An Approach to Identify Services Provided by IETF Transport Protocols and Congestion Control Mechanisms

draft-welzl-taps-transports-00

Michael Welzl, Michael Tuexen and Naeem Khademi
TAPS WG, IETF 94 -- Yokohama
Scope of the I-D

• TAPS WG charter item (1): Define a set of Transport Services, identifying the services provided by existing IETF protocols and congestion control mechanisms. As a starting point, consider services used between two endpoints.

• As a supplement/complement to draft-ietf-taps-transports

• -00 includes TCP and SCTP (but there will be more in -01)
Goal of the I-D

• Using a generic approach, develop a document that can be used by TAPS system designers/API developers to map the services between the protocols and know how to use them in each protocol.

• Answers questions that arise when building a TAPS system.
Transport Service Identification Approach

- **draft-welzl-taps-transports** follows a three-pass approach

- **Pass 1**: relevant parts of the protocol’s RFCs are summarized, focusing on *what* a protocol provides to the upper layer and *how* it is used.

- **Pass 2**: categorizes the services from Pass 1 based on whether they relate to a *connection* or to *data transmission*.

- **Pass 3**: presents the superset of all services in all protocols, based on the list in Pass 2 but also on text in pass 1 to include services that can be configured in one protocol and are static properties in another.
Approach: Pass 1

- **Identify** services provided by TCP/SCTP. Here *service* is every form of defined interaction between a transport protocol and its user (ULP or application).
- **Exclude** some services that SHOULD NOT be implemented (e.g. URGENT mechanism (RFC6093)), or are optional or implementation-dependent, or already provided elsewhere.

<table>
<thead>
<tr>
<th>TCP (RFC1122, RFC0793)</th>
<th>SCTP (RFC4960)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open</td>
<td>Associate</td>
</tr>
<tr>
<td>Send</td>
<td>Send</td>
</tr>
<tr>
<td>Receive</td>
<td>Receive</td>
</tr>
<tr>
<td>Close</td>
<td>Shutdown</td>
</tr>
<tr>
<td>Abort</td>
<td>Abort</td>
</tr>
<tr>
<td>Close event</td>
<td>Change-Heartbeat/Request-Heartbeat</td>
</tr>
<tr>
<td>Abort event</td>
<td>Set Protocol Parameters</td>
</tr>
<tr>
<td>...</td>
<td>Set Primary Status</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
**Approach: Pass 2**

- Categorize the services from Pass 1 based on whether they relate to a *connection* or to *data transmission*
- **Format:** CATEGORY.[SUBCATEGORY].SERVICENAME.PROTOCOL

<table>
<thead>
<tr>
<th>TCP</th>
<th>SCTP</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONNECTION.ESTABLISHMENT.CONNECT.TCP</td>
<td>CONNECTION.ESTABLISHMENT.CONNECT.SCTP</td>
</tr>
<tr>
<td>CONNECTION.AVAILABILITY.LISTEN.TCP</td>
<td>CONNECTION.AVAILABILITY.LISTEN.SCTP</td>
</tr>
<tr>
<td>CONNECTION.MAINTENANCE.CHANGE-TIMEOUT.TCP</td>
<td>CONNECTION.MAINTENANCE.CHANGE-TIMEOUT.SCTP</td>
</tr>
<tr>
<td>DATA.SEND.TCP</td>
<td>DATA.SEND.SCTP</td>
</tr>
<tr>
<td>DATA.RECEIVE.TCP</td>
<td>DATA.RECEIVE.SCTP</td>
</tr>
<tr>
<td>CONNECTION.MAINTENANCE.DISABLE-NAGLE.TCP</td>
<td>CONNECTION.MAINTENANCE.REQUESTHEARTBEAT.SCTP</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
## Approach: Pass 2

- **Format:** CATEGORY.[SUBCATEGORY].SERVICENAME.PROTOCOL

- For every *service Pass 2* defines command/event, [parameters], [returns] and comments according to the generic/abstract APIs.

### CONNECT.TCP

**Command / event:** 'open' (active) or 'open' (passive) with destination transport address, followed by 'send'

**Parameters:**
- 1 local IP address (optional);
- 1 destination transport address (for active open; else the destination transport address and the local IP address of the succeeding incoming connection request will be maintained);
- timeout (optional);
- options (optional)

**Comments:** If the local IP address is not provided, a default choice will automatically be made. The timeout can also be a retransmission count. The options are IP options to be used on all segments of the connection. At least the Source Route option is mandatory for TCP to provide.

### CONNECT.SCTP

**Command / event:** 'initialize', followed by 'associate'

**Parameters:**
- list of local transport addresses (initialize);
- 1 destination transport address;
- outbound stream count

**Returns:** destination transport address list

**Comments:** 'initialize' needs to be called only once per local transport address list. One destination transport address will automatically be chosen; it can later be changed in MAINTENANCE.
## Approach: Pass 3

- Present the superset of all services in all protocols, based on the list in Pass 2 but also on text in Pass 1 to include services that can be configured in one protocol and are static properties in another.

<table>
<thead>
<tr>
<th><strong>e.g. CONNECTION.AVAILABILITY</strong></th>
<th><strong>e.g. DATA.RECEIVE</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AVAILABILITY:</strong> preparing to receive incoming connection requests.</td>
<td><strong>DATA.RECEIVE:</strong> fills a buffer provided to the application, with what we here call a &quot;message&quot;.</td>
</tr>
</tbody>
</table>
|  o Listen, 1 specified local interface  
  Protocols: TCP, SCTP |  o Receive data  
  Protocols: TCP, SCTP |
|  o Listen, N specified local interfaces  
  Protocols: SCTP |  o Choice of stream to receive on  
  Protocols: SCTP |
|  o Listen, all local interfaces (unspecified)  
  Protocols: TCP, SCTP |  o Message identification  
  Protocols: SCTP  
  Comments: in SCTP, this is optionally achieved with a "stream sequence number". The stream sequence number is always provided in case of partial message arrival. |
|  o Obtain requested number of streams  
  Protocols: SCTP |  o Information about partial message arrival  
  Protocols: SCTP  
  Comments: in SCTP, partial messages are combined with a stream sequence number so that the application can restore the correct order of data blocks an entire message consists of. |
Next steps for -01 and beyond?

• Using 3-pass approach we can derive services from any text that talks about what protocol provides and how it’s used.

• What protocols to include? Widely implemented protocols
  – From the TAPS ML discussions: UDP, UDP-Lite, MPTCP, DTLS, TLS
  – DCCP doesn’t have a well-defined API, (maybe it doesn’t matter, use anything from RFC4340, RFC4336?)
  – Other protocols from draft-ietf-taps-transports?
    • ICMP
    • RTP
    • Multicast protocols (FLUTE/ALC, NORM)
    • HTTP/TCP

• Adopting as WG item?