with Path MTU Discovery", RFC 2923, September 2000.

[RFC2988] Paxson, V. and M. Allman, "Computing TCP's Retransmission Timer", RFC 2988, November 2000.

[RFC2990] Huston, G., "Next Steps for the IP QoS Architecture", RFC 2990, November 2000.

[RFC3048] Whetten, B., Vicisano, L., Kermode, R., Handley, M., Floyd, S. and M. Luby, "Reliable Multicast Transport Building Blocks for One-to-Many Bulk-Data Transfer", RFC 3048, January 2001.

[RFC3095] Bormann, C., Ed., Burmeister, C., Degermark, M., Fukushima, H., Hannu, H., Jonsson, L-E., Hakenberg, R., Koren, T., Le, K., Liu, Z., Martensson, A., Miyazaki, A., Svanbro, K., Wiebke, T., Yoshimura, T. and H. Zheng, "RObust Header Compression (ROHC): Framework and four profiles: RTP, UDP, ESP, and uncompressed", RFC 3095, July 2001.

[RFC3096] Degermark, M., Ed., "Requirements for robust IP/UDP/RTP header compression", RFC 3096, July 2001.

[RFC3150] Dawkins, S., Montenegro, G., Kojo, M. and V. Magret, "End-to-end Performance Implications of Slow Links", BCP 48, RFC 3150, July 2001.

[RFC3155] Dawkins, S., Montenegro, G., Kojo, M., Magret, V. and N. Vaidya, "End-to-end Performance Implications of Links with Errors", BCP 50, RFC 3155, August 2001.

[RFC3168] Ramakrishnan, K., Floyd, S. and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, September 2001.

[RFC3173] Shacham, A., Monsour, B., Pereira, R. and M. Thomas, "IP Payload Compression Protocol (IPComp)", RFC 3173, September 2001.

[RFC3246] Davie, B., Charny, A., Bennet, J.C.R., Benson, K., Le Boudec, J.Y., Courtney, W., Davari, S., Firoiu, V. and D. Stiliadis, "An Expedited Forwarding PHB (Per-Hop Behavior)", RFC 3246, March 2002.

[RFC3248] Armitage, G., Carpenter, B., Casati, A., Crowcroft, J., Halpern, J., Kumar, B. and J. Schnizlein, "A Delay Bound alternative revision of RFC 2598", RFC 3248, March 2002.

[RFC3344] Perkins, C., Ed., "IP Mobility Support for IPv4", RFC 3344, August 2002.

[RFC3366] Fairhurst, G. and L. Wood, "Advice to link designers on link Automatic Repeat reQuest (ARQ)", BCP 62, RFC 3366, August 2002.

[RFC3376] Cain, B., Deering, S., Kouvelas, I., Fenner, B. and A. Thyagarajan, "Internet Group Management Protocol, Version 3", RFC 3376, October 2002.

[RFC3449] Balakrishnan, H., Padmanabhan, V., Fairhurst, G. and M. Sooriyabandara, "TCP Performance Implications of Network Path Asymmetry", BCP 69, RFC 3449, December 2002.

[RFC3450] Luby, M., Gemmell, J., Vicisano, L., Rizzo, L. and J. Crowcroft, "Asynchronous Layered Coding (ALC) Protocol Instantiation", RFC 3450, December 2002.

[RFC3451] Luby, M., Gemmell, J., Vicisano, L., Rizzo, L., Handley, M. and J. Crowcroft, "Layered Coding Transport (LCT) Building Block", RFC 3451, December 2002.

[RFC3452] Luby, M., Vicisano, L., Gemmell, J., Rizzo, L., Handley, M. and J. Crowcroft, "Forward Error Correction (FEC) Building Block", RFC 3452, December 2002.

[RFC3453] Luby, M., Vicisano, L., Gemmell, J., Rizzo, L., Handley, M. and J. Crowcroft, "The Use of Forward Error Correction (FEC) in Reliable Multicast", RFC 3453, December 2002.

[RFC3488] Wu, I. and T. Eckert, "Cisco Systems Router-port Group Management Protocol (RGMP)", RFC 3488, February 2003.

[RFC3501] Crispin, M., "INTERNET MESSAGE ACCESS PROTOCOL - VERSION 4rev1", RFC 3501, March 2003.

[RFC3828] Larzon, L-A., Degermark, M., Pink, S., Jonsson, L-E., Ed. and G. Fairhurst, Ed., "The User Datagram Protocol (UDP)-Lite Protocol", RFC 3828, June 2004.

[Schneier95] Schneier, B., Applied Cryptography: Protocols, Algorithms and Source Code in C (John Wiley and Sons, October 1995).

[Schneier00] Schneier, B., Secrets and Lies: Digital Security in a Networked World (John Wiley and Sons, August 2000).

[SP2000] Stone, J. and C. Partridge, "When the CRC and TCP Checksum Disagree", ACM SIGCOMM, September 2000. http://www.acm.org/sigcomm/sigcomm2000/conf/paper/sigcomm2000-9-1.pdf

[SRC81] Saltzer, J., Reed D. and D. Clark, "End-to-End Arguments in System Design". Second International Conference on Distributed Computing Systems (April, 1981) pages 509-512. Published with minor changes in ACM Transactions in Computer Systems 2, 4, November, 1984, pages 277-288. Reprinted in Craig Partridge, editor Innovations in internetworking. Artech House, Norwood, MA, 1988, pages 195-206. ISBN 0-89006-337-0.

[SSL2] Hickman, K., "The SSL Protocol", Netscape Communications Corp., Feb 9, 1995.

[SSL3] Frier, A., Karlton, P. and P. Kocher, "The SSL 3.0 Protocol", Netscape Communications Corp., Nov 18, 1996.

[TCPF98] Lin, D. and H.T. Kung, "TCP Fast Recovery Strategies: Analysis and Improvements", IEEE Infocom, March 1998. http://www.eecs.harvard.edu/networking/papers/infocom-tcp-final-198.pdf

[WFBA2000] Wagner, D., Foster, J., Brewer, E. and A. Aiken, "A First Step Toward Automated Detection of Buffer Overrun Vulnerabilities", Proceedings of NDSS2000. http://www.isoc.org/isoc/conferences/ndss/2000/proceedings/039.pdf

[Wilbur89] Wilbur, Steve R., Jon Crowcroft, and Yuko Murayama. "MAC layer Security Measures in Local Area Networks", Local Area Network Security, Workshop LANSEC '89 Proceedings, Springer-Verlag, April 1989, pp. 53-64.

21. Contributors' Addresses

Aaron Falk
USC/Information Sciences Institute
4676 Admiralty Way
Marina Del Rey, CA 90292

Phone: 310-448-9327
EMail: falk@isi.edu

Saverio Mascolo
Dipartimento di Elettrotecnica ed Elettronica,
Politecnico di Bari Via Orabona 4, 70125 Bari, Italy

Phone: +39 080 596 3621
EMail: mascolo@poliba.it
URL: http://www-dee.poliba.it/dee-web/Personale/mascolo.html

Marie-Jose Montpetit
MJMontpetit.com

EMail: marie@mjmontpetit.com

22. Authors' Addresses

Phil Karn, Editor
Qualcomm 5775 Morehouse Drive
San Diego CA 92121

Phone: 858 587 1121
EMail: karn@qualcomm.com

Carsten Bormann
Universitaet Bremen TZI
Postfach 330440
D-28334 Bremen, Germany

Phone: +49 421 218 7024
Fax: +49 421 218 7000
EMail: cabo@tzi.org

Godred (Gorry) Fairhurst
Department of Engineering, University of Aberdeen,
Aberdeen, AB24 3UE, United Kingdom

EMail: gorry@erg.abdn.ac.uk
URL: http://www.erg.abdn.ac.uk/users/gorry

Dan Grossman
Motorola, Inc.
111 Locke Drive
Marlboro, MA 01752

EMail: Dan.Grossman@motorola.com

Reiner Ludwig
Ericsson Research
Ericsson Allee
1 52134 Herzogenrath, Germany

Phone: +49 2407 575 719
EMail: Reiner.Ludwig@ericsson.com

Jamshid Mahdavi
Novell, Inc.

EMail: jmahdavi@earthlink.net

Gabriel Montenegro
Sun Microsystems Laboratories, Europe
180, Avenue de l'Europe
38334 Saint Ismier CEDEX
France

EMail: gab@sun.com

Joe Touch
USC/Information Sciences Institute
4676 Admiralty Way
Marina del Rey CA 90292

Phone: 310 448 9151
EMail: touch@isi.edu
URL: http://www.isi.edu/touch

Lloyd Wood
Cisco Systems
9 New Square Park, Bedfont Lakes
Feltham TW14 8HA
United Kingdom

Phone: +44 (0)20 8824 4236
EMail: lwood@cisco.com
URL: http://www.ee.surrey.ac.uk/Personal/L.Wood/

23. Full Copyright Statement

Intellectual Property

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.