Improving Routing Scalability with Name-Oriented Sockets

Christian Vogt  Ericsson Research

IRTF Routing research group meeting. November 2009
2 Types of Identifier-Locator Separation

- **Type 1**: 'surrogate' address serves as ID and hides real addresses.
- **Type 2**: name-oriented sockets abstraction hides addresses.

- Main difference is application transparency.
- Growing deployment for type 2, not for type 1.
Transparency Implies Deployment Hurdle

transparency to application → extra namespace → extra resolution → extra security → extra infrastructure → more complex implementation → more administration → more dependencies → limited benefits for applications → deployment hurdle

application transparency does not aid deployment as commonly believed
Name-Oriented Sockets

- Applications use DNS names bilaterally
- IP address management at IP layer
- Standard IP packets

use DNS names

use addresses

name-oriented interface

name resolution

address selection

NAT traversal

address updates

applications use DNS names bilaterally

IP address management at IP layer

standard IP packets
New Interface For Applications

- **Listen method** — prepare for incoming session
  
  \[ \text{service handle} = \text{Listen} \left( \text{source name, destination name, local port, transport} \right) \]

- **Open method** — initiate outgoing session
  
  \[ \text{session handle} = \text{Open} \left( \text{source name, destination name, remote port, transport} \right) \]

- **Accept method** — receive incoming session
  
  \[ \left( \text{source name, destination name, session handle} \right) = \text{Accept} \left( \text{handle} \right) \]

- **Write method** — send data
  
  \[ \text{Write} \left( \text{session handle, data} \right) \]

- **Read method** — receive data
  
  \[ \text{data} = \text{Read} \left( \text{session handle} \right) \]

- **Close method** — close session
  
  \[ \text{Close} \left( \text{session handle} \right) \]
Other Components

- initial name exchange
- address updates
- backwards compatibility
- hosts without registered DNS name
- security
Initial Name Exchange

- **host.left.net** has address **a.b.c.d**
- **peer.right.net** has address **v.x.y.z**

**Application**

- Open from **host.left.net** to **peer.right.net**
- Resolve **peer.right.net**

**Operating System**

- Source **a.b.c.d** destination **v.x.y.z** from **host.left.net** to **peer.right.net**
- Source **v.x.y.z** destination **a.b.c.d** from **peer.right.net** to **host.left.net**
- Accept from **host.left.net** to **peer.right.net**
- Listen from any to **peer.right.net**
- Verify **host.left.net**
Initial Name Exchange

host.left.net has address a.b.c.d

peer.right.net has address v.x.y.z

**application**

**operating system**

open from host.left.net to peer.right.net
write

send names until receive packet

read
write

source a.b.c.d destination v.x.y.z
from host.left.net to peer.right.net
accept from host.left.net to peer.right.net
read
write

source v.x.y.z destination a.b.c.d
from peer.right.net to host.left.net

listen from any to peer.right.net

send names until receive packet without names
Address Updates

**host.left.net** starts with address `a.b.c.d`

**peer.right.net** has address `v.x.y.z`

---

**Application**

- **Write** from **host.left.net** to **peer.right.net**

**Operating System**

- **Dynamic DNS Update**

- **Address Update**
  - Source: `a.b.c.d`
  - Destination: `v.x.y.z`

- **Verification**

**Application**

- **Read** from **host.left.net** to **peer.right.net**

---

**Update Address for** **host.left.net**

- **Write** from **host.left.net** to **peer.right.net**
Backwards Compatibility

two types

1. legacy local application
   - old interface alongside new interface
   - remote peer sees legacy host

2. legacy remote peer
   - unilaterally name-oriented
   - address-derived name for peer
Backwards Compatibility

Host left.net has address a.b.c.d

Legacy host has address v.x.y.z

Application

Operating system

Open from host.left.net to v.x.y.z.names

Resolve peer.right.net

Read from host.left.net to v.x.y.z.names

Source a.b.c.d destination v.x.y.z

Source v.x.y.z destination a.b.c.d

Accept from a.b.c.d

Bind

Listen

Write
Security

- initial name exchange: DNS lookup by peer
  - same security as for initiating host
  - security strength depends on DNSSEC

- address update: return routability end to end
  - retains security of non-mobile Internet

- dynamic DNS update: crypto authentication
  - provisioned by hosting provider
New Dependencies On the DNS

- name-oriented stack will increase DNS load
  - more lookups
  - more dynamic updates

- scalability and convergence perhaps problematic
  - load increase never tested
  - low time-to-live values often not supported

- analysis results so far promising
  - load increase affects only lowest-level servers
  - missing time-to-live support fixable
Conclusion

- name-oriented sockets improve routing scalability
  - enable multi-homing and mobility
  - simplify renumbering

- good deployment prerequisites
  - backwards compatibility
  - advantages for application developers
  - incentives to change operating systems
  - no new infrastructure or administrative procedures
  - no dependency between stakeholders

- early prototype at Ericsson