Recursive InterNetwork Architecture

A policy-based recursive approach to building a better Internet

Abraham Matta
College of Arts & Sciences
Computer Science
Boston University

March 2014
Collaborators

PhD Students:
- Yuefeng Wang
- Nabeel Akhtar

Alumni:
- Flavio Esposito, Exegy
- Karim Mattar, Akamai
- Vatche Ishakian, BBN
- Gowtham Boddapati, Akamai
- Gonca Gursun, Ozyegin U. (Turkey)
- Joseph Akinwumi

Faculty
- John Day
- Lou Chitkushev

Outside Collaborators:
- Eduard Grasa, i2CAT, Spain
- Eleni Trouva, i2CAT, Spain
- Steve Bunch, TRIA Network Systems
- Peter DeWolf, TRIA
- Miguel Ponce de Leon, TSSG, Ireland
- Patrick Phelan, x-TSSG, Ireland
- Xavier Hesselbach-Serra, UPC, Spain
- Louis Pouzin, father of “connectionless” networking

PSOC pouzin society
What’s wrong with today’s Internet?

- The brave new world
  - larger scale, more diverse technologies
  - new services: content-driven, cloud-based, context-aware, mobile, socially-driven, secure, profitable, …

- Custom point-solutions: No or little “science”

- Lots of problems: Denial-of-service attacks, bad performance, hard to manage, …
Questions

- Is the Internet’s architecture fundamentally broken that we need to “clean slate”?
- Can we find a new architecture that is complete, yet minimal? If so, what is it?
- Can we transition to it without requiring everyone to adopt it?

- YES
- RINA (Recursive InterNetwork Architecture)?
  - Based on “Networking is inter-process communication” --Robert Metcalfe ’72
- YES
Talk Outline

- Problems with today’s Internet architecture
- Our Recursive IPC-based Net Architecture
  - one IPC layer that repeats over different scopes
- One Data Transfer Protocol (DTP)
  - soft-state (ala Delta-t) approach
- One Common Distributed Application Protocol (CDAP)
  - stateless (ala CMIP), used by management applications
  - naming & addressing
  - multihoming, mobility
- Prototyping, evaluation, conclusions
Talk Outline

- Problems with today’s Internet architecture
  - Our Recursive IPC-based Net Architecture
    - one IPC layer that repeats over different scopes
  - One Data Transfer Protocol (DTP)
    - soft-state (ala Delta-t) approach
  - One Common Distributed Application Protocol (CDAP)
    - stateless (ala CMIP), used by management applications
    - naming & addressing
    - multihoming, mobility
- Prototyping, evaluation, conclusions
Internet’s view: one big, flat, open net

<table>
<thead>
<tr>
<th>Application</th>
<th>Web, email, ftp, …</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transport</td>
<td>TCP, UDP, …</td>
</tr>
<tr>
<td>Network</td>
<td></td>
</tr>
<tr>
<td>Data Link</td>
<td></td>
</tr>
<tr>
<td>Physical</td>
<td></td>
</tr>
</tbody>
</table>

- There’s no building block
- The “hour-glass” model imposed a least common denominator
- We named and addressed the wrong things (i.e., interfaces)

![Diagram showing network layers and IP addresses]
Internet’s view: one big, flat, open net

- We exposed addresses to applications
- We hacked in “middleboxes”
- Built a network of boxes, rather than networks of processes

- IP protocol: TCP, UDP, …
Ex1: Bad Addressing & Routing

Want to send message to “Bob”

Alice → “Bob” → I₁

To: I₁

Bob multi-homed destination

- Naming “interfaces”
  - application bound to a path (point-of-attachment address)
  - huge routing tables

- Hard to deal with multihoming and mobility
Network Address Translator aggregates private addresses

NAT acts as firewall
- preventing attacks on private addresses & ports
- causing so-called “layer violations”

Hard to coordinate communication across domains when we want to

Network Address Translator (NAT) aggregates private addresses. It acts as a firewall by preventing attacks on private addresses and ports, and it causes so-called “layer violations.” It’s hard to coordinate communication across domains when we want to initiate a connection.
Talk Outline

- Problems with today’s Internet architecture
- Our Recursive IPC-based Net Architecture
  - one IPC layer that repeats over different scopes
  - One Data Transfer Protocol (DTP)
    - soft-state (ala Delta-t) approach
  - One Common Distributed Application Protocol (CDAP)
    - stateless (ala CMIP), used by management applications
    - naming & addressing
    - multihoming, mobility
- Prototyping, evaluation, conclusions
Our Solution: divide-and-conquer [ReArch/CoNEXT’08]

- Application processes communicate over Distributed IPC Facility (DIF)
  - a distributed application that does IPC
  - DIF management is internal ➔ better security

- IPC processes are application processes to lower DIF’s
  - Recurse as needed ➔ better management & scalability

- Well-defined interfaces ➔ predictable service
Recursive Architecture based on IPC

DIF = Distributed IPC Facility (locus of shared state=scope)
Policies are tailored to scope of DIF
RINA allows scoping of services

- The DIF is the building block (layer) and can be composed
  - A DIF has all that is needed to manage a network, i.e. it integrates routing, transport and management

- **E2E (end-to-end principle) is not relevant!**
  - Each DIF layer provides transport flow service/QoS over its scope

- **IPv6 is/was a waste of time!**
  - Each DIF layer has its private addresses
What goes into a DIF?

- Processing at 3 timescales, decoupled by either a **State Vector** or a **Resource Information Base**
  - IPC Transfer actually moves the data
  - IPC Control (optional) for transmission, error, flow control, etc.
  - IPC Management for routing, resource allocation, locating applications, access control, monitoring lower layer, etc.
Talk Outline

- Problems with today’s Internet architecture
  - one IPC layer that repeats over different scopes

- Our Recursive IPC-based Net Architecture
  - One Data Transfer Protocol (DTP)
    - soft-state (ala Delta-t) approach
  - One Common Distributed Application Protocol (CDAP)
    - stateless (ala CMIP), used by management applications
    - naming & addressing
    - multihoming, mobility

- Prototyping, evaluation, conclusions
- RINA decouples port allocation and access control from data synchronization and transfer
- At each end, port and conn ID are allocated dynamically and bound to each other by management (using CDAP) in a hard-state fashion
Once allocated, Data Transfer can start following Delta-t [Watson’ 81], a soft-state protocol

- **Timers are necessary and sufficient** for data synchronization and transfer
- **Flows without data transfer control** are UDP-like. Different **policies** support different requirements
- If there is a long idle period, conn state is discarded, but ports remain
- Conn IDs can be changed during data transfer and bound to same ports
RINA: Good Transport leads to Better Security [NPSec/ICNP’12]

- In RINA, requesting applications never see addresses nor conn IDs
  - No well-known ports
  - Ports, dynamically allocated, are not part of conn IDs
  - Service requested by application name
  - Traditional port scanning attacks not possible
    - Scanning application names is much more difficult, far larger name space
RINA: Good Transport leads to Better Security

- In RINA, state of data transfer is soft, and conn IDs are allocated dynamically (and can change on the fly)
  - Need to be authenticated and member of the DIF
  - No explicit control messages to fabricate
  - Conn IDs are hard to guess
  - Conn opening and data transfer attacks are harder to mount
Talk Outline

- Problems with today’s Internet architecture
- Our Recursive IPC-based Net Architecture
  - one IPC layer that repeats over different scopes
- One Data Transfer Protocol (DTP)
  - soft-state (ala Delta-t) approach
- One Common Distributed Application Protocol (CDAP)
  - stateless (ala CMIP), used by management applications
  - naming & addressing
  - multihoming, mobility
- Prototyping, evaluation, conclusions
Only One Application Protocol

- For management applications, need only one “stateless” (soft-state) application protocol to access objects
  - It does Read/Write, Create/Delete, Start/Stop
- The objects are outside the protocol
Each DIF is privately managed

- It assigns private node addresses to IPC processes
- It internally maps app/service name to node address

• An address is a synonym for an IPC process whose scope is limited to the DIF and may be structured to be “useful” within the DIF
RINA: Good Addressing – recursive …

Want to send message to “Bob”

- Node address mapped to PoA (point-of-attachment) address
- Roles are relative: node address is name for lower DIF, and PoA for higher DIF
- Processes on a system are members of various DIF layers
RINA: Better Scalability & Security – secure containers

- Nothing more than applications establishing communication
  - Authenticating that A is a valid member of the DIF
  - Initializing it with current DIF information
  - Assigning it an internal address for use in coordinating IPC
  - This is enrollment, i.e., explicit negotiation to join DIF (access control)
- RINA decouples authentication from connection management and integrity/confidentiality
Good Design leads to Better Security

- In RINA, underlying IPC processes must be authenticated to join DIF
  - only “insider” attacks possible
  - a hurdle that is not present in TCP/IP networks

- Authentication and encryption are applied recursively – no “shim” sublayers
Back to naming-addressing basics [Saltzer ’82]
- Service/app name (location-independent)
- Node address (location-dependent)
- PoA address (path-dependent)
- Path

We clearly distinguish the last 2 mappings

Route: sequence of node addresses

Next-hop node address is mapped to PoA by lower DIF
Mobility is Inherent

- Mobility is a dynamic form of multihoming
- Mobile joins new DIF layers and leaves old ones
- Local movement results in local routing updates
Mobility is Inherent

- Mobility is a dynamic form of multihoming
- Mobile joins new DIF layers and leaves old ones
- Local movement results in local routing updates
Mobility is Inherent

- Mobility is a dynamic form of multihoming
- Mobile joins new DIF layers and leaves old ones
- Local movement results in local routing updates
Simulation Results: RINA vs. LISP vs. … [FutureNet-III’10, CC’12]

- RINA inherently limits the scope of location update & inconsistency
  - LISP (loc/id split): “loc” is still path-dependent!
- RINA uses “direct” routing to destination node
Talk Outline

- Problems with today’s Internet architecture
  - one IPC layer that repeats over different scopes
- One Data Transfer Protocol (DTP)
  - soft-state (ala Delta-t) approach
- One Common Distributed Application Protocol (CDAP)
  - stateless (ala CMIP), used by management applications
  - naming & addressing
  - multihoming, mobility

- Prototyping, evaluation, conclusions
ProtoRINA [TR-2013-013,NSDI’13,GREE’13,GREE’14]

- Overview
  - Boston University’s user-space prototype of the RINA architecture
  - Experimental tool for new (non-IP) applications / policies
  - Teaching tool for networking and distributed systems classes

- Status
  - cross-debugging with two other RINA prototypes (IRATI and TRIA)
  - around 55,000 lines of Java code
  - not complete; we continue to modify/add elements
  - code and user manual available online
RINA Node

- Distributed Application Facility (DAF): Distributed Application Processes cooperating to perform a certain function: communication, weather forecast, genomics, etc.

- A DIF is a specific DAF whose job is only to provide IPC
IPC Process

Provides communication service for application processes or higher level IPC processes
Applications use it to allocate/deallocate flows, send/receive data, register/unregister their service.

IRM Interface

```java
public int allocateFlow(Flow flow);
public void deallocateFlow(int handleID);
public void send(int handleID, byte[] msg) throws Exception;
public byte[] receive(int handleID);

public void registerApplication(ApplicationProcessNamingInfo apInfo, Flow);
public void deregisterApplication(ApplicationProcessNamingInfo apInfo);
```
RINA API: RIB Daemon

- Applications use this publish/subscribe API to access objects in a local RIB, or remote RIB (using CDAP)
- Configuration files allow for selecting different policies (authentication, routing, etc.)

RIB Daemon Interface

```java
public int createEvent(SubscriptionEvent subscriptionEvent);
public void deleteEvent(int subscriptionID);
public Object readSub(int subID);
public void writePub(int pubID, byte[] obj);
```

```java
rina.routing.protocol = linkState
rina.routingEntrySubUpdatePeriod = 2
rina.linkCost.policy = hop
```
Dynamic Formation of a Virtual Private Cloud DIF

- In RINA, Flow Allocation may involve instantiation of an underlying DIF, if one does not exist
A new application acts as a “relay” IPC
Dynamic Formation of a Virtual Private Cloud DIF
Policy-based Dynamic Service Management
[TR-2013-014]
Policy-based Dynamic Service Management

Network Management
Policy-based Dynamic Service Management

Customer Application DAF

Virtual Private Cloud DIF

Enterprise DIF

App1

VPC1

E1

E2

VPC2

VPC3

Cloud Provider DIF

CP1

CP2

Naming Management
Policy-based Dynamic Service Management

Customer Application DAF

App1 -> VPC1

E1
Enterprise DIF

E2

VPC2

CP1
Cloud Provider DIF

App2 -> VPC3

CP2

Application Management
ProtoRNA over the GENI Testbed

- Large-scale experimentation for correctness and performance
- Run ProtoRNA within a long-lived “slice” over GENI
  - researchers and educators can opt-in and experiment with programmable management policies
Example: One-level DIF Topology

- Link-state updates sent every 10 seconds
Example: Two-level DIF Topology

- 0-DIFs: Link-state updates sent every 10 seconds
- 1-DIF: Link-state updates sent every 5 seconds
GENI Resources

- each RINA node on one VM from NYU aggregate
  - nine VM’s (one runs a “naming” service)
Effect of DIF Mgmt & Routing Policies

Scoping (2-level DIF) yields faster convergence and less OOO packets with similar routing overhead*

* Generally lower routing overhead for larger multi-level DIF topologies
How does RINA compare?

- **Related work**: many, but not holistic
  - We claim RINA is *more complete and minimal*
  - RINA subsumes the mechanisms and policies of other architectural proposals

- **Security**: DIF is a *secure container* of coordinated IPC processes
  - *RINA supports secure address spaces as in XIA*

- **Manageability**: DIF defines a *scope* that is locally managed
  - RINA separates mechanisms from policies
  - RINA has one recursive layer configurable with policies
    - beyond “middleware”, “tunneling”, “cross-layer” approaches
  - *RINA supports virtual “slices” as in Nebula*
How does RINA compare? (2)

- **Scalability**: the multi-level DIF structure limits the scope of control and management
  - Routing table size of a system depends on only those DIFs to which its IPC processes join

- **Content-based**: service/app name is location-independent, node address is not path-dependent
  - beyond “loc/id split” approaches
  - RINA supports multihoming and mobility as in Serval or Mobility-First, but inherently via local routing updates
  - RINA supports content discovery as in NDN

- **Socially-driven / Cloud-based**: DIF is dynamically formed to enable IPC among cloud / peer processes
How does RINA compare? (3)

- **Adoption**: DIF is an overlay, internally managed using only two protocols: data transfer and management.

- **Network neutrality**: not relevant
  - User has a choice of which DIF to join
  - DIF differentiates its services via policies – mechanisms are the same

- **Marketplace**: individual services of DIFs can be (recursively) composed to offer new user services


References


