Service-Centric Networking with SCAFFOLD

Michael J. Freedman, Matvey Arye, Prem Gopalan, Steven Y. Ko, Erik Nordström, Jennifer Rexford, and David Shue
Princeton University

Presented by: Justin Meden
Outline

● Motivation for Content Centric Networking
● Related Work
  ● DONA
  ● Networking Named Content
  ● Triad
● SCAFFOLD
● Summary
Motivation

- The Internet started out to facilitate remote resource sharing
- Today's Internet mostly consists of HTTP requests, media streaming, and DNS traffic
- Service connections are also important
  - Services can be replicated and moved like data
- There is a disconnect between how the network operates and how it is used
  - Use: Retrieve a specific data item or service
  - Operates: Retrieve a connection with a specific host
Goals CCN

- Remove the need to make DNS lookups
  - New naming system for services and data
  - Place the name lookup scheme in the network
- Route to one of many possible service instances
  - Any-cast routing to a service instance
  - Find closest instance
  - Balance load of instances
- Allow for service instances to move locations
- Allow for self-certifying name
Related Work

• **An Architecture for Content Routing Support in the Internet** USITS 2001
  Mark Gritter, David R. Cheriton

• **Networking Named Content** CoNext 2009
  Van Jacobson, Diana K. Smetters, James D. Thornton, Michael F. Plass, Nicholas H. Briggs, Rebecca L. Braynard

• **A Data-Oriented (and Beyond) Network Architecture** SIGCOMM 2007
  Mohit Chawla, Teemu Koponen, Byung-Gon Chun, Kye Hyun Kim, Scott Shenker, Andrey Ermolinskiy, Ion Stoica
An Architecture for Content Routing Support in the Internet

Gritter and Cheriton

- URL naming scheme of services and data
  - e.g. bar.foo.com

- Network is made up of Content Routers (CRs) and Network Routers
  - Content Routers map URL names to a next hop
  - Content Routers also forward IP packets as usual
  - All other network routers are left unchanged
Content Routers

- Route based on URL suffixes
  - Content Routers map a longest-suffix match to a next hop location
  - Content Routers make use of a protocol similar to BGP to announce paths along Content Routers to services
- Routing ends at a CR adjacent to the content server
  - Adjacent CR returns the IP address of the content server to the user
- Once the IP address has been returned to the initiating client, standard IP communication will commence
Networking Named Content
Jacobson et al.

- Routers in the network route as well as distribute content
  - Cache recently seen data
  - Route data names to a next hop location
  - Keep track of pending requests for data
- Hierarchical naming
  - e.g. `/parc.com/videos/WidgetA.mpg/v3/s0`
- Routers in the network map longest-prefix matches to next hop locations
Networking Named Content
Jacobson et al.

- Hosts send one of two packet types
  - Interest – A request for a specifically named data
  - Data – The data requested by an Interest packet
- Routers in the network have three main parts
  - Content Store
    - Cached data that the router has recently seen
  - Pending Interest Table
    - Interests, and where they came from, that have not been fulfilled by a data packet
  - Forward Information Base
    - A name prefix mapping to a next hop
## Networking Named Content

Jacobson et al.

### Content Store

<table>
<thead>
<tr>
<th>Name</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>/parc.com/videos/WidgetA.mpg/v3/s0</td>
<td>...</td>
</tr>
</tbody>
</table>

### Pending Interest Table (PIT)

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Requesting Face(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>/parc.com/videos/WidgetA.mpg/v3/s1</td>
<td>0</td>
</tr>
</tbody>
</table>

### Index

<table>
<thead>
<tr>
<th>ptr</th>
<th>type</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>

- C = Content store
- P = PIT
- F = FIB

### FIB

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Face list</th>
</tr>
</thead>
<tbody>
<tr>
<td>/parc.com</td>
<td>0, 1</td>
</tr>
</tbody>
</table>

### Faces

- Face 0
- Face 1
- Face 2

- Application
DONA

- Service Identifiers are pairs: P:L
  - The first element, P, is a hash of the owner's public key
  - The second element is a label assigned by the owner
- Networks maintain Resolution Handlers (RHs)
  - Services issue REGISTER(P:L) commands to RHs
  - Clients issue FIND(P:L) commands to RHs
Resolution Handlers

• A single logical Resolution Handler exists in a local network

• Maps service identifiers to a next hop and a distance to the copy of the service
  • If no mapping exists, the FIND request is sent to the parent RH (towards the core)
  • Tier 1 ASes must maintain a mapping for all service identifiers

• RHs can also be used to cache recently requested data
Name Resolution
SCAFFOLD

- Want to provide similar functionality to related work
  - Multiple service instances represented by one name
  - Service-selection through named based Anycast routing
- Also want to allow dynamics in the network
  - Server mobility
  - Client mobility
  - Connection recovery from failures
Naming

• SCAFFOLD makes use of serviceIDs
  • Fixed length
  • Location-independent
  • One serviceID can map to multiple service instances
• Like most flat named systems, serviceIDs allows for self-certifying instances of a service
• ServiceIDs are mapped to service instances by Service Routers
Service Routers

- Similar to Resolution Handlers and Content Routers
- Service routers resolve serviceIDs into end host IP addresses
  - Service router only has knowledge of services in the local network
  - Forwards requests towards the service after name resolution
- Services send register and unregister requests so service routers can keep track of instances of a service
Service Router

Network Routers

HostAddr A

HostAddr B

HostAddr C

HostAddr D
SCAFFOLD Packets

• serviceID: The flat-name for the given service

• hostAddr: Network attachment location.

• sockID: A specific flow's connection identifier
Setting up a connection

1. SYN
2. SYN
3. SYN-ACK
4. ACK
Setting up a connection cont.

**User-Space Process**

<table>
<thead>
<tr>
<th>Socket Descriptor</th>
<th>Local Service ID</th>
<th>Remote Service ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>U</td>
<td>X</td>
</tr>
<tr>
<td>9</td>
<td>U</td>
<td>X</td>
</tr>
<tr>
<td>47</td>
<td>U</td>
<td>Y</td>
</tr>
</tbody>
</table>

**Network Stack**

<table>
<thead>
<tr>
<th>Socket State</th>
<th>Local SrvID</th>
<th>Local Addr : SockID</th>
<th>Remote SrvID</th>
<th>Remote Addr : SockID</th>
</tr>
</thead>
<tbody>
<tr>
<td>bound</td>
<td>U</td>
<td>A : p</td>
<td>X</td>
<td>B : r</td>
</tr>
<tr>
<td>bound</td>
<td>U</td>
<td>A : q</td>
<td>X</td>
<td>C : r</td>
</tr>
<tr>
<td>connecting</td>
<td>U</td>
<td>A : r</td>
<td>Y</td>
<td>---</td>
</tr>
</tbody>
</table>
End-to-end design

- Unlike the presented related work, SCAFFOLD introduces a partial end-to-end design.
- By keeping track of connections with specific service instances on the network stack, a stable host can maintain a connection with service that is dynamic.
- The authors call this flow affinity: Maintaining a flow between service instances even when a service changes hosts.
Mobility

- If one of the end hosts changes addresses
  - The changed host sends an RSYN packet to the stationary host
  - The RSYN contains the new address and a new socket ID
  - The stationary host updates its network stack and sends an ACK for the RSYN
- The end-to-end design allows for persistent flows even when hosts are mobile
Mobility Update

HostID A

HostID B

HostID C

HostID D

RSYN-ACK

RSYN

SRC X D q

DST U A p

U A w

X D q
SCAFFOLD in the WAN

• Introduce the notion of a Service System
  • Manages a subset of serviceIDs that an identity owns
  • Assigned a globally unique identifier called Service System IDs (SSids)
  • SSid form the higher order bits of a ServiceID

• A Service System is an administrative identity
  • e.g. Microsoft, Google, Amazon
  • Not physically constrained like an AS
SCAFFOLD in the WAN
SCAFFOLD in the WAN

- SCAFFOLD does not implement how a client determines which AS to send a request to.

- The authors suggest the following:
  - A DNS like structure that maps ssIDs onto ASes that host that ssID's services.
  - Use a global routing protocol where service routers announce the ssIDs they associate with.
Experimental Setup
Figure 7: High availability with two clients and two servers, showing how a client is transparently redirected to another service instance as failure happens.
Figure 9: Memcached Server Throughput. Server 2 joins the network after around 15 seconds; server 1 leaves after 30 seconds. In both cases, the network transparently redistributes the data partitions (named by unique serviceIDs) over the available servers.
Figure 8: Replicated service support with 2 clients and 3 servers showing load-balancing as additional servers are added, request shedding for planned maintenance, and the residual effects of lingering requests with draining.
## Comparisons

<table>
<thead>
<tr>
<th></th>
<th>SCAFFOLD</th>
<th>DONA</th>
<th>Cheriton/Gritter</th>
<th>Jacobson</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Naming System</strong></td>
<td>Flat names</td>
<td>Flat names</td>
<td>URL</td>
<td>Hierarchical</td>
</tr>
<tr>
<td><strong>Routers Return</strong></td>
<td>Connection</td>
<td>Connection/Content</td>
<td>Address of Host</td>
<td>Content</td>
</tr>
<tr>
<td><strong>Content</strong></td>
<td>At Hosts</td>
<td>At Hosts/Cached</td>
<td>At Hosts</td>
<td>At Hosts/Cached</td>
</tr>
<tr>
<td><strong>Host Mobility</strong></td>
<td>RSYN/Net. Stack</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td><strong>Mainly Concerns</strong></td>
<td>Services</td>
<td>Services/Data</td>
<td>Services</td>
<td>Data</td>
</tr>
<tr>
<td><strong>Content Routers</strong></td>
<td>1 logical router</td>
<td>1 logical router</td>
<td>Many</td>
<td>Many in a network</td>
</tr>
</tbody>
</table>
Questions?