Overview of NDN Platform, Application Libraries, and API

NDNComm 2014

September 4, 2014

jburke@ucla.edu

Motivation

"The NDN project's approach is to design and build a variety of applications on NDN to **drive the development and deployment of the architecture and its supporting modules, to test prototype implementations**, and to encourage community use, experimentation, and feedback into the design.

"Application-driven development also allows **verification and validation of performance and functional advantages of NDN**, such as how routing on names promotes efficient authoring of sophisticated distributed applications, by reducing complexity, opportunities for error, and **time and expense of design and deployment**."

- Afanasyev et al., "Named Data Networking," CCR July 2014.

NDN Platform

- **Provide a coherent, usable, and well-documented "platform" for exploring NDN in practical applications – for the NDN project team and external users.**

- Use a release "heartbeat" to stimulate interoperability testing and discussion of how the various moving parts work together.

- Along the way, improve access to and consistency of various NDN code projects.

- Open and lightweight process, with no unrealistic centralization or over-management but clear ownership of each component project.

- Managed nodes on the testbed run the Platform.

NDN Platform 0.1 (Aug '13)

- NDNx – Team fork of CCNx
  - C API support only, no Java, limited Android support
  - Include NDNLP
  - Break in API compatibility with CCNx
- NDN-CCL – Common Client Libraries
  - ndn-cpp, ndn-js, PyNDN (also break in compatibility?)
  - Naming conv change for Interest/ContentObject?
- NDN on Node
- ndnSIM v.5 – Simulator
- NDN Network Tools – ping, ndndump
- Package manager support – Macports

NDN Platform 0.3 (August 2014)

- **NFD** NDN Forwarding Daemon, version 0.2.0
- **ndn-cxx** library, version 0.2.0
  - The NDN C++ library with eXperimental eXtensions (CXX)
  - The ndnsec security tools to manage security identities and certificates
- **NDN-CCL** - NDN Common Client libraries suite, version 0.3
  - NDN-CPP C++ / C library
  - PyNDN2 Python library
  - NDN-JS JavaScript library (with Node.js support)
  - jNDN Java library (preliminary)
- **NLSR** - Named Data Link State Routing Protocol , version 0.1.0
- **repo-ng** - next generation of NDN repository , version 0.1.0
- **ndn-tlv-ping** - ping application for NDN , version 0.2.0
- **ndn-traffic-generator** - traffic generator for NDN , version 0.2.0
- **ndndump** - packet capture and analysis tool for NDN , version 0.5

- Preliminary binary package support on Ubuntu, MacOS X, others…

Supported platforms

- Required
  - Two most recent Ubuntu LTS releases with the gcc which comes with apt, both 32-bit* and 64-bit, 2GB memory
  - Latest three OS X releases with the clang which comes with XCode, 64-bit, 2GB memory
- Optional
  - Latest Ubuntu release (if not LTS) with the gcc which comes with apt, 64-bit, 2GB memory
  - Latest Windows with Visual Studio, 64 bit, 2GB
  - Latest FreeBSD "RELEASE" with the clang which comes with ports, 64 bit, 2GB memory

http://named-data.net/codebase/platform/documentation/ndn-platform-development-guidelines/#Build_support

Licensing

- GPLv3 applications (mostly)
- LGPLv3 libraries
- Open and cost-free

Community support

- One Github repo for all code
- Public redmine
- Per-component wiki
- Code review
- Continuous integration
- Technical reports (NFD, NDN-CCL, etc.)
- Mailing lists

Open to contributors and collaborators!

Evolution of the libraries

- All libraries now reflect fundamental architectural abstractions directly in objects, and wire format manipulation is abstracted.
  - Name
  - Interest
  - Data
  - Face
  - KeyChain

- Two library efforts available to community
  - NDN-CXX: “C++ for eXtended eXperimentation”
  - NDN-CCL: “Common Client Libraries”
  - *Enables diversity of coding choice*
  - *Drives us towards specification (and not just implementation)*

ndn-cxx: **NDN C**++ library for e**X**tended e**X**perimentation

- Application-driven iterative library extension and evolution
  - Stable but evolving API based on application needs
    - New and extended APIs to support application patterns
  - Playground for experimental features
- Prioritizing developer productivity for experimentation
  - encouragement and extensive use of Boost libraries and modern application patterns
    - leveraging > 7000 person years of high-quality code*
  - multiple utility classes and helpers to simplify common operations
- Purity
  - Pure C++ implementation, adherent to OOP principles
  - Simplified maintenance and extensibility

* http://www.boost.org/development/index.html

ndn-cxx: Available Functionality

- Base
  - Fully asynchronous communication model based on Boost.Asio
  - Single-threaded, but thread-safe Face operations
  - Explicit time management based on Boost.Chrono
  - Test-driven development with continuous integration
- Security library with latest extensions and experimental features
  - Security primitives to simplify development of NDN applications
  - Flexible trust model for packet validation
    - set of built-in trust models
    - policy-based custom trust model definition
- Utility classes
  - Scheduler, NDN regular expressions, NFD management protocols helpers, random number generator, digest calculation, routines to work with time, NFD management support, security credentials IO, security library, etc
- Miscellaneous tools
  - ndnsec tools to manage security, tlvdump to visualize NDN-TLV

ndn-cxx: Usage Cases

- The library is currently being used as part of the following projects:
  - NFD - NDN Forwarding Daemon
  - NLSR - Named-data Link-State Routing protocol
  - repo-ng - Next generation of NDN repository
  - ChronoChat - Multi-user NDN chat application
  - ChronoSync - Sync library for multiuser realtime applications for NDN
  - ndn-tlv-ping - Ping Application For NDN
  - ndn-traffic-generator - Traffic Generator For NDN

NDN-CCL “Common Client Library”

- Encourage development and experimentation with NDN for a large audience of developers
- Reasonably consistent, stable API across multiple platforms and languages, plus language-specific syntax (typically more concise)
- Minimal dependencies or assumptions about threading/memory management for easier integration with applications
- Track updates to message protocols and the TLV wire format
- Incorporate advances from NDN research projects as library modules to speed adoption by applications (Security, Sync, etc.)
- Provide install packages where possible so applications can deploy easily

Features

- Languages
  - C++ with C core
  - Python (2 and 3),
  - JavaScript (browser and Node.js)
  - Java (preliminary)

- Helper functions
  - Basic interaction with NFD including Signed Interests
  - Protobuf+TLV based cross-platform message description
  - MemoryContent Cache

- Port of security library developed for ndn-cxx (Y. Yu)
  - Full support in C++ and Python
  - Preliminary support in other languages

- Port of ChronoSync 2013 (experimental feature)
  - Full support in C++ and Javascript

NDN-CCL Applications

- **CCNx Federated Wiki**, an NDN port of the Smallest Federated wiki (NDN-JS)
- **Chronochat-js**, a javascript implementation of the ChronoChat demonstration application (NDN-JS)
- **Matryoshka**, an experimental multi-player online game using NDN and the Unity3D game engine. (jndn as the basis of the .NET port of CCL used in this project.)
- **ndn-bms**, a building management system prototype being developed as part of the NDN-NP project (PyNDN, NDN-JS)
- **ndn-lighting**, lighting control application using NDN (PyNDN, NDN-JS)
- **ndn-protocol**, a firefox browser plug-in supporting an ndn:/ retrieval scheme (NDN-JS)
- **ndnfs** and **ChronoShare**, NDN file sharing platforms (PyNDN, NDN-JS – with ndn-cxx)
- **NDNoT**, the Named Data Network of Things toolkit for the Raspberry PI (PyNDN, NDN-JS)
- **ndnrjs**, a javascript implementation of an NDN repository (NDN-JS)
- **ndnrtc**, a peer-to-peer multiparty audio, video, and chat application over NDN. (NDN-CPP, NDN-JS)
- **ndnstatus**, the NDN routing status web page (PyNDN, NDN-JS)
- **NDNVideo**, a video playout application for NDN (PyNDN)

Coming:

- **NDNEx**, an NDN-based mobile health application being developed as part of the NDN-NP research project. (jndn)
- **OpenPTrack-NDN** an open source person tracking system that will add NDN support in Fall 2014. (NDN-CPP)

Research: New Protocols + Advanced APIs

- Consumer / Producer API

  Moiseenko & Zhang. NDN Technical report #17, 2014.

- SYNC: Efficient synchronization of namespaces

  Zhu & Afanasyev. "Let's ChronoSync: Decentralized dataset state synchronization in Named Data Networking." ICNP, 2013.

Security approach

- Consistent across ndn-cxx and NDN-CCL
- Reference implementation is ndn-cxx security components
- Data packets typically signed and verified with a default key of RSA 2048-bit. ECDSA also supported. (Probably other more efficient verification techniques in the future.)
- Most flexibility, power, and challenges in how trust is managed – still an open area of research.

ndn-cxx Security Library

Abstractions

- **Certificate** (same as signing certificate or identity certificate)
  - identified by NDN identity certificate name
  - carries "real-word" identity and other meta information
  - associated with the private key
- **Key** (same as signing key)
  - identified by a "logical" name of a private key
  - "container" for the public (derived from the private) key certificates
- **Identity**
  - defines signing namespace and is identified by this namespace
  - container for one or multiple keys

http://named-data.net/doc/ndn-cxx/current/tutorials/security-library.html

Data and Interest Validation

- Two-part Validation
  - Check if key is authorized to sign Data/Interest
    - name of the key matches Data/Interest name based on some rule
    - until reaching a trust anchor or step limit
  - Check signature validity
- Application defines when to do when packet is received
  - Either manually or using “Validator”-derived class
  - ValidatorNull: null-validation
  - ValidatorRegex
    - compile-time defined “rules” and trust anchor
  - ValidatorConfig
    - run-time defined “rules” and trust anchors

ValidatorRegex

- Compile-time configuration
  - Set of NDN regular expression rules
  - Set of trust anchors
  - Lifetime of trusted certificate cache
  - Limit on certification chain depth

```
Face face;
ValidatorRegex validator(face);

// Hierarchical Trust Anchor
IdentityCertificate anchor = ...;
validator.addTrustAnchor(anchor);

// Hierarchical Trust Rules
SecRuleRelative rule(
  // Extract authority namespace from data
name
  "(<>*)",
  "\\1",
  // Extract authority namespace from key
name
  "(<>*)<KEY><ksk-.*><ID-CERT><>$",
  "\\1",
  // Key's authority namespace must be
parent of data's namespace
  ">"));

validator.addRule(rule);
```

Configuration File Based Validator

- Compile-time configuration
  - Lifetime of trusted certificate cache
  - Limit on certification chain depth
- Run-time configuration (configuration file)
  - Set of NDN regular expression rules
  - Set of trust anchors

```
; One or more "rule"
rule
{
  id "<id>"
  for data ; or  "for interest"

  ; Apply the rule only for packet that
match the filter
  filter
  {
     ...
  }

  ; Make a decision of valid/invalid
based on the checker configuration
  checker
  {
     ...
  }
}

; One or more "trust-anchor"
trust-anchor
{
  ...
}
```

Naming Conventions

- Namespace design is a critical component of application development.
- *"Where possible, put it in the name"* philosophy is expressing some packet requirements / features in the name.
- Three areas covered here
  - Scope control obeyed by NFD: /localhost, /localhop
  - Signed interest format
  - Versioning, segmenting, etc.

Naming Conventions

- Namespace design is a critical component of application development.
- *"Where possible, put it in the name"* philosophy is expressing some packet requirements / features in the name.
- Three areas *of library support* covered
  - Scope control obeyed by NFD: /localhost, /localhop
  - Signed interest format
  - Versioning, segmenting, etc.

For scope control

- http://redmine.named-data.net/projects/nfd/wiki/ScopeControl

- **`/localhost`**

  Limits propagation to the applications on the originating host.
  It is equivalent to Scope=1.

- **`/localhop`**

  The localhop scope limits propagation to no further than the next node.
  It is equivalent to Scope=2.

For groups of objects

- Technical Report #22, “Naming Conventions”
- **Marker** (Platform v0.3) vs. **Hierarchy**-based encoding (Platform v.0.4 targeted)
- Segmenting
  - Cut large data (e.g. video frame) into packet-sized pieces
  - Final segment indicated by MetaInfo FinalBlockID
  - appendSegment/toSegment, appendSegmentOffset/toSegmentOffset
- Versioning
  - Data packet is immutable: a new version needs a new name
  - Suggest millisecond time stamp but not required
  - appendVersion/toVersion
- Time-stamping
  - When data packet was produced
  - Microseconds since January 1, 1970
  - appendTimestamp/toTimestamp
- Sequencing
  - Sequential items in a collection
  - 0, 1, … X. Assume item X + 1 may be produced
  - appendSequenceNumber/toSequenceNumber

For Interest signing

- **`/command/params/<timestamp>/<random-value>/<SignatureInfo>/<SignatureValue>`**
  - `params` may be TLV-encoded data
- The signature covers name components through SignatureInfo
- SignatureInfo and SignatureValue the same as a Data packet
- Example: Register
  - **`/localhost/nfd/rib/register/<control-parameters>/<timestamp>`**…
  - ControlParameters ::= CONTROL-PARAMETERS-TYPE TLV-LENGTH Name? FaceId? Uri? LocalControlFeature? Origin? Cost? Flags? Strategy? ExpirationPeriod?

What's next

"A few years of designing and developing prototype applications on NDN has revealed five key areas of application research that map to important features of the architecture:

(1) namespaces;

(2) trust models;

(3) in-network storage;

(4) data synchronization;

(5) rendezvous, discovery, and bootstrapping."

- Afanasyev et al., "Named Data Networking," CCR July 2014.

Also, see Tech Report #17 for ideas on API evolution.

Where to find things

http://named-data.net

http://github.com/named-data

http://named-data.net/doc/NFD/
http://named-data.net/doc/NLSR/
http://named-data.net/doc/ndn-cxx/
http://named-data.net/codebase/platform/ndn-ccl/

Overview of NDN Platform, Application Libraries, and API

NDNComm 2014

September 4, 2014

jburke@ucla.edu