Multipath TCP Signaling Options or Payload?

Costin Raiciu
c.raiciu@cs.ucl.ac.uk
Motivation

• MPTCP needs to signal control data to function correctly
  – Data sequence mapping
  – Data ACK
  – Add/remove IP
  – Security Information
• What’s the best way to signal this information?
What’s at stake

• We are extending TCP signaling and changing the endpoints to do multipath
• If we really succeed nearly all TCP will be multipath
  – Hence we are changing TCP signaling too
Considerations

• Control Message
  – Size
  – Requirements

• Middlebox behavior
  – Remove options *
  – Duplicate options
  – Normalize payload

* Our data shows that only 15/10000 websites do not respond to SYNs with new options
Options Encoding

- 40B in TCP header, some already used
- Traditional way to signal control information in TCP
- Not reliably delivered
- Cons
  - Limited size
  - Messed with by middleboxes
Payload Encoding

• Use Type-Length-Value encoding to add control messages in the payload
• No more space issues
Connection/Subflow Setup

• Options encoding:
  – Add MPTCP capable/token on initial connection (12B)
  – Add JOIN option on subsequent connections (8B)

• Payload encoding impossible
  – Must use options encoding, otherwise receiver can’t demux SYNs
Data Sequence Mapping

• 14B with options, 10B with payload
• Options encoding:
  – Reliability issue: may be stripped by middleboxes
    • Using data ACK we can infer it was lost => drop path and use others
• Payload Encoding
  – If segment is ACKed, mapping is ACKed too.
Data ACK

- 8B
- Options - just piggyback on ACKs, cuts 1 SACK segment
- Payload
  - Data ACKs are sent reliably, in order, and are congestion controlled!
  - We could just not CC packets with data ACKs
    - Bad interaction between CC and reliability
    - What about pacing retransmissions?
  - Implications
    - Cannot piggyback them on all ACKs
    - Can trigger timeouts at the sender on connection level data
    - Can delay delivery of useful data
Add/Remove IP

• 6-18B
• Need in order, reliable delivery
• Options
  – Must echo options explicitly and send one/RTT or
  – Add sequence numbers & acks
• Payload
  – Send one/RTT or
  – Add sequence numbers & acks.
Security Information

• Typically bootstrapped when connection begins

• Options
  – Public keys (if used) will not fit in options, must encode in payload
  – We could add option describing where in the payload the security data is

• Payload: no issues
Middleboxes

• That strip options on data packets but not on SYNs
  • Options
    – Will not use the subflow
  • Payload
    • Will use the subflow
Future Middleboxes

• Middleboxes will be deployed to optimize various aspects of multipath TCP

• Options
  – Allows all (stateless/stateful) middleboxes to see the state of the connection and the fact its multipath

• Payload
  – Stateless middleboxes cannot differentiate between TCP and MPTCP
  – Stateful middleboxes will have to parse the data stream; when they get desynchronized the can’t tell if it’s TCP or multipath anymore.
Future middleboxes (2)

• What about signaling by middleboxes?
  – E.g.: insert “add ip” to reroute flow through another middlebox?
  – Or strip/change some options (e.g. “add ip”, policy, etc.)

• Problems
  – The middlebox needs to rewrite sequence numbers on both data and acks
  – Can’t do it if you don’t see ACK stream
Single path MPTCP

• Many MPTCP connections will still be single path
• Shouldn’t fall back to TCP if at the beginning there are no multiple paths
  – What if a new path becomes available?
  – What if an endpoint moves?
• With options encoding: flow looks roughly the same, albeit with some weird options
• With payload: TCP is now something else!
## Comparison Summary

<table>
<thead>
<tr>
<th>Topic</th>
<th>Options Signaling</th>
<th>Payload Signaling</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connection, Subflow Setup</td>
<td>Use options</td>
<td>Use options</td>
</tr>
<tr>
<td>Data ACK</td>
<td>Same as regular ACKs</td>
<td>Reliable, in order, congestion controlled =&gt; use options</td>
</tr>
<tr>
<td>TCP = Single path MPTCP</td>
<td>Nearly the same as TCP today</td>
<td>Uses TLV encoding</td>
</tr>
<tr>
<td>Future middleboxes</td>
<td>Plays nice</td>
<td>More difficult</td>
</tr>
<tr>
<td>Security Negotiation (and other big signaling)</td>
<td>Use payload</td>
<td>Use payload</td>
</tr>
<tr>
<td>Mboxes that allow new options on SYNs but not on data packets</td>
<td>Stops using path</td>
<td>Keeps going</td>
</tr>
<tr>
<td>Add/Remove IP</td>
<td>Echo option, send max one/RTT or Sequence numbers &amp; acks</td>
<td>Send max one/RTT or Sequence numbers &amp; acks</td>
</tr>
</tbody>
</table>
Options: What Next?

• We can fit all options needed for multipath in the existing options space

• However we can’t fit:
  – Security negotiation
  – Any other options we might need in the future

• Moving forward:
  – Option 1: do nothing 😊
  – Option 2: we could try to extend the options space (tcp-edy-loo). We could mandate that using multipath implies using extending options.
  – Option 3: we could allow chunks of the payload to be used as extensions of the options space
Payload Encoding: What Next?

• We need to allow options encoding to avoid having to give reliability to signaling that doesn’t require it (data ACK obvious example)

• Payload encoding that allows options encoding:
  a) TLV type that says: look at TCP options. How do you tie it with packets?
  b) Receiver always parses TCP options too
     • Breaks the layering
     • What do you do about reliability here?

• How do we play nice with middleboxes?
  – Must use option (b)
What should we standardize?