Sockets
Performance Tuning

Infrastructure Perspective
Ken Gottry
May - 2001
Table of Contents

• Review
• ndd Command
• netstat Command
• SE Toolkit
• Denial of Service (DoS) Attacks
Last Time We Talked About…

• A socket is a way for two programs (processes) to communicate.

• **Socket** = IP Address + Port Number
  
  • Uniquely identifies every program in the world
  
  • 192.168.1.200:80  -- web server (port 80) on 192.168.1.200

• Use `netstat -na` to view sockets and their state

• Sockets can be tested with **ttcp** utility

• Sockets are affected by firewalls and load balancers
We Also Talked About Ports

- **Well-known ports** — between 0-1024. Same on all UNIX computers around the world. For example, ftp=21, telnet=23, smtp=25, http=80, ldap=389

- **Commonly-used ports** — 1024-65535. Usually less than 32767. Generally accepted *default* port numbers. OracleSQL=1521, WebLogic=7001, iPlanet Admin=8888

- **Anonymous (ephemeral) ports** — the socket at the client end has to have a port number also. The TCP/IP stack assigns one temporarily. When the socket is closed, this port becomes available for use by another program

```
Computer A 192.168.1.200
Web Browser 33186
Port 33186
Socket 192.168.1.200:33186

Computer B 66.66.34.202
Web Server 80
Port 80
Socket 66.66.34.202:80

Web server LISTEN'ing on port 80 on Computer B
Browser asks for socket connection to port 80 on 66.66.34.202 by saying: (http://ComputerB)
The PC on which the browser is running assigns an anonymous port (33186) that the browser can use
```
# netstat -na (netstat -na -P tcp)

## UDP

<table>
<thead>
<tr>
<th>Local Address</th>
<th>Remote Address</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>.</em></td>
<td><em>.</em></td>
<td>Idle</td>
</tr>
<tr>
<td><em>.</em></td>
<td><em>.</em></td>
<td>Idle</td>
</tr>
<tr>
<td>127.0.0.1.53</td>
<td></td>
<td>Idle</td>
</tr>
<tr>
<td>192.168.1.200.53</td>
<td></td>
<td>Idle</td>
</tr>
</tbody>
</table>

## TCP

<table>
<thead>
<tr>
<th>Local Address</th>
<th>Remote Address</th>
<th>Swind</th>
<th>Send-Q</th>
<th>Rwind</th>
<th>Recv-Q</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>.</em></td>
<td><em>.</em></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0 LISTEN</td>
</tr>
<tr>
<td><em>.</em></td>
<td><em>.</em></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0 LISTEN</td>
</tr>
<tr>
<td><em>.</em></td>
<td><em>.</em></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0 LISTEN</td>
</tr>
</tbody>
</table>

| 192.168.1.200.23 | 192.168.1.150.1714 | 8732 | 1 | 9520 | 0 | ESTABLISHED |
| 192.168.1.200.80 | 192.168.1.150.1716 | 9400 | 0 | 9520 | 0 | TIME_WAIT    |
| 192.168.1.200.80 | 192.168.1.200.1717 | 9315 | 0 | 9520 | 0 | ESTABLISHED  |
| 192.168.1.200.80 | 192.168.1.150.1719 | 9300 | 0 | 9520 | 0 | ESTABLISHED  |

## Active UNIX domain sockets

<table>
<thead>
<tr>
<th>Address</th>
<th>Type</th>
<th>Vnode</th>
<th>Conn</th>
<th>Local Addr</th>
<th>Remote Addr</th>
</tr>
</thead>
<tbody>
<tr>
<td>300000a2bba8</td>
<td>stream-ord</td>
<td>00000000</td>
<td>00000000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>300000a2bd48</td>
<td>stream-ord</td>
<td>30000374300</td>
<td>00000000</td>
<td>/tmp/.X11-unix/X0</td>
<td></td>
</tr>
</tbody>
</table>
ndd Command
**ndd Overview**

- Network device drivers (/dev/xxx): tcp, ip, udp, icmp, arp
- To display all parameters for a driver
  ```
  ndd /dev/tcp 
  ```
- To display a value for a parameter
  ```
  ndd /dev/tcp tcp_time_wait_interval 
  ```
- To change a value for a parameter
  ```
  ndd -set /dev/tcp tcp_slow_start_initial 2 
  ```
- Can also be used for NIC’s, such as hme0 or qfe0
  ```
  ndd /dev/hme instance 0 
  ```
<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>tcp_conn_grace_period</td>
<td>0</td>
</tr>
<tr>
<td>tcp_conn_req_max_q</td>
<td>128</td>
</tr>
<tr>
<td>tcp_conn_req_max_q0</td>
<td>1024</td>
</tr>
<tr>
<td>tcp_conn_req_min</td>
<td>1</td>
</tr>
<tr>
<td>tcp_cwnd_max</td>
<td>262144</td>
</tr>
<tr>
<td>tcp_debug</td>
<td>0</td>
</tr>
<tr>
<td>tcp_deferred_ack_interval</td>
<td>100</td>
</tr>
<tr>
<td>tcp_deferred_acks_max</td>
<td>8</td>
</tr>
<tr>
<td>tcp_dupack_fast_retransmit</td>
<td>3</td>
</tr>
<tr>
<td>tcp_host_param</td>
<td>Hash</td>
</tr>
<tr>
<td>tcp_ignore_path_mtu</td>
<td>0</td>
</tr>
<tr>
<td>tcp_ip_abort_cinterval</td>
<td>180000</td>
</tr>
<tr>
<td>tcp_ip_abort_interval</td>
<td>480000</td>
</tr>
<tr>
<td>tcp_ip_abort_linterval</td>
<td>180000</td>
</tr>
<tr>
<td>tcp_ip_notify_cinterval</td>
<td>10000</td>
</tr>
<tr>
<td>tcp_ip_notify_interval</td>
<td>10000</td>
</tr>
<tr>
<td>tcp_ip_ttl</td>
<td>255</td>
</tr>
<tr>
<td>tcp_keepalive_interval</td>
<td>720000</td>
</tr>
<tr>
<td>tcp_largest_anon_port</td>
<td>65535</td>
</tr>
<tr>
<td>tcp_max_buf</td>
<td>1048576</td>
</tr>
<tr>
<td>tcp_mss_def</td>
<td>536</td>
</tr>
<tr>
<td>tcp_mss_max</td>
<td>65495</td>
</tr>
<tr>
<td>tcp_mss_min</td>
<td>108</td>
</tr>
<tr>
<td>tcp_naglim_def</td>
<td>4095</td>
</tr>
<tr>
<td>tcp_rcv_push_wait</td>
<td>16384</td>
</tr>
<tr>
<td>tcp_recv_hiwat</td>
<td>8192</td>
</tr>
<tr>
<td>tcp_recv_hiwat_minmss</td>
<td>4</td>
</tr>
<tr>
<td>tcp_rexmit_interval_extra</td>
<td>0</td>
</tr>
<tr>
<td>tcp_rexmit_interval_initial</td>
<td>3000</td>
</tr>
<tr>
<td>tcp_rexmit_interval_max</td>
<td>60000</td>
</tr>
<tr>
<td>tcp_rexmit_interval_min</td>
<td>400</td>
</tr>
<tr>
<td>tcp_slow_start_after_idle</td>
<td>2</td>
</tr>
<tr>
<td>tcp_slow_start_initial</td>
<td>2</td>
</tr>
<tr>
<td>tcp_smallest_anon_port</td>
<td>32768</td>
</tr>
<tr>
<td>tcp_smallest_nonpriv_port</td>
<td>1024</td>
</tr>
<tr>
<td>tcp_snd_lowat_fraction</td>
<td>0</td>
</tr>
<tr>
<td>tcp_sth_rcv_hiwat</td>
<td>0</td>
</tr>
<tr>
<td>tcp_sth_rcv_lowat</td>
<td>0</td>
</tr>
<tr>
<td>tcp_strong_iss</td>
<td>1</td>
</tr>
<tr>
<td>tcp_time_wait_interval</td>
<td>240000</td>
</tr>
<tr>
<td>tcp_trace</td>
<td>0</td>
</tr>
<tr>
<td>tcp_wroff_xtra</td>
<td>32</td>
</tr>
<tr>
<td>tcp_wscale_always</td>
<td>0</td>
</tr>
<tr>
<td>tcp_xmit_hiwat</td>
<td>8192</td>
</tr>
<tr>
<td>tcp_xmit_lowat</td>
<td>2048</td>
</tr>
<tr>
<td>tcp_zero_win_probesize</td>
<td>1</td>
</tr>
</tbody>
</table>
noodle “others”

```
icmp_bsd_compat     1
icmp_def_ttl         255
icmp_max_buf         262144
icmp_recv_hiwat      8192
icmp_wroff_extra     32
icmp_xmit_hiwat      8192
icmp_xmit_lowat      1024

arp_cleanup_interval 300000
arp_debug             0

udp_def_ttl          255
udp_do_checksum      1
udp_largest_anon_port 65535
udp_max_buf          262144
udp_recv_hiwat       8192
udp_smallest_anon_port 32768
udp_smallest_nonpriv_port 1024
udp_trust_optlen     0
udp_wroff_extra      32
udp_xmit_hiwat       8192
udp_xmit_lowat       1024
```

```
ip_addrs_per_if                  256
ip_broadcast_ttl                 1
ip_debug                          0
ip_def_ttl                        255
ip_enable_group_ifs               0
ip_forward_directed_broadcasts    1
ip_forward_src_routed             1
ip_forwarding                     0
ip_icmp_err_interval              500
ip_icmp_return_data_bytes         64
ip_ignore_redirect                0
ip_ire_cleanup_interval           30000
ip_ire_flush_interval             1200000
ip_ire_pathmtu_interval           600000
ip_ire_redirect_interval          60000
ip_mrtdebug                       0
ip_output_queue                    1
ip_path_mtu_discovery             1
ip_respond_to_address_mask_broadcast 0
ip_respond_to_echo_broadcast      1
ip_respond_to_timestamp           1
ip_respond_to_timestamp_broadcast 1
ip_send_redirects                  1
ip_strict_dst_multihoming          0
```

```
icmp_bsd_compat        1
icmp_def_ttl           255
icmp_max_buf           262144
icmp_recv_hiwat        8192
icmp_wroff_extra       32
icmp_xmit_hiwat        8192
icmp_xmit_lowat        1024

arp_cleanup_interval       300000
arp_debug                  0

udp_def_ttl                255
udp_do_checksum            1
udp_largest_anon_port      65535
udp_max_buf                262144
udp_recv_hiwat             8192
udp_smallest_anon_port     32768
udp_smallest_nonpriv_port  1024
udp_trust_optlen           0
udp_wroff_extra            32
udp_xmit_hiwat             8192
udp_xmit_lowat             1024
```

www.gotry.com

© 2001 Ken Gottry
Parameters to Care About
Parameters To Care About

**tcp_time_wait_interval** — length of time socket is remembered after it is closed. RFC default is 4 minutes. Reduce to 1 minute (60000ms). At 100 connections/sec, the default means that 24,000 closed connections are still in memory waiting to be freed. May run out of anonymous ports and refuse incoming connections. Also fills up **tcp_conn_hash** with closed connections.

**tcp_conn_hash_size** — hash helps quickly locate TCP data structure in the kernel. If default of 256 entries is exceeded, then a slow linear search is required. Increase to at least 8K. Web benchmarks set it to 262144. Must be power of 2. If **tcp_time_wait_interval** is not adjusted, then many of the entries in the hash point to closed connections. This parameter is changed by adding the following line to /etc/system:

```
set tcp:tcp_conn_hash_size=8192
```

**tcp_slow_start_initial** — number of packets to send when establishing connection. RFC says 1. Windows TCP clients hesitate unless 2 are sent. RFC expected to be changed to 2. Impacts LoadRunner tests.
Parameters To Care About

tcp_conn_req_max_q0 — default=1024; recommend=4096. See if tcpListenDropQ0 in “netstat –s” display is nonzero. Tune time_wait and conn_hash parameters first.

tcp_conn_req_max_q — default=128; recommend=1024. See if tcpListenDrop in “netstat –s” display is nonzero. Tune time_wait and conn_hash parameters first.

tcp_xmit_hiwat — how much data can be sent at one time. Default of 8K not large enough for fast networks (100mbps). Increase to 32K

tcp_recv_hiwat — same as xmit

Changes made by ndd take effect immediately. They are lost during next reboot. To make changes permanent, add the ndd commands to /etc/rc2.d/S69inet. Some parameters can only be changed in /etc/system.
You Can Never Have Too Many ...

Each end of a socket connection has to have a port number. When you connect to a server, the TCP stack on your client assigns a temporary port number to your end of the socket connection. If you don’t relinquish the temporary port numbers fast enough, then there may not be any available to establish new connections.

Ever wonder why LoadRunner can’t simulate very many virtual clients when running on Windows NT? The maximum file descriptors is 2048 ... and sockets count as files.

```
tcp_smallest_anon_port  --  default=32768; recommend=1024;
```

Anonymous (ephemeral) ports — the socket at the client end has to have a port number also. The TCP/IP stack assigns one temporarily. When the socket is closed, this port becomes available for use by another program.
Sockets are Files

One cool thing about UNIX is that everything is a file. When a program opens a socket, it uses a file descriptor since that’s how files are accessed. The following parameters must always be set correctly.

**rlim_fd_max** — default=1024; recommend=8192; Controls the number of file descriptors a process can have open

**rlim_fd_cur** — default=64; recommend=8192; Allows a process to ratchet down the limit if desired; A process can never ratchet the value up beyond the system-wide setting for rlim_fd_max

While you can change the values for these parameters on-the-fly using adb, you really aren’t changing anything. You need to add these lines to /etc/system:

```
set rlim_fd_max=8192
set rlim_fd_cur=8192
```
TCP Connection Pool
The `tcp_time_wait_interval` measures the time from when both application ends have closed the TCP connection and when the TCP stack software releases the resources. The default is 2MSL (maximum segment life). This allows time for any remaining packets to arrive before the anonymous port is reused by another TCP connection. 240000ms (4 minutes) was a reasonable setting when 300 baud modems were in vogue. Today 60000ms (1 minute) is a reasonable setting.
TCP Connection Pool

1) To connect an App Server to a Web Server you install an NSAPI plug-in on the Web Server.

2) Every time an HTTP request hits the Web Server it is passed to the NSAPI plug-in.

3) The NSAPI plug-in sends the request to the App Server.

4) Before the request can be sent from the NSAPI plug-in to the App Server, a socket connection must be established.

The big question. How is the socket connection established??

Ugly way – establish a socket connection, send the request, close the connection … and have it hang around for tcp_time_wait_interval

Better way – establish a pool of socket connections when the NSAPI plug-in starts, use a connection from the pool to send the request.
netstat Command
netstat -s -P tcp

tcpActiveOpens  = 123  tcpPassiveOpens  = 319
tcpAttemptFails =  8   tcpEstabResets   = 26
tcpCurrEstab    =  1   tcpOutSegs      = 186140
tcpOutDataSegs  = 85552 tcpOutDataBytes = 16673629
tcpRetransSegs  =  58  tcpRetransBytes = 26604
tcpOutAck       =100588 tcpOutAckDelayed = 1599
tcpOutUrg       =  0   tcpOutWinUpdate =  0
tcpOutWinProbe  =  0   tcpOutControl   =  879
tcpOutRsts     =  6    tcpOutFastRetrans =  0
tcpInSegs       =317808 tcpInAckBytes    =16669469
tcpInAckSegs    = 79865 tcpInAckUnsent    =  0
tcpInDupAck     = 317   tcpInAckUnsent    =  0
tcpInInorderSegs=300013 tcpInInorderBytes=315104568
tcpInUnorderSegs=  4    tcpInUnorderBytes = 4323
tcpInDupSegs    =  8    tcpInDupBytes     =  0
tcpInPartDupSegs=  0    tcpInPartDupBytes =  0
tcpInPastWinSegs=  0    tcpInPastWinBytes =  0
tcpInWinProbe   =  0    tcpInWinUpdate   =  0
tcpTimRetransDrop =  0    tcpTimKeepalive = 145
tcpTimKeepaliveProbe=  0    tcpTimKeepaliveDrop =  0
tcpListenDrop   =  0    tcpListenDropQ0 =  0
tcpHalfOpenDrop =  0    tcpOutSackRetrans =  0
Thresholds

<table>
<thead>
<tr>
<th>Metric</th>
<th>Warning</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Retransmission</td>
<td>&gt; 15%</td>
<td>&gt; 25%</td>
</tr>
<tr>
<td>Duplicate Packets</td>
<td>&gt; 15%</td>
<td>&gt; 25%</td>
</tr>
<tr>
<td>ListenDrop</td>
<td>&gt; 0</td>
<td>&gt; 0.5/sec</td>
</tr>
</tbody>
</table>

- **Retransmissions** — your system is not receiving ACK’s fast enough. May be bad network hardware or a congested route (maybe you have a hub and not a switch).

- **Duplicates** — the remote system is retransmitting too fast (i.e. not receiving your ACK’s fast enough). May be bad network hardware or a congested route

- **ListenDrop** — your system is receiving more incoming packets than it can handle. TCP parameter tuning required. May need another server to balance the workload
Other Interesting Counters

• **Out of Order** — packets are not arriving at your system in the same order they were sent by the remote system. May be a routing problem.

• **Denial of Service (Dos) attack** — some computer is starting a TCP connection with you but never completing the initial connection phase. Your computer becomes so busy dealing with the phoney connections that it can’t process valid connections
  
  • Solaris implemented a separate queue (Q0) to hold connections while the initial 3-step handshake is being negotiated. This allows Solaris to continue to process valid connections in the normal queue while under DoS attack.
  
  • \((\text{tcpListenDropQ0} > 0)\) or \((\text{tcpHalfOpenDrop} > 0)\)
  
• Your server received SYN (step 1 of 3), you sent ACK plus your SYN (step 2 of 3) but you never received the ACK from the remote server (step 3 of 3)
Calculate Percents

- Retransmission Percent = \( \frac{\text{tcpRetransBytes}}{\text{tcpOutDataBytes}} \)
  \[
  \frac{26604}{16673629} = 0.16\%
  \]
- Duplicates Percent = \( \frac{\text{tcpInDupBytes}}{\text{tcpInDataBytes}} \)
  \[
  \frac{0}{16729} = 0.00\%
  \]
- ListenDrop = tcpListenDrop = 0
- Out of Order = tcpInUnorderBytes = 4323
### Script and Graph

- Import the log file into MS Excel, delimited by space, tab, and other ("=")
- Compute connections/second using an MS Excel formula
  \[
  \text{connections/second} = \frac{(D2-D1)}{\text{sleep}}
  \]
  where sleep is the number of seconds between samplings in your script (i.e. the value of variable SLEEP)
- Graph using MS Excel Insert → Chart menu bar option

```
#!/bin/ksh
export LOG=/tmp/netstat_s.log
export CTR=0
export SLEEP=1
export LIMIT=60
while [ $(CTR) -lt $(LIMIT) ]
do
  netstat -s -P tcp | grep tcpPassiveOpens >> ${LOG}
  CTR=$(expr ${CTR} + 1)
sleep ${SLEEP}
done
cat ${LOG} | mailx -s "netstat-s-log" kgottry@nervewire.com
```

<table>
<thead>
<tr>
<th>tcpActiveOpens</th>
<th>tcpPassiveOpens</th>
<th>Connections/Sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>124</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>124</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>124</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>124</td>
<td>14</td>
<td>14</td>
</tr>
<tr>
<td>124</td>
<td>13</td>
<td>13</td>
</tr>
<tr>
<td>124</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>124</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>124</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>124</td>
<td>14</td>
<td>14</td>
</tr>
</tbody>
</table>
Denial of Service Attacks
Denial of Service (DoS) Attack

DoS attack instigated by Harry against Ken.

Harry pretends to be Jack when starting the 3-way handshake.
Ken ends up with so many half-open sockets (no ACK to his SYN) that he can’t process valid incoming TCP connections.
netstat -s on Ken will show tcpHalfOpenDrop > 0

www.gottry.com