From Shannon to Recursive Nets: Multihop/Multiparty Influences on Net Arch.

Joe Touch
USC/ISI

with Venkata Pingali, Yu-Shun Wang, Lars Eggert
Outline

- Background
- Principles of multihop/multiparty comms
- RNA
  - Concept
  - Design / Implementation
  - Related Work
- Conclusions
Background
What makes an architecture new?

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

- What defines success?
  - fixing what's 'broken'
  - doing something new/different
  - the Internet / circuits as a degenerate case
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
Shannon Channel

- Two preselected parties
  - Homogenous endpoints

- Unidirectional channel
  - Preselected sender, preselected receiver
What is communication?

- Shannon: shared bits
  - Between fixed endpoints, known \textit{a priori}

- Shared bits between two parties
  - How do we find the party to talk to?
What SCs Ignore

- What if you’re not directly connected?
  - A) multihop
  - B) multilayer

- Why are multihop/multilayer interesting?
  - Scalable = multihop
  - Ubiquitous = multilayer
  - I.e., all scalable, ubiquitous comms!
Exploring Invariants

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

- $M$ different interacting parties need
  - $M^2$ translators

  or

  - $M$ translators + common format

... *i.e.*, a layer
Layering leads to resolution

- IDs are local to a layer
  - Whether names, paths, locations
- Need to resolve IDs between layers
  - Google, DNS, ARP, LISP encap tables
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
Recursion unifies layering, forwarding, & resolution

- Layering (left)
  - Heterogeneity via $O(N)$ translators
  - *Supported by successive recursive resolution*
- Forwarding (right)
  - $N^2$ connectivity via $O(N)$ links
  - *Supported by successive iterative resolution* (tail recursion)
Recursion requires new layers – where? Why?

- Wedge between (IPsec, left) or replicate (virtualization, right)
What if…

- Über-protocols are the right idea…
  - A single configurable protocol with
    - Hard/soft state management
    - Congestion control, error management
    - Security
  - E.g., XTP, TP++
- But they went too far…
  - Keep layering – because of first principles
RNA – concept
RNA

- One metaprotocol, many instances
  - Needed layers, with needed services
  - Layers limit scope, enable context sensitivity
  - Scope defined by reach, layer above, layer below
  - Resolution connects the layers (red/green)
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
What makes this an architecture?

- General template (metaprotocol + MDCM)
  - Instantiates as different layers or forwarding
- Abstraction for virtualization
  - Tunnel as link
  - Partitioned router as virtual router
  - Partitioned host + internal router as virtual host
- Abstraction for recursion
  - Recursive router implemented as a network of vrouters with vhosts at the router interfaces
RNA MP Unifies...

- "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'
        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
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?

- Explains and details invariants
  - Layering as more than a SW Engr. artifact
- Integrate current architecture
  - ‘stack’ (IP, TCP) vs. ‘glue’ (ARP, DNS)
- Support needed improvements
  - Recursion (AS-level LISP, L3 BARP, L2 TRILL)
  - Revisititation (X-Bone)
  - Concurrence (VPNs, multipath TCP)
- Supports “old horse” challenges natively
  - Dynamic ‘dual-stack’ (or more)
The Hourglass Principle

- Common interchange format between layers
Multiple hourglasses

- “Waist” is relative
  - The common interchange = the waist
Click Implementation

Conf File
Compose What

Composition Graph
mux
demux
buffer
Scheduler
Data API
Utilities
Parser
Composition Logic
Control AP

Click

Protocol
m1
m2
e3
e4
Compose Recursively
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
  ....
Building a Stack

(a) Source Sink
(b) Instance 2
(c) Source Sink
Composition Process

1. Discover Context/Goal
2. Alter Composition?
   - Yes: Feasible?
     - Yes: Select Template
     - No: (Re)Compose
   - No: Send/Recv Data
Related Work
Related Work

- 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
Other Components

- Dynamic negotiation protocol
  - Cross-layer negotiation, IETF TAE
- Composable/recursive extensions
  - Network management/SLAs
  - Security (user/infrastructure)
  - Non-comm services (storage, computation)
- Integrated optimization
  - Caching, precompute/prefetch
  - Pinning, dampening
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
Discussion Questions
Define a "science of networking" (SON)

- Informally:
  - Principles we’d teach to besides “here’s an artifact we built”

- Formally:
  - Abstract principles and fundamentals of multiparty communication
Fundamental of a SON

- State coordination
  - 3-way handshake, soft state, delta-T
  - *All as “convergence of shared state”*

- Error control and recovery
  - FEC, ACK/NAK, sliding window
  - *All as “refinement of shared state”*

- Flow and policy control
  - Pacing, SLA enforcement, authorization, window scale
  - *All as “maintenance of shared state”*
Contributions to SON

- Latency management
  - Trading information structure, predictability, and capacity for delay
- Virtualization
  - Unifying strong/weak models of addressing
- Recursion
  - Unifying forwarding, layering, recursion, resolution
Ignored SON Aspects

- Almost everything...
  - Most comm work is artifact, not architecture
  - Teaching focuses on tools, not principles
- Foundational principles missing
  - Lack of generalized concepts
- Expand Shannon
  - Shared state as more than symbol sequence
  - Extend shared state to determining endpoints
SON Changes What?

- Teaching
  - See current textbooks to see why

- Tools
  - Start to build reusable components based on key concepts, not forced playgrounds

- Testbeds
  - Helps us focus effort on shared utility

- Architectures and Protocols
  - Won’t confuse artifacts with approaches