A Recursive Network Architecture

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

- Towards future network architectures
- Background on X-Bone VNks
- RNA
  - Intro.
  - Design
  - Implementation
  - Implications
  - Related work
Towards future network architectures
What makes an architecture new?

- Shaking the Hourglass (CCW 08)
  - All exchanges are 1 packet
  - Collosograms > RTT*delay
  - No LANs? (L2 is only pt-pt)

- What defines success?
  - Fixing what's 'broken'
  - Doing something new/different
  - The Internet / circuits as a degenerate case
Internet Architecture

Accused of ossification, but:

- Ossification = stability
- Flexibility is abundant:
  - Shim layers:
    - HIP, SHIM6, IPsec, TLS
  - Muxing layers:
    - SCTP, RDDP, BEEP
  - Connections:
    - MPLS, GRE, IKE, BEEP, SCTP
  - Virtualization:
    - L2VPN, L3VPN/X-Bone/RON/Detour, L7-DHTs
Motivation

- Desire to support new capabilities
  - Interlayer cooperation, dynamic layer selection, layering created by virtualization
- Desire to support emerging abstractions
  - Overlay layers don’t map to 1-7
  - Support for recursive nodes (BARP, LISP, TRILL)
- Desire to coordinate services in diff. places
  - Security, soft-state, pacing, retransmission
Observations

- Networking is *groups of interacting parties*
  - Groups are heterogeneous
  - All members want to interact
  - Groups can be dynamic (*i.e.*, virtual)
- Need an architecture that supports:
  - Heterogeneity
  - Interaction
  - Virtualization
Heterogeneity leads to layering

- M different interacting parties need
  - $M^2$ translators
  - $M$ translators + common format

... i.e., a layer
Interaction leads to forwarding

- N parties need
  - $N^2$ circuits
  - $O(N)$ links + forwarding

or
Virtualization leads to recursion

- N parties want to group in arbitrary, dynamic ways.
  ... such groups are inherently virtual

... and virtualization is inherently recursive
What makes this an architecture?

- Abstraction for virtualization
  - Tunnel as link
  - Partitioned router as virtual router
  - Partitioned host + internal router as virtual host
- Abstractions for recursion
  - Recursive router implemented as a network of vouters with vhosts at the router interfaces
  - Recursion within the protocol stack
- General template (metaprotocol + MDCM)
  - Instantiates as different layers or forwarding
X-Bone Virtual Nets
Virtual Net Req’ts

- **Internet-Compliant Architecture**
  - Hosts add/delete headers
  - Routers transit (constant # headers)

- **Supports New Capabilities**
  - Concurrence (multiprocessing)
  - Revisitation (multiple roles in one net)
  - Recursion (to hide topology and/or mgt.)
VN Principles

- **TENET 1. Internet-like**
  - VIs = VRs + VHs + tunnels
  - Emulating the Internet

- **TENET 2. All-Virtual**
  - Decoupled from their base network

- **TENET 3. Recursion-as-router**
  - Some of VRs are VI networks
VN Corollaries

- **Behavior:**
  - VH adds/deletes headers
  - VRs transit (constant # headers)

- **Structure:**
  - VIs support concurrence
  - VIs support revisitation
  - Each VI has its own names, addresses
    - Address indicates overlay context
VN Architecture

- Components:
  - VH -> hosts include a hidden router
  - VL -> 2 layers of encaps. (strong link, weak net)
  - VR -> partitioned forwarding

- Capabilities:
  - Revisitation -> multihoming for VNs
  - Recursion -> router as network, i.e., Rbridges, LISP

>> RUNNING CODE (FreeBSD, Linux, Cisco)
Recursive Internet

Control / deployment
- Recursion as a router (vs. ASes)
- Network recursion examples
  - L3 = BARP (X-Bone), LISP (IRTF)
  - L2 = Rbridges/TRILL

Network
Recursion requires new layers – where? Why?

- Wedge between (IPsec, left) or replicate (virtualization, right)
Challenges of Layering

- Which to add…
  - IPv4/IPv6, TCP/DCCP/SCTP
- When to add…
  - Security, muxing, cong. control
- Real vs. virtual
  - What’s the difference?
Scope defines a layer

- Its endpoints
  - A “hop” @layer N = E2E extent of layer N-1
- The layer above
  - What services this layer provides
- The layer below
  - What services this layer requires
- E.g.: Shared state at diff. layers for diff. services
  - Application binding
  - Transport delivery
  - Net security

The difference is scope
RNA Intro.
Motivation for RNA

- 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, LISP, TRILL)

Is layering more than a coding artifact?
Observations

1. Services are relative

2. A template can avoid recapitulation

3. Composition requires coordination
Recapitulation

- Component services repeat:
  - handshake / state management
  - security
  - policy (admission control, filtering)
  - multiplexing and demultiplexing
  - retransmission
  - reordering
  - pacing / congestion control
  - switching / forwarding
- Compounded by virtualization
  - Layer on layer on layer
Composition Requires Coordination

- Many services integrate layers
  - Congestion control
  - Message boundaries
  - Security
  - State establishment
- Current interlayer interface is limited
  - Defined by each layer
  - No general security, state, etc. interface
RNA Stack

- One MP, many instances
  - Needed layers, with needed services
  - Layers limit scope, enable context sensitivity
  - Scope defined by reach, layer above, layer below
What does RNA enable?

- Integrate current architecture
  - ‘stack’ (IP, TCP) vs. ‘glue’ (ARP, DNS)
- Support needed improvements
  - Recursion (AS-level LISP, L3 BARP, L2 TRILL)
  - Revisitation
- Supports “old horses” natively
  - Dynamic ‘dual-stack’ (or more)
RNA Design
Structured template w/plug-in functions

- 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

```plaintext
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 Metaprotocol

- Template of basic protocol service:
  - Establish / refresh state
  - Encrypt / decrypt message
  - Apply filtering
  - Pace output via flow control
  - Pace input to allow reordering
  - Multiplex/demultiplex
    - includes switching/forwarding
Components of RNA MP

Instantiate MDCM’s “Process DATA”

- Establish / refresh state
- Encrypt / decrypt message
- Apply filtering
- Pace output via flow control
- Pace input to allow reordering
- Multiplex/demultiplex as indicated
  - includes switching/forwarding
RNA Implementation
START PATTERN MIN

# This simply specifies a buffer. no reordering etc.
PATTERN MIN
   REQ MUST BUFFER 1
   ARG BUFFER 1 VAR size 1000
   LINK ADD SELF 0 BUFFER 1
...

# Next use this pattern if MIN is successful
PATTERN ORDERED_DELIVERY
   follows MIN
   REQ MUST REORDERING 1
   LINK DEL ....
   LINK ADD ....
...

# If reordering successful, try more stuff…
PATTERN ENCRYPTED_ORDERED_DELIVERY
   follows ORDERED_DELIVERY
   REQ MUST ENCRYPTION 1
   ARG ENCRYPTION 1 VAR algo des
   ARG ENCRYPTION 1 VAR keysize 512
   ....
Instantiation

Standardized Protocol

RNA Meta-Protocol

Protocol Instance

Implicit, Design-time Context, Goals, Constraints

Explicit, Run-time Context, Goals, Template
Click Implementation

Composition Graph

Conf File
Compose What

mux
demux
buffer
Scheduler
Data API
Utilities
Composi-
tion Logic
Control API
Parser

Click

Protocol
Compose Recursively

m1
m2
e3
e4
Building a Stack

(a) Instance 1

(b) Instance 1

(c) Instance 2

Source

Sink

m1

m2

e1

e2

e3

e4
Composition Process
RNA Implications
RNA – fills the gaps

- Between layers (left, from *Choices*)
  - Affects next-layer
- Between stacks (right, from Padlipsky)
  - Affects next-hop
Recursion supports Layering and Forwarding

- **Layering (left)**
  - Heterogeneity via $O(N)$ translators
  - *Requires successive recursive discovery*

- **Forwarding (right)**
  - $N^2$ connectivity via $O(N)$ links
  - *Requires successive iterative discovery*
Challenges

- MP design
  - Building a sensible, generic template
- Stack management
  - Supporting instantiation and composition
- Supporting interlayer coordination
  - Designing a sensible, recursive API
  - Makes it easier to interface (to yourself, e.g., LEGO)
- Supporting context sensitivity
  - Detecting environment and autotuning
Related Work
Related Work Summary

- Recursion in networking
  - X-Bone/Virtual Nets, Spawning Nets, TRILL, Network IPC, LISP
  - RNA natively includes resolution and discovery

- Protocol environments
  - Modular systems: Click, x-Kernel, Netgraph, Flexible Stacks
  - Template models: RBA, MDCM
  - RNA adds a constrained template with structured services

- Context-sensitive components
  - PEPs, Shims, intermediate overlay layers, etc.
  - RNA incorporates this into the stack directly

- Configurable über-protocols
  - XTP, TP++, SCTP
  - RNA makes every layer configurable, but keeps multiple layers.
RNA and Network IPC

- **Similarities**
  - Recursive protocol stack
  - Unified communication mechanism
  - Focus on process-to-process interaction

- **Differences**
  - RNA uses MDCM to define IPC as combining a Shannon-style channel with namespace coordination
  - RNA provides a detailed (and demonstrated) mechanism that achieves unification and recursion
  - RNA supports both recursion and forwarding in a single mechanism
Protocol & Transit Domains

Protocol Domain (H1 → H2)

Multi-Hop Protocol Domain (S → D)

Transit Domain T1

Transit Domain T2

Protocol Domain M1

Protocol Domain M2
Conclusions

- Virtualization requires recursion
- Recursion supports layering
- Recursion supports forwarding

One recurrence to bind them all...

- Recursion is a native network property
  - Integrates and virtualization, forwarding and layering in a single mechanism