NDN Common Client Libraries

November 13, 2013 – jefft0@ucla.edu

Agenda

- Motivation/goals of the libraries
- Using the libraries
- Example fetch
- Conclusion/future development

Motivation/goals

- NDN Project report, section 3.1.4 “Libraries”
- Clean slate/portable libraries
- Make it easy for developers to create applications
- Multiple languages: C++, JavaScript, Python, and (soon) Java
- With Common Client Library API
  - named-data.net/codebase/platform/documentation/ndn-ccl-api
- C++: Few assumptions about memory management, linked libraries, threading

Main Classes

- Name
- Interest (Name + selectors)
- Data (Name + Content + MetaInfo + Signature)
- Face (expressInterest, registerPrefix)
- KeyChain (see Yingdi Yu's security talk)

Example: Name and Interest

```
var name = new Name("/test/hello.txt");
var interest = new Interest(name);
Interest.setChildSelector(1);
Interest.setInterestLifetimeMilliseconds(1000);
console.log(interest.toUri());
```

/test/hello.txt?ndn.ChildSelector=1&ndn.InterestLifetime=1000

Data (CCL documentation)

Data Constructor

Create a new Data with the optional values.

```
[C++]
Data(
    [const Name& name]
);
[JavaScript]
var Data = function Data(
    [name           // Name]
)
[Python]
def __init__(self
    [, name         // Name]
)
```

Parameters

name

(optional) The name for the data packet.

Face (CCL table of contents)

Face expressInterest

Face.expressInterest Method (from Interest)

Send the interest through the transport, read the entire response and call onData. If the interest times out according to interest lifetime, call onTimeout (if not omitted).

[C++]

```
    unsigned int expressInterest(
    const Interest& interest,
    const OnData& onData,
    [, const OnTimeout& onTimeout]
    [, WireFormat& wireFormat]
);
```

Parameters

interest

The Interest to send which includes the interest lifetime for the timeout.

onData

When a matching data packet is received, this calls onData(*interest*, *data*) where:
interest is the interest given to expressInterest.
data is the received Data object.

onTimeout

(optional) If the interest times out according to the interest lifetime, this calls onTimeout(*interest*) where:
interest is the interest given to expressInterest.

wireFormat

(optional) A WireFormat object used to encode the message. If omitted, use WireFormat getDefaultWireFormat ().

Returns

The pending interest ID which can be used with removePendingInterest.

KeyChain (from NDNFS)

```
shared_ptr<KeyChain> keychain(new KeyChain
  (make_shared<IdentityManager>
    (make_shared<BasicIdentityStorage>(),
     make_shared<OSXPrivateKeyStorage>()),
   make_shared<NoVerifyPolicyManager>()));

keychain->signByIdentity(data, signer);
SignedBlob wire_data = data.wireEncode();
…
sqlite3_bind_blob(s, wire_data.buf(), wire_data.size());
```

WireEncoding

- expressInterest, etc. take optional WireEncoding argument
- WireEncoding processes abstract Interest, Data to wire format
- Currently supports BinaryXmlEncoding. TLV support soon.
- Example: use a different wire format on a certain face
- Example: use a different wire encoding to sign a Data packet:

```
keyChain.sign(data, certificateName, wireFormat);
transport.send(data.wireEncode(wireFormat));
```

Example fetch (NDN-JS)

```
function onData(interest, data) {
  console.log("Received: " + data.name.toUri());
  console.log(data.content.toString());
}

var face = new Face
  ({host: "C.hub.ndn.ucla.edu", port: 9695});
face.expressInterest
  (new Name("/ndn/ucla.edu/apps/ndn-js-test/hello.txt”),
   onData);
```

Example fetch (NDN-CPP)

```
void onData
   (const shared_ptr<const Interest>& interest,
    const shared_ptr<Data>& data) {
   cout << "Got " << data->getName().to_uri() << endl;
}

void fetch() {
   Face face("C.hub.ndn.ucla.edu”, 9695);
   face.expressInterest
    (Name("/ndn/ucla.edu/apps/ndn-js-test/hello.txt/"),
     onData);
}
```

Getting the Libraries

- git clone https://github.com/named-data/**ndn-cpp**.git
- git clone https://github.com/named-data/**ndn-js**.git
- git clone https://github.com/named-data/**jndn**.git (soon)
- git clone https://github.com/named-data/**PyNDN**.git

Development status

- NDN-CPP v0.2 (C++): Mature
  - Applications: RTC Video, NDNFS file system
- NDN-JS v0.2 (JavaScript): Mature
  - Browser and Node.js support
  - Applications: Ping, Testbed monitor, web ndnls, Smallest federated wiki
- PyNDN v0.2 (Python): Mature but with old API on top of NDNx
- JNDN (Java): In progress. Plan first release in Dec.

- How could the library be improved to use in your project?

Conclusion/future work

- Libraries are available now for development
- Libraries will track research progress on security, usage patterns
- Future work
  - Release JNDN Java/Android library
  - Implement TLV wire format
  - Support for synchronization protocols
  - NDN-JS security library (trust policy)
  - Python: stand-alone independent of C bindings?
  - Others?

Quarterly releases: named-data.net/codebase/platform

Devel: github.com/named-data