I-D.tiesel-taps-socketintentents-01

TAPS • IETF • Singapore 2017–11–14
Automated Transport Option Selection

• Choosing paths, endpoints and protocol stack as part of a TAPS system

• Socket Intents provide input on application needs
  • Optimize for bandwidth, latency, or cost
  • Traffic characteristics
  • Tolerance towards packet/data/connection loss
  • Cost preferences
Socket Intents 101

Socket Intents are pieces of information that allow an *application* to express *what it know* about the application’s communication.

Socket Intents are applications’ *hints* for Transport Option Selection

- Intuitive
- Generic
- Protocol Independent
- Best Effort

➢ Standardized but extendable set of hints that could be used by transport option selection
Socket Intents Integration

• BSD Sockets: Socket Options (prototype exists)
• Post-Socket: Address Resolution Configuration
• NEAT: Neat Properties
Socket Intents Granularity

Socket Intents can be applied at any granularity

- Message
- Stream
- Association
- Association Set
Socket Intents Representation

- Key-Value Pairs
- Key: Lower case ASCII string (needs IANA registry)
- Value: Fixed type per key
  - Enum
  - Int
  - Float
  - String
  - Sequence

Philipp S. Tiesel <philipp@inet.tu-berlin.de>
## Socket Intents Types

<table>
<thead>
<tr>
<th>Intent Type</th>
<th>Data Type</th>
<th>Applicable Message</th>
<th>Granularity Stream</th>
<th>Assoc</th>
<th>Assoc.Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Traffic Category</td>
<td>Enum</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Size to be Sent</td>
<td>Int (bytes)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Size to be Received</td>
<td>Int (bytes)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Duration</td>
<td>Int (msec)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Bitrate Sent</td>
<td>Int (bytes/sec)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Bitrate Received</td>
<td>Int (bytes/sec)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Burstiness</td>
<td>Enum</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Timeliness</td>
<td>Enum</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Disruption Resilience</td>
<td>Enum</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Cost Preferences</td>
<td>Enum</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
Open Questions

• Are Socket Intents easy enough?

• Is the structure of Socket Intent Types sufficient to express all anticipated future non-requirement performance preferences and application knowledge?

• What Socket Intent Types / kinds of information are missing?
Open Questions

• Does the current abstract definition of Socket Intents fit the way the IETF specifies abstract APIs?
Interactions Socket Intents vs. QoS

*Socket Intents are not QoS labels – they are orthogonal.*

- Socket Intents SHALL be purely advisory.
- Socket Intents MUST NOT be used to derive IntServ / RSVP style guarantees.
- Socket Intents SHOULD be taken into account on a best-effort basis and MAY be used to derive DiffServ Service.