“Networking is IPC”:
A Guiding Principle to a Better Internet

Internet 1.0 is broken
Internet 2.0 is a repeat with more b/w
How about Internet 3.0?

Ibrahim Matta
Computer Science
Boston University

Joint work with John Day, Karim Mattar, Gonca Gursun,
Vatche Ishakian, Joseph Akinwumi
What this talk is (NOT) about

- NOT about specific protocols, algorithms, interfaces, implementation
  - about protocol structure, performance models

- It’s about architecture, i.e., objects and how they relate to each other

- It’s based on the **IPC model**, not a specific implementation

“Networking is inter-process communication”
--Robert Metcalfe '72
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
  - soft-state (ala Delta-t) approach
- One Common Application Protocol
  - stateless, used by management applications
- Naming & addressing
  - multihoming, mobility
- Security, adoptability, 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
  - soft-state (ala Delta-t) approach
- One Common Application Protocol
  - stateless, used by management applications
- Naming & addressing
  - multihoming, mobility
- Security, adoptability, conclusions
A Fundamentally Broken Architecture

- Bunch of hacks
  - No or little “science”

- Lots of problems
  - Denial-of-service attacks, bad performance, hard to manage, ...

- Why?
  - Too big, too flat, too open
  - We’re seeing what happened with Wall Street...
Ex1: Bad Addressing & Routing

Want to send message to “Bob”

Alice → “Bob” → I₁

Bob

multi-homed destination

I₁

I₂

To: I₁

- Naming “interfaces” - i.e., binding objects to their attributes (Point-of-Attachment addresses) - makes it hard to deal with multihoming and mobility

- Destination application process identified by a well-known (static) port number
Ex2: Ad hoc Scalability & Security

- Network Address Translator aggregates private addresses
- NAT acts as firewall
  - preventing attacks on private addresses & ports
- But, hard to coordinate communication across domains when we want to
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**
  - soft-state (ala Delta-t) approach
- **One Common Application Protocol**
  - stateless, used by management applications
- Naming & addressing
  - multihoming, mobility
- Security, adoptability, conclusions
Our Solution: divide-and-conquer

- Application processes communicate over IPC facility
- How IPC managed is hidden → better security

- IPC processes are application processes of lower IPC facilities
- Recurse as needed
  → better management & scalability

- Well-defined interfaces → predictable service
Architecture based on IPC

DIF = Distributed IPC Facility (locus of shared state=scope)
Policies are tailored to scope of DIF
What Goes into an IPC Layer?

- 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 error, flow control, etc.
  - **IPC Management** for routing, resource allocation, locating applications, access control, monitoring lower layer, etc.
Two-system Case
Multi-system Case

Host 1

- Application Process
- Application Protocol
- Port ID

IPC Process
Mux

Host 2

- Application Process
- Application Protocol
- Port ID

IPC Process
Mux

Relaying Element
Relay-and-mux instances

Physical Link
Only 3 Kinds of Systems

- **Host**, **internal router**, **border router**
- No middleboxes, no NATs, ....
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
  - soft-state (ala Delta-t) approach
- One Common Application Protocol
  - stateless, used by management applications
- Naming & addressing
  - multihoming, mobility
- Security, adoptability, conclusions
Good we split TCP, but we split TCP in the wrong direction!

We artificially isolated functions of same IPC / scope

We artificially limited the number of layers / levels
TCP was partly split to separate “hard-state” from “soft-state”

- Hard-state must be explicitly discarded
- But we don’t need it to be [Watson ‘81]
- Watson proves that if 3 timers are bounded:
  - Maximum Packet Lifetime \( (MPL) \)
  - Maximum number of Retries \( (G) \)
  - Maximum time before Ack \( (UAT) \)
  - That no explicit state synchronization, i.e., hard-state, is necessary
    - SYN, FINs are unnecessary
- In fact, TCP uses all these timers and more
- TCP is really hybrid HS+SS
Five-Packet Protocol (ala TCP)

- Explicit handshaking: SYN and SYN+ACK messages
- For single-message communication, TCP uses five-packet protocol + timers (HS+SS)
- Vulnerability: Aborted connections 😞

\[ 4 \ast \text{channel-delay} \]
Delta-t Protocol (Watson '81)

- A pure SS approach
- Two-Packet Protocol (Belsnes '76) with timers
  - Assumes all connections exist all the time
  - TCBs are simply caches of state of ones with recent activity
- $G = n \times RTO$
- $R_{time} = 2MPL + G + UAT$
- $S_{time} = 3MPL + G + UAT$

$R_{time} \sim 2 \cdot MPL > 4 \text{ channel-delay}$
**Analytical Model**

- **Worst-case single-message communication**
- **Only the initial messages (with DATA) can get lost**
- **A new conn starts when previous one ends**
- **p:** loss prob.  **D:** channel delay

<table>
<thead>
<tr>
<th></th>
<th>Delta-t Protocol</th>
<th>Five-Packet (TCP) Protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\lambda_r$</td>
<td>$(1-p)/D$</td>
<td>$(1-p)/D$</td>
</tr>
<tr>
<td>$\lambda_i$</td>
<td>$p / D$</td>
<td>$p / D$</td>
</tr>
<tr>
<td>$\lambda_i$</td>
<td>$(1-p)/RTO$</td>
<td>$(1-p)/RTO$</td>
</tr>
<tr>
<td>$\omega$</td>
<td>$1/MPL$</td>
<td>$1/MPL$</td>
</tr>
<tr>
<td>$\mu_r$</td>
<td>$1/\text{Rtime}$</td>
<td>$1/4D$</td>
</tr>
</tbody>
</table>

$\lambda_r$ = arrival rate of initial message at the receiver  
$\lambda_i$ = loss rate of initial message  
$\lambda_i$ = successful retransmission rate of initial message  
$\mu_r$ = connection state removal at the receiver  
$\omega$ = connection state removal at the sender
Analytical Model

- Goodput
  \[ v = \pi_{(\ast,-)} / D \]

- Message Rate
  \[ \phi_{\text{five}} = \frac{1}{D} \pi_{(*)} + \frac{1}{RTO} \pi_{(*)} + \frac{1}{Rtime} \pi_{=} \]
  \[ \phi_{\text{delta}} = \frac{1}{D} \pi_{(*)} + \frac{1}{RTO} \pi_{(*)} + \frac{1}{Rtime} \pi_{=} \]

- Receiver State Lifetime
  \[ \eta = \frac{1}{v} \pi_{=} \]
Analytical Model Results

- Tradeoff between memory overhead/goodput and message rate
Simulation Results: Correctness

- Two-state channel-delay model, random initial sequence numbers

- SS (Delta-t) is more robust to bad net conditions
Simulation Results: Performance

- Goodput won’t be limited given a reasonable conn ID space
- Memory requirement is not a concern
  - only 1.2MB needed at Delta-t receiver (server) in a typical setting

May 11, 2009
Only one Transfer Protocol

- Allocating conn ID (ports) is done by management, IPC Access Protocol (IAP)
- Once allocated Data Transfer can start, ala Delta-t
  - Flows without data transfer control are UDP-like. Different policies support different requirements
- If there is a long lull, state is discarded, but ports remain
**All protocols are soft-state**

- 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
  - Other “protocols” may access the same objects
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
  - soft-state (ala Delta-t) approach
- One Common Application Protocol
  - stateless, used by management applications
- Naming & addressing
  - multihoming, mobility
- Security, adoptability, conclusions
We exposed addresses to applications
We named and addressed the wrong things
Compare to Current Stack (3)

- E2E (end-to-end principle) is not relevant
  - Each IPC layer provides service / QoS over its scope
- IPv6 is/was a waste of time!
  - We don't need too many addresses within a DIF
Good Addressing

want to send message to “Bob”

- Destination application is identified by “name”
- App name mapped to node name (address)
- Node addresses are private within IPC layer
- Need a global namespace, but not address space
- Destination application process is assigned a port number dynamically
Good Addressing

want to send message to “Bob”

- Late binding of node name to a PoA address
- PoA address is “name” at the lower IPC level
- Node subscribes to different IPC layers
Good Routing

- Back to naming-addressing basics [Saltzer '82]
  - Service name (location-independent) →
    node name (location-dependent) →
    PoA address (path-dependent) → path
- We clearly distinguish the last 2 mappings
- Route: sequence of node names (addresses)
- Map next-hop's node name to PoA at lower IPC level
Mobility is Inherent

- Mobile joins new IPC layers and leaves old ones
- Local movement results in local routing updates
Mobility is Inherent

- Mobile joins new IPC layers and leaves old ones
- Local movement results in local routing updates
Mobility is Inherent

- Mobile joins new IPC layers and leaves old ones
- Local movement results in local routing updates
Compare to loc/id split (1)

- Basis of any solution to the multihoming issue
- **Claim**: the IP address semantics are overloaded as both location and identifier
- LISP (Location ID Separation Protocol) '06

**Diagram:**
- **EID^x → EID^y**
- **RLOC^1_{EID_y} →**
- **RLOC^2_{EID_y} →**
- **EID^y → RLOC^2_y**

**Mapping:** EID^y → RLOC^2_y
Compare to loc/id split (2)

- Ingress Border Router maps ID to loc, which is the location of destination BR
- **Problem**: loc is path-dependent, does not name the ultimate destination

![Diagram showing network mapping with EIDs and RLOCs]

Mapping: EID\(^x\) \(\rightarrow\) EID\(^y\)

RLOC\(^1\)\(_{EID\(^y\)}\) RLOC\(^2\)\(_{EID\(^y\)}\)

EID\(^x\) \(\rightarrow\) EID\(^y\)

**Mapping**: EID\(^y\) \(\rightarrow\) RLOC\(^2\)\(_{EID\(^y\)}\)**
LISP vs. Our approach

- Total Cost per loc change = Cost of Loc Update +
  \( \rho [P_{cons} \times \text{DeliveryCost} + (1-P_{cons}) \times \text{InconsistencyCost}] \)

\( \rho \): expected packets per loc change

\( P_{cons} \): probability of no loc change since last pkt delivery

![Graph showing Total Cost (N = 15) vs. Arrival Rate for different protocols, including RINA, MobileIP, LISP, LISP-MIP, LISP-Cache, and LISP-CACHE-MIP. The graph highlights Our approach and the NxN Grid Topology.](image-url)
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
  - soft-state (ala Delta-t) approach
- One Common Application Protocol
  - stateless, used by management applications
- Naming & addressing
  - multihoming, mobility
- Security, adoptability, conclusions
Better Scalability & Security

- 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
Adoptability

- ISPs get into the IPC business and compete with host providers
- A user joins any IPC network she chooses
- All IPC networks are private
- We could still have a public network with weak security properties, i.e., the current Internet
- Many IPC providers can join forces and compete with other groups
Related Work

- Back to networking basics
  - Networking is IPC and only IPC [Metcalfe '72]
  - We apply this principle all the way!

- Back to naming-addressing basics
  - Extend [Saltzer '82] to next-hop routing on node addresses

- Back to connection management basics
  - Use soft-state approach [Watson '81] within a complete arch.

- Recursive [Touch et al. '06] but we recurse IPC over different scopes
  - Beyond existing stack, “middleware”, and “tunneling”

- Loc/id split [LISP '06] approaches
  - “loc” does not name the dest!
Current / Future Work

- Complete specification of IPC mechanism (data transfer & control) and management (routing, security, resource allocation, ...)

- Fast implementation
  - Minimize data copying, context switching, ...

- Declarative specification of policies
The Pouzin Society was formed ...

- [http://pouzinsociety.org/](http://pouzinsociety.org/)
- Email me ([matta@cs.bu.edu](mailto:matta@cs.bu.edu)) for more info
Thank You

Questions?