parc®
A Xerox Company

CCN 1.x Tutorial

Nacho Solis
Glenn Scott

What is CCN?

Content-centric Networking (CCN)

is a

communication architecture

based on

transferring named data

parc
A Xerox Company

The CCN communication architecture

Applications

User facing programs

Services

Base services required for network operation

APIs

Abstractions for interacting with the network

Transport

Structured and secure “end-to-end” communication

Messaging

Name-based, network wide communication using CCN messages

Framing

Transport for messages over layer 2

parc
A Xerox Company

The CCN project

Specifications

Description of protocols and algorithms

Software

Reference software implementation

Hardware

Hardware prototypes (big and small)

Commercial community

Commercial companies developing CCN

Research community

Researchers, faculty and students working on the CCN technology

Developer community

Application developers (big and small) using CCN

History

"A CABLE-TREE ETHER"
ALTO
ALTO
ALTO
DYNABOOK
VTS
NOVA
NOVA
XGP
PDP-11
ETHER!
"CABLE ETHER"
TELEPHONE COAX BOOSTER
TELEPHONE
TELEPHONE RADIO BOOSTER
RADIO ETHER
TELEPHONE ETHER
-1-

parc
A Xerox Company

Internet stack trouble areas

Data Center
High speed data transfers between nearby nodes

Internet of Things
Data collection, sensing and actuation on small devices

Web
Efficient and secure communication channels

Enterprise
Isolation, management, provisioning and containers

Ad-hoc / DTN
Non-traditional edge networks with no infrastructure

De-centralized Applications
New network abstractions to deal with communication

parc
A Xerox Company

rethink the network
rethink the stack

parc
A Xerox Company

Timeline

ICN Research

CCN 0.x

CCN 1.x

NDN Formed (2010)
1st Dagstuhl Seminar (2010)
1st SIGCOMM workshop (2011)
1st INFOCOM workshop (2012)
1st ICNRG meeting (2012)

1st NDNCom meeting (2014)
1st ICN Conference (2014)

parc
A Xerox Company

Basic concepts

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

parc
A Xerox Company

name data

parc
A Xerox Company

A large object like a file has a CCN name

parc
A Xerox Company

CCN also names the network sized chunks

parc
A Xerox Company

CCN creates a manifest describing the file

parc
A Xerox Company

A large object like a file has a CCN name

parc
A Xerox Company

The stream manifest is the metadata for the stream.

parc
A Xerox Company

CCN name uses

Routing/Forwarding
Find the source of the data

Demultiplex
Choose between options

Matching
Determine equivalence

Structure
Inherent information

parc
A Xerox Company

CCN name format

Name Segment based
Labeled binary segments

Hierarchical structure
Ordered sequence of segments

/seg1/seg2/seg3/seg4

parc
A Xerox Company

CCN name example

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

globally
routable
name segments

application
dependent
name segments

protocol
dependent
name segments

parc
A Xerox Company

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)

parc
A Xerox Company

CCN name qualifier

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

ContentObjectHash

hash of the content object message

KeyId

identifier of key used to sign the object

parc
A Xerox Company

secure data

parc
A Xerox Company

Secure single chunks

CCN names and signs every chunk

parc
A Xerox Company

Secure whole object via manifest

CCN names and signs the file via a manifest

parc
A Xerox Company

Secure via manifest

Indirectly sign every chunk through the manifest

parc
A Xerox Company

transfer data

parc
A Xerox Company

Core Protocol

One interest packet gets one content packet

parc
A Xerox Company

Interest

/parc/ccnx/slide1/c=5 ?

Content Object

/parc/ccnx/slide1/c=5

Signature

Data
(optionally encrypted)

parc
A Xerox Company

Interest

/parc/ccnx/slide1/c=5 ?

Content Object

/parc/ccnx/slide1/c=5

Signature

Data
(optionally encrypted)

parc
A Xerox Company

Interest

/parc/ccnx/slide1/c=5 ?

KeyID=<keyId>

Content Object

/parc/ccnx/slide1/c=5

Signature by <keyId>

Data
(optionally encrypted)

parc
A Xerox Company

Interest

/parc/ccnx/slide1/c=5 ?

COHash=<hash>

Content Object

/parc/ccnx/slide1/c=5

Data
(optionally encrypted)

hash

parc
A Xerox Company

Transport Protocols

Transport protocols are built on core exchanges

parc
A Xerox Company

Transport Protocols

Transport protocols can do parallel requests

parc
A Xerox Company

Transport Protocols

Transport protocols can use manifests

parc
A Xerox Company

Manifest structure

parc
A Xerox Company

Manifest structure

parc
A Xerox Company

Manifest structure

parc
A Xerox Company

Manifest structure

parc
A Xerox Company

CCN

CCN - Forwarder

parc
A Xerox Company

parc
A Xerox Company

parc
A Xerox Company

parc
A Xerox Company

parc
A Xerox Company

parc
A Xerox Company

parc
A Xerox Company

parc
A Xerox Company

CCN - Base Protocol

Name routes are advertised

Node F serves /parc content.
Advertises a route to /parc

Name routes are set up in forwarders

Routes are set up pointing to F for prefix /parc

parc
A Xerox Company

Core Protocol

One interest packet gets one content packet

parc
A Xerox Company

Interests follow the routes

A issues interest for /parc/a

parc
A Xerox Company

Interests leave breadcrumbs

Interest leaves reverse path state in the network

parc
A Xerox Company

Interest matches content

Interest for /parc/a finds a match at F

parc
A Xerox Company

Content follows reverse path

Content packet follows reverse path and consumes state

parc
A Xerox Company

Any node can cache

Node G requests same content, gets it from a cache in D
G authenticates content via the content signature, not the sender

parc
A Xerox Company

Interests are aggregated

Node C aggregates the interests (only sends one upstream)

parc
A Xerox Company

Content is sent in all interest paths

Node C sends the content packet to both B and G

parc
A Xerox Company

Multiple nodes can advertise same prefix

Both F and I can provide and advertise the content for /parc

parc
A Xerox Company

Nodes can forward on either path

Nodes can choose where to send each interest

Forward on “best” path

Choose one path to forward on

Forward on alternating paths

CCN is not connection oriented
Each interest can go to a different content provider

Forward on both paths

Node D will only reply to node C once
This can provide redundancy (at a high price)

Strategy Layer determines forwarding

The Strategy Layer at a forwarder determines what forwarding policy a node follows.

CCN - 1.x

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.

CCN 1.x

parc
A Xerox Company

Packet Format

CCN 0.x

CCN 1.x

parc
A Xerox Company

Packet Format - Why change

Static header

enables fast parsing
contains common needs
allows versioning

Optional headers

allows network elements to add/modify information

parc
A Xerox Company

Message Organization

Single message format

Name in front

Payload

General Validation info in back

CCN 1.x

parc
A Xerox Company

Message Organization

CCN 0.x

CCN 1.x

parc
A Xerox Company

Message Organization - Why change

Name comes first

fast parsing

Separate validation from metadata at the end

modular security

Unified packet format

simplified, fast parsing

parc
A Xerox Company

Packet Encoding

Easy to parse

Easy to skip

Unambiguous

CCN 1.x

parc
A Xerox Company

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	

parc
A Xerox Company

Packet Encoding - Why change

ccnb

flexible but complicated
relies on meta-structure
bit efficient

TLV

easy to parse
well understood
parse efficient

parc
A Xerox Company

2 x 2 encoding

parc
A Xerox Company

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

parc
A Xerox Company

Label-based names

CCN 1.x

All segments have types

/parc/ccn.zip/app<id>=1234/v=12/c=2/

Types specified by protocols

Type space for applications

parc
A Xerox Company

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/

parc
A Xerox Company

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

parc
A Xerox Company

Matching (exact)

CCN 1.x

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

Exact binary match

parc
A Xerox Company

Matching (exact)

CCN 0.x

/parc/ccn.zip ==
/parc/ccn.zip/v2/s3 ==
/parc/ccn.zip/v2/s4 ==
/parc/ccn.zip/v7/s6 ==
/parc/ccn.zip/v100/s1 ==
/parc/ccn.zip/meta/v1/s8 ==
/parc/ccn.zip/discussion ==
cn.zip/.acl/group/owner/key/v1/s9 ==

/parc/ccn.zip

CCN 1.x

== /parc/ccn.zip

parc
A Xerox Company

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

parc
A Xerox Company

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

parc
A Xerox Company

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

parc
A Xerox Company

Core Protocol Primitives

parc
A Xerox Company

Matching (restrictions) - Why change

No ‘real’ change

Functionally the same

Explicit contentObjectHash matching

Simpler matching (not intermingled)

parc
A Xerox Company

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

parc
A Xerox Company

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

parc
A Xerox Company

Nonces

Regular path

parc
A Xerox Company

Nonces and loops

Loop path

parc
A Xerox Company

Nonces and loops

Need hop-limit to stop loop.
Nonces dont help

parc
A Xerox Company

Nonces and multi-copy

Node B sends multiple copies of interest

parc
A Xerox Company

Nonces and multi-copy

Nonce not needed to keep entry

parc
A Xerox Company

Nonces and multi-copy

Nonce detects copy

parc
A Xerox Company

Nonces and multi-copy

Dangling PIT entry

parc
A Xerox Company

Nonces and multi-copy

Concurrent requests

parc
A Xerox Company

Nonces and multi-copy

Dangling (valid) PIT entry

parc
A Xerox Company

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)

parc
A Xerox Company

Interest Payload

Interests can carry payload

CCN 1.x

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

payload = abcdefg<1k component>xyz

parc
A Xerox Company

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

parc
A Xerox Company

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

parc
A Xerox Company

Protocol Separation

Core protocol runs everywhere

Protocol specified individually

Separation of concerns

Modularity

parc
A Xerox Company

Caching

Universal

Optional

Restricted

parc
A Xerox Company

Layer 2

Next hop

Mapping to link layer (ARP)

Fragmentation

parc
A Xerox Company

CCN - Encoding

CCN - Packets

Type-Length-Value (TLV) encoding

Type	Length	Value …

parc
A Xerox Company

2 x 2 encoding

parc
A Xerox Company

CCN Packet Structure

parc
A Xerox Company

Static Header

parc
A Xerox Company

Static Header

parc
A Xerox Company

CCN Headers

parc
A Xerox Company

Optional Headers

parc
A Xerox Company

CCN Packet Structure

parc
A Xerox Company

CCN Message

parc
A Xerox Company

CCN Message

parc
A Xerox Company

CCN Packet

```
+---------------+---------------+---------------+---------------+
|       0       |       1       |       2       |       3       |
|0 1 2 3 4 5 6 7|8 9 0 1 2 3 4 5|6 7 8 9 0 1 2 3|4 5 6 7 8 9 0 1|
+---------------+---------------+---------------+---------------+
|   Version     |  PacketType   |          PayloadLength        |
+---------------+---------------+---------------+---------------+
|   HopLimit†   |    reserved   |          HeaderLength         |
+---------------+---------------+---------------+---------------+
~ Optional Header TLVs                                          ~
+---------------+---------------+---------------+---------------+
~ CCN Message TLV                                               ~
+---------------+---------------+---------------+---------------+
~ Optional CCN  Validation Algorithm TLV                        ~
+---------------+---------------+---------------+---------------+
~ Optional CCN  Validation Payload TLV (required ValidationAlg) ~
+---------------+---------------+---------------+---------------+
```

Validation Range

Self-certified Name

parc
A Xerox Company

Interest Message Packet

```
+---------------+---------------+---------------+---------------+
|       0       |       1       |       2       |       3       |
|0 1 2 3 4 5 6 7|8 9 0 1 2 3 4 5|6 7 8 9 0 1 2 3|4 5 6 7 8 9 0 1|
+---------------+---------------+---------------+---------------+
|    Version    |   PacketType  |         PayloadLength         |
+---------------+---------------+---------------+---------------+
|    HopLimit   |    reserved   |         HeaderLength          |
+---------------+---------------+---------------+---------------+
~ Optional Header TLVs                                          ~
+---------------+---------------+---------------+---------------+
|           T_INTEREST          |         Interest Length       |
+---------------+---------------+---------------+---------------+
|            T_NAME             |           Name Length         |
+---------------+---------------+---------------+---------------+
~ Name segment TLVs                                             ~
+---------------+---------------+---------------+---------------+
~ Optional Metadata TLV                                         ~
+---------------+---------------+---------------+---------------+
~ Optional Payload TLV                                          ~
+---------------+---------------+---------------+---------------+
~ Optional CCN  Validation Algorithm TLV                        ~
+---------------+---------------+---------------+---------------+
~ Optional CCN  Validation Payload TLV (required ValidationAlg) ~
+---------------+---------------+---------------+---------------+
```

CCN Message

parc
A Xerox Company

Content Object Packet

```
+---------------+---------------+---------------+---------------+
|       0       |       1       |       2       |       3       |
|0 1 2 3 4 5 6 7|8 9 0 1 2 3 4 5|6 7 8 9 0 1 2 3|4 5 6 7 8 9 0 1|
+---------------+---------------+---------------+---------------+
|     Version   |  PacketType   |         PayloadLength         |
+---------------+---------------+---------------+---------------+
|            reserved           |         HeaderLength          |
+---------------+---------------+---------------+---------------+
~ Optional Header TLVs                                          ~
+---------------+---------------+---------------+---------------+
|        T_CONTENT_OBJECT       |    Content Object Length      |
+---------------+---------------+---------------+---------------+
|            T_NAME             |           Name Length         |
+---------------+---------------+---------------+---------------+
~ Name segment TLVs                                             ~
+---------------+---------------+---------------+---------------+
~ Optional Metadata TLV                                         ~
+---------------+---------------+---------------+---------------+
~ Optional Payload TLV                                          ~
+---------------+---------------+---------------+---------------+
~ Optional CCN  Validation Algorithm TLV                        ~
+---------------+---------------+---------------+---------------+
~ Optional CCN  Validation Payload TLV (required ValidationAlg) ~
+---------------+---------------+---------------+---------------+
```

CCN Message

parc
A Xerox Company

CCN - Names

CCN Name structure

/segment1/segment2/segment3

/parc/ccnx/presentation
/parc/ccnx/presentation/serial=2/
/parc/ccnx/presentation/serial=2/chunk=6
/parc/ccnx/presentation/serial=2/app:slide=80

parc
A Xerox Company

CCN name segment structure

label=segment

parc
A Xerox Company

CCN name segment structure

/ccn/presentation

/name=ccn/name=presentation

Default label = name

parc
A Xerox Company

CCN name segment structure

chunk=6

serial=6

Chunk is used for chunked objects
Serial is used for versioning

parc
A Xerox Company

CCN name segment structure

app:<param>=2014

app:year=2014

Application specific labels used by apps

parc
A Xerox Company

Labeled Content Identifiers (CCN URIs)

```
lci:/parc/ccn/spec/serial=3/chunk=2
```


parc
A Xerox Company

Labeled Content Identifiers

```
lci:/parc/ccn/spec/serial=3/chunk=2
lci:/name=parc/name=ccn/name=spec/serial=3/chunk=2
```


parc
A Xerox Company

Labeled Content Identifiers

```
lci:/parc/ccn/spec/serial=3/chunk=2
lci:/name=parc/name=ccn/name=spec/serial=3/chunk=2
lci:/N=parc/N=ccn/N=spec/V=3/C=2
```


parc
A Xerox Company

Labeled Content Identifiers

```
lci:/parc/ccn/spec/serial=3/chunk=2
lci:/name=parc/name=ccn/name=spec/serial=3/chunk=2
lci:/N=parc/N=ccn/N=spec/V=3/C=2
lci:/0x0010=parc/0x0010=ccn/0x0010=spec/0x00A1=3/0x00A3=2
```


parc
A Xerox Company

Labeled Content Identifiers

```
lci:/parc/pics/spec/app:year=2014
```


parc
A Xerox Company

Labeled Content Identifiers

```
lci:/parc/pics/spec/app:year=2014
lci:/name=parc/name=pics/app:year=2014/app:03=bob
```


parc
A Xerox Company

Labeled Content Identifiers

```
lci:/parc/pics/spec/app:year=2014
lci:/name=parc/name=pics/app:year=2014/app:03=bob
lci:/0x0010=parc/0x0010=pics/0xFF01=2014/0xFF03=bob
```


parc
A Xerox Company

lci:/ scheme based off of URIs (RFC 3986)

```
LS-URI           = scheme ":" ls-hier-part ["?" ls-query]
                   ["#" fragment]
ls-hier-part     = ["//" authority] ls-path-absolute
ls-path-absolute = "/" [ ls-segment *( "/" ls-segment ) ]
ls-segment       = lpv-segment / v-segment
lpv-segment      = label [":" param] "=" s-value
v-segment        = s-value
label            = alpha-t / num-t
param            = alpha-t / num-t
s-value          = *(s-pchar)

ls-query         = *1 ( lpv-component / v-component
                       *( "&" (lpv-component / v-component) ) )
lpv-component    = label [":" param] "=" q-value
v-component      = q-value
q-value          = *(q-pchar)

alpha-t          = ALPHA *(ALPHA / DIGIT)
num-t            = dec-t / hex-t
dec-t            = 1*(DIGIT)
hex-t            = "0x" 1*(HEXDIG)
ls-pchar         =  unreserved / pct-encoded / ls-sub-delims
s-pchar          = ls-pchar / ":" / "@" / "&"
q-pchar          = ls-pchar / ":" / "@" / "/"
ls-sub-delims    = "!" / "$" / "'" / "(" / ")"
                     / "*" / "+" / "," / ";"
```


parc
A Xerox Company

CCN name TLV encoding

parc
A Xerox Company

CCN name TLV encoding

TYPE_NAME	Name Length

Segment Label	Segment Length	Segment Value
Segment Label	Segment Length	Segment Value
Segment Label	Segment Length	Segment Value

parc
A Xerox Company

CCN - Fragmentation

Fragmentation

Content Objects can be up to 64KB in size

To transmit Content Objects over network links with smaller MTU (Maximum Transmission Unit) size we need to fragment Content Objects

parc
A Xerox Company

Fragmentation Types

Hop-by-Hop

Link specific, not part of spec

Mid-to-End

Not supported

End-to-End

Interest discovers MTU

None

Preferred mode, potentially more efficient

parc
A Xerox Company

End-to-End Fragmentation

Interests discover the MTU

End-to-End Fragmentation

Interests are fragmented to minimum MTU
First Interest fragment must be routable

parc
A Xerox Company

End-to-End Fragmentation

Content is fragmented to discovered MTU

End-to-End Fragmentation

Local networks might have large MTU

parc
A Xerox Company

End-to-End Fragmentation

Reply might be fragmented smaller than MTU discovered

parc
A Xerox Company

End-to-End Fragmentation Characteristics

Interests fragmented to minimum MTU at Requester

First Interest fragment must be routable

Content is fragmented to MTU size

Nodes reassemble if checking Content-Object Hash

Fragments are packed ("MTU" size, except last)

parc
A Xerox Company

Fragmentation Optional Header

parc
A Xerox Company

Fragmentation Optional Header

parc
A Xerox Company

End-to-End Fragmentation Summary

Cut through forwarding of interests

Cut through forwarding of content via Interest Id

Works in conjunction with Express Headers

Advanced fragmentation scheme in the works

parc
A Xerox Company

Advanced Topics

Advanced topics

Routing (loop free)

Retransmissions

Placeless objects

Push

Trust models

Hash based forwarding

SDN

Return channel overhead

Asymmetric paths

parc
A Xerox Company

Hardware

Penn

11 Terabit non-blocking fabric

14 slot chassis

Teller

4.4 Terabit non-blocking fabric

6 slot chassis

parc
A Xerox Company

Software Architecture

The CCN software architecture

parc
A Xerox Company

The CCN transport stack

parc
A Xerox Company

Transport Stack

parc
A Xerox Company

CCN Services and API characteristics

parc
A Xerox Company

CCN - 1.0 Software

CCN 1.0 Software

What Is It?

What Is Different?

How Do I Use It?

parc
A Xerox Company

CCN 1.0 Software

What Is It?

- Application Programs
- Software Stack
- Common Libraries
- Documentation
- Instructional Videos

parc
A Xerox Company

CCN 1.0 Software

CCN Software Stack

- Portal API
- Transport Framework

parc
A Xerox Company

CCN 1.0 Software

CCN Portal API

- Streaming
- Datagram
- Blocking
- Non-blocking

Portal

```
ccnxPortal_Listen
ccnxPortal_Read
ccnxPortal_Write
ccnxPortal_Release
ccnxPortal_Acquire
ccnxPortal_Ignore
ccnxPortal_IsEOF
ccnxPortal_IsError
```

parc
A Xerox Company

CCN 1.0 Software

Transport Framework

- Component Based Design
- Dynamically Plumbed
- Dynamically Loaded Modules (future)

parc
A Xerox Company

CCN 1.0 Software

Forwarder

Built For Experimenting

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

parc
A Xerox Company

CCN 1.0 Software

Common Libraries

Used By Everything

Reference Down,
No Reference Up

parc
A Xerox Company

CCN 1.0 Software

CCN Library

Application Writer's View

CCNxNameSegmentType
CCNxKeystoreUtilities

CCNxContentObject

CCNxName

CCNxInterest

CCNxPortalMessage

parc
A Xerox Company

CCN 1.0 Software

CCN Library

API Writer's View

CCNxControl

CCNxMessage

CCNxName

CCNxInterest

CCNxContentObject

CCNxKeyLocator

parc
A Xerox Company

CCN 1.0 Software

CCN Library

Component Writer's View

CCNxValidationFacade
CCNxContentObject
CCNxControlFacade
CCNxTlv
CCNxControl
CCNxName
CCNxTlvDictionary
CCNxMessage
CCNxWireFormatFacade
CCNxJson
CCNxNetworkBuffer

parc
A Xerox Company

CCN 1.0 Software

PARC Library

- C Utility Functions
- General Purpose

parc
A Xerox Company

CCN 1.0 Software

LongBow

Write Better C Programs

Runtime Assertions and Traps

`assertTrue`	`trapIllegalValue`
`assertFalse`	`trapNotImplemented`
`assertNull`	`TrapOutOfBounds`
`assertNotNull`	`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

parc
A Xerox Company

CCN 1.0 Software

What Is Different?

- Written for Extension and Experimentation
- Written for People
- Measured

parc
A Xerox Company

CCN 1.0 Software

Written for Experimentation

Interface Based Architecture

- Clear Separation of Concerns
- Simple to Substitute Different Implementations

Modular Design

- Promotes Extensibility
- Interoperability Testing
- Graduated/Progressive Implementation

parc
A Xerox Company

CCN 1.0 Software

Written For People

- Human Factors Emphasis

Documentation

- Function Documentation
- Tutorial Guides

Consistent Design and Style

Measured Software

parc
A Xerox Company

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
```

Declaration	PARCBuffer *tutorialFileIO_GetFileChunk(const char * fileName, size_t chunkSize, uint64_t 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_FileIO.h

parc A Xerox Company

CCN 1.0 Software

Consistent Design and Style

- Uniform Code Style
- Clean and Clear Naming
- High Cohesion
- Low Coupling

Palo Alto Research Center
Computing Science Laboratory

PARC C Style Guide

Glenn Scott[1]

Palo Alto Research Center
Computing Science Laboratory

PARC Library
Canonical C Function Name Conventions

Glenn Scott[1]

Abstract
This is a design guide for the uniform naming of functions in C language modules the PARC Library. A uniform naming convention means quicker understanding for developers and better interoperability between modules.

Keywords
Design Guide — C Language — Software Quality

[1] *Computing Science Laboratory, Palo Alto Research Center*

Contents

1 Function Names and Semantics 1
1.1 General Guidelines . . . 1
Create • CreateFrom • To • Get
1.2 Acquire . . . 3
1.3 AssertValid . . . 3
1.4 BuildString . . . 4
1.5 Compare . . . 4
1.6 Copy . . . 5
1.7 Display . . . 5
1.8 Equals . . . 6
1.9 HashCode . . . 6
1.10 Release . . . 7
1.11 ToString . . . 7
Acknowledgments 8

1. Function Names and Semantics

A consistent function naming scheme leads to a better understanding of exactly what is meant reduces the size of the vocabulary necessary to understand a collection of functions by their names, permits functions to be grouped and recognized at sight, and makes the composition of new function names more straightforward.

1.1 General Guidelines

Names always adhere to the namespace naming scheme consisting of a root domain name, followed by a concatenation of subcomponent names.

For the PARC Library in particular, the root domain name is `parc` and subcomponent names are the names of modules implementing functions manipulating a specific type like a linked-list, or buffer of bytes, or a group of functions addressing the needs of a particular concept, like

Palo Alto Research Center
Computing Science Laboratory

PARC Design
Namespaces for C Programs

Glenn Scott[1]

Palo Alto Research Center
Computing Science Laboratory

PARC Design
Managing Modularity and Coupling in C Programs

Glenn Scott[1]

Abstract
This is a design guide for the management of modularity and coupling in the C programming language.

Keywords
Design Guide — C Language — Software Quality

[1] *Computing Science Laboratory, Palo Alto Research Center*

Contents

1 Introduction 1
2 Name Spaces in C 1
3 Name Spaces and Files 1
4 Name Spaces and Functions 2
5 Name Spaces and Constants 2
5.1 Naming Functions . . . 2
Acknowledgments 2

1. Introduction

The C programming language offers no provisions for managing the names of functions, structures, type definitions, and enumerations. Each of these kinds of things exist in their own flat-namespace shared with every other named element of their kind. For example, within a C program there can only be one function named `write`, one structure named `result`, and so forth.

Software architects and implementors must manage these namespaces manually (mentally). As the source code for a software programming system grows, the cognitive burden also grows, leading to a increasingly difficult system to learn and use. Typically, this culminates as a system that is understood and maintained by a small group and is rarely used to its fullest extent.

This memo is a description of an obvious technique to manage namespaces in C programs and describes some conventions for that technique to increase usability.

2. Name Spaces in C

The namespace scheme is simply a linearized form of a path through a tree, where the path is a concatenation of the names of the nodes of the tree terminated by the name of the leaf, where a node is a portion of the C source file name, and a leaf is the name of a function, structure, enumeration, or type definition name.

The convention is to first choose the root name for the namespace. In the following examples, the namespace for the PARC Library is simply `PARC` and the namespace for the LongBow unit-test system is `LongBow`. The namespace name appears as both its upper- and lower-case (`PARC` and `parc`, `LongBow` and `longBow`) representations to give lexical and visual differentiation in specific contexts which are described below.

In all cases file, functions, structures, enumerations, and type definition names are divided into two components: A prefix comprised of the linearized form of the namespace hierarchy and suffix consisting of the name of the element being named. By convention these two components are always separated by a single _ (underscore) character.

namespace | typeName | .c

3. Name Spaces and Files

All C source and header files are named with the lower-case form of the name as the prefix `parc` followed by a single _ character separating the name of the namespace from the the name of the module. The Buffer implementation in the PARC Library has the name `parc_Buffer` and the corresponding source and header files are named `parc_Buffer.c` and `parc_Buffer.h`.

Sometimes sub-modules are a necessary design element and naming sub-modules follow the same pattern. For example, `longBow_Report.c` and `longBow_Report.h`, implements a facade to a set of functions implementing a reporting mechanism for LongBow. These functions have different implementations depending upon the output format. As a consequence, the submodules implementing these functions use the namespace 'longBowReport' followed by the _ character followed by the component name, where a component

parc
A Xerox Company

CCN 1.0 Software

Measured

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

Developer Tools

- Readiness and Acceptance

parc
A Xerox Company

CCNx Dashboard

Source Code

	Main	Testing	Test Coverage
Forwarder	19	16	
APIs	14	8	
Transport	27	15	
CCN Library	25	14	
PARC Library	38	21	
LongBow	10	…	
Total Lines	***136k***	***77k***	***74%***

Developer Documentation

	Print	HTML	Coverage
Forwarder	315	354	
APIs	301	239	
Transport	438	471	
CCN Library	604	234	
PARC Library	907	413	
LongBow	225	175	
Total Pages	**2,864**	**1,972**	***92%***

Code Complexity

	Cyclomatic	Vocabulary
Forwarder	1.77	55
APIs	1.88	51
Transport	2.43	70
CCN Library	2.29	45
PARC Library	1.97	43
LongBow	2.00	39
Average	**2.00**	***50***

Design: Modularity

	Modularity
Forwarder	
APIs	
Transport	
CCN Library	
PARC Library	
LongBow	
Average	***63%***

parc
A Xerox Company

CCN 1.0 Software

parc
A Xerox Company

CCN 1.0 Software

How Do I Use It?

- Install CCN 1.0 Software
- Use Portal API
- Link Application with Libraries
- Run
- Sit back, enjoy

parc
A Xerox Company

parc®

A Xerox Company

Thank you

http://www.ccnx.org/

CCN - Extra Slides

CCN - Examples

Sample use case

Web page

A simplified example

parc
A Xerox Company

Web page

Step 1 - Request web page manifest

Retrieve the manifest of the web page

Step 2 - Request web page content

Download the web page content (html)

Step 3 - Request references

Download embedded references (images, css, etc)

parc
A Xerox Company

Request page manifest

Request the list of chunks for the main webpage
Each chunk identified with a hash

Request page content

Request each chunk using name and hash

parc
A Xerox Company

Retrieve references

Sample use case

Netflix

A simplified example

parc
A Xerox Company

Netflix

Step 1 - Login

Contact Netflix to login, retrieve session key

Step 2 - Request movie network name

Get the actual network name

Step 3 - Request personal movie key

Get a key to decrypt the movie

Step 4 - Request movie manifest

Get the movie manifest object

Step 5 - Request movie stream

Download the actual movie contents based on the manifest.

parc
A Xerox Company

Login to service

Request a session ID from Netflix using our login ID
Authenticate sending our password, encrypted

parc
A Xerox Company

Get movie network name

Search Netflix for the movie Wall-e
Get back a network name of the movie

parc
A Xerox Company

Request movie key

Request a personal key for the movie we want to watch
Send session ID and device ID

parc
A Xerox Company

Request movie manifest

Request a list of segments for the movie
Each segment identified with a hash

parc
A Xerox Company

Request movie content

parc
A Xerox Company

Request more manifest

Request a list of segments for the movie
Each segment identified with a hash

parc
A Xerox Company

Request more movie content

parc
A Xerox Company

Sample use case

Phone call

A simplified example

parc
A Xerox Company

Phone Call

Step 1 - Setup call

Contact call agent at callee

Step 2 - Request caller stream

Request audio stream from caller

Step 3 - Request callee stream

Request audio stream from callee

parc
A Xerox Company

Setup call - Bob calls Alice

Setup a call from Bob to Alice. Bob tells Alice of the caller stream, Alice tells bob of the callee stream.

Request caller stream

Alice requests Bob's audio stream

parc
A Xerox Company

Request callee stream

Bob requests Alice's audio stream

Protocol Changes - Structural

Improved packet flexibility

Segmented packet - Using a static header, optional headers and message allows data to be carried efficiently at the right abstraction layer.

TLV packet format - Efficient and easy to parse packet structure with enough flexibility to extend and update the protocol.

Protocol Changes - Cleanup

Streamlined matching

Exact matching - Efficiently and unambiguously match an interest to a content object. Deliver deterministic behavior as much as possible.

Simple restrictions - Removed overhead created by selectors and replaced them with equality based restrictions.

Protocol Changes - Correctness

Fixed old protocol shortcomings

Loop halting and aggregation - Enable loop halting using hop-limit to fix flaw in nonce-based aggregation.

Type-based names - Enabled meaning in name components without aliasing issues in binary data.

parc
A Xerox Company

Protocol Changes - Functionality

Increased protocol functionality

Payload in Interests - Efficiently carry data in an interest without requiring the overhead of large names.

Validation Algorithm - Allow the efficient use of variable-length payload for validation algorithms in both Interests and Content Objects.

parc
A Xerox Company

Protocol Changes - New Features

Introduced new data structures

Manifests - Provide a structure for larger data objects and the transport layer.

Control Messages - A way for the network to communicate about what's happening, return errors, etc.

parc
A Xerox Company

Protocol Changes - Security

Specified security semantics

Self-certified name restriction - Defined required node behavior to deliver secure self-certified data.

Network caching restriction - Defined required cache behavior to prevent attack amplification.

parc
A Xerox Company

Protocol Changes - Native

Modified to run over Layer 2

Native Ethernet - Updated protocol to enabled CCN to run on the wire.

Fragmentation - Enabled protocol to work under different MTU situations.

Glossary

Interest
A request for a piece of content.

Content Object
A response to an interest. A piece of data, signed.

Message
An Interest or Content Object

Chunk
A Content Object that is part of a larger piece of Data.

Fragment
A network encoded partial CCN Message
Includes static and per-hop-headers

Packet
A network encoded CCN Message (partial or complete)
Includes static and per-hop-headers

parc
A Xerox Company

Glossary

Name Segment
A labeled name element (“year=2014”)

Name
A ordered set of CCN Name Segments

Name Prefix
A “left” subset of Name Segments in a Name

Content Object Hash
A hash over an encoded Content Object

Similarity Hash
A hash over the matching portion of an Interest

Interest Fragment Identifier
A Similarity Hash used to identify a fragmented Interest

Content Object Fragment Identifier
A Content Object Hash used to identify a fragmented Content Object

parc
A Xerox Company

Other changes - Summary

Software (all C)

Coding style

Modular transport

Modular API

Test driven

parc
A Xerox Company

Source Code

	Main	Testing	Test Coverage
Forwarder	19	16	
APIs	14	8	
Transport	27	15	
CCN Library	25	14	
PARC Library	38	21	
LongBow	10	…	
Total Lines	***136k***	***77k***	***74%***

Developer Documentation

	Print	HTML	Coverage
Forwarder	315	354	
APIs	301	239	
Transport	438	471	
CCN Library	604	234	
PARC Library	907	413	
LongBow	225	175	
Total Pages	**2,864**	**1,972**	***92%***

Code Complexity

	Cyclomatic	Vocabulary
Forwarder	1.77	55
APIs	1.88	51
Transport	2.43	70
CCN Library	2.29	45
PARC Library	1.97	43
LongBow	2.00	39
Average	**2.00**	***50***

Design: Modularity

	Modularity
Forwarder	
APIs	
Transport	
CCN Library	
PARC Library	
LongBow	
Average	***63%***

parc
A Xerox Company

PRIMARY

White	PARC blue	Dark blue	Light blue	Orange	Dark gray
R 255	R 32	R 0	R 58	R 255	R 55
G 255	G 84	G 35	G 110	G 102	G 55
B 255	B 105	B 50	B 143	B 0	B 55

Red	Lt. Orange	Lt. Gray	Lt. Green	Faded Blue
R 191	R 255	R 213	R 168	R 165
G 29	G 170	G 213	G 188	G 255
B 10	B 10	B 213	B 50	B 250

parc
A Xerox Company

Workable area
(1920 - 2*83) = 1754px

Two column proportional
83 + 653+ 1098 + 83

Body Level One

Body Level Two

Body Level Three

Body Level Four

Body Level Five

parc
A Xerox Company

Other changes - Summary

Manifest as core object

Control messages

Storage (vs Repo)

Sync as protocol+service

Flow control (vs static pipelining)

parc
A Xerox Company

CCN - Express

Express Headers

Used for fast forwarding

Similar to CCN enabled MPLS

Can be used together with fragmentation

Simple implementation via hash

Can be done global or local in scope

parc
A Xerox Company

Forwarding Label

/parc/ccnx/presentations/slide10/v=2/c=0

Globally
Routable
Label

parc
A Xerox Company

Forwarding Label

/parc/ccnx/presentations/slide10/v=2/c=0

Globally
Routable
Label

parc
A Xerox Company

Similarity Label

/parc/ccnx/presentations/slide10/v=2/c=0

?KeyId

?ContentObjectHash

Similarity
Label

parc
A Xerox Company

Express Headers

parc
A Xerox Company

CCN - Messages

Interest

Base CCN Message used to request Content Objects

parc
A Xerox Company

Interest elements

Name
Hierarchical resource identifier. Used for routing and matching.

KeyIdRestriction (optional)
Matching restriction based on a specific key

ContentObjectHashRestriction (optional)
Matching restriction based on a content hash

LifetimeRestriction (optional with default value)
Lifetime restriction of the Interest in-network.

Payload (optional)
Data

parc
A Xerox Company

Content Object

Base CCN Message used to answer an Interest

parc
A Xerox Company

Content Object elements

Name
Hierarchical resource identifier.

Validation Information (optional)
Validation Algorithm
Validation Information

Payload (optional)
The data.

Metadata (optional)
The data.

parc
A Xerox Company

Content Object elements

Protocol Information (optional)

Structured information about the content object. Protocols store information here.

Payload

The data.

Signature Block

Cryptographic signature of the Content Object.

parc
A Xerox Company