An Architectural Perspective on MPTCP

Janardhan Iyengar

Franklin and Marshall College

jiyengar@fandm.edu

Bryan Ford

Max Planck Institute for Software Systems and Yale University

baford@mpi-sws.org

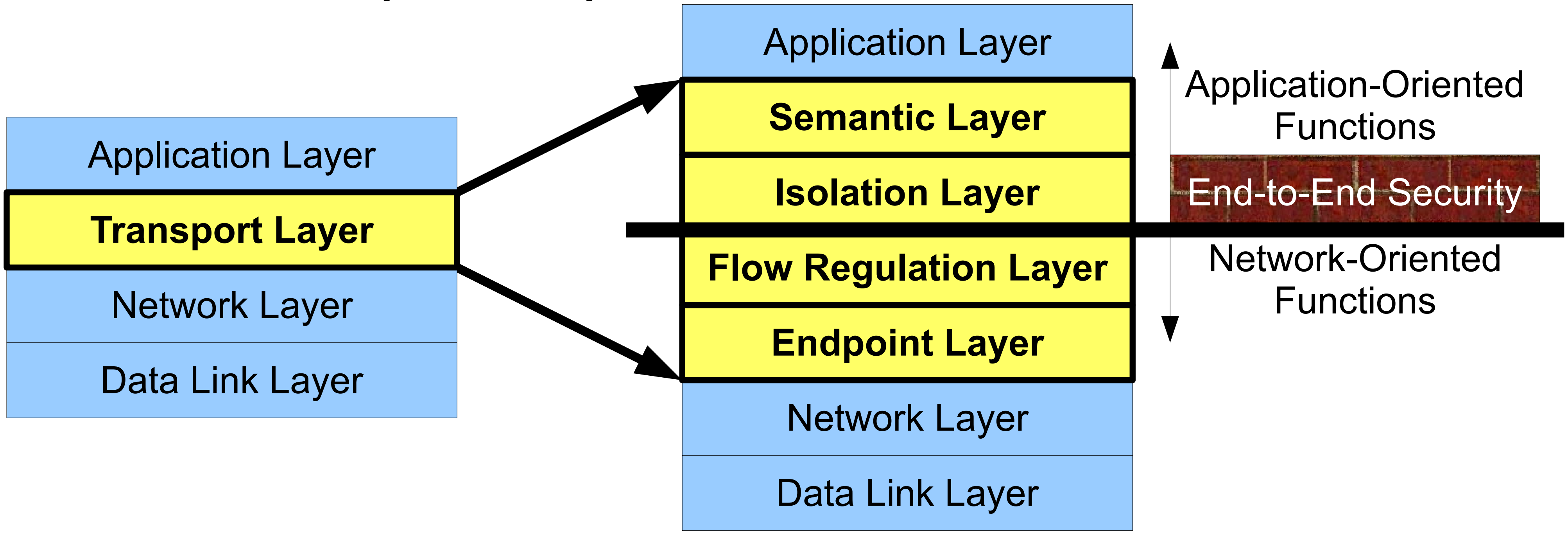
Draft: http://tools.ietf.org/html/draft-iyengar-ford-tng-00

Presentation for MPTCP BoF at IETF75, 30 July 2009

Tng: *Transport next-generation*

Refactor transport layer to match reality

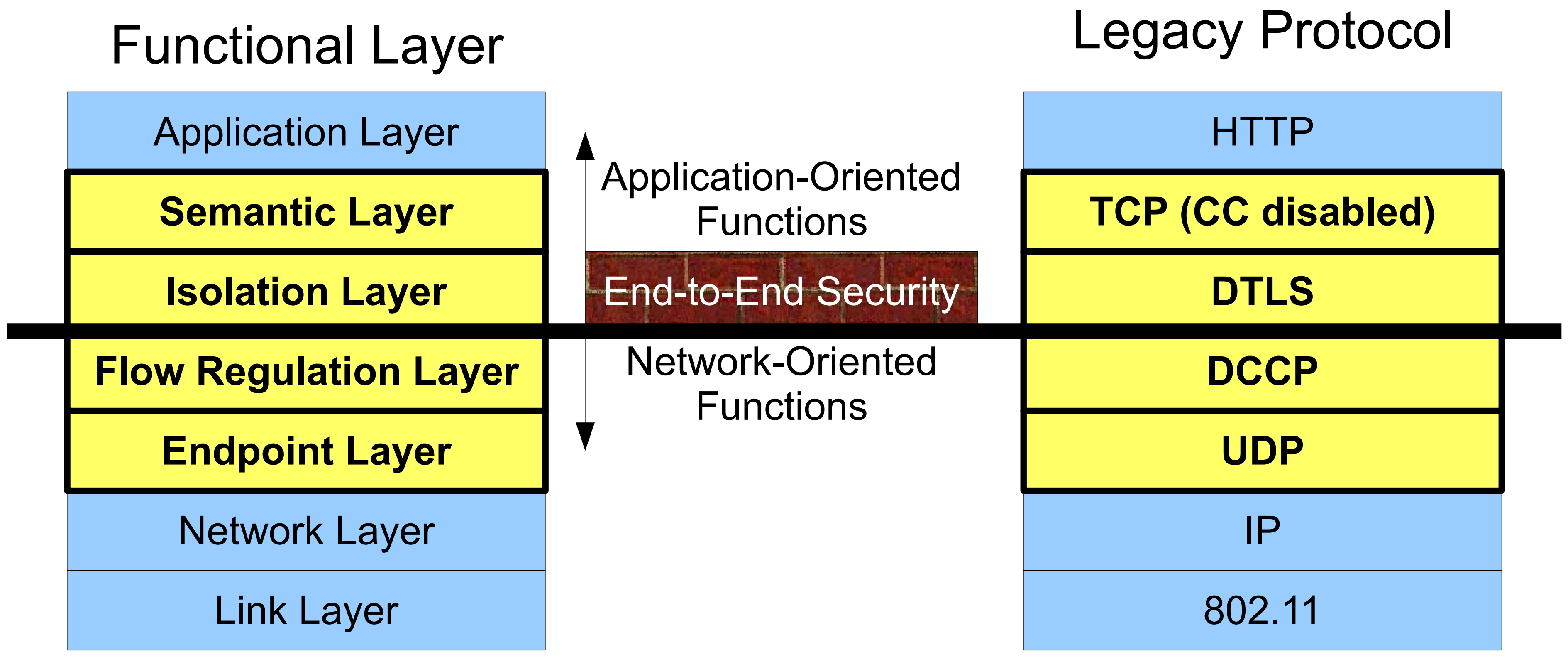
- **Network-oriented** functions of interest to middleboxes
  - Endpoints (ports); flow regulation (congestion control)
- **Application-oriented** functions serving the endpoints
  - Reliability/security



Example T*ng* Protocol Stack

Can implement Tng using only "legacy" protocols

– Workable design; not ideal in function or efficiency



Multipath in T*ng*

- The Semantic Layer
  - creates separate flows over multiple paths
  - manages e2e state across these flows, and bundles flows for shared cong. control

```
                   +---------+----------+
Application        |  HTTP, SMTP, etc.  |
                   +---------+----------+
                             |
                   +---------+----------+
Semantic           |    Byte Stream     |
                   +---------+----------+
                             |
            Path 1    Path 2 |  Path 3    Path 4
              /---------/----+----\---------\
              |         |         |         |
           +------+  +------+  +------+  +------+
           | Per- |  | Per- |  | Per- |  | Per- |
Flow       | Path |  | Path |  | Path |  | Path |
           |  CC  |  |  CC  |  |  CC  |  |  CC  |
           +------+  +------+  +------+  +------+
              |         |         |         |
           +------+  +------+  +------+  +------+
Endpoint   | Port |  | Port |  | Port |  | Port |
           +------+  +------+  +------+  +------+
              |         |         |         |
           +------------------------------------+
Network    |            IP Forwarding           |
           +------------------------------------+
```

Figure 2

An architectural perspective on MPTCP

- We can find MPTCP's **functions** in our stack:
  - MPTCP functionally fits in the Semantic Layer
  - TCP functionally satisfies the requirements of the Flow and Endpoint Layers

```
+-----------------------------+     +-----------------------------+
|         Application         |     |         Application         |
+-----------------------------+     +-----------------------------+
|            MPTCP            |     |          Semantic           |
+ - - - - - - + - - - - - - - +     +-----------------------------+
|     TCP     |      TCP      |     |  Flow+Endp   |  Flow+Endp   |
+-----------------------------+     +-----------------------------+
|      IP     |      IP       |     |      IP      |      IP      |
+-----------------------------+     +-----------------------------+
```

Why should this perspective be important to the proposed MPTCP wg?

1. Sheds light on implications of MPTCP's design

2. Cleanly separates functional units in MPTCP, opening up space for further exploration and for other protocols in the multipath architecture

MPTCP design implications: Example 1

MPTCP currently does not use end-to-end acks, relying on flow-level acks instead

- Middleboxes are known to ack optimistically
- MPTCP's goal is to recover from path failures
- If such a middlebox fails after acking and before successfully transmitting the acked data forward, MPTCP will not be able to recover from this failure

- This is a specific MPTCP design choice; a *Semantic Protocol* wanting e2e reliability MUST do e2e acks.

MPTCP design implications: Example 2

With apps that open multiple MPTCP connections, number of open TCP flows multiplies

- For example, HTTP client opens 4 MPTCP connections and each MPTCP connection opens up 2 flows
- Even with properly weighted congestion control:
  - Port numbers consumed faster: esp. problematic for NATs
  - Increased state: esp. problematic at webserver
  - As number of synchronized concurrent TCP connections increases, CC behavior can get more noisy/erratic

MPTCP design implications

- Again, a different *Semantic Protocol*, such as SCTP or SST, might provide a clue: multiplex lightweight “streams” (ala SCTP and SST) over several underlying flows

```
             +------------------------------------------+
Application  |                    HTTP                  |
             +----------+---------------------+---------+
                        |                     |
                 +------+------+       +------+------+
Semantic         |   Stream 1  |       |   Stream 2  |
                 +------+------+       +------+------+
                        |                     |
                        \----------+----------/
                                   |
               Path 1      Path 2  |   Path 3      Path 4
                 /-----------/-----+-----\-----------\
                 |           |           |           |
             +------+    +------+    +------+    +------+
             | Per- |    | Per- |    | Per- |    | Per- |
Flow         | Path |    | Path |    | Path |    | Path |
             |  CC  |    |  CC  |    |  CC  |    |  CC  |
             +------+    +------+    +------+    +------+
                 |           |           |           |
             +------+    +------+    +------+    +------+
Endpoint     | Port |    | Port |    | Port |    | Port |
             +------+    +------+    +------+    +------+
                 |           |           |           |
             +------------------------------------------+
Network      |               IP Forwarding              |
             +------------------------------------------+
```

Why should this perspective be important to the proposed MPTCP wg?

1. Sheds light on implications of MPTCP's design

2. Cleanly separates functional units in MPTCP, opening up space for further exploration and for other protocols in the multipath architecture

Further exploration in multipath transport

- Where does security belong in MPTCP?
- We propose using TLS above the individual TCP subflows:
  - Each subflow appears as a regular TLS/TCP flow in the network
  - If rowdy middlebox on one path messes with TLS, MPTCP simply drops that path and session continues
  - End-to-end MPTCP state is protected
- This is merely one design, several others will likely be proposed

Further exploration in multipath transport (... 2)

- MPTCP preserves compatibility with TCP at the API AND at the network levels.
- Cleanly separating these has implications for further evolution:
  - SCTP is multihome-capable and, with appropriate mods, can be used at the *Semantic Layer* with TCP subflows
  - SCTP and SST have richer APIs and can be used at the *Semantic Layer* above TCP subflows
  - DCCP can be used as a better *Flow Layer* with MPTCP, SST or SCTP at the *Semantic Layer*

Conclusion

- We would like to see an architectural document be part of this proposed wg's efforts.
  - Describes MPTCP's arch, relevant design choices
  - Shows how other protocols can fit in this arch
- *Simply exposing the architecture and design is sufficient; other parties can use it to build from other protocols.*
- Starting points:
  - MPTCP design draft: *http://www.cs.ucl.ac.uk/staff/C.Raiciu/files/mtcp-design.pdf*
  - Tng draft: *draft-iyengar-ford-tng-00.txt*