

# (CCN) Naming

Deep dive discussion

April 24, 2013

# What is a name?

*A way to identify something*

# Categories of names

## Structured

You can interpret/parse the structure and infer some relationship between the elements

## Flat

No way to interpret the name other than by exact match

# Types of name components

## Structured

You can interpret/parse some meaning from the component

## Flat

No way to interpret the component other than by exact match

Application

Vertical solutions to problems  
Normally do not contain general  
networking code

Library

Used by many applications  
Only required at endpoints (hosts)  
Inner nodes might implement some of it

Core

Exists at every node in the network  
Forwarders and routers

Naming

**ELEMENTS**

# Controlled Elements

- Identifier
- Date/Time
- Version
- Organization/Relationship to other content
- Security/Permissions/Signatures
- Domain/Scope (for interpretation)
- Size (?)

# Inherited Elements

- Author
- Membership
- Location
- Date/Time (?)
- Size (?)



Naming

**FEATURES**

# Match

- Can it provide exact content match?
- Can it provide partial content match?
- Can it provide exact name match?
- Can it provide partial name match?
- Prefix-match? Regex? Set match? Ordered match?
- Can you scope the match?

# Uniqueness

- Is it required?
- Is it useful?
- Is it harmful?
- Can we work around not having it?
- Does it require namespace coordination?

# Routing

- Can it determine the next hop?
- Can it determine the best hop?
- Who knows about the name?
- When should we send the packet?
- What's the cost of sending the packet?

# Aggregation

- Does it make it possible?
- Does it make it easy?
- Does it imply more efficient routing?
- Who defines the aggregation point?
- Does it provide scalability? (number of global names)

# Human Readable

- Is it Human Friendly?
- Is it Printable?
- Is it Shareable?
- Does it pass the bus test?

# Network Friendly

- Is it Router Friendly?
- Is it Short?

# Persistence

- Can it be short lived?
- Can it be long lived?



# Provenance

- Does it provide an author?
- Does it provide an author group?
- Is the author human friendly?

# Location

- For routing

# Multiplexing

- Does the name imply an application?
- Does it imply a Handler?
- Does it imply a High level protocol?
- What can be named? What type of things?
- Does it imply/have a type?

# Format

- Type
- Globally defined
- User defined

# Relationships

- Next object
- Previous object
- All objects in a certain space

# Organization

- Contents (what other objects are inside this name space)
- Containership (what other name/content acts as the container of this)

# Scope

- Can you determine under what condition the name should be interpreted?

# Description

- Does it provide a description of the content?
- Does it provide attributes of the content?



# Mutability

- Is it mutable?
- Is it appendable?
- Who can change it?

# Binding

- Who does the binding?
- When is the binding made?
- Is binding expensive?
- Is late binding allowed?

# Privacy and Security

- Does the name give away information?
- Who can interpret this information?

# Layering

- At what Layer does this naming apply?
- Does the lower layer need to provide something?
- How does the lower layer affect naming?
- Is overlay a special case?

# Evolution

- How hard is it to change?
- What data/structure is fixed?
- What algorithms are fixed? (global constants)

# Discussion

- What is part of the name?
- Who creates the name?
- What is required to “interpret” the name?

Old

**IP NAMING**

# Naming

An IP packet contains a lot of names:

74.125.45.100 13.7.8.14 | tcp 80 23759 41746833 watch?v=RxPZh4AnVWyk  
\_\_\_\_\_ destination \_\_\_\_\_ transport signature \_\_\_\_\_ packet-in-stream \_\_\_\_\_ DPI  
\_\_\_\_\_ URL  
\_\_\_\_\_ content

- ✌ IP separated the problems of forwarding & routing.
- ✌ IP kept the structure & meaning of names out of forwarding.
- ✌ IP used its name hierarchy to elegantly solve some tricky discovery and bootstrap problems.



# Naming

- ☠ IP names name only communication endpoints. It takes a warehouse full of equipment to fix this (amazon.com, google.com, ...).
- ☠ IP spread parts of its names all over the packet (and multiple packets). It takes a lot of very expensive equipment to deal with this (load balancers, deep packet inspection engines, ...)

Current

**CCN NAMING**

# What's in a Name (user/app view)

App supplied name	Versioning & segmentation	Content or proxy (e.g., SHA256 checksum)
<code>/parc.com/van/cal/417.vcf/v3/s0/0x3fdc96a4...</code>		
<div><div>Signed by parc.com</div><div>Signed by /parc.com/van</div></div>		

This binding is *immutable*  
(the data associated with the name can't change)

# Characteristics

- Provides organization + aggregation
- Location independent
- Needs keys for verification
- Provenance + Binding via Signature/Key
- No type (or type of name)
- No layering

Naming

**PROPOSAL**



Application

Library

Core

# @ Core

- Slow evolution
- Efficient for forwarding
- Flexible for routing and provide aggregation
- Simple matching
- Simple binding (Routing binding)
- Organization (only for routing)

# @ Library

- Paced evolution
- Flexible for security
- Flexible for interpretation
- Flexible for binding (related to security)
- Allow organization and relationships
- Human friendly?



Version/Magic	Flags/size/etc...
Flat-Element	Flat-Element
Flat-Element	Flat-Element
CCN Name + Sig + etc	
Data	

Version/Magic

Flags/size/etc...

Flat-Element

Flat-Element

Flat-Element

Flat-Element

CORE

LIB

CCN Name + Sig + etc

Data

# Flat-elements

- Ordered
- Identify (explicitly)
  - actual data
  - name of data
  - location of data

# Discussion

- Take advantage of layering
- Add type
- Let lower layer solve “lower” networking
- Higher layers don’t need to change
- Using vs ignoring topology

Old

**EXTRA SLIDES**

# Naming Schemes

- Names are an aspect of communication (language) systems.
- Names have meaning only within some (social, political, economic, technical, ...) domain.
- The rules (syntax and semantics) for naming in some domain are that domain's naming scheme.

# There is no universal Naming Scheme

- European vs. Scandanavian family names
- Library classification / ISBN
- URL / URI / URN / Handles
- E.164 phone numbers / IP addresses
- IP routing / IP transport

# CCN is naming scheme agnostic

- CCN is a communications framework that can be adapted to many different naming schemes to solve particular problems.
- It deals with some 'generic' issues by appending info to naming scheme names:
  - \* uniqueness is guaranteed by appending hash of data+publisher to name.
  - \* profiles [communication abstractions] can append version, segment id, ..., to name.



# SEN has a naming scheme

- SEN operates within a specific social (family and extended family) and communications context (mobiles & computers talking over public & private infrastructure).
- Its naming is designed to facilitate CCN-based communication among the significant entities within this context.

# SEN naming constraints

- The SEN communications context implies there is no central authority to assign names and all-to-all communication for conflict detection is impossible.
- This suggests autonomously generated names with enough entropy to avoid 'birthday problem' collisions ( $>10^{12}$  bits).
- Since signing keys are long, random bit strings, using them (or their fingerprints) for names simultaneously solves both autonomous generation and trust issues.

# Context-sensitive names

- Since context-sensitive names automatically change their target based on context, they can simplify application development and user interaction.
  - ▶ camera app puts new pictures in '/mypictures'.
  - ▶ TV asks for '/todaysPictures' to solicit recent photos from local phone(s).

# Implications

- In almost all circumstances, contextual names are implemented via a prefix substitution on the local machine and *never* appear on the wire.
  - ▶ Inbound substitution is done as last step before name presented to user/app (after verification and trust model)
  - ▶ Outbound substitution is done as first step before name handed to protocol stack.