NDN Common Client Libraries API

Jeff Thompson, Jeff Burke
Center for Research in Engineering, Media and Performance (REMAP)
University of California, Los Angeles (UCLA)

Introduction

Design Goals

The NDN Common Client Libraries (CCL) provide a common application programming interface (API) across several languages for building applications that communicate using Named Data Networking (NDN), with these goals:

- Encourage development and experimentation with NDN for a larger audience of developers
- Multiple platforms and languages: C++ (std or boost), Python (version 2 and 3), JavaScript (browser and Node.js) and Java
- Consistent, stable API across all languages
- Minimal dependencies or assumptions about threading/memory management for easier integration with applications
- Provide install packages so applications can deploy easily
- Track updates to message protocols and the TLV wire format
- Incorporate advances from NDN research projects as library modules to speed adoption by applications (Sync, etc.)

Example Code

Following are examples of the CCL API in several programming languages. The sample creates a Face, calls expressInterest, then loops calling processEvents. The onData callback displays the data packet's Name and content. We also handle the onTimeout callback.

C++ (ndn-cpp)

```
#include <iostream>
#include <ndn-cpp/face.hpp>
using namespace ndn;
using namespace ndn::func_lib;

class Counter {
public:
  int callbackCount_;
  Counter() { callbackCount_ = 0; }

  void onData(const ptr_lib::shared_ptr<const Interest>& interest,
              const ptr_lib::shared_ptr<Data>& data) {
    ++callbackCount_;
    std::cout << "Name: " << data->getName().toUri() << std::endl;
    std::cout << "Content: " << data->getContent() << std::endl;
  }
  void onTimeout(const ptr_lib::shared_ptr<const Interest>& interest) {
    ++callbackCount_;
    std::cout << "Timed out: " << interest->getName().toUri() << std::endl;
  }
};

int main(int argc, char** argv) {
  Face face("memoria.ndn.ucla.edu");
  Counter counter;
  Name name("/ndn/edu/ucla/remap/ndn-js-test/howdy.txt");
  face.expressInterest(name, bind(&Counter::onData, &counter, _1, _2),
                       bind(&Counter::onTimeout, &counter, _1));
  while (counter.callbackCount_ < 1) {
    face.processEvents();
    usleep(10000);
  }
}
```

JavaScript (ndn-js for Node.js)

```
var Face = require('../..').Face;
var Name = require('../..').Name;

var onData = function(interest, data) {
  console.log("Name: " + data.getName().toUri());
  console.log("Content: " + data.getContent().buf().toString());
};

var onTimeout = function(interest) {
  console.log("Interest timed out: " + interest.getName().toUri());
};

var face = new Face({host: "memoria.ndn.ucla.edu"});
var name = new Name("/ndn/edu/ucla/remap/ndn-js-test/howdy.txt");
face.expressInterest(name, onData, onTimeout);

// Don't need a processEvents loop. JavaScript is already asynchronous.
```

Python (PyNDN)

```
import time, trollius as asyncio
from pyndn import ThreadsafeFace, Name

class Counter(object):
    def __init__(self):
        self._callbackCount = 0
    def onData(self, interest, data):
        self._callbackCount += 1
        print("Name: " + data.getName().toUri())
        print("Content: " + str(data.getContent()))
    def onTimeout(self, interest):
        self._callbackCount += 1
        print("Interest timed out: " + interest.getName().toUri())

loop = asyncio.get_event_loop()
face = ThreadsafeFace(loop, "memoria.ndn.ucla.edu")
counter = Counter()
face.stopWhen(lambda: counter._callbackCount >= 1)
name = Name("/ndn/edu/ucla/remap/ndn-js-test/howdy.txt")
face.expressInterest(name, counter.onData, counter.onTimeout)
loop.run_forever()
```

Java (jndn)

```
package net.named_data.jndn.tests; // (Imports omitted to save space.)
class Counter implements OnData, OnTimeout {
  public int callbackCount_ = 0;

  public void onData(Interest interest, Data data) {
    ++callbackCount_;
    System.out.println("Name: " + data.getName().toUri());
    System.out.println("Content: " + data.getContent().toString());
  }
  public void onTimeout(Interest interest) {
    ++callbackCount_;
    System.out.println("Timed out: " + interest.getName().toUri());
  }
}

public class TestGetAsync { public static void main(String[] args) {
  try {
    Face face = new Face("memoria.ndn.ucla.edu");
    Counter counter = new Counter();
    Name name = new Name("/ndn/edu/ucla/remap/ndn-js-test/howdy.txt");
    face.expressInterest(name, counter, counter);
    while (counter.callbackCount_ < 1) {
      face.processEvents();
      Thread.sleep(5);
    }
  } catch (Exception e) {}
} }
```

C# (ndn-dot-net) *in development*

```
// (using directives omitted to save space.)
class Counter : OnData, OnTimeout {
  public int callbackCount_ = 0;

  public void onData(Interest interest, Data data) {
    ++callbackCount_;
    System.Diagnostics.Debug.WriteLine("Name: " + data.getName().toUri());
    System.Diagnostics.Debug.WriteLine("C: " + data.getContent().toString());
  }
  public void onTimeout(Interest interest) {
    ++callbackCount_;
    System.Diagnostics.Debug.WriteLine("T/O: " + interest.getName().toUri());
  }
}

public class TestGetAsync { public static void Main(string[] args) {
  try {
    var face = new Face("memoria.ndn.ucla.edu");
    var counter = new Counter();
    var name = new Name("/ndn/edu/ucla/remap/ndn-js-test/howdy.txt");
    face.expressInterest(name, counter, counter);
    while (counter.callbackCount_ < 1) {
      face.processEvents();
      System.Threading.Thread.Sleep(5);
    }
  } catch (Exception e) {}
} }
```

UCLA

UCLA remap