"Sharp Close": Elimination of TIME-WAIT state of TCP connections
<draft-kitamura-tcp-sharp-close-02.txt>

Hiroshi KITAMURA
NEC Corporation
kitamura@da.jp.nec.com
Index

- Introduction
- Analysis of current TIME-WAIT state
- Why TIME-WAIT state is needed?
  If no TIME-WAIT state, what will happen?
- Design of Sharp Close
  (elimination or minimize of TIME-WAIT state)
- TIME-WAIT state can be eliminated by setsockopt()
- For Next Steps
Introduction

TIME-WAIT state will NOT be up-to-date functions anymore.

• From the viewpoints of current high-speed and high-multiplicity communication styles that require highly resource recycling, it is thought that TIME-WAIT state will be one of evil functions.

• In order to provide efficient communications that match current styles, an idea Sharp Close that eliminates or minimizes TIME-WAIT state of TCP connections is proposed.
Current **ACTIVE-PASSIVE** Close Sequence

**ACTIVE** CLOSE

- ESTABLISH
- FIN
- FIN-WAIT-1
- FIN-WAIT-2
- TIME-WAIT
- 2MSL

**PASSIVE** CLOSE

- ESTABLISH
- FIN
- ACK
- FIN
- ACK
- LAST-ACK
- CLOSED
Current **ACTIVE-ACTIVE** Close Sequence

- **ACTIVE** CLOSE
  - ESTABLISH
  - FIN-WAIT-1
  - CLOSING
  - TIME-WAIT
  - 2MSL

- **ACTIVE** CLOSE
  - ESTABLISH
  - FIN-WAIT-1
  - CLOSING
  - TIME-WAIT
  - 2MSL
Actual **2MSL** values used by major OS implementation

<table>
<thead>
<tr>
<th>RFC / OS</th>
<th>2MSL value</th>
</tr>
</thead>
<tbody>
<tr>
<td>[RFC0793]</td>
<td>240 sec.</td>
</tr>
<tr>
<td>Windows2000</td>
<td>240 sec.</td>
</tr>
<tr>
<td>Windows (after Win2K)</td>
<td>120 sec.</td>
</tr>
<tr>
<td>Unix/Linux</td>
<td>60 sec.</td>
</tr>
<tr>
<td>net.ipv4.tcp_fin_timeout = 60</td>
<td></td>
</tr>
</tbody>
</table>
Why TIME-WAIT state is needed? If no TIME-WAIT state, what will happen? (1/2)

- Basically, TIME-WAIT state will be designed for **fail-safe** purpose.

- When FIN-WAIT-2 state is finished (FIN packet is received):
  - All of data packets from a corresponding node are received
  - No data packets will be received after that.
  (if it is assumed that packets transferring order is not changed)

- At TIME-WAIT state (on an **ACTIVE CLOSE** node):
  - Waits for a *resending control* packet **FIN only** from the corresponding node for the case of the sent ACK (for the FIN) is lost.
  - (No data packets are waited for.)
Why TIME-WAIT state is needed? If no TIME-WAIT state, what will happen? (2/2)

- Only when the last sent ACK from the ACTIVE CLOSE node is lost, 
  *resending control* packet FIN from the corresponding node is issued.
- It is *rare* case to happen this event at current stable network environment.
- It is *less significant* issue to wait for *resending control* packet FIN, 
  because all data from the corresponding node is received by 
  the ACTIVE CLOSE node at that time

- If *resending control* packet FIN is *NOT* waited on ACTIVE CLOSE node 
  and *resending control* packet FIN is issued from the corresponding node, 
  *significant problem will NOT be happened.*
- After *resending control* packet FIN is received on ACTIVE CLOSE node, 
  only RST packet (to notify receiving unexpected packet) will be issued to the 
  corresponding node.
Design of **Sharp Close**

- It is easy to design **Sharp Close** function.

- **Sharp Close** function is achieved by eliminating or minimizing TIME-WAIT state of TCP connections.
Proposed Sharp ACTIVE-PASSIVE Close Sequence

ACTIVE CLOSE

ESTABLISH

FIN

FIN-WAIT-1

FIN-WAIT-2

(No or Short TIME-WAIT ) CLOSED

ACK

ACK

PASSIVE CLOSE

ESTABLISH

CLOSE-WAIT

FIN

LAST-ACK

CLOSED
Proposed **Sharp** **ACTIVE-ACTIVE** Close Sequence

**ACTIVE** CLOSE

ESTABLISH

FIN-WAIT-1

CLOSING

(No or Short TIME-WAIT) CLOSED

(No or Short TIME-WAIT) CLOSED
TIME-WAIT state can be eliminated by setsockopt()

- Under current implementation, TIME-WAIT (close()) action can be controlled by setsockopt() function.
- SO_LINGER option of setsockopt() can eliminate TIME-WAIT state and close connections immediately.

```c
<sys/socket.h>
struct linger {
    int l_onoff;    /* linger active */
    int l linger;   /* how many seconds to linger for */
};
```

Procedures to eliminate TIME-WAIT state (close connection immediately):

1. makes linger active(on)
   ```c
   l_onoff = on;
   ```
2. sets linger time to 0
   ```c
   l linger = 0;
   ```

It is possible to eliminate TIME-WAIT state by these procedures. However, this behavior is **NOT** default operation. In order to utilize this feature, it is necessary to modify huge number of communication applications.
For Next Steps

Q1: Do you think \textbf{TIME-WAIT} state will \textbf{NOT} be \textbf{up-to-date} functions?

Q2: If so, what do you want to do?
   A2-1: \textbf{Eliminate} \textbf{TIME-WAIT} state
   A2-2: \textbf{Minimize} \textbf{TIME-WAIT} state

Q3: If you choose “Minimize”, which actual value is appropriate for \textbf{TIME-WAIT} state?