Abstractions for Network Routing

Impressions of Network Routing

Neo-Dadaisms for Network Routing

Absurdisms for Network Routing

See also: Postmodern Routing

[Bhattacharjee, Calvert, Griffioen, Spring, & Sterbenz]
What routing abstractions facilitate flexibility and evolvability?

How can we quantitatively compare architectures and abstractions? rather than just performance of an implementation
Setting the Stage
Routing Defined

Selection of path in network along which to send message
Routing Defined

Selection of services in network along which to send message
Components of Routing

Data plane

forwarding service

Control plane

routing service

service advertisement

service selection

forwarding action
Components of Routing

“Just” a distributed systems problem...

service
advertisement

service
selection

forwarding
action

forwarding
service

routing
service

follows from forwarding

Most fundamental abstraction

Who, where, how?
Components of Routing

Today: all components coupled at each router
Summary so far

Key questions:

• What’s the right abstraction of forwarding service?
• Who should choose the services and how?

Traditional (next-hop-style) networking: coupled

• Each router locally selects service, installs forwarding service, advertises directly to all recipients (neighbors)

Software Defined Network: decoupled

• [forwarding] [service advertisement] [service selection]

Interdomain: ???
What problem are we solving?

• What’s the **right** abstraction of forwarding service?
• Who **should** choose the services and how?

What do these this mean??
“Flexibility”
Today’s inflexible routing: BGP

Routing fixed within the network, leading to:

- Unreliability (long convergence)
- Inefficient resource allocation (prefix-level load balancing)
- Insecurity
  - Even with Secure BGP, traffic attraction attacks
  - Each domain’s security is dependent on the actions of many other domains between it and the destination

You get one path to each IP prefix, and this path may be broken, inefficient, or insecure.
Source routing for flexibility

Separate route computation from the network

- Route (i.e., selected services) is parameter given to the network
Source routing for flexibility

Reliability
source can switch quickly or use many

Path quality
source knows what it wants

Security
Each domain can independently protect itself

Lowest latency path
Highest bandwidth path
Path the network would have picked for you
Source routing challenges

Security

- Can attackers exploit route control? (Can defenders?)

Scalability

- How do sources quickly pick good paths without huge amounts of dynamic state distribution?
- “Eh.”

Route control tussle

- How can an architecture enable source control yet still provide sufficient network owner control of routing?
Solving the route control tussle

Pick one “reasonable” tradeoff between source and network control?

- then get everyone to agree...
- then standardize it...

Better solution: design for variation
“Design for variation in outcome, so that the outcome can be different in different places, and the tussle takes place within the design, not by distorting or violating it.”

— Clark, Wroclawski, Sollins & Braden, 2002
“Tussle in Cyberspace”
Pathlet routing

[Godfrey, Ganichev, Shenker, Stoica, SIGCOMM 2009]

**vnode** virtual node

**pathlet** fragment of a path: a sequence of vnodes

**Source routing** over pathlets.

**virtual graph:** flexible way to define policy constraints

provides many path choices for senders
vnode: virtual node within an AS

Walla Walla  New York
Crumstown
San Diego  Roosterville
vnode: virtual node within an AS

designated ingress vnode for each neighbor

Internally: a forwarding table at one or more routers
Pathlets

Packet route field

Forwarding table

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>7,2</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

... ...

3 push 7,2; fwd to B

... ...

7 fwd to C

... ...

2 fwd to D

delivered!
So what?

For network owners, **flexibility** to define how the network can be used.

For users, **flexibility** to choose paths or services.
Choice for senders

source

destination
Example: allow all valley free routes

E.g., all valley free routes
("customers can go anywhere; anyone can route to customer")
Example: flexible granularity

BGP

Pathlet routing

Router

AS

BGP announcement message
Flexible policies

128.2.0.0/16
Flexible policies
Flexible policies

128.2.0.0/16
Quantifying policy flexibility

“ We don’t know how to figure out whether one of our ideas is better than another. ”

— David Clark
Quantifying policy flexibility

Feedback-based routing

Pathlet routing

NIRA

MIRO

LISP

Routing deflections, path splicing

Loose source routing

Strict source routing

IP (BGP)
Quantifying policy flexibility

Feedback-based routing

Strict source routing

Loose source routing

MIRO

Pathlet routing

NIRA

Routing deflections, path splicing

LISP

IP (BGP)
“Evolvability”
Evolvability

Goal:

• Communication infrastructure for all of humanity

Only hope: evolve across time

• Ratnasamy, Shenker, McCanne [SIGCOMM’05]
• FII [CCR’11]
• OPAE [Ghodsi, Koponen, Raghavan, Shenker, Singla, Wilcox, HotNets’11]
• XIA [Anand, Dogar, Han, Li, Lim, Machado, Wu, Akella, Andersen, Byers, Seshan, Steenkiste, HotNets’11 & NSDI’12]

What is an evolvable architecture?
Our history: Not Good

IP options? Usually dropped

UDP? Sometimes dropped

Not HTTP? Sometimes dropped

...
Attacks on evolution

Useful frame of mind: Some parties will act to hinder evolution

• Apathy
• Security
• Government control

Therefore, should design architecture to defend against evolution attacks

• What abstraction yields “defensive evolvability”? 
Quantifying evolvability (Toy Model)

Node state

- Legacy
- Attacker
- Deployed New Protocol

When can we run the New Protocol along a path?

- Source runs N.P. and no attacker on path

Utility of a path to source

- 0 for old protocol
- ~ (#new hops) for new protocol
Attacks kill evolution: simulation

Simplistic simulation on CAIDA AS-level Internet topology (2011)
36,878 nodes, 103,485 edges
Attacks kill evolution: dynamics

Simplistic simulation on 500-node degree-5 random graph
1% initial deployment
Attacks kill evolution: dynamics

Simplistic simulation on 500-node degree-5 random graph
1% initial deployment
Case Study #1: Next-Hop Fwd’ing

Traditional IP routing & forwarding

- Each router selects one hop of path (= service)

**Result:** all routers along path know, agree to, and select the end-to-end service
Case Study #2: XIA

“How should a legacy router in the middle of the network handle a new principal type that it does not recognize?”

- Fallbacks:

Result: Each router is explicitly aware of novel services being deployed

- Analogous to IP options
- Potential result: drop anything “weird” (e.g., security risk)

XIA is flexible, but is it really evolvable?
Defensive Evolvability

Hammer: Modularity

- Hide functionality from those who need not see it

AS/user should be able to unilaterally deploy a new type of connectivity service

- ...without approval of parties used to reach that service
- ...and without them even knowing!

Rough solution: pathlets++

- Each segment is a general “function” rather than just a link between two vnodes
Putting together the pieces
Observations

1. Flexibility and evolvability come from modularity
   - “the degree to which a system's components may be separated and recombined” – wikipedia

2. The principal function of networks is connectivity

3. Need clean abstraction to recombine connectivity

4. Hypothesis: The current architecture lacks such an abstraction
   - Instead of one reusable abstraction, we keep inventing special-purpose tunnels: overlay networks, VPNs, ports, ...
Vasily Kandinsky
“Small Worlds”
1922