A Roadmap for Transmission Control Protocol (TCP) Specification Documents

Abstract

This document contains a roadmap to the Request for Comments (RFC) documents relating to the Internet’s Transmission Control Protocol (TCP). This roadmap provides a brief summary of the documents defining TCP and various TCP extensions that have accumulated in the RFC series. This serves as a guide and quick reference for both TCP implementers and other parties who desire information contained in the TCP-related RFCs.

This document obsoletes RFC 4614.

Status of This Memo

This document is not an Internet Standards Track specification; it is published for informational purposes.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Not all documents approved by the IESG are a candidate for any level of Internet Standard; see Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc7414.
Table of Contents

1. Introduction ....................................................4
2. Core Functionality ..............................................6
3. Strongly Encouraged Enhancements ...............................8
   3.1. Fundamental Changes .......................................9
   3.2. Congestion Control Extensions .............................10
   3.3. Loss Recovery Extensions ................................11
   3.4. Detection and Prevention of Spurious Retransmissions ......13
   3.5. Path MTU Discovery .......................................14
   3.6. Header Compression ......................................15
   3.7. Defending Spoofing and Flooding Attacks ..................15
4. Experimental Extensions ........................................17
   4.1. Architectural Guidelines ..................................18
   4.2. Fundamental Changes .......................................18
   4.3. Congestion Control Extensions .............................19
   4.4. Loss Recovery Extensions ................................20
   4.5. Detection and Prevention of Spurious Retransmissions ......21
   4.6. TCP Timeouts ............................................22
   4.7. Multipath TCP ............................................22
5. TCP Parameters at IANA .........................................23
6. Historic and Undeployed Extensions .............................24
7. Support Documents ..............................................27
   7.1. Foundational Works .......................................27
   7.2. Architectural Guidelines ..................................29
   7.3. Difficult Network Environments ............................30
   7.4. Guidance for Developing, Analyzing, and Evaluating TCP ...33
   7.5. Implementation Advice ....................................34
   7.6. Tools and Tutorials ......................................36
   7.7. MIB Modules .............................................37
   7.8. Case Studies .............................................39
8. Undocumented TCP Features ......................................40
9. Security Considerations ........................................41
10. References ..................................................42
    10.1. Normative References ...................................42
    10.2. Informative References ................................53
Acknowledgments ....................................................56
Authors’ Addresses ................................................57
1. Introduction

A correct and efficient implementation of the Transmission Control Protocol (TCP) is a critical part of the software of most Internet hosts. As TCP has evolved over the years, many distinct documents have become part of the accepted standard for TCP. At the same time, a large number of experimental modifications to TCP have also been published in the RFC series, along with informational notes, case studies, and other advice.

As an introduction to newcomers and an attempt to organize the plethora of information for old hands, this document contains a roadmap to the TCP-related RFCs. It provides a brief summary of the RFC documents that define TCP. This should provide guidance to implementers on the relevance and significance of the standards-track extensions, informational notes, and best current practices that relate to TCP.

This document is not an update of RFC 1122 [RFC1122] and is not a rigorous standard for what needs to be implemented in TCP. This document is merely an informational roadmap that captures, organizes, and summarizes most of the RFC documents that a TCP implementer, experimenter, or student should be aware of. Particular comments or broad categorizations that this document makes about individual mechanisms and behaviors are not to be taken as definitive, nor should the content of this document alone influence implementation decisions.

This roadmap includes a brief description of the contents of each TCP-related RFC. In some cases, we simply supply the abstract or a key summary sentence from the text as a terse description. In addition, a letter code after an RFC number indicates its category in the RFC series (see BCP 9 [RFC2026] for explanation of these categories):

S - Standards Track (Proposed Standard, Draft Standard, or Internet Standard)

E - Experimental

I - Informational

H - Historic

B - Best Current Practice

U - Unknown (not formally defined)
Note that the category of an RFC does not necessarily reflect its current relevance. For instance, RFC 5681 [RFC5681] is considered part of the required core functionality of TCP, although the RFC is only a Draft Standard. Similarly, some Informational RFCs contain significant technical proposals for changing TCP.

Finally, if an error in the technical content has been found after publication of an RFC (at the time of this writing), this fact is indicated by the term "(Errata)" in the headline of the RFC’s description. The contents of the errata can be found through the RFC Errata page [Errata].

This roadmap is divided into three main sections. Section 2 lists the RFCs that describe absolutely required TCP behaviors for proper functioning and interoperability. Further RFCs that describe strongly encouraged, but nonessential, behaviors are listed in Section 3. Experimental extensions that are not yet standard practices, but that potentially could be in the future, are described in Section 4.

The reader will probably notice that these three sections are broadly equivalent to MUST/SHOULD/MAY specifications (per RFC 2119 [RFC2119]), and although the authors support this intuition, this document is merely descriptive; it does not represent a binding Standards Track position. Individual implementers still need to examine the Standards Track RFCs themselves to evaluate specific requirement levels.

Section 5 describes both the procedures that the Internet Assigned Numbers Authority (IANA) uses and an RFC author should follow when new TCP parameters are requested and finally assigned.

A small number of older experimental extensions that have not been widely implemented, deployed, and used are noted in Section 6. Many other supporting documents that are relevant to the development, implementation, and deployment of TCP are described in Section 7.

A small number of fairly ubiquitous important implementation practices that are not currently documented in the RFC series are listed in Section 8.

Within each section, RFCs are listed in the chronological order of their publication dates.
2. Core Functionality

A small number of documents compose the core specification of TCP. These define the required core functionalities of TCP’s header parsing, state machine, congestion control, and retransmission timeout computation. These base specifications must be correctly followed for interoperability.

(Errata)

This is the fundamental TCP specification document [RFC793]. Written by Jon Postel as part of the Internet protocol suite’s core, it describes the TCP packet format, the TCP state machine and event processing, and TCP’s semantics for data transmission, reliability, flow control, multiplexing, and acknowledgment.

Section 3.6 of RFC 793, describing TCP’s handling of the IP precedence and security compartment, is mostly irrelevant today. RFC 2873 (discussed later in Section 2 below) changed the IP precedence handling, and the security compartment portion of the API is no longer implemented or used. In addition, RFC 793 did not describe any congestion control mechanism. Otherwise, however, the majority of this document still accurately describes modern TCPS. RFC 793 is the last of a series of developmental TCP specifications, starting in the Internet Experimental Notes (IENs) and continuing in the RFC series.

RFC 1122 S: "Requirements for Internet Hosts - Communication Layers"
(October 1989)

This document [RFC1122] updates and clarifies RFC 793 (see above in Section 2), fixing some specification bugs and oversights. It also explains some features such as keep-alives and Karn’s and Jacobson’s RTO estimation algorithms [KP87][Jac88][JK92]. ICMP interactions are mentioned, and some tips are given for efficient implementation. RFC 1122 is an Applicability Statement, listing the various features that MUST, SHOULD, MAY, SHOULD NOT, and MUST NOT be present in standards-conforming TCP implementations. Unlike a purely informational roadmap, this Applicability Statement is a standards document and gives formal rules for implementation.
RFC 2460 S: "Internet Protocol, Version 6 (IPv6) Specification"  
(December 1998)  (Errata)

This document [RFC2460] is of relevance to TCP because it defines how the pseudo-header for TCP’s checksum computation is derived when 128-bit IPv6 addresses are used instead of 32-bit IPv4 addresses. Additionally, RFC 2675 (see Section 3.1 of this document) describes TCP changes required to support IPv6 jumbograms.

RFC 2873 S: "TCP Processing of the IPv4 Precedence Field" (June 2000)  
(Errata)

This document [RFC2873] removes from the TCP specification all processing of the precedence bits of the TOS byte of the IP header. This resolves a conflict over the use of these bits between RFC 793 (see above in Section 2) and Differentiated Services [RFC2474].

RFC 5681 S: "TCP Congestion Control" (August 2009)

Although RFC 793 (see above in Section 2) did not contain any congestion control mechanisms, today congestion control is a required component of TCP implementations. This document [RFC5681] defines congestion avoidance and control mechanism for TCP, based on Van Jacobson’s 1988 SIGCOMM paper [Jac88].

A number of behaviors that together constitute what the community refers to as "Reno TCP" is described in RFC 5681. The name "Reno" comes from the Net/2 release of the 4.3 BSD operating system. This is generally regarded as the least common denominator among TCP flavors currently found running on Internet hosts. Reno TCP includes the congestion control features of slow start, congestion avoidance, fast retransmit, and fast recovery.

RFC 5681 details the currently accepted congestion control mechanism, while RFC 1122, (see above in Section 2) mandates that such a congestion control mechanism must be implemented. RFC 5681 differs slightly from the other documents listed in this section, as it does not affect the ability of two TCP endpoints to communicate; however, congestion control remains a critical component of any widely deployed TCP implementation and is required for the avoidance of congestion collapse and to ensure fairness among competing flows.
RFCs 2001 and 2581 are the conceptual precursors of RFC 5681. The most important changes relative to RFC 2581 are:

(a) The initial window requirements were changed to allow larger Initial Windows as standardized in [RFC3390] (see Section 3.2 of this document).
(b) During slow start and congestion avoidance, the usage of Appropriate Byte Counting [RFC3465] (see Section 3.2 of this document) is explicitly recommended.
(c) The use of Limited Transmit [RFC3042] (see Section 3.3 of this document) is now recommended.

RFC 6093 S: "On the Implementation of the TCP Urgent Mechanism" (January 2011)

This document [RFC6093] analyzes how current TCP stacks process TCP urgent indications, and how the behavior of widely deployed middleboxes affects the urgent indications processing. The document updates the relevant specifications such that it accommodates current practice in processing TCP urgent indications. Finally, the document raises awareness about the reliability of TCP urgent indications in the Internet, and recommends against the use of urgent mechanism.

RFC 6298 S: "Computing TCP’s Retransmission Timer" (June 2011)

Abstract of RFC 6298 [RFC6298]: "This document defines the standard algorithm that Transmission Control Protocol (TCP) senders are required to use to compute and manage their retransmission timer. It expands on the discussion in Section 4.2.3.1 of RFC 1122 and upgrades the requirement of supporting the algorithm from a SHOULD to a MUST." RFC 6298 updates RFC 2988 by changing the initial RTO from 3s to 1s.

RFC 6691 I: "TCP Options and Maximum Segment Size (MSS)" (July 2012)

This document [RFC6691] clarifies what value to use with the TCP Maximum Segment Size (MSS) option when IP and TCP options are in use.

3. Strongly Encouraged Enhancements

This section describes recommended TCP modifications that improve performance and security. Section 3.1 represents fundamental changes to the protocol. Sections 3.2 and 3.3 list improvements over the congestion control and loss recovery mechanisms as specified in RFC 5681 (see Section 2). Section 3.4 describes algorithms that allow a TCP sender to detect whether it has entered loss recovery spuriously.
Section 3.5 comprises Path MTU Discovery mechanisms. Schemes for TCP/IP header compression are listed in Section 3.6. Finally, Section 3.7 deals with the problem of preventing acceptance of forged segments and flooding attacks.

3.1. Fundamental Changes

RFCs 2675 and 7323 represent fundamental changes to TCP by redefining how parts of the basic TCP header and options are interpreted. RFC 7323 defines the Window Scale option, which reinterprets the advertised receive window. RFC 2675 specifies that MSS option and urgent pointer fields with a value of 65,535 are to be treated specially.

RFC 2675 S: "IPv6 Jumbograms" (August 1999) (Errata)

IPv6 supports longer datagrams than were allowed in IPv4. These are known as jumbograms, and use with TCP has necessitated changes to the handling of TCP's MSS and Urgent fields (both 16 bits). This document [RFC2675] explains those changes. Although it describes changes to basic header semantics, these changes should only affect the use of very large segments, such as IPv6 jumbograms, which are currently rarely used in the general Internet.

Supporting the behavior described in this document does not affect interoperability with other TCP implementations when IPv4 or non-jumbogram IPv6 is used. This document states that jumbograms are to only be used when it can be guaranteed that all receiving nodes, including each router in the end-to-end path, will support jumbograms. If even a single node that does not support jumbograms is attached to a local network, then no host on that network may use jumbograms. This explains why jumbogram use has been rare, and why this document is considered a performance optimization and not part of TCP over IPv6's basic functionality.

RFC 7323 S: "TCP Extensions for High Performance" (September 2014)

This document [RFC7323] defines TCP extensions for window scaling, timestamps, and protection against wrapped sequence numbers, for efficient and safe operation over paths with large bandwidth-delay products. These extensions are commonly found in currently used systems. The predecessor of this document, RFC 1323, was published in 1992, and is deployed in most TCP implementations. This document includes fixes and clarifications based on the gained deployment experience. One specific issued addressed in
this specification is a recommendation how to modify the algorithm for estimating the mean RTT when timestamps are used. RFCs 1072, 1185, and 1323 are the conceptual precursors of RFC 7323.

3.2. Congestion Control Extensions

Two of the most important aspects of TCP are its congestion control and loss recovery features. TCP treats lost packets as indicating congestion-related loss and cannot distinguish between congestion-related loss and loss due to transmission errors. Even when ECN is in use, there is a rather intimate coupling between congestion control and loss recovery mechanisms. There are several extensions to both features, and more often than not, a particular extension applies to both. In these two subsections, we group enhancements to TCP’s congestion control, while the next subsection focus on TCP’s loss recovery.

RFC 3168 S: "The Addition of Explicit Congestion Notification (ECN) to IP" (September 2001)

This document [RFC3168] defines a means for end hosts to detect congestion before congested routers are forced to discard packets. Although congestion notification takes place at the IP level, ECN requires support at the transport level (e.g., in TCP) to echo the bits and adapt the sending rate. This document updates RFC 793 (see Section 2 of this document) to define two previously unused flag bits in the TCP header for ECN support. RFC 3540 (see Section 4.3 of this document) provides a supplementary (experimental) means for more secure use of ECN, and RFC 2884 (see Section 7.8 of this document) provides some sample results from using ECN.

RFC 3390 S: "Increasing TCP’s Initial Window" (October 2002)

This document [RFC3390] specifies an increase in the permitted initial window for TCP from one segment to three or four segments during the slow start phase, depending on the segment size.

RFC 3465 E: "TCP Congestion Control with Appropriate Byte Counting (ABC)" (February 2003)

This document [RFC3465] suggests that congestion control use the number of bytes acknowledged instead of the number of acknowledgments received. This change improves the performance of TCP in situations where there is no one-to-one relationship between data segments and acknowledgments (e.g., delayed ACKs or ACK loss) and closes a security hole TCP receivers can use to
induce the sender into increasing the sending rate too rapidly
(ACK-division [SCWA99] [RFC3449]). ABC is recommended by RFC 5681
(see Section 2 of this document).

RFC 6633 S: "Deprecation of ICMP Source Quench Messages" (May 2012)

This document [RFC6633] formally deprecates the use of ICMP Source
Quench messages by transport protocols and recommends against the
implementation of [RFC1016].

3.3. Loss Recovery Extensions

For the typical implementation of the TCP fast recovery algorithm
described in RFC 5681 (see Section 2 of this document), a TCP sender
only retransmits a segment after a retransmit timeout has occurred,
or after three duplicate ACKs have arrived triggering the fast
retransmit. A single RTO might result in the retransmission of
several segments, while the fast retransmit algorithm in RFC 5681
leads only to a single retransmission. Hence, multiple losses from a
single window of data can lead to a performance degradation.

Documents listed in this section aim to improve the overall
performance of TCP’s standard loss recovery algorithms. In
particular, some of them allow TCP senders to recover more
effectively when multiple segments are lost from a single flight of
data.

RFC 2018 S: "TCP Selective Acknowledgment Options" (October 1996)
(Errata)

When more than one packet is lost during one RTT, TCP may
experience poor performance since a TCP sender can only learn
about a single lost packet per RTT from cumulative
acknowledgments. This document [RFC2018] defines the basic
selective acknowledgment (SACK) mechanism for TCP, which can help
to overcome these limitations. The receiving TCP returns SACK
blocks to inform the sender which data has been received. The
sender can then retransmit only the missing data segments.

RFC 3042 S: "Enhancing TCP’s Loss Recovery Using Limited Transmit"
(January 2001)

Abstract of RFC 3042 [RFC3042]: "This document proposes a new
Transmission Control Protocol (TCP) mechanism that can be used to
more effectively recover lost segments when a connection’s
congestion window is small, or when a large number of segments are
lost in a single transmission window." This algorithm described
in RFC 3042 is called "Limited Transmit". Tests from 2004 showed
RFC 7414                       TCP Roadmap                 February 2015

that Limited Transmit was deployed in roughly one third of the web servers tested [MAF04]. Limited Transmit is recommended by RFC 5681 (see Section 2 of this document).

RFC 6582 S: "The NewReno Modification to TCP’s Fast Recovery Algorithm" (April 2012)

This document [RFC6582] specifies a modification to the standard Reno fast recovery algorithm, whereby a TCP sender can use partial acknowledgments to make inferences determining the next segment to send in situations where SACK would be helpful but isn’t available. Although it is only a slight modification, the NewReno behavior can make a significant difference in performance when multiple segments are lost from a single window of data.

RFCs 2582 and 3782 are the conceptual precursors of RFC 6582. The main change in RFC 3782 relative to RFC 2582 was to specify the Careful variant of NewReno’s Fast Retransmit and Fast Recovery algorithms and advance those two algorithms from Experimental to Standards Track status. The main change in RFC 6582 relative to RFC 3782 was to solve a performance degradation that could occur if FlightSize on Full ACK reception is zero.

RFC 6675 S: "A Conservative Loss Recovery Algorithm Based on Selective Acknowledgment (SACK) for TCP" (August 2012)

This document [RFC6675] describes a conservative loss recovery algorithm for TCP that is based on the use of the selective acknowledgment (SACK) TCP option [RFC2018] (see above in Section 3.3). The algorithm conforms to the spirit of the congestion control specification in RFC 5681 (see Section 2 of this document), but allows TCP senders to recover more effectively when multiple segments are lost from a single flight of data.

RFC 6675 is a revision of RFC 3517 to address several situations that are not handled explicitly before. In particular,

(a) it improves the loss detection in the event that the sender has outstanding segments that are smaller than Sender Maximum Segment Size (SMSS).
(b) it modifies the definition of a "duplicate acknowledgment" to utilize the SACK information in detecting loss.
(c) it maintains the ACK clock under certain circumstances involving loss at the end of the window.
3.4. Detection and Prevention of Spurious Retransmissions

Spurious retransmission timeouts are harmful to TCP performance and multiple algorithms have been defined for detecting when spurious retransmissions have occurred, but they respond differently with regard to their manners of recovering performance. The IETF defined multiple algorithms because there are trade-offs in whether or not certain TCP options need to be implemented and concerns about IPR status. The Standards Track RFCs in this section are closely related to the Experimental RFCs in Section 4.5 also addressing this topic.

RFC 2883 S: "An Extension to the Selective Acknowledgement (SACK) Option for TCP" (July 2000)

This document [RFC2883] extends RFC 2018 (see Section 3.3 of this document). It enables use of the SACK option to acknowledge duplicate packets. With this extension, called DSACK, the sender is able to infer the order of packets received at the receiver and, therefore, to infer when it has unnecessarily retransmitted a packet. A TCP sender could then use this information to detect spurious retransmissions (see [RFC3708]).

RFC 4015 S: "The Eifel Response Algorithm for TCP" (February 2005)

This document [RFC4015] describes the response portion of the Eifel algorithm, which can be used in conjunction with one of several methods of detecting when loss recovery has been spuriously entered, such as the Eifel detection algorithm in RFC 3522 (see Section 4.5), the algorithm in RFC 3708 (see Section 4.5 of this document), or F-RTO in RFC 5682 (see below in Section 3.4).

Abstract of RFC 4015 [RFC4015]: "Based on an appropriate detection algorithm, the Eifel response algorithm provides a way for a TCP sender to respond to a detected spurious timeout. It adapts the retransmission timer to avoid further spurious timeouts and (depending on the detection algorithm) can avoid the often unnecessary go-back-N retransmits that would otherwise be sent. In addition, the Eifel response algorithm restores the congestion control state in such a way that packet bursts are avoided."

RFC 5682 S: "Forward RTO-Recovery (F-RTO): An Algorithm for Detecting Spurious Retransmission Timeouts with TCP" (September 2009)

The F-RTO detection algorithm [RFC5682], originally described in RFC 4138, provides an option for inferring spurious retransmission timeouts. Unlike some similar detection methods (e.g., RFCs 3522
and 3708, both listed in Section 4.5 of this document), F-RTO does not rely on the use of any TCP options. The basic idea is to send previously unsent data after the first retransmission after a RTO. If the ACKs advance the window, the RTO may be declared spurious.

3.5. Path MTU Discovery

The MTUs supported by different links and tunnels within the Internet can vary widely. Fragmentation of packets larger than the supported MTU on a hop is undesirable. As TCP is the segmentation layer for dividing an application's byte stream into IP packet payloads, TCP implementations generally include Path MTU Discovery (PMTUD) mechanisms in order to maximize the size of segments they send, without causing fragmentation within the network. Some algorithms may utilize signaling from routers on the path to determine that the MTU on some part of the path has been exceeded.

RFC 1191 S: "Path MTU Discovery" (November 1990)

Abstract of RFC 1191 [RFC1191]: "This memo describes a technique for dynamically discovering the maximum transmission unit (MTU) of an arbitrary internet path. It specifies a small change to the way routers generate one type of ICMP message. For a path that passes through a router that has not been so changed, this technique might not discover the correct Path MTU, but it will always choose a Path MTU as accurate as, and in many cases more accurate than, the Path MTU that would be chosen by current practice."


Abstract of RFC 1981 [RFC1981]: "This document describes Path MTU Discovery for IP version 6. It is largely derived from RFC 1191, which describes Path MTU Discovery for IP version 4."

RFC 4821 S: "Packetization Layer Path MTU Discovery" (March 2007)

Abstract of RFC 4821 [RFC4821]: "This document describes a robust method for Path MTU Discovery (PMTUD) that relies on TCP or some other Packetization Layer to probe an Internet path with progressively larger packets. This method is described as an extension to RFC 1191 and RFC 1981, which specify ICMP-based Path MTU Discovery for IP versions 4 and 6, respectively."
3.6. Header Compression

Especially in streaming applications, the overhead of TCP/IP headers could correspond to more than 50% of the total amount of data sent. Such large overheads may be tolerable in wired LANs where capacity is often not an issue, but are excessive for WANs and wireless systems where bandwidth is scarce. Header compression schemes for TCP/IP like RObust Header Compression (ROHC) can significantly compress this overhead. It performs well over links with significant error rates and long round-trip times.

RFC 1144 S: "Compressing TCP/IP Headers for Low-Speed Serial Links" (February 1990)

This document [RFC1144] describes a method for compressing the headers of TCP/IP datagrams to improve performance over low-speed serial links. The method described in this document is limited in its handling of TCP options and cannot compress the headers of SYNs and FINs.

RFC 6846 S: "RObust Header Compression (ROHC): A Profile for TCP/IP (ROHC-TCP)" (January 2013)

From the Abstract of RFC 6846 [RFC6846]: "This document specifies a RObust Header Compression (ROHC) profile for compression of TCP/IP packets. The profile, called ROHC-TCP, provides efficient and robust compression of TCP headers, including frequently used TCP options such as selective acknowledgments (SACKs) and Timestamps." RFC 6846 is the successor of RFC 4996. It fixes a technical issue with the SACK compression and clarifies other compression methods used.

3.7. Defending Spoofing and Flooding Attacks

By default, TCP lacks any cryptographic structures to differentiate legitimate segments from those spoofed from malicious hosts. Spoofing valid segments requires correctly guessing a number of fields. The documents in this subsection describe ways to make that guessing harder or to prevent it from being able to affect a connection negatively.
RFC 4953 I: "Defending TCP Against Spoofing Attacks" (July 2007)

This document [RFC4953] discusses the recently increased vulnerability of long-lived TCP connections, such as BGP connections, to reset (send RST) spoofing attacks. The document analyzes the vulnerability, discussing proposed solutions at the transport level and their inherent challenges, as well as existing network level solutions and the feasibility of their deployment.

RFC 5461 I: "TCP’s Reaction to Soft Errors" (February 2009)

This document [RFC5461] describes a nonstandard but widely implemented modification to TCP’s handling of ICMP soft error messages that rejects pending connection-requests when such error messages are received. This behavior reduces the likelihood of long delays between connection-establishment attempts that may arise in some scenarios.

RFC 4987 I: "TCP SYN Flooding Attacks and Common Mitigations" (August 2007)

This document [RFC4987] describes the well-known TCP SYN flooding attack. It analyzes and discusses various countermeasures against these attacks, including their use and trade-offs.

RFC 5925 S: "The TCP Authentication Option" (June 2010)

This document [RFC5925] describes the TCP Authentication Option (TCP-AO), which is used to authenticate TCP segments. TCP-AO obsoletes the TCP MD5 Signature option of RFC 2385. It supports the use of stronger hash functions, protects against replays for long-lived TCP connections (as used, e.g., in BGP and LDP), coordinates key exchanges between endpoints, and provides a more explicit recommendation for external key management. Cryptographic algorithms for TCP-AO are defined in [RFC5926] (see below in Section 3.7).

RFC 5926 S: "Cryptographic Algorithms for the TCP Authentication Option (TCP-AO)" (June 2010)

This document [RFC5926] specifies the algorithms and attributes that can be used in TCP Authentication Option’s (TCP-AO) [RFC5925] (see above in Section 3.7) current manual keying mechanism and provides the interface for future message authentication codes (MACs).
RFC 5927 I: "ICMP Attacks against TCP" (July 2010)

Abstract of RFC 5927 [RFC5927]: "This document discusses the use of the Internet Control Message Protocol (ICMP) to perform a variety of attacks against the Transmission Control Protocol (TCP). Additionally, this document describes a number of widely implemented modifications to TCP's handling of ICMP error messages that help to mitigate these issues."

RFC 5961 S: "Improving TCP’s Robustness to Blind In-Window Attacks" (August 2010)

This document [RFC5961] describes minor modifications to how TCP handles inbound segments. This renders TCP connections, especially long-lived connections such as H-323 or BGP, less vulnerable to spoofed packet injection attacks where the 4-tuple (the source and destination IP addresses and the source and destination ports) has been guessed.

RFC 6528 S: "Defending against Sequence Number Attacks" (February 2012)

Abstract of RFC 6528 [RFC6528]: "This document specifies an algorithm for the generation of TCP Initial Sequence Numbers (ISNs), such that the chances of an off-path attacker guessing the sequence numbers in use by a target connection are reduced. This document revises (and formally obsoletes) RFC 1948, and takes the ISN generation algorithm originally proposed in that document to Standards Track, formally updating RFC 793"

4. Experimental Extensions

The RFCs in this section are either Experimental and may become Proposed Standards in the future or are Proposed Standards (or Informational), but can be considered experimental due to lack of wide deployment. At least part of the reason that they are still experimental is to gain more wide-scale experience with them before a standards track decision is made.

If the Experimental RFC is a proposal for a new protocol capability or service, i.e., it requires a new TCP option code point, the implementation and experimentation should follow [RFC6994] (see Section 5 of this document), which describes how the experimental TCP option code points can concurrently support multiple TCP extensions.

By their publication as Experimental RFCs, it is hoped that the community of TCP researchers will analyze and test the contents of these RFCs. Although experimentation is encouraged, there is not yet
formal consensus that these are fully logical and safe behaviors. Wide-scale deployment of implementations that use these features should be well thought out in terms of consequences.

4.1. Architectural Guidelines

As multiple flows may share the same paths, sections of paths, or other resources, the TCP implementation may benefit from sharing information across TCP connections or other flows. Some experimental proposals have been documented and some implementations have included the concepts.

RFC 2140 I: "TCP Control Block Interdependence" (April 1997)

This document [RFC2140] suggests how TCP connections between the same endpoints might share information, such as their congestion control state. To some degree, this is done in practice by a few operating systems; for example, Linux currently has a destination cache. Although this RFC is technically Informational, the concepts it describes are in experimental use, so we include it in this section.

RFC 3124 S: "The Congestion Manager" (June 2001)

This document [RFC3124] is a related proposal to RFC 2140 (see above in Section 4.1). The idea behind the Congestion Manager, moving congestion control outside of individual TCP connections, represents a modification to the core of TCP, which supports sharing information among TCP connections. Although a Proposed Standard, some pieces of the Congestion Manager support architecture have not been specified yet, and it has not achieved use or implementation beyond experimental stacks, so it is not listed among the standard TCP enhancements in this roadmap.

4.2. Fundamental Changes

Like the Standards Track documents listed in Section 3.1, there also exist new Experimental RFCs that specify fundamental changes to TCP. At the time of writing, the only example so far is TCP Fast Open that deviates from the standard TCP semantics of [RFC793].

RFC 7413 E: "TCP Fast Open" (December 2014)

This document [RFC7413] describes TCP Fast Open that allows data to be carried in the SYN and SYN-ACK packets and consumed by the receiver during the initial connection handshake. It saves up to one RTT compared to the standard TCP, which requires a three-way handshake to complete before data can be exchanged.
4.3. Congestion Control Extensions

TCP congestion control has been an extremely active research area for many years (see RFC 5783 discussed in Section 7.6 of this document), as it determines the performance of many applications that use TCP. A number of Experimental RFCs address issues with flow start up, overshoot, and steady-state behavior in the basic algorithms of RFC 5681 (see Section 2 of this document). In these subsections, enhancements to TCP’s congestion control are listed. The next subsection focuses on TCP’s loss recovery.

RFC 2861 E: "TCP Congestion Window Validation" (June 2000)

This document [RFC2861] suggests reducing the congestion window over time when no packets are flowing. This behavior is more aggressive than that specified in RFC 5681 (see Section 2 of this document), which says that a TCP sender SHOULD set its congestion window to the initial window after an idle period of an RTO or greater.

RFC 3540 E: "Robust Explicit Congestion Notification (ECN) Signaling with Nonces" (June 2003)

This document [RFC3540] describes an optional addition to ECN that protects against accidental or malicious concealment of marked packets from the TCP sender.

RFC 3649 E: "HighSpeed TCP for Large Congestion Windows" (December 2003)

This document [RFC3649] proposes a modification to TCP’s congestion control mechanism for use with TCP connections with large congestion windows, to allow TCP to achieve a higher throughput in high-bandwidth environments.

RFC 3742 E: "Limited Slow-Start for TCP with Large Congestion Windows" (March 2004)

This document [RFC3742] describes a more conservative slow-start behavior to prevent massive packet losses when a connection uses a very large congestion window.
This document [RFC4782] specifies the optional Quick-Start mechanism for TCP. This mechanism allows connections to use higher sending rates at the beginning of the data transfer or after an idle period, provided that there is significant unused bandwidth along the path, and the sender and all of the routers along the path approve this higher rate.

This document [RFC5562] describes an experimental modification to ECN [RFC3168] (see Section 3.2 of this document) for the use of ECN in TCP SYN/ACK packets. This would allow to ECN-mark rather than drop the TCP SYN/ACK packet at an ECN-capable router, and to avoid the severe penalty of a retransmission timeout for a connection when the SYN/ACK packet is dropped.

This document [RFC5690] describes a congestion control mechanism for acknowledgment (ACKs) traffic in TCP. The mechanism is based on the acknowledgment congestion control of the Datagram Congestion Control Protocol’s (DCCP’s) [RFC4340] Congestion Control Identifier (CCID) 2 [RFC4341].

This document [RFC6928] proposes to increase the TCP initial window from between 2 and 4 segments, as specified in RFC 3390 (see Section 3.2 of this document), to 10 segments with a fallback to the existing recommendation when performance issues are detected.

4.4. Loss Recovery Extensions

This document [RFC5827] proposes the "Early Retransmit" mechanism for TCP (and SCTP) that can be used to recover lost segments when a connection’s congestion window is small. In certain special circumstances, Early Retransmit reduces the number of duplicate acknowledgments required to trigger fast retransmit to recover segment losses without waiting for a lengthy retransmission timeout.
This document [RFC6069] describes how standard ICMP messages can be used to disambiguate true congestion loss from non-congestion loss caused by connectivity disruptions. It proposes a reversion strategy of TCP’s retransmission timer that enables a more prompt detection of whether or not the connectivity has been restored.

RFC 6937 E: "Proportional Rate Reduction for TCP" (May 2013)

This document [RFC6937] describes an experimental Proportional Rate Reduction (P RR) algorithm as an alternative to the widely deployed Fast Recovery algorithm, to improve the accuracy of the amount of data sent by TCP during loss recovery.

4.5. Detection and Prevention of Spurious Retransmissions

In addition to the Standards Track extensions to deal with spurious retransmissions in Section 3.4, Experimental proposals have also been documented.

RFC 3522 E: "The Eifel Detection Algorithm for TCP" (April 2003)

The Eifel detection algorithm [RFC3522] allows a TCP sender to detect a posteriori whether it has entered loss recovery unnecessarily by using the TCP timestamp option to solve the ACK ambiguity.

RFC 3708 E: "Using TCP Duplicate Selective Acknowledgement (DSACKs) and Stream Control Transmission Protocol (SCTP) Duplicate Transmission Sequence Numbers (TSNs) to Detect Spurious Retransmissions" (February 2004)

Abstract: "TCP and Stream Control Transmission Protocol (SCTP) provide notification of duplicate segment receipt through Duplicate Selective Acknowledgement (DSACKs) and Duplicate Transmission Sequence Number (TSN) notification, respectively. This document presents conservative methods of using this information to identify unnecessary retransmissions for various applications."

RFC 4653 E: "Improving the Robustness of TCP to Non-Congestion Events" (August 2006)

In the presence of non-congestion events, such as packet reordering, an out-of-order segment does not necessarily indicate a lost segment and congestion. This document [RFC4653] proposes
to increase the threshold used to trigger a fast retransmission
from the fixed value of three duplicate ACKs to about one
congestion window of data in order to disambiguate true segment
loss from segment reordering.

4.6. TCP Timeouts

Besides the well-known retransmission timeout the TCP standard
[RFC793] defines other timeouts. This section lists documents that
deal with TCP’s various timeouts.

RFC 5482 S: "TCP User Timeout Option" (March 2009)

As a local per-connection parameter, the TCP user timeout controls
how long transmitted data may remain unacknowledged before a
connection is forcefully closed. This document [RFC5482]
specifies the TCP User Timeout Option that allows one end of a TCP
connection to advertise its current user timeout value. This
information provides advice to the other end of the TCP connection
to adapt its user timeout accordingly.

4.7. Multipath TCP

MultiPath TCP (MPTCP) is an ongoing effort within the IETF that
allows a TCP connection to simultaneously use multiple IP addresses /
interfaces to spread their data across several subflows, while
presenting a regular TCP interface to applications. Benefits of this
include better resource utilization, better throughput and smoother
reaction to failures. The documents listed in this section specify
the Multipath TCP scheme, while the documents in Sections 7.2, 7.4,
and 7.5 provide some additional background information.

RFC 6356 E: "Coupled Congestion Control for Multipath Transport
Protocols" (October 2011)

This document [RFC6356] presents a congestion control algorithm
for multipath transport protocols such as Multipath TCP. It
couples the congestion control algorithms running on different
subflows by linking their increase functions, and dynamically
controls the overall aggressiveness of the multipath flow. The
result is an algorithm that is fair to TCP at bottlenecks while
moving traffic away from congested links.
This document [RFC6824] presents protocol changes required to add multipath capability to TCP; specifically, those for signaling and setting up multiple paths ("subflows"), managing these subflows, reassembly of data, and termination of sessions.

5. TCP Parameters at IANA

RFCs listed here describes both the procedures that the Internet Assigned Numbers Authority (IANA) uses when handling assignments and the procedures an RFC author should follow when requesting new TCP option code points.

RFC 2780 B: "IANA Allocation Guidelines For Values In the Internet Protocol and Related Headers" (March 2000)

Abstract of RFC 2780 [RFC2780]: "This memo provides guidance for the IANA to use in assigning parameters for fields in the IPv4, IPv6, ICMP, UDP and TCP protocol headers."

RFC 4727 S: "Experimental Values in IPv4, IPv6, ICMPv4, ICMPv6, UDP, and TCP Headers" (November 2006)

This document [RFC4727] reserves both TCP options 253 and 254 for experimentation purposes. When such experiments are deployed in the Internet, they should follow the additional requirements in RFC 6994 (see below in Section 5).

RFC 6335 B: "Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry" (August 2011)

From the Abstract of RFC 6335 [RFC6335]: "This document defines the procedures that the Internet Assigned Numbers Authority (IANA) uses when handling assignment and other requests related to the Service Name and Transport Protocol Port Number registry."

RFC 6994 S: "Shared Use of Experimental TCP Options (August 2013)

This document [RFC6994] describes how the experimental TCP option code points can concurrently support multiple TCP extensions, even within the same connection. It creates an IANA registry for extensions to the experimental code points.
6. Historic and Undeployed Extensions

The RFCs listed here define extensions that have thus far failed to arouse substantial interest from implementers and have never seen widespread deployment or were found to be defective for general use. Most of them were reclassified by [RFC6247] to Historic status.

RFC 721 U: "Out-of-Band Control Signals in a Host-to-Host Protocol" (September 1976): lack of interest

RFC 721 [RFC721] addresses the problem of implementing a reliable out-of-band signal (interrupts) for use in a host-to-host protocol. The proposal was not included in the final TCP specification.

RFC 1078 U: "TCP Port Service Multiplexer (TCPMUX)" (November 1988): lack of interest

This document [RFC1078] proposes a protocol to contact multiple services on a single well-known TCP port using a service name instead of a well-known number.

RFC 1106 H: "TCP Big Window and Nak Options" (June 1989): found defective

This RFC [RFC1106] defined an alternative to the Window Scale option for using large windows and described the "negative acknowledgment" or NAK option. There is a comparison of NAK and SACK methods and early discussion of TCP over satellite issues. RFC 1110 (see below in Section 6) explains some problems with the approaches described in RFC 1106. The options described in this document have not been adopted by the larger community, although NAKs are used in the SCPS-TP adaptation of TCP for satellite and spacecraft use, developed by the Consultative Committee for Space Data Systems (CCSDS).

RFC 1110 H: "A Problem with the TCP Big Window Option" (August 1989): deprecates RFC 1106

Abstract of RFC 1110 [RFC1110]: "The TCP Big Window option discussed in RFC 1106 will not work properly in an Internet environment which has both a high bandwidth * delay product and the possibility of disordered and duplicating packets. In such networks, the window size must not be increased without a similar increase in the sequence number space. Therefore, a different approach to big windows should be taken in the Internet."
RFC 1146 H: "TCP Alternate Checksum Options" (March 1990): lack of interest

This document [RFC1146] defined more robust TCP checksums than the 16-bit ones-complement in use today. A typographical error in RFC 1145 is fixed in RFC 1146; otherwise, the documents are the same.

RFC 1263 I: "TCP Extensions Considered Harmful" (October 1991): lack of interest

This document [RFC1263] argues against "backwards compatible" TCP extensions. Specifically mentioned are several TCP enhancements that have been successful, including timestamps, window scaling, PAWS, and SACK. RFC 1263 presents an alternative approach called "protocol evolution", whereby several evolutionary versions of TCP would exist on hosts. These distinct TCP versions would represent upgrades to each other and could be header incompatible. Interoperability would be provided by having a virtualization layer select the right TCP version for a particular connection. This idea did not catch on with the community, while the type of extensions RFC 1263 specifically targeted as harmful did become popular.


See RFC 1644, in Section 6 below.


The inventors of TCP believed that cached connection state could have been used to eliminate TCP’s three-way handshake, to support two-packet request/response exchanges. RFC 1379 [RFC1379] (see above in Section 6) and RFC 1644 [RFC1644] show that this is far from simple. Furthermore, T/TCP floundered on the ease of denial-of-service attacks that can result. One idea pioneered by T/TCP lives on in RFC 2140 (see Section 4.1 of this document), in the sharing of state across connections.
RFC 1693 H: "An Extension to TCP: Partial Order Service" (November 1994): lack of interest

This document [RFC1693] defines a TCP extension for applications that do not care about the order in which application-layer objects are received. Examples are multimedia and database applications. In practice, these applications either accept the possible performance loss because of TCP’s strict ordering or use specialized transport protocols other than TCP, such as PR-SCTP [RFC3758].

RFC 1705 I: "Six Virtual Inches to the Left: The Problem with IPng" (October 1994): lack of interest

To overcome the exhaustion of the IP class B address space, this document [RFC1705] suggests that a new version of TCP (TCPng) needs to be developed and deployed. It proposes that a globally unique address be assigned to the transport layer to uniquely identify an Internet host without specifying any routing information. Later work on splitting locator and identifier values is summarized well in [RFC6115], but no resulting changes to TCP have occurred.

RFC 6013 E: "TCP Cookie Transactions (TCPCT)" (January 2011): lack of interest

This document [RFC6013] describes a method to exchange a cookie (nonce) during the connection establishment to negotiate elimination of receiver state. These cookies are later used to inhibit premature closing of connections and reduce retention of state after the connection has terminated.

Since the cookie pair is too large to fit with the other TCP options in the 40 bytes of TCP option space, the document further describes a method to extent the option space after the connection establishment.

Although RFC 6013 was published in 2011, the authors of this document places it in this section of the roadmap document due to two factors.

(a) The authors are not aware of any wide deployment and use of RFC 6013.
(b) RFC 6013 uses experimental TCP option code points, which prohibits a large-scale deployment.
7. Support Documents

This section contains several classes of documents that do not necessarily define current protocol behaviors but that are nevertheless of interest to TCP implementers. Section 7.1 describes several foundational RFCs that give modern readers a better understanding of the principles underlying TCP’s behaviors and development over the years. Section 7.2 contains architectural guidelines and principles for TCP architects and designers. The documents listed in Section 7.3 provide advice on using TCP in various types of network situations that pose challenges above those of typical wired links. Guidance for developing, analyzing, and evaluating TCP is given in Section 7.4. Some implementation notes and implementation advice can be found in Section 7.5. RFCs that contain high-level tutorials on the protocol are listed Section 7.6. The TCP Management Information Bases are described in Section 7.7, and Section 7.8 lists a number of case studies that have explored TCP performance.

7.1. Foundational Works

The documents listed in this section contain information that is largely duplicated by the standards documents previously discussed. However, some of them contain a greater depth of problem statement explanation or other context. Particularly, RFCs 813 - 817 (known as the "Dave Clark Five") describe some early problems and solutions (RFC 815 only describes the reassembly of IP fragments and is not included in this TCP roadmap).

RFC 675 U: "Specification of Internet Transmission Control Program" (December 1974)

This document [RFC675] is a very early precursor of the fundamental RFC 793 (see Section 2 of this document), which already contained the three-way handshake in its final form and the concept of sliding windows for reliable data transmission. Apart from that, the segment layout is totally different and the specified API differs from the latter RFC 793 (see Section 2 of this document).
This document [RFC761] is the immediate precursor of RFC 793 (see Section 2 of this document). The header format, the connection establishment (including the different connection states), and the overall API correspond mostly to the final Standard RFC 793 (see Section 2 of this document).

This document [RFC813] contains an early discussion of Silly Window Syndrome and its avoidance and motivates and describes the use of delayed acknowledgments.

Suggestions and guidance for the design of tables and algorithms to keep track of various identifiers within a TCP/IP implementation are provided by this document [RFC814].

In this document [RFC816], TCP's response to indications of network error conditions such as timeouts or received ICMP messages is discussed.

This document [RFC817] contains implementation suggestions that are general and not TCP specific. However, they have been used to develop TCP implementations and describe some performance implications of the interactions between various layers in the Internet stack.

Conclusion of RFC 872 [RFC872]: "The sometimes-expressed fear that using TCP on a local net is a bad idea is unfounded."
RFC 896 U: "Congestion Control in IP/TCP Internetworks" (January 1984)

This document [RFC896] contains some early experiences with congestion collapse and some initial thoughts on how to avoid it using congestion control in TCP. Furthermore, it defined an algorithm for efficient transmission of small packets that is today known as the Nagle algorithm.


This document [RFC964] points out several specification bugs in the US Military’s MIL-STD-1778 document, which was intended as a successor to RFC 793 (see Section 2 of this document). This serves to remind us of the difficulty in specification writing (even when we work from existing documents!).

7.2. Architectural Guidelines

Some documents in this section contain architectural guidance and concerns, while others specify TCP- and congestion-control-related mechanisms that are broadly applicable and have impacts on TCP’s congestion control techniques. Some of these documents are direct products of the Internet Architecture Board (IAB) giving their guidance on specific aspects of congestion control in the Internet.

RFC 1958 I: "Architectural Principles of the Internet" (June 1996)

This document [RFC1958] describes the underlying principles of the Internet architecture. It provides guidelines for network systems designs that have proven useful in the evolution of the Internet.

RFC 2914 B: "Congestion Control Principles" (September 2000)

This document [RFC2914] motivates the use of end-to-end congestion control for preventing congestion collapse and providing fairness to TCP. Later work on TCP has included several more aggressive mechanisms than Reno TCP includes, and RFC 5033 (see Section 7.4 of this document) provides additional guidance on use of such algorithms. The fundamental architectural discussion in RFC 2914 remains valid, regarding the standards process role in defining protocol aspects that are critical to performance and avoiding congestion collapse scenarios.
RFC 3360 B: "Inappropriate TCP Resets Considered Harmful" (August 2002)

This document [RFC3360] is a plea that firewall vendors not send gratuitous TCP RST (Reset) packets when unassigned TCP header bits are used. This practice prevents desirable extension and evolution of the protocol and thus is potentially harmful to the future of the Internet.

RFC 3439 I: "Some Internet Architectural Guidelines and Philosophy" (December 2002)

This document [RFC3439] updates RFC 1958 (see above in Section 7.2) by outlining some philosophical guidelines for architects and designers of Internet backbone networks. The document describes the Simplicity Principle, which states that complexity is the primary impediment to efficient scaling.

RFC 4774 B: "Specifying Alternate Semantics for the Explicit Congestion Notification (ECN) Field" (November 2006)

This document [RFC4774] discusses some of the issues in defining alternate semantics for the ECN field and specifies requirements for a safe coexistence with routers that do not understand the defined alternate semantics.

RFC 6182 I: "Architectural Guidelines for Multipath TCP Development" (March 2011)

Abstract of RFC 6182 [RFC6182]: "This document outlines architectural guidelines for the development of a Multipath Transport Protocol, with references to how these architectural components come together in the development of a Multipath TCP (MPTCP) (see Section 4.7 of this document). This document lists certain high-level design decisions that provide foundations for the design of the MPTCP protocol, based upon these architectural requirements"

7.3. Difficult Network Environments

As the internetworking field has explored wireless, satellite, cellular telephone, and other kinds of link-layer technologies, a large body of work has built up on enhancing TCP performance for such links. The RFCs listed in this section describe some of these more challenging network environments and how TCP interacts with them.
RFC 2488 B: "Enhancing TCP Over Satellite Channels using Standard Mechanisms" (January 1999)

From the Abstract of RFC 2488 [RFC2488]: "While TCP works over satellite channels there are several IETF standardized mechanisms that enable TCP to more effectively utilize the available capacity of the network path. This document outlines some of these TCP mitigations. At this time, all mitigations discussed in this document are IETF standards track mechanisms (or are compliant with IETF standards)."

RFC 2757 I: "Long Thin Networks" (January 2000)

Several methods of improving TCP performance over long thin networks (i.e., networks with low bandwidth and high delay), such as geosynchronous satellite links, are discussed in this document [RFC2757]. A particular set of TCP options is developed that should work well in such environments and be safe to use in the global Internet. The implications of such environments have been further discussed in RFCs 3150 and 3155 (see below in Section 7.3), and these documents should be preferred where there is overlap between them and RFC 2757 (see Section 7.3 of this document).

RFC 2760 I: "Ongoing TCP Research Related to Satellites" (February 2000)

This document [RFC2760] discusses the advantages and disadvantages of several different experimental means of improving TCP performance over long-delay or error-prone paths. These include T/TCP, larger initial windows, byte counting, delayed acknowledgments, slow start thresholds, NewReno and SACK-based loss recovery, FACK [MM96], ECN, various corruption-detection mechanisms, congestion avoidance changes for fairness, use of multiple parallel flows, pacing, header compression, state sharing, and ACK congestion control, filtering, and reconstruction. Although RFC 2488 (see above in Section 7.3) looks at standard extensions, this document focuses on more experimental means of performance enhancement.

RFC 3135 I: "Performance Enhancing Proxies Intended to Mitigate Link-Related Degradations" (June 2001)

From the Abstract of RFC 3135 [RFC3135]: "This document is a survey of Performance Enhancing Proxies (PEPs) often employed to improve degraded TCP performance caused by characteristics of specific link environments, for example, in satellite, wireless
WAN, and wireless LAN environments. Different types of Performance Enhancing Proxies are described as well as the mechanisms used to improve performance.

RFC 3150 B: "End-to-end Performance Implications of Slow Links" (July 2001)

From the Abstract of RFC 3150 [RFC3150]: "This document makes performance-related recommendations for users of network paths that traverse "very low bit-rate" links....This recommendation may be useful in any network where hosts can saturate available bandwidth, but the design space for this recommendation explicitly includes connections that traverse 56 Kb/second modem links or 4.8 Kb/second wireless access links - both of which are widely deployed."

RFC 3155 B: "End-to-end Performance Implications of Links with Errors" (August 2001)

From the Abstract of RFC 3155 [RFC3155]: "This document discusses the specific TCP mechanisms that are problematic in environments with high uncorrected error rates, and discusses what can be done to mitigate the problems without introducing intermediate devices into the connection."

RFC 3366 B: "Advice to link designers on link Automatic Repeat reQuest (ARQ)" (August 2002)

From the Abstract of RFC 3366 [RFC3366]: "This document provides advice to the designers of digital communication equipment and link-layer protocols employing link-layer Automatic Repeat reQuest (ARQ) techniques. This document presumes that the designers wish to support Internet protocols, but may be unfamiliar with the architecture of the Internet and with the implications of their design choices for the performance and efficiency of Internet traffic carried over their links."

RFC 3449 B: "TCP Performance Implications of Network Path Asymmetry" (December 2002)

From the Abstract of RFC 3449 [RFC3449]: "This document describes TCP performance problems that arise because of asymmetric effects. These problems arise in several access networks, including bandwidth-asymmetric networks and packet radio subnetworks, for different underlying reasons. However, the end result on TCP performance is the same in both cases: performance often degrades significantly because of imperfection and variability in the ACK feedback from the receiver to the sender.

The document details several mitigations to these effects, which have either been proposed or evaluated in the literature, or are currently deployed in networks.

RFC 3481 B: "TCP over Second (2.5G) and Third (3G) Generation Wireless Networks" (February 2003)

From the Abstract of RFC 3481 [RFC3481]: "This document describes a profile for optimizing TCP to adapt so that it handles paths including second (2.5G) and third (3G) generation wireless networks."

RFC 3819 B: "Advice for Internet Subnetwork Designers" (July 2004)

This document [RFC3819] describes how TCP performance can be negatively affected by some particular lower-layer behaviors and provides guidance in designing lower-layer networks and protocols to be amicable to TCP. RFC 3366 (see above in Section 7.3) specifically focuses on ARQ mechanisms, while RFC 3819 more widely covers additional aspects of the underlying layers.

7.4. Guidance for Developing, Analyzing, and Evaluating TCP

Documents in this section give general guidance for developing, analyzing, and evaluating TCP. Some of the documents discuss, for example, the properties of congestion control protocols that are "safe" for Internet deployment as well as how to measure the properties of congestion control mechanisms and transport protocols.

RFC 5033 B: "Specifying New Congestion Control Algorithms" (August 2007)

This document [RFC5033] considers the evaluation of suggested congestion control algorithms that differ from the principles outlined in RFC 2914 (see Section 7.2 of this document). It is useful for authors of such algorithms as well as for IETF members reviewing the associated documents.

RFC 5166 I: "Metrics for the Evaluation of Congestion Control Mechanisms" (March 2008)

This document [RFC5166] discusses metrics that need to be considered when evaluating new or modified congestion control mechanisms for the Internet. Among other topics, the document discusses throughput, delay, loss rates, response times, fairness, and robustness for challenging environments.
RFC 6077 I: "Open Research Issues in Internet Congestion Control" (February 2011)

This document [RFC6077] summarizes the main open problems in the domain of Internet congestion control. As a good starting point for newcomers, the document describes several new challenges that are becoming important as the network grows, as well as some issues that have been known for many years.

RFC 6181 I: "Threat Analysis for TCP Extensions for Multipath Operation with Multiple Addresses" (March 2011)

This document [RFC6181] describes a threat analysis for Multipath TCP (MPTCP) (see Section 4.7 of this document). The document discusses several types of attacks and provides recommendations for MPTCP designers how to create an MPTCP specification that is as secure as the current (single-path) TCP.

RFC 6349 I: "Framework for TCP Throughput Testing" (August 2011)

From the Abstract of RFC 6349 [RFC6349]: "This framework describes a practical methodology for measuring end-to-end TCP Throughput in a managed IP network. The goal is to provide a better indication in regard to user experience. In this framework, TCP and IP parameters are specified to optimize TCP Throughput."

7.5. Implementation Advice

RFC 794 U: "PRE-EMPTION" (September 1981)

This document [RFC794] clarifies that operating systems need to manage their limited resources, which may include TCP connection state, and that these decisions can be made with application input, but they do not need to be part of the TCP protocol specification itself.

RFC 879 U: "The TCP Maximum Segment Size and Related Topics" (November 1983)

Abstract of RFC 879 [RFC879]: "This memo discusses the TCP Maximum Segment Size Option and related topics. The purposes [sic] is to clarify some aspects of TCP and its interaction with IP. This memo is a clarification to the TCP specification, and contains information that may be considered as ‘advice to implementers’."
This document [RFC1071] lists a number of implementation techniques for efficiently computing the Internet checksum (used by TCP).

Incrementally updating the Internet checksum is useful to routers in updating IP checksums. Some middleboxes that alter TCP headers may also be able to update the TCP checksum incrementally. This document [RFC1624] expands upon the explanation of the incremental update procedure in RFC 1071 (see above in Section 7.5).

This document [RFC1936] describes the motivation for implementing the Internet checksum in hardware, rather than in software, and provides an implementation example.

From the Abstract of RFC 2525 [RFC2525]: "This memo catalogs a number of known TCP implementation problems. The goal in doing so is to improve conditions in the existing Internet by enhancing the quality of current TCP/IP implementations."

From abstract: "This memo catalogs several known Transmission Control Protocol (TCP) implementation problems dealing with Path Maximum Transmission Unit Discovery (PMTUD), including the long-standing black hole problem, stretch acknowledgments (ACKs) due to confusion between Maximum Segment Size (MSS) and segment size, and MSS advertisement based on PMTU." [RFC2923]

This document [RFC3493] describes the de facto standard sockets API for programming with TCP. This API is implemented nearly ubiquitously in modern operating systems and programming languages.
RFC 6056 B: "Recommendations for Transport-Protocol Port Randomization" (December 2010)

This document [RFC6056] describes a number of simple and efficient methods for the selection of the client port number. It reduces the possibility of an attacker guessing the correct five-tuple (Protocol, Source/Destination Address, Source/Destination Port).

RFC 6191 B: "Reducing the TIME-WAIT State Using TCP Timestamps" (April 2011)

This document [RFC6191] describes the usage of the TCP Timestamps option (RFC 7323, see Section 3.1 of this document) to perform heuristics to determine whether or not to allow the creation of a new incarnation of a connection that is in the TIME-WAIT state.

RFC 6429 I: "TCP Sender Clarification for Persist Condition" (December 2011)

This document [RFC6429] clarifies the actions that a TCP can take on connections that are experiencing the Zero Window Probe (ZWP) condition.

RFC 6897 I: "Multipath TCP (MPTCP) Application Interface Considerations" (March 2013)

This document [RFC6897] characterizes the impact that Multipath TCP (MPTCP) (see Section 4.7 of this document) may have on applications. It further discusses compatibility issues of MPTCP in combination with non-MPTCP-aware applications. Finally, it describes a basic API that is a simple extension of TCP’s interface for MPTCP-aware applications.

7.6. Tools and Tutorials

RFC 1180 I: "TCP/IP Tutorial" (January 1991) (Errata)

This document [RFC1180] is an extremely brief overview of the TCP/IP protocol suite as a whole. It gives some explanation as to how and where TCP fits in.
A few of the tools that this document [RFC1470] describes are still maintained and in use today, for example, ttcp and tcpdump. However, many of the tools described do not relate specifically to TCP and are no longer used or easily available.

This document [RFC2398] describes a number of TCP packet generation and analysis tools. Although some of these tools are no longer readily available or widely used, for the most part they are still relevant and usable.

This document [RFC5783] provides an overview of RFCs related to congestion control that had been published at the time. The focus of the document is on end-host-based congestion control.

7.7. MIB Modules

The first MIB module defined for use with Simple Network Management Protocol (SNMP) was a single monolithic MIB module, called MIB-I, defined in RFC 1156. This evolved over time to the MIB-II specification in RFC 1213, which obsoletes RFC 1156. It then became apparent that having a single monolithic MIB module was not scalable, given the number and breadth of MIB data definitions that needed to be included. Thus, additional MIB modules were defined, and those parts of MIB-II that needed to evolve were split off. Eventually, the remaining parts of MIB-II were also split off, the TCP-specific part being documented in RFC 2012. RFC 2012 was obsoleted by RFC 4022, which is the primary TCP MIB document at the time of writing. For current TCP implementers, RFC 4022 should be supported.

This document [RFC1156] describes the required MIB fields for TCP implementations with minor corrections and no technical changes from RFC 1066, which it obsoletes. This is the Standards Track RFC for MIB-I.

This document [RFC1213] describes the second version of the MIB in a monolithic form. It is the immediate successor of RFC 1158, with minor modifications. It obsoletes the MIB-I, defined in RFC 1156 (see above in Section 7.7).


In an update to RFC 1213 (see Section 7.7 of this document), this document [RFC2012] defines the TCP MIB by splitting out the TCP-specific portions. It is now obsoleted by RFC 4022 (see below in Section 7.7).


This document [RFC2452] augments RFC 2012 (see Section 7.7 of this document) by adding an IPv6-specific connection table. The rest of RFC 2012 holds for any IP version. RFC 2452 is now obsoleted by RFC 4022 (see below in Section 7.7).

Although it is a Standards Track RFC, RFC 2452 is considered a historic mistake by the MIB community, as it is based on the idea of parallel IPv4 and IPv6 structures. Although IPv6 requires new structures, the community has decided to define a single generic structure for both IPv4 and IPv6. This will aid in definition, implementation, and transition between IPv4 and IPv6.


This document [RFC4022] obsoletes RFCs 2012 and 2452 (see above in Section 7.7) and specifies the current standard for the TCP MIB that should be deployed.

RFC 4898 S: "TCP Extended Statistics MIB" (May 2007)

This document [RFC4898] describes extended performance statistics for TCP. They are designed to use TCP’s ideal vantage point to diagnose performance problems in both the network and the application.
7.8. Case Studies

RFC 700 U: "A Protocol Experiment" (August 1974)

This document [RFC700] presents a field report about the deployment of a very early version of TCP, the so-called INWN #39 protocol, which is originally described by Cerf and Kahn in INWG Note #39 [CK73] to use a PDP-11 line printer via the ARPANET.

RFC 889 U: "Internet Delay Experiments" (December 1983)

This document [RFC889] is a status report about experiments concerning the TCP retransmission timeout calculation and also provides advice for implementers.

RFC 1337 I: "TIME-WAIT Assassination Hazards in TCP" (May 1992)

This document [RFC1337] points out a problem with acting on received reset segments while one is in the TIME-WAIT state. The main recommendation is that hosts in TIME-WAIT ignore resets. This recommendation might not currently be widely implemented.

RFC 2415 I: "Simulation Studies of Increased Initial TCP Window Size" (September 1998)

This document [RFC2415] presents results of some simulations using TCP initial windows greater than 1 segment. The analysis indicates that user-perceived performance can be improved by increasing the initial window to 3 segments.

RFC 2416 I: "When TCP Starts Up With Four Packets Into Only Three Buffers" (September 1998)

This document [RFC2416] uses simulation results to clear up some concerns about using an initial window of 4 segments when the network path has less provisioning.

RFC 2884 I: "Performance Evaluation of Explicit Congestion Notification (ECN) in IP Networks" (July 2000)

This document [RFC2884] describes experimental results that show some improvements to the performance of both short- and long-lived connections due to ECN.
8. Undocumented TCP Features

There are a few important implementation tactics for the TCP that have not yet been described in any RFC. Although this roadmap is primarily concerned with mapping the TCP RFCs, this section is included because an implementer needs to be aware of these important issues.

Header Prediction

Header prediction is a trick to speed up the processing of segments. Van Jacobson and Mike Karels developed the technique in the late 1980s. The basic idea is that some processing time can be saved when most of a segment’s fields can be predicted from previous segments. A good description of this was sent to the TCP-IP mailing list by Van Jacobson on March 9, 1988 (see [Jacobson] for the full message):

Quite a bit of the speedup comes from an algorithm that we ('we' refers to collaborator Mike Karels and myself) are calling "header prediction". The idea is that if you’re in the middle of a bulk data transfer and have just seen a packet, you know what the next packet is going to look like: It will look just like the current packet with either the sequence number or ack number updated (depending on whether you’re the sender or receiver). Combining this with the "Use hints" epigram from Butler Lampson’s classic "Epigrams for System Designers", you start to think of the tcp state (rcv.nxt, snd.una, etc.) as "hints" about what the next packet should look like.

If you arrange those "hints" so they match the layout of a tcp packet header, it takes a single 14-byte compare to see if your prediction is correct (3 longword compares to pick up the send & ack sequence numbers, header length, flags and window, plus a short compare on the length). If the prediction is correct, there’s a single test on the length to see if you’re the sender or receiver followed by the appropriate processing. E.g., if the length is non-zero (you’re the receiver), checksum and append the data to the socket buffer then wake any process that’s sleeping on the buffer. Update rcv.nxt by the length of this packet (this updates your "prediction" of the next packet). Check if you can handle another packet the same size as the current one. If not, set one of the unused flag bits in your header prediction to guarantee that the prediction will fail on the next packet and force you to go through full protocol processing. Otherwise, you’re done with this packet. So, the "total" tcp protocol processing, exclusive of checksumming, is on the order of 6 compares and an add.
Forward Acknowledgement (FACK)

FACK [MM96] includes an alternate algorithm for triggering fast retransmit [RFC5681], based on the extent of the SACK scoreboard. Its goal is to trigger fast retransmit as soon as the receiver’s reassembly queue is larger than the duplicate ACK threshold, as indicated by the difference between the forward most SACK block edge and SND.UNA. This algorithm quickly and reliably triggers fast retransmit in the presence of burst losses -- often on the first SACK following such a loss. Such a threshold-based algorithm also triggers fast retransmit immediately in the presence of any reordering with extent greater than the duplicate ACK threshold. FACK is implemented in Linux and turned on per default.

Congestion Control for High Rate Flows

In the last decade significant research effort has been put into experimental TCP congestion control modifications for obtaining high throughput with reduced startup and recovery times. Only a few RFCs have been published on some of these modifications, including HighSpeed TCP [RFC3649], Limited Slow-Start [RFC3742], and Quick-Start [RFC4782] (see Section 4.3 of this document for more information on each), but high-rate congestion control mechanisms are still considered an open issue in congestion control research. Some other schemes have been published as Internet-Drafts, e.g. CUBIC [CUBIC] (the standard TCP congestion control algorithm in Linux), Compound TCP [CTCP], and H-TCP [HTCP] or have been discussed a little by the IETF, but much of the work in this area has not been adopted within the IETF yet, so the majority of this work is outside the RFC series and may be discussed in other products of the IRTF Internet Congestion Control Research Group (ICCRG).

9. Security Considerations

This document introduces no new security considerations. Each RFC listed in this document attempts to address the security considerations of the specification it contains.
10. References

10.1. Normative References


10.2. Informative References


SIGCOMM 1988 Proceedings, in ACM Computer Communication

[Jacobson] Jacobson, V., "TCP-IP Mailing List", Article 167 of
comp.protocols.tcp-ip, March 1988,

ACM SIGCOMM 1987 Proceedings, in ACM Computer

Evolution of Transport Protocols in the Internet", ACM

[MM96]     Mathis, M. and J. Mahdavi, "Forward Acknowledgement:
Refining TCP Congestion Control", ACM SIGCOMM 1996
Proceedings, in ACM Computer Communication Review 26 (4),
pp. 281-292, October 1996.

[RFC1016]  Prue, W. and J. Postel, "Something a host could do with
source quench: The Source Quench Introduced Delay
(SQuID)", RFC 1016, July 1987,

[RFC2026]  Bradner, S., "The Internet Standards Process -- Revision
3", BCP 9, RFC 2026, October 1996,

[RFC2119]  Bradner, S., "Key words for use in RFCs to Indicate
Requirement Levels", BCP 14, RFC 2119, March 1997,

[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

[RFC3758]  Stewart, R., Ramalho, M., Xie, Q., Tuexen, M., and P.
Conrad, "Stream Control Transmission Protocol (SCTP)
Partial Reliability Extension", RFC 3758, May 2004,

Congestion Control Protocol (DCCP)", RFC 4340, March 2006,


Acknowledgments

This document grew out of a discussion on the end2end-interest mailing list, the public list of the End-to-End Research Group of the IRTF, and continued development under the IETF’s TCP Maintenance and Minor Extensions (TCPM) working group. We thank Mark Allman, Yuchung Cheng, Ted Faber, Gorry Fairhurst, Sally Floyd, Janardhan Iyengar, Reiner Ludwig, Pekka Savola, and Joe Touch for their contributions, in particular. Keith McCloghrie provided some useful notes and clarification on the various MIB-related RFCs.
Authors’ Addresses

Martin Duke
F5 Networks
401 Elliott Ave W
Seattle, WA  98119
United States

Phone: 206-272-7537
EMail: m.duke@f5.com

Robert Braden
USC Information Sciences Institute
Marina del Rey, CA  90292-6695
United States

Phone: 310-448-9173
EMail: braden@isi.edu

Wesley M. Eddy
MTI Systems
18013 Cleveland Parkway
Suite 170
Cleveland, OH  44135
United States

Phone: 216-433-6682
EMail: wes@mti-systems.com

Ethan Blanton
Interrupt Sciences

EMail: elb@interruptsciences.com

Alexander Zimmermann
NetApp, Inc.
Sonnenallee 1
Kirchheim  85551
Germany

Phone: +49 89 900594712
EMail: alexander.zimmermann@netapp.com