Quick tutorial
Content-centric Networking (CCN) is a communication architecture based on transferring named data.
CCN overview

**Step 1 - Name the data**
Name every piece of network data

**Step 2 - Secure the data**
Secure every piece of network data

**Step 3 - Transfer the data**
Move the data to interested recipients
name data
A large object like a file has a CCN name
CCN also names the network sized chunks
CCN creates a manifest describing the file.
A large object like a file has a CCN name
The stream manifest is the metadata for the stream.
CCN name uses

Routing/Forwarding
Find the source of the data

Demultiplex
Choose between options

Matching
Determine equivalence

Structure
Inherent information
CCN name format

Name Segment based
Labeled binary segments

Hierarchical structure
Ordered sequence of segments

/seg1/seg2/seg3/seg4
CCN name example

/parc/ccnx/presentations/slide20/v=2/c=0

- globally routable name segments
- application dependent name segments
- protocol dependent name segments
CCN name origins

** Routable name segments **
Globally coordinated namespace (like domain names)
Locally coordinated namespace (like subdomains)

** Application name segments **
Applications can name their data any way they want (like filenames)

** Protocol name segments **
Protocols use conventions in naming to transmit information (like version, chunk number, etc)
secure data
Secure single chunks

CCN names and signs every chunk
Secure whole object via manifest

CCN names and signs the file via a manifest

manifest

/parc/ccnx/spec/manifest

Signature

List of chunks (optionally encrypted)
Secure single chunks

Keyld

Key used to sign

/parc/ccnx/spec/c=1

Signature

Data (optionally encrypted)

Signature binds the name to a Key
CCN name qualifier

/parc/ccnx/presentations/slide20/v=2/c=0

Keyld

identifier of key used to sign the object
Secure single chunks

Content Object Hash

Signature binds the name to a key

/parc/ccnx/spec/c=1

Signature

Data
(optionally encrypted)
CCN name qualifier

/parc/ccnx/presentations/slide20/v=2/c=0

ContentObjectHash

- hash of the content object message

Keyld

- identifier of key used to sign the object
Secure whole object via manifest

CCN names and signs the file via a manifest

Manifest

/parc/ccnx/spec/manifest

Signature

List of chunks
(optionally encrypted)
Secure via manifest

Indirectly sign every chunk through the manifest
transfer data
Core Protocol

One interest packet gets one content packet
Interest

/parc/ccnx/slide1/c=5

Content Object

/parc/ccnx/slide1/c=5

Signature

Data
(optionally encrypted)
Interest

/parc/ccnx/slide1/c=5 ?

Content Object

/parc/ccnx/slide1/c=5

Signature

Data
(optionally encrypted)
Interest

/parc/ccnx/slide1/c=5

KeyID=<keyId>

Content Object

/parc/ccnx/slide1/c=5

Signature by <keyId>

Data (optionally encrypted)
**Interest**

/parc/ccnx/slide1/c=5

COHash=<hash>

---

**Content Object**

/parc/ccnx/slide1/c=5

Data (optionally encrypted)

hash
Transport protocols are built on core exchanges.
Transport Protocols

Transport protocols can do parallel requests
Transport Protocols

Transport protocols can use manifests
Manifest structure
Manifest structure
Manifest structure

/parc/index.html/manifest

C

C

C

C

C

M

A Xerox Company
Manifest structure
CCN
The CCN software architecture

- Applications
- API
- Transport Framework
- Forwarder
- Services
CCN Services and API characteristics

CCN Portal | Key-Value | Messaging | Streaming
---|---|---|---
Applications

Network agnostic APIs
The CCN transport stack

Base Stack

Application
  API
  API
  API

API Adaptor
Flow Control
Encode/Sign
Decode/Verify
Fragmentation
Forwarder Adaptor
Forwarder

Optional Components

Storage
Permissions / Policies
  Local Keychain
  Radius / LDAP / etc
Packet Format

**Static Header**
Information required on every packet. May be modified at intermediate hops.

**Optional Header**
Optional information. May be modified at intermediate hops.

**Message**
End-to-end protocol message. Unmodified by intermediate elements.
Packet Format

CCN 0.x

Message

CCN 1.x

Static Header
Optional Headers
Message
Packet Format - Why change

Static header
- enables fast parsing
- contains common needs
- allows versioning

Optional headers
- allows network elements to add/modify information
Message Organization

Single message format

Name in front

Payload

General Validation info in back

CCN 1.x

Content Object / Interest

Name

Meta

Payload

Validation
Message Organization

CCN 0.x

- Content Object
- Signature
- Name
- Signed Info + Meta
- Payload

CCN 1.x

- Content Object / Interest
- Name
- Meta
- Payload
- Validation
Message Organization - Why change

Name comes first
   fast parsing

Separate validation from metadata at the end
   modular security

Unified packet format
   simplified, fast parsing
Packet Encoding

Easy to parse

Easy to skip

Unambiguous

CCN 1.x

tlv

type-length-value
(tag-length-value)

(2B) Type       (2B) Length
(Length) Value
Packet Encoding

**CCN 0.x**

- `ccnb`
- “Custom binary encoding format for XML to meet specific needs of CCNx”
- `<block><block><block>`

**CCN 1.x**

- `tlv`
- `type-length-value (tag-length-value)`
- `(2B) Type  (2B) Length  (Length) Value`
Packet Encoding - Why change

ccnb
flexible but complicated
relies on meta-structure
bit efficient

TLV
easy to parse
well understood
parse efficient
2 x 2 encoding

<table>
<thead>
<tr>
<th>0</th>
<th>8</th>
<th>16</th>
<th>24</th>
<th>32</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Length</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Value ...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
2 x 2 encoding

2-Byte Type
- Enough for expansion and experiments
- Inherently hierarchical (not many types)
- Fast to jump over (no parsing needed)
- No aliasing (0 vs 00)
- Canonical sorting

2-Byte Length
- Don't cover what doesn't fit in a packet
- Enough for large fields like payload
- Fast to jump over (no parsing)
- No aliasing (0 vs 00)
- Canonical sorting
Label-based names

All segments have types

Types specified by protocols

Type space for applications

CCN 1.x

/parc/ccn.zip/app<id>=1234/v=12/c=2/
Label-based names

CCN 0.x
/parc/ccn.zip/%C1.M.K%01%02/
%FD%04%62/%00%02/

CCN 1.x
/parc/ccn.zip/app<id>=1234/v=12/c=2/
Label-based names - Why change

No aliasing
  Defining structure eliminates aliasing

Cleaner representation
  More human readable

More powerful
  Structure allows network elements to make choices
Matching (exact)

/parc/ccn.zip == /parc/ccn.zip

Exact binary match

CCN 1.x
Matching (exact)

CCN 0.x


CCN 1.x

/parc/ccn.zip == /parc/ccn.zip
Matching (no selectors)

CCN 0.x

Interest:
name = /parc/ccn.zip
minSuffixComponents=x
maxSuffixComponents=y
exclude=xxx,xxx,xxx,…
childSelector=Left/Right

CCN 1.x

Interest:
name = /parc/ccn.zip
Matching - Why change

**Exact match is deterministic**
You get what you ask for

**Efficient match**
Fast forwarding on single match

**No rummaging of caches / traffic**
Better privacy
Matching (restrictions)

CCN 0.x

Interest:
name = /parc/ccn.zip
pubKeyDigest=xxx

Interest:
name = /parc/ccn.zip/abcd
minSuffixComponents=0
maxSuffixComponents=0

CCN 1.x

Interest:
name = /parc/ccn.zip
keyIdRestriction=xxx

Interest:
name = /parc/ccn.zip
contentObjectHash=abcd
Core Protocol Primitives

**Interest**
- /parc/ccnx/slide1/c=5
- KeyID=<keyId> (optional)
- COHash=<hash> (optional)

**Content Object**
- /parc/ccnx/slide1/c=5
- Signature by <keyId>
- hash
- Data (optionally encrypted)
Matching (restrictions) - Why change

No ‘real’ change
  Functionally the same

Explicit contentObjectHash matching
  Simpler matching (not intermingled)
Loop halting

Loops halted by PIT

Hop-Limit is stop-gap

CCN 1.x

Interest:
name = /parc/ccn.zip
hop-limit = 16

PIT
/parc/ccn.zip
Loop halting

CCN 0.x

Interest:
name = /parc/ccn.zip
nonce = 1234

PIT
/parc/ccn.zip : 1234

CCN 1.x

Interest:
name = /parc/ccn.zip
hop-limit = 16

PIT
/parc/ccn.zip
Loop halting - Why change

**Less overhead**
- No need to carry large nonce in packet
- No need to keep nonces at router (large at fast speed)

**PIT takes care of most loops**
- PIT halts loops, hop-limit is a stop-gap

**Nonce-for-loops break aggregation**
- Interests can’t be aggregated
  - (if node can treat same nonce interests as equal)
Interest Payload

Interests can carry payload

CCN 1.x

Interest:
name = /store/cart/id=1234/checkout

payload = abcdefg<1k component>xyz
Interest Payload

CCN 0.x

Interest:
name = /store/cart/abcdefg...
...<1k component>xyz/checkout

CCN 1.x

Interest:
name = /store/cart/id=1234/checkout
payload = abcdefg<1k component>xyz
Interest Payload - Why change

**Less processing at routers**
- No need to parse long names all the time

**Less storage at routers**
- No need to keep large names at routers

**Less traffic overhead**
- No need to carry copy of state back in the response
Protocol Separation

- Core protocol runs everywhere
- Protocol specified individually
- Separation of concerns
- Modularity

<table>
<thead>
<tr>
<th>Discovery</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chunking</td>
</tr>
<tr>
<td>Versioning</td>
</tr>
<tr>
<td>Core Messaging</td>
</tr>
<tr>
<td>Framing</td>
</tr>
</tbody>
</table>
Caching

Universal

Optional

Restricted
Layer 2

Next hop

Mapping to link layer (ARP)

Fragmentation
CCN Benefits

Security
Data is always secure, in transit and at rest.

Control
The network works in conjunction with the clients

Interoperable
Applications can interoperate transparently

Resilience
The network can operate with minimal interruption

Efficient
Low overhead under heavy demand

Composable
Integrate storage, communication, processing
Production quality
Deployable
Feasible
Realistic for all hardware
Hardware
CCN - 1.0 Software
## CCNx 1.0 Releases

<table>
<thead>
<tr>
<th>Release</th>
<th>Content</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>R0</td>
<td>Core Protocol</td>
<td>Ability to port base core protocol to new platforms (embedded, desktop or server (Mac OS X &amp; Linux)</td>
</tr>
<tr>
<td>R1</td>
<td>Services</td>
<td>Ability to install, start, stop and manage a CCN local network</td>
</tr>
<tr>
<td>R2</td>
<td>Storage, Devices, Mobility &amp; Performance</td>
<td>Ability to store &amp; retrieve objects in a CCN network, use express headers &amp; run on Android devices</td>
</tr>
<tr>
<td>R3</td>
<td>Platforms, Usability &amp; Applications Framework</td>
<td>Ability to run on Web Browsers, Windows, iOS &amp; FreeBSD, write &amp; test “network aware” applications and add functionality to CCN</td>
</tr>
</tbody>
</table>
CCN 1.0 Software

What Is It?

codename - Distillery
Application Programs
Software Stack
Common Libraries
Documentation
Instructional Videos
CCN 1.0 Software - Common Libs

C Support

codename - LongBow

Write Better C Programs

Runtime Assertions and Traps

assertTrue
assertFalse
assertNull
assertNotNull

trapIllegalValue
trapNotImplemented
TrapOutOfBounds
trapOutOfMemory
trapUnexpectedState

Native C Unit Test Framework

xUnit-style testing for C (in C)
Integrates with runtime assertions and traps
Integrated with automake, Xcode, Eclipse
CCN 1.0 Software - Common Libs

C Library

codename - PARCLib
Data Structures
Algorithms
Utility Functions
CCN 1.0 Software - Common Libs

CCN Library

- Abstractions
- Data Structures
- Algorithms
- Utility Functions

CCN Packets
- Headers, TLVs

CCN Names
- Components, Types

CCN Messages
- Content Objects
- Interests
- Manifests

...
CCN 1.0 Software

CCN Stack
Transport Framework
Portal API
Transport Framework

Component Based Design
Dynamically Plumbed
Dynamically Loaded Modules (planned)
CCN 1.0 Software - App View

Portal API

Native
Simple

CCNxContentObject
CCNxName
CCNxInterest
CCNxPortalMessage

CCNxNameSegmentType
CCNxKeystoreUtilities
CCN 1.0 Software - API View

APIs

Straight forward
Control transport
Components

Data manipulation
Stack communication
CCN 1.0 Software

Easy to Use
- Simple application
- No special privileges

Configurable Cache Size
- In memory cache, 0 to …

CCN 1.0 Packet Format
- TCP/IP encapsulation
- Native ethernet

Forwarder
- codename - Metis
- Built for experimentation
- Efficient
CCN 1.0 Software - Goals

Written for Experimentation

Interface Based Architecture
- Clear Separation of Concerns
- Simple to Substitute Different Implementations

Modular Design
- Promotes Extensibility
- Interoperability Testing
- Graduated/Progressive Implementation
CCN 1.0 Software - Goals

Written for People

- Human Factors Emphasis
- Documentation
  - Function Documentation
  - Tutorial Guides
- Consistent Design and Style
- Measured Software
CCN 1.0 Software

Documentation

100% Coverage
Module,
Function,
Enumeration
Type
IDE Integration
Printed and Online

// Get the actual contents of the specified chunk of the file.
PARCBuffer *payload = tutorialFileIO_GetFileChunk(fullFilePath,
chunkSize, chunkNumber);

Declaration PARCBuffer *tutorialFileIO_GetFileChunk(const char *
fileName, size_t chunkSize,
chunkNumber);

Description Given a fileName and chunk number, retrieve that chunk from the
specified file. The contents of the chunk are returned in a PARCBuffer
that must eventually be released via a call to parcBuffer_Release(buf).
The chunkNumber is 0-based.

Parameters

fileName[in] A pointer to a string containing the name of the file to read from.
chunkSize[in] The maximum number of bytes to be returned in each chunk.
chunkNumber[in] The 0-based number of chunk to return from the file.

Returns A newly created PARCBuffer containing the contents of the specified
chunk.

Declared In tutorial_FIIO.h
CCN 1.0 Software

Consistent Design and Style

Uniform Code Style
Clean and Clear Naming
High Cohesion
Low Coupling
CCN 1.0 Software

Measured

Quality Control
- Style Conformance
- Naming Conformance
- Unit Testing
- Documentation Coverage
- Complexity Management

Developer Tools
- Readiness and Acceptance
## CCNx Dashboard

### Source Code

<table>
<thead>
<tr>
<th>Module</th>
<th>Main</th>
<th>Testing</th>
<th>Test Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Forwarder</td>
<td>19</td>
<td>17</td>
<td></td>
</tr>
<tr>
<td>APIs</td>
<td>14</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>Transport</td>
<td>27</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>CCN Library</td>
<td>25</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>C Library</td>
<td>38</td>
<td>22</td>
<td></td>
</tr>
<tr>
<td>C Support</td>
<td>10</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td><strong>Total Lines</strong></td>
<td><strong>136k</strong></td>
<td><strong>77k</strong></td>
<td><strong>74%</strong></td>
</tr>
</tbody>
</table>

### Developer Documentation

<table>
<thead>
<tr>
<th>Module</th>
<th>Print</th>
<th>HTML</th>
<th>Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Forwarder</td>
<td>315</td>
<td>354</td>
<td></td>
</tr>
<tr>
<td>APIs</td>
<td>301</td>
<td>239</td>
<td></td>
</tr>
<tr>
<td>Transport</td>
<td>438</td>
<td>471</td>
<td></td>
</tr>
<tr>
<td>CCN Library</td>
<td>604</td>
<td>234</td>
<td></td>
</tr>
<tr>
<td>C Library</td>
<td>907</td>
<td>413</td>
<td></td>
</tr>
<tr>
<td>C Support</td>
<td>225</td>
<td>175</td>
<td></td>
</tr>
<tr>
<td><strong>Total Pages</strong></td>
<td><strong>2,864</strong></td>
<td><strong>1,972</strong></td>
<td><strong>92%</strong></td>
</tr>
</tbody>
</table>

### Code Complexity

<table>
<thead>
<tr>
<th>Module</th>
<th>Cyclomatic</th>
<th>Vocabulary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Forwarder</td>
<td>1.77</td>
<td>55</td>
</tr>
<tr>
<td>APIs</td>
<td>1.88</td>
<td>51</td>
</tr>
<tr>
<td>Transport</td>
<td>2.43</td>
<td>70</td>
</tr>
<tr>
<td>CCN Library</td>
<td>2.29</td>
<td>45</td>
</tr>
<tr>
<td>C Library</td>
<td>1.97</td>
<td>43</td>
</tr>
<tr>
<td>C Support</td>
<td>2.00</td>
<td>39</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td><strong>2.00</strong></td>
<td><strong>50</strong></td>
</tr>
</tbody>
</table>

### Design: Modularity

<table>
<thead>
<tr>
<th>Module</th>
<th>Modularity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Forwarder</td>
<td></td>
</tr>
<tr>
<td>APIs</td>
<td></td>
</tr>
<tr>
<td>Transport</td>
<td></td>
</tr>
<tr>
<td>CCN Library</td>
<td></td>
</tr>
<tr>
<td>C Library</td>
<td></td>
</tr>
<tr>
<td>C Support</td>
<td></td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td><strong>84%</strong></td>
</tr>
</tbody>
</table>
CCN 1.0 Software

**Cyclomatic Complexity**

- CCN 0.8: 7
- Apache: 6
- Linux 3.x: 3
- CCN 1.0: 1

**Vocabulary**

- CCN 0.8: 175
- Linux 3.x: 75
- CCN 1.0: 25
CCN - Future
Control is good
Security is good
Isolation is good
Virtualizing Operating Systems
Virtualizing Network

OpenFlow

parc
A Xerox Company
Virtualizing Cloud

OpenStack

Your Applications

APIs

OpenStack Dashboard

Compute

Networking

Storage

OpenStack Shared Services

Standard Hardware

parc

A Xerox Company
Virtualizing Applications

Docker Engine

Host OS

Server

App A

Bins/Libs

App B

Bins/Libs
Solitary confinement is not for everybody
Integration
Robust and secure Integration
Virtual Platform

(the next logical step)
Platform

The base on which we build applications and services
Platform

Identity
Authentication
Security

Storage
Archival
Communication

Integration
Coordination
Federation
CCN Virtual Platform
Codename: COPA
CCN Virtual Platform
Codename: COPA

Identity
Understands individuals, groups and organizations

Authentication
Validates and verifies identities

Security
Enables access control at various granularities
CCN Virtual Platform
Codename: COPA

Storage
Abstracts storage from applications

Archival
Keeps versions, histories and backups

Communication
Abstracts messaging between applications
CCN Virtual Platform
Codename: COPA

Integration
Allows applications to share data and resources

Coordination
Enables applications to work together

Federation
Bridges administrative domains
Simple Example
Authenticate
Store data
Notify Services
Augment data
Notify Services

A

S1

S2
Augment data

A
S1
S2
Notify outputs

A  S1  S2  O
Form scan

Rotate OCR SSN Redact Mailer

A SO S1 S2 S3 O
A  S0  S1  S2  S3  O

Ecosystem

Uncoordinated
Dynamic
Federated
PARC Experience

Benefits from research into workflows
Benefits from research in APIs and languages
Benefits from research in data representation
Benefits from research in planning

Built on research in networking - CCN
CCN enablers

Secure communication (data at rest and in transit)

Efficient content transfer (eliminates redundancy)

Data storage in coordination with the network (storage abstraction)

Data reuse (same data, multiple applications)

Smart network rulesets (communication decisions based on metadata)
CCN Virtual Platform
Codename: COPA

Platform

Identity
Authentication
Security

Storage
Archival
Communication

Integration
Coordination
Federation
Thank you

http://www.ccnx.org/