Recursion and the Transport Tussle

Joe Touch
Postel Center Director
USC/ISI
Research Assoc. Prof.
USC CS and EE/Systems Depts.
Overview

- Internet stack isn’t forked (Ford)
- Flows separate from interfaces (Kalim)
- Networking is recursive (Touch, Day/Matta)

- Conclusion: *info. that is missing or merged right now should be added*
  - *at every layer*
Why recursive?

- Layers of a stack becoming more similar
  - Security, soft-state, pacing, retransmission
- Desire to support new capabilities
  - Interlayer cooperation, dynamic layer selection
- Desire to support emerging abstractions
  - Overlay layers don’t map to 1-7
  - Support for recursive nodes (BARP, LI SP, TRI LL)

Is layering more than a coding artifact?
**Recursive Network Architecture**

- **“Resolve” unifies:**
  - Layer address translate/resolution
    - ARP, IP forwarding lookup
    - BARP/LISP/TRILL lookup
  - Layer alternates selection
    - IPv4/IPv6, TCP/SCTP/DCCP/UDP
  - Iterative forwarding
    - IP hop-by-hop, DNS recursive queries
- **“Process data” unifies:**
  - Shared state, security, management
  - Flow control, error control

---

```
 LAYER(DATA, SRC, DST)
 Process DATA, SRC, DST into MSG
 WHILE (Here <> DST)
   IF (exists(lower layer))
     Select a lower layer
     Resolve SRC/DST to next layer S’, D’
     LAYER(MSG, S’, D’)
   ELSE
     FAIL /* can’t find destination */
   ENDIF
 ENDWHILE
/* message arrives here */
RETURN {up the current stack}
```
RNA Stack

- One MP, many instances – all **LATE BOUND**
  - Unifies routing, forwarding
  - Unifies connections, provisioning
  - Unifies name resolution (Google, DNS, ARP, etc.)

RNA mp-1
RNA mp-2
RNA mp-3
RNA mp-4

wireless

RNA mp-1’
RNA mp-2
RNA mp-3
RNA mp-4

optical
A picture is worth...

- Two kinds of layer info:
  - **THRU**: Edge/path-relevant
    - More than ID
    - Also state start/end
  - **TO**: End-relevant
    - Flow/cong., etc.
- Also unify the layers
  - At least green boxes ;-)
Implications

- Allow network to see flow, flow state
  - Not for correctness; only for performance
  - Only endpoints NEED this info*
- A service is an ENTIRE stack
  - No such thing as “transport indep.” apps
  - Need full stack agreement (svc discovery)
- Legacy implications?
  - No new answers here
*What’s an endpoint?

- Difference between HW/SW?
  - HW = that which can be kicked
  - SW = everything else

- End vs. middle?
  - End = that which *I* can kick
  - Middle = everything else (to me, at least)
NAT Implications

- To the public side, it IS the endpoint
  - MUST avoid IP ID reuse, obey TIME-WAIT…
- To the private side, it is a router (mostly)
  - SHOULD decrement the TTL, manage ICMPs

- Implications: protocols that modify the endpoint will need to modify NATs
  - Except if you encaps., but that’s M.A.D.