Increasing TCP’s Extensibility Using Second Grade Math

Eric Kaminsky (BWC), Ken Atchinson (BWC), Mark Allman (ICSI)

Baldwin-Wallace College
October 2011

“I want to put a ding the universe.” –Steve Jobs
Increasing TCP’s Extensibility Using Second Grade Math

Eric Kaminsky (BWC), Ken Atchinson (BWC), Mark Allman (ICSI)

Baldwin-Wallace College
October 2011

“I want to put a ding the universe.” —Steve Jobs
The Internet is made up of myriad technologies

Network architecture provides ....

abstractions

services

organizing principles

i.e., the basic structure of the system

Primary goal: flexibility and generality
## Layering

<table>
<thead>
<tr>
<th>OSI Model</th>
<th>Internet Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>Application</td>
</tr>
<tr>
<td>Presentation</td>
<td></td>
</tr>
<tr>
<td>Session</td>
<td></td>
</tr>
<tr>
<td>Transport</td>
<td>Transport</td>
</tr>
<tr>
<td>Network</td>
<td>Network</td>
</tr>
<tr>
<td>Data Link</td>
<td>Data Link</td>
</tr>
<tr>
<td>Physical</td>
<td>Physical</td>
</tr>
</tbody>
</table>
### Layering (cont.)

<table>
<thead>
<tr>
<th>Layer</th>
<th>Examples of Communication Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>Physical</td>
<td>analog to digital conversion (EE stuff!)</td>
</tr>
<tr>
<td>Data Link</td>
<td>host-to-host comm (e.g., Ethernet)</td>
</tr>
<tr>
<td>Network</td>
<td>direct host-to-host comm (e.g., IP)</td>
</tr>
<tr>
<td>Transport</td>
<td>app-to-app comm (e.g., HTTP, XMPP, SMTP)</td>
</tr>
<tr>
<td>Application</td>
<td>process-to-process comm (e.g., TCP, UDP)</td>
</tr>
</tbody>
</table>

**Allman**
Layering (cont.)

HTTP  SMTP  IMAP  DNS  VoIP

TCP

UDP  DCCP

IP

PPP  Ethernet  WiFi

Twisted Pair  Fiber  Coaxial  RF  Satellite
TCP

- Provides reliable in-order byte stream between processes
- Implements congestion control to prevent collapse
- Very general
- Usually suboptimal
- Carries most of the traffic on the Internet
TCP Usage

(courtesy of Matt Sargent)
Packets

0000111010101010011101010011111001101......

LL hdr  Net hdr  Trans hdr  App data  LL trlr

Eth  IP  TCP  HTTP hdr  HTTP data  Eth trlr

22B  20B  20B  269B  (3812B)  4B

Allman
TCP Header

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Src. Port</td>
<td>Source Port Number</td>
</tr>
<tr>
<td>Dest. Port</td>
<td>Destination Port Number</td>
</tr>
<tr>
<td>Sequence</td>
<td>Sequence Number</td>
</tr>
<tr>
<td>Acknowledgment</td>
<td>Acknowledgment Number</td>
</tr>
<tr>
<td>Window</td>
<td>Window Size</td>
</tr>
<tr>
<td>Checksum</td>
<td>Checksum</td>
</tr>
<tr>
<td>Urg. Ptr.</td>
<td>Urgent Pointer</td>
</tr>
</tbody>
</table>

We’re out of space!
### TCP Header (cont.)

<table>
<thead>
<tr>
<th>Field</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Src. Port</td>
<td></td>
</tr>
<tr>
<td>Dest. Port</td>
<td></td>
</tr>
<tr>
<td>Sequence Number</td>
<td></td>
</tr>
<tr>
<td>Acknowledgment Number</td>
<td></td>
</tr>
<tr>
<td>HLen</td>
<td>RB</td>
</tr>
<tr>
<td>Checksum</td>
<td></td>
</tr>
</tbody>
</table>

Option space to the rescue!  
40 more bytes!
# TCP Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Size (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSS</td>
<td>3</td>
</tr>
<tr>
<td>SACK Permitted</td>
<td>2</td>
</tr>
<tr>
<td>Window Scale</td>
<td>3</td>
</tr>
<tr>
<td>SACK</td>
<td>2 + multiples of 8</td>
</tr>
<tr>
<td>Timestamp</td>
<td>10</td>
</tr>
<tr>
<td>Auth</td>
<td>18</td>
</tr>
<tr>
<td>.....</td>
<td></td>
</tr>
</tbody>
</table>

**UGH! We’re really out of space!**
How To Fix TCP?

• Two approaches:
  • TCPng
  • More options
More Options

• Approach #1: Negotiate a different way to convey header length
  • scale factor on the current field
  • new field as the first option
• Approach #2: Introduce a magic “end-of-options” option and walk the options
• Approach #3: Define unused code-points in the current header length field
  • E.g., HLen == 0 may yield a 96 byte header
  • E.g., HLen == 1 may yield a 128 byte header
  • Etc.
More Options (cont.)

• Engineering costs and benefits to all these schemes

• Maybe TCP is just the wrong size!
Our Approach

• The previously voiced two options:
  • TCPng or more options

• Maybe we need a third general approach ....

• Foolishly multiply the TCP header size by two
  • aka, “TCPx2”
TCPx2: Pros

- Conceptually simple
- Extends a known useful protocol
- The protocol semantics don’t change; only the syntax
- Code “should be” straightforward
- Solves all issues at once, not piecemeal
TCPx2: Cons

- New header format
- All hosts have to change
- All firewalls have to change
- All NATs have to change
- ..... 
- Not backward compatible
  - (a point we address and will return to ....)
Implementation

• Modifying the Linux kernel
• Kernel: lots of C code
• 18 files modified for TCPx2

<table>
<thead>
<tr>
<th></th>
<th>Amount of Code</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TCP</strong></td>
<td>23K-30K LOC</td>
</tr>
<tr>
<td><strong>TCPx2</strong></td>
<td>2,851 LOC  (added or changed)</td>
</tr>
<tr>
<td></td>
<td>10-12% change</td>
</tr>
</tbody>
</table>
Strategy

- Modify TCP packet structure
- Packets in the system will have TCPx2 headers
- Accept both types of packets, but keep one structure internally

```
Within System
```

```
Header x2
Data

Modify

Data

Data
```

```
Data
```

```
Header x2
Data

Shrink

Data

Data
```

```
Header
Data
```

```
Data
```

```
Data
```

```
Header
Data
```

```
Data
```
Further Changes

• Needed to fix many bugs caused by struct change

• Migrated a number of TCP features:
  • SACK (Selective Acknowledgement)
  • Expanded SYN cookies
  • Disabled window scaling for TCPx2 packets
    • (unneeded with larger window size)
  • Modified tcpdump to parse TCPx2 packets
Migration

• How can we transition to the new protocol?

• TCPx2 needs to be able to:
  • Accept incoming connections from TCP and TCPx2 clients
  • Connect to servers, not knowing whether they support TCPx2 or not
  • … without sacrificing any speed
Migration (cont.)

Original TCP Handshake

Client

Server

Server is waiting for connections...

SYN

SYN-ACK

ACK

Connection established!

New TCPx2 Handshake

Client With tcpx2

Server With ???

Two SYNs

SYN-ACK

ACK

Whichever SYN the server gets first becomes the protocol.
Future Work

• Performance testing
• Modify a NAT/firewall to support TCPx2
• Utilize expanded port space
Questions? Comments?

Mark Allman, Eric Kaminsky, Ken Atchinson

mallman@icir.org
http://www.icir.org/mallman/