```
**********************************************************
   19581 Mon May 26 17:48:17 2014
new/usr/src/man/man7p/tcp.7p
4774 Typos in tcp(7P) manpage
**********************************************************
'\" te
.\" Copyright (c) 2006, Sun Microsystems, Inc. All Rights Reserved.
.\" Copyright (c) 2011 Nexenta Systems, Inc. All rights reserved.
.\" Copyright 1989 AT&T
.\" The contents of this file are subject to the terms of the Common Development
.\" You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE or http:
.\" When distributing Covered Code, include this CDDL HEADER in each file and in
.TH TCP 7P "Jun 30, 2006"
.SH NAME
tcp, TCP \- Internet Transmission Control Protocol
.SH SYNOPSIS
.LP
.nf
\fB#include <sys/socket.h>\fR
.fi

.LP
.nf
\fB#include <netinet/in.h>\fR
.fi

.LP
.nf
\fBs = socket(AF_INET, SOCK_STREAM, 0);\fR
.fi

.LP
.nf
\fBs = socket(AF_INET6, SOCK_STREAM, 0);\fR
.fi

.LP
.nf
\fBt = t_open("/dev/tcp", O_RDWR);\fR
.fi

.LP
.nf
\fBt = t_open("/dev/tcp6", O_RDWR);\fR
.fi

.SH DESCRIPTION
.sp
.LP
\fBTCP\fR is the virtual circuit protocol of the Internet protocol family. It
provides reliable, flow-controlled, in order, two-way transmission of data. It
is a byte-stream protocol layered above the Internet Protocol (\fBIP\fR), or
the Internet Protocol Version 6 (\fBIPv6\fR), the Internet protocol family's
internetwork datagram delivery protocol.
.sp
.LP
Programs can access \fBTCP\fR using the socket interface as a \fBSOCK_STREAM\fR
socket type, or using the Transport Level Interface (\fBTLI\fR) where it
supports the connection-oriented (\fBT_COTS_ORD\fR) service type.
.sp
.LP
\fBTCP\fR uses \fBIP\fR's host-level addressing and adds its own per-host
collection of "port addresses." The endpoints of a \fBTCP\fR connection are
identified by the combination of an \fBIP\fR or IPv6 address and a \fBTCP\fR
port number. Although other protocols, such as the User Datagram Protocol
(UDP), may use the same host and port address format, the port space of these
```



```
protocols is distinct. See \fBinet\fR(7P) and \fBinet6\fR(7P) for details on
the common aspects of addressing in the Internet protocol family.
.sp
.LP
Sockets utilizing \fBTCP\fR are either "active" or "passive." Active sockets
initiate connections to passive sockets. Both types of sockets must have their
local \fBIP\fR or IPv6 address and \fBTCP\fR port number bound with the
\fBbind\fR(3SOCKET) system call after the socket is created. By default,
\fBTCP\fR sockets are active. A passive socket is created by calling the
\fBlisten\fR(3SOCKET) system call after binding the socket with \fBbind()\fR.
This establishes a queueing parameter for the passive socket. After this,
connections to the passive socket can be received with the
\fBaccept\fR(3SOCKET) system call. Active sockets use the
\fBconnect\fR(3SOCKET) call after binding to initiate connections.
.sp
.LP
By using the special value \fBINADDR_ANY\fR with \fBIP\fR, or the unspecified
address (all zeroes) with IPv6, the local \fBIP\fR address can be left
unspecified in the \fBbind()\fR call by either active or passive \fBTCP\fR
sockets. This feature is usually used if the local address is either unknown or
irrelevant. If left unspecified, the local \fBIP\fR or IPv6 address will be
bound at connection time to the address of the network interface used to
service the connection.
.sp
.LP
Note that no two TCP sockets can be bound to the same port unless the bound IP
addresses are different.  IPv4 \fBINADDR_ANY\fR and IPv6 unspecified addresses
compare as equal to any IPv4 or IPv6 address. For example, if a socket is bound
to \fBINADDR_ANY\fR or unspecified address and port X, no other socket can bind
to port X, regardless of the binding address. This special consideration of
\fBINADDR_ANY\fR and unspecified address can be changed using the socket option
\fBSO_REUSEADDR\fR. If \fBSO_REUSEADDR\fR is set on a socket doing a bind, IPv4
\fBINADDR_ANY\fR and IPv6 unspecified address do not compare as equal to any IP
address. This means that as long as the two sockets are not both bound to
\fBINADDR_ANY\fR/unspecified address or the same IP address, the two sockets
can be bound to the same port.
.sp
.LP
If an application does not want to allow another socket  using the
\fBSO_REUSEADDR\fR option to bind to a port its socket is bound to, the
application can set the socket level option \fBSO_EXCLBIND\fR on a socket. The
option values of 0 and 1 mean enabling and disabling the option respectively.
Once this option is enabled on a socket, no other socket can be bound to the
same port.
.sp
.LP
Once a connection has been established, data can be exchanged using the
\fBread\fR(2) and \fBwrite\fR(2) system calls.
.sp
.LP
Under most circumstances, \fBTCP\fR sends data when it is presented. When
outstanding data has not yet been acknowledged, \fBTCP\fR gathers small amounts
of output to be sent in a single packet once an acknowledgement has been
received. For a small number of clients, such as window systems that send a
stream of mouse events which receive no replies, this packetization may cause
significant delays. To circumvent this problem, \fBTCP\fR provides a
socket-level boolean option, \fBTCP_NODELAY.\fR \fBTCP_NODELAY\fR is defined in
\fB<netinet/tcp.h>\fR, and is set with \fBsetsockopt\fR(3SOCKET) and tested
with \fBgetsockopt\fR(3SOCKET). The option level for the \fBsetsockopt()\fR
call is the protocol number for \fBTCP,\fR available from
\fBgetprotobyname\fR(3SOCKET).
.sp
.LP
For some applications, it may be desirable for TCP not to send out data unless
a full TCP segment can be sent. To enable this behavior, an application can use
the \fBTCP_CORK\fR socket option. When \fBTCP_CORK\fR is set with a non-zero
```

value, TCP sends out a full TCP segment only. When \fBTCP_CORK\fR is set to zero after it has been enabled, all buffered data is sent out (as permitted by the peer's receive window and the current congestion window). \fBTCP_CORK\fR is defined in <\fBnetinet/tcp.h\fR>, and is set with \fBsetsockopt\fR(3SOCKET) and tested with \fBgetsockopt\fR(3SOCKET). The option level for the \fBsetsockopt()\fR call is the protocol number for TCP, available from \fBgetprotobyname\fR(3SOCKET).

.sp
.LP
The SO_RCVBUF socket level option can be used to control the window that TCP advertises to the peer. IP level options may also be used with TCP. See \fBip\fR(7P) and \fBip6\fR(7P).

.sp
.LP
Another socket level option, \fBSO_RCVBUF,\fR can be used to control the window that \fBTCP\fR advertises to the peer. \fBIP\fR level options may also be used with \fBTCP.\fR See \fBip\fR(7P) and \fBip6\fR(7P).

.sp
.LP
\fBTCP\fR provides an urgent data mechanism, which may be invoked using the out-of-band provisions of \fBsend\fR(3SOCKET). The caller may mark one byte as "urgent" with the \fBMSG_OOB\fR flag to \fBsend\fR(3SOCKET). This sets an "urgent pointer" pointing to this byte in the \fBTCP\fR stream. The receiver on the other side of the stream is notified of the urgent data by a \fBSIGURG\fR signal. The \fBSIOCATMARK\fR \fBioctl\fR(2) request returns a value indicating whether the stream is at the urgent mark. Because the system never returns data across the urgent mark in a single \fBread\fR(2) call, it is possible to advance to the urgent data in a simple loop which reads data, testing the socket with the \fBSIOCATMARK\fR \fBioctl()\fR request, until it reaches the mark.

.sp
.LP
Incoming connection requests that include an \fBIP\fR source route option are noted, and the reverse source route is used in responding.

.sp
.LP
A checksum over all data helps \fBTCP\fR implement reliability. Using a window-based flow control mechanism that makes use of positive acknowledgements, sequence numbers, and a retransmission strategy, \fBTCP\fR can usually recover when datagrams are damaged, delayed, duplicated or delivered out of order by the underlying communication medium.

.sp
.LP
If the local \fBTCP\fR receives no acknowledgements from its peer for a period of time, (for example, if the remote machine crashes), the connection is closed and an error is returned.

.sp
.LP
TCP follows the congestion control algorithm described in \fIRFC 2581\fR, and also supports the initial congestion window (cwnd) changes in \fIRFC 3390\fR. The initial cwnd calculation can be overridden by the socket option TCP_INIT_CWND. An application can use this option to set the initial cwnd to a specified number of TCP segments. This applies to the cases when the connection first starts and restarts after an idle period. The process must have the PRIV_SYS_NET_CONFIG privilege if it wants to specify a number greater than that calculated by \fIRFC 3390\fR.

.sp
.LP
SunOS supports \fBTCP\fR Extensions for High Performance (\fIRFC 1323\fR) which **includes the window scale and timestamp options, and Protection Against Wrap** *includes the window scale and time stamp options, and Protection Against Wrap* Around Sequence Numbers (PAWS). SunOS also supports Selective Acknowledgment (SACK) capabilities (RFC 2018) and Explicit Congestion Notification (ECN) mechanism (\fIRFC 3168\fR).

.sp
.LP

Turn on the window scale option in one of the following ways:
.RS +4
.TP
.ie t \(bu
.el o
An application can set \fBSO_SNDBUF\fR or \fBSO_RCVBUF\fR size in the \fBsetsockopt()\fR option to be larger than 64K. This must be done \fIbefore\fR the program calls \fBlisten()\fR or \fBconnect()\fR, because the window scale option is negotiated when the connection is established. Once the connection has been made, it is too late to increase the send or receive window beyond the default \fBTCP\fR limit of 64K.
.RE
.RS +4
.TP
.ie t \(bu
.el o
For all applications, use \fBndd\fR(1M) to modify the configuration parameter \fBtcp_wscale_always\fR. If \fBtcp_wscale_always\fR is set to \fB1\fR, the window scale option will always be set when connecting to a remote system. If \fBtcp_wscale_always\fR is \fB0,\fR the window scale option will be set only if the user has requested a send or receive window larger than 64K. The default value of \fBtcp_wscale_always\fR is \fB1\fR.
.RE
.RS +4
.TP
.ie t \(bu
.el o
Regardless of the value of \fBtcp_wscale_always\fR, the window scale option will always be included in a connect acknowledgement if the connecting system has used the option.
.RE
.sp
.LP
Turn on \fBSACK\fR capabilities in the following way:
.RS +4
.TP
.ie t \(bu
.el o
Use \fBndd\fR to modify the configuration parameter \fBtcp_sack_permitted\fR. If \fBtcp_sack_permitted\fR is set to \fB0\fR, \fBTCP\fR will not accept \fBSACK\fR or send out \fBSACK\fR information. If \fBtcp_sack_permitted\fR is set to \fB1\fR, \fBTCP\fR will not initiate a connection with \fBSACK\fR permitted option in the \fBSYN\fR segment, but will respond with \fBSACK\fR permitted option in the \fBSYN|ACK\fR segment if an incoming connection request has the \fBSACK \fR permitted option. This means that \fBTCP\fR will only accept \fBSACK\fR information if the other side of the connection also accepts \fBSACK\fR information. If \fBtcp_sack_permitted\fR is set to \fB2\fR, it will both initiate and accept connections with \fBSACK\fR information. The default for \fBtcp_sack_permitted\fR is \fB2\fR (active enabled).
.RE
.sp
.LP
Turn on \fBTCP ECN\fR mechanism in the following way:
.RS +4
.TP
.ie t \(bu
.el o
Use \fBndd\fR to modify the configuration parameter \fBtcp_ecn_permitted\fR. If \fBtcp_ecn_permitted\fR is set to \fB0\fR, \fBTCP\fR will not negotiate with a peer that supports \fBECN\fR mechanism. If \fBtcp_ecn_permitted\fR is set to \fB1\fR when initiating a connection, TCP will not tell a peer that it supports ECN mechanism. However, it will tell a peer that it supports \fBECN\fR mechanism when accepting a new incoming connection request if the peer indicates that it supports \fBECN\fR mechanism in the SYN segment. If tcp_ecn_permitted is set to 2, in addition to negotiating with a peer on ECN mechanism when accepting connections, TCP will indicate in the outgoing SYN

segment that it supports \fBECN\fR mechanism when \fBTCP\fR makes active outgoing connections. The default for \fBtcp_ecn_permitted\fR is 1.
.RE
.sp
.LP
Turn on the timestamp option in the following way:
Turn on the time stamp option in the following way:
.RS +4
.TP
.ie t \(bu
.el o
Use \fBndd\fR to modify the configuration parameter \fBtcp_tstamp_always\fR. If **\fBtcp_tstamp_always\fR is \fB1\fR, the timestamp option will always be set**
\fBtcp_tstamp_always\fR is \fB1\fR, the time stamp option will always be set when connecting to a remote machine. If \fBtcp_tstamp_always\fR is \fB0\fR, the timestamp option will not be set when connecting to a remote system. The default for \fBtcp_tstamp_always\fR is \fB0\fR.
.RE
.RS +4
.TP
.ie t \(bu
.el o
Regardless of the value of \fBtcp_tstamp_always\fR, the timestamp option will
Regardless of the value of \fBtcp_tstamp_always\fR, the time stamp option will always be included in a connect acknowledgement (and all succeeding packets) if **the connecting system has used the timestamp option.**
the connecting system has used the time stamp option.
.RE
.sp
.LP
Use the following procedure to turn on the timestamp option only when the
Use the following procedure to turn on the time stamp option only when the window scale option is in effect:
.RS +4
.TP
.ie t \(bu
.el o
Use \fBndd\fR to modify the configuration parameter \fBtcp_tstamp_if_wscale\fR. **Setting \fBtcp_tstamp_if_wscale\fR to \fB1\fR will cause the timestamp option**
Setting \fBtcp_tstamp_if_wscale\fR to \fB1\fR will cause the time stamp option to be set when connecting to a remote system, if the window scale option has **been set. If \fBtcp_tstamp_if_wscale\fR is \fB0\fR, the timestamp option will**
been set. If \fBtcp_tstamp_if_wscale\fR is \fB0\fR, the time stamp option will not be set when connecting to a remote system. The default for \fBtcp_tstamp_if_wscale\fR is \fB1\fR.
.RE
.sp
.LP
Protection Against Wrap Around Sequence Numbers (PAWS) is always used when the **timestamp option is set.**
time stamp option is set.
.sp
.LP
SunOS also supports multiple methods of generating initial sequence numbers. One of these methods is the improved technique suggested in \fBRFC\fR 1948. We \fBHIGHLY\fR recommend that you set sequence number generation parameters as close to boot time as possible. This prevents sequence number problems on connections that use the same connection-ID as ones that used a different sequence number generation. The \fBsvc:/network/initial:default\fR service configures the initial sequence number generation. The service reads the value contained in the configuration file \fB/etc/default/inetinit\fR to determine which method to use.
.sp
.LP
The \fB/etc/default/inetinit\fR file is an unstable interface, and may change in future releases.

.sp
.LP
\fBTCP\fR may be configured to report some information on connections that terminate by means of an \fBRST\fR packet. By default, no logging is done. If the \fBndd\fR(1M) parameter \fItcp_trace\fR is set to 1, then trace data is collected for all new connections established after that time.
.sp
.LP
The trace data consists of the \fBTCP\fR headers and \fBIP\fR source and destination addresses of the last few packets sent in each direction before RST occurred. Those packets are logged in a series of \fBstrlog\fR(9F) calls. This trace facility has a very low overhead, and so is superior to such utilities as \fBsnoop\fR(1M) for non-intrusive debugging for connections terminating by means of an \fBRST\fR.
.sp
.LP
SunOS supports the keep-alive mechanism described in \fIRFC 1122\fR. It is enabled using the socket option SO_KEEPALIVE. When enabled, the first keep-alive probe is sent out after a TCP is idle for two hours If the peer does not respond to the probe within eight minutes, the TCP connection is aborted. You can alter the interval for sending out the first probe using the socket option TCP_KEEPALIVE_THRESHOLD. The option value is an unsigned integer in milliseconds. The system default is controlled by the TCP ndd parameter tcp_keepalive_interval. The minimum value is ten seconds. The maximum is ten days, while the default is two hours. If you receive no response to the probe, you can use the TCP_KEEPALIVE_ABORT_THRESHOLD socket option to change the time threshold for aborting a TCP connection. The option value is an unsigned integer in milliseconds. The value zero indicates that TCP should never time out and abort the connection when probing. The system default is controlled by the TCP ndd parameter tcp_keepalive_abort_interval. The default is eight minutes.
.sp
.LP
socket options TCP_KEEPIDLE, TCP_KEEPCNT and TCP_KEEPINTVL are also supported for compatibility with other Unix Flavors. TCP_KEEPIDLE option specifies the interval in seconds for sending out the first keep-alive probe. TCP_KEEPCNT specifies the number of keep-alive probes to be sent before aborting the connection in the event of no response from peer. TCP_KEEPINTVL specifies the interval in seconds between successive keep-alive probes.
.SH SEE ALSO
.sp
.LP
\fBsvcs\fR(1), \fBndd\fR(1M), \fBioctl\fR(2), \fBread\fR(2), \fBsvcadm\fR(1M), \fBwrite\fR(2), \fBaccept\fR(3SOCKET), \fBbind\fR(3SOCKET), \fBconnect\fR(3SOCKET), \fBgetprotobyname\fR(3SOCKET), \fBgetsockopt\fR(3SOCKET), \fBlisten\fR(3SOCKET), \fBsend\fR(3SOCKET), \fBsmf\fR(5), \fBinet\fR(7P), \fBinet6\fR(7P), \fBip\fR(7P), \fBip6\fR(7P)
.sp
.LP
Ramakrishnan, K., Floyd, S., Black, D., RFC 3168, \fIThe Addition of Explicit Congestion Notification (ECN) to IP\fR, September 2001.
.sp
.LP
Mathias, M. and Hahdavi, J. Pittsburgh Supercomputing Center; Ford, S. Lawrence Berkeley National Laboratory; Romanow, A. Sun Microsystems, Inc. \fIRFC 2018, TCP Selective Acknowledgement Options\fR, October 1996.
.sp
.LP
Bellovin, S., \fIRFC 1948, Defending Against Sequence Number Attacks\fR, May 1996.
.sp
.LP
Jacobson, V., Braden, R., and Borman, D., \fIRFC 1323, TCP Extensions for High Performance\fR, May 1992.
.sp
.LP

```
383 Postel, Jon, \fIRFC 793, Transmission Control Protocol - DARPA Internet Program
384 Protocol Specification\fR, Network Information Center, SRI International, Menlo
385 Park, CA., September 1981.
386 .SH DIAGNOSTICS
387 .sp
388 .LP
389 A socket operation may fail if:
390 .sp
391 .ne 2
392 .na
393 \fB\fBEISCONN\fR\fR
394 .ad
395 .RS 17n
396 A \fBconnect()\fR operation was attempted on a socket on which a
397 \fBconnect()\fR operation had already been performed.
398 .RE

400 .sp
401 .ne 2
402 .na
403 \fB\fBETIMEDOUT\fR\fR
404 .ad
405 .RS 17n
406 A connection was dropped due to excessive retransmissions.
407 .RE

409 .sp
410 .ne 2
411 .na
412 \fB\fBECONNRESET\fR\fR
413 .ad
414 .RS 17n
415 The remote peer forced the connection to be closed (usually because the remote
416 machine has lost state information about the connection due to a crash).
417 .RE

419 .sp
420 .ne 2
421 .na
422 \fB\fBECONNREFUSED\fR\fR
423 .ad
424 .RS 17n
425 The remote peer actively refused connection establishment (usually because no
426 process is listening to the port).
427 .RE

429 .sp
430 .ne 2
431 .na
432 \fB\fBEADDRINUSE\fR\fR
433 .ad
434 .RS 17n
435 A \fBbind()\fR operation was attempted on a socket with a network address/port
436 pair that has already been bound to another socket.
437 .RE

439 .sp
440 .ne 2
441 .na
442 \fB\fBEADDRNOTAVAIL\fR\fR
443 .ad
444 .RS 17n
445 A \fBbind()\fR operation was attempted on a socket with a network address for
446 which no network interface exists.
447 .RE
```



```
449 .sp
450 .ne 2
451 .na
452 \fB\fBEACCES\fR\fR
453 .ad
454 .RS 17n
455 A \fBbind()\fR operation was attempted with a "reserved" port number and the
456 effective user \fBID\fR of the process was not the privileged user.
457 .RE

459 .sp
460 .ne 2
461 .na
462 \fB\fBENOBUFS\fR\fR
463 .ad
464 .RS 17n
465 The system ran out of memory for internal data structures.
466 .RE

468 .SH NOTES
469 .sp
470 .LP
471 The \fBtcp\fR service is managed by the service management facility,
472 \fBsmf\fR(5), under the service identifier:
473 .sp
474 .in +2
475 .nf
476 svc:/network/initial:default
477 .fi
478 .in -2
479 .sp

481 .sp
482 .LP
483 Administrative actions on this service, such as enabling, disabling, or
484 requesting restart, can be performed using \fBsvcadm\fR(1M). The service's
485 status can be queried using the \fBsvcs\fR(1) command.
```