Security in ChronoShare

Leaders:

Yukai Tu, Zhiyi Zhang

Security in ChronoShare

ChronoShare is a completely decentralized distributed file sharing application builds on top of the Named Data Networking(NDN) architecture.

Using ChronoSync, users exchange their knowledge about the file state digest through an interest, called sync interest. And fetch the actions and corresponds files through request Interest.

※If sync replies are not secured, an attacker may force users to fetch malicious data from a fake “user”. If information request are not secured, attackers can easily launch impersonation attack. File access control is also an important security part in ChronoShare. For better usage purpose we need to implement security part in ChronoShare.

Security in ChronoShare

1. ChronoShare may need user authentication module. The group initiator can authenticate people into group and information can only be shared within the group.

2. For data level, when users require the secure data transmission, the data encrypt is also needed in ChronoShare.

3. The file access control can be introduced and guarantee the file permissions for certain users.

Initial Implementation of Interest Digest in NFD

- In NFD 0.5.1, when Data arrives:
  - NFD finds all PIT entries that can be satisfied by this Data, through name-based lookup
  - This lookup has significant computation cost.
- Interest digest proposal:
  - PIT is indexed by SHA1 digest over Interest Name+Selectors+Link.
  - Link layer header of Data packet carries a digest of the Interest, so NFD can find PIT entry quickly.
- This project starts the implementation of Interest digest in NFD forwarding,
  - serves as a starting point of further studies on protocol semantics and forwarding performance.
  - We will focus on the performance aspect of Interest digest. While Interest digest is proposed as a solution to prevent cache poisoning attacks with Link objects, we will not study that aspect.

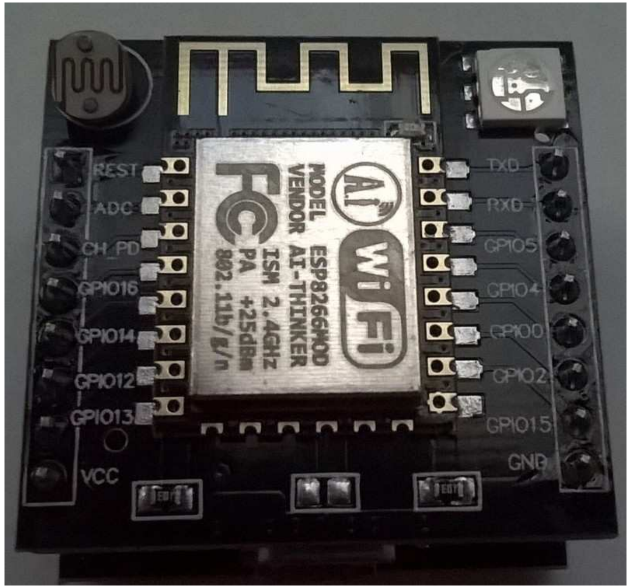
Initial Implementation of Interest Digest in NFD

- Tasks:
  - Change ndn-cxx Face to add Interest digest onto Interest/Data/Nack packets.
  - Index PIT with Interest digest instead of names. Modify Data/Nack forwarding pipelines.
- You need:
  - C++11, ndn-cxx Face API, NFD forwarding pipelines
  - a computer capable of running 3 NFD instances, OR access to Open Network Laboratory (ONL)
- Project author will loan 2x ESP8266 Witty Cloud boards.
  - Max team size is 3. Some tasks do not require using an ESP8266 all the time.
- Project outcome:
  - forwarding performance benchmark
  - a better understanding of Interest digest implications

- **Junxiao Shi**

NDN Stack for ESP8266 Microcontroller

- Meet the ESP8266
  - 80MHz RISC CPU, 96KB data RAM
  - 802.11 b/g/n WiFi chip, full TCP/IP stack
  - GPIO pins, analog input, etc
  - small package, USD $3 each
- esp8266ndn library:
  - Arduino library to bring NDN to ESP8266
  - Face with UDP transport, Interest+Data, no dispatch
  - ndnping client and server, no prefix registration
- This project: let ESP8266 do more NDN
  - prefix registration
  - ECDSA signing and verification
  - Nack



- Junxiao Shi

NDN Stack for ESP8266 Microcontroller

- The little ESP8266 is western maker’s favorite microcontroller with built-in WiFi.
  - Giving it NDN makes it more powerful, and makes NDN more appealing.
- You need:
  - knowledge about signed Interests
  - C++11, ndn-cxx or ndn-cpp-lite API
  - Arduino IDE and NFD on your computer, a full size USB 2.0 port
- Project author will loan 3x ESP8266 units (Witty Cloud or NodeMCU).
  - Max team size is 4. Some tasks do not require using an ESP8266 all the time.
- Project demos:
  - ndnping server with prefix registration and Nack
  - light control with ECDSA verification

- **Junxiao Shi**

Mini-NDN Wi-Fi

- Motivation and problem statement
  - Currently no Wi-Fi emulation program exists to test NDN over Wifi. Mininet-Wifi can be used to provide such a functionality

Contribution to NDN
  - Mini-NDN Wi-Fi would provide a rapid prototyping environment for testing wireless networks
- Tasks
  - Determine how to use Mininet-Wifi, how it works.
  - Run NDN over the Wifi-nodes and enable mobility.
  - Write/Modify framework to experiment and test NDN scenarios over wireless networks.
  - Enable Ad-Hoc functionality in the Wifi nodes.
  - Set-up a vehicular ad-hoc networking environment using the Wifi-Nodes.
- Required knowledge for participants
  - NDN, Mininet, Mininet-Wifi
  - Python, Shell
- Expected outcome
  - NDN is working over virtual Wi-Fi interfaces provided by Mininet-Wifi
  - Experimental framework is provided
  - Some sample experiments and scenarios are developed

- **Ashlesh Gawande,** Muktadir Chowdhury

Namespace syncing with the CNL

- Motivation and problem statement
  - The experimental Common Name Library (CNL) maintains an abstraction of the application's namespace and alerts the application new names are or content is attached.
  - If Interest selectors (rightmost child) are not available for name discovery, the CNL could enable applications to update local copies of names in a namespace and search locally.
- Contribution to NDN
  - An API for sync-based name discovery as an alternative to Interest selectors.
  - A ChronoSync protocol for syncing names (adaptation of existing approaches?)
  - A demo of the CNL as an abstraction for NDN applications.

- **Jeff Thompson,** Jeff Burke

Namespace syncing with the CNL

- Tasks

  1. Familiarize participants with the Python implementation of the CNL, PyCNL.

  2a. Make a stub client application to gets CNL notifications of a new name, fetch and display it.

  2b. Make a ChronoSync-based protocol to synchronize a names and update the CNL namespace.

  3. Merge tasks 2a and 2b to display the latest version of a document from a producer.

  4. Time permitting: Use the CNL namespace to "time travel" to earlier versions.

- Required knowledge for participants
  - Python. Basic NDN networking. (optional: ChronoSync)

- Expected outcome
  - The ChronoSync-based name synchronization protocol
  - A handler for the Common Name Library (Python) that implements it
  - Demonstrate an application to sync the latest version of a document

- **Jeff Thompson,** Jeff Burke

NDNProSe - NDN for Proximity Services

- Motivation and problem statement
  - Exploit NDN communication protocol for Proximity-based Services (ProSe)
- Tasks
  - Modify the base NDN library to support BLE advertisement
  - Develop Android library and app for basic single-hop discovery
  - Extension (and optimization) for multi-hop discovery
- Required knowledge for participants
  - C/C++ and/or Java language, Android development
  - General software engineering and wireless communication background
- Expected outcome
  - NDN module to support proximity discovery
  - Live demo with Android smartphones

- **Jacopo De Benedetto**

NFD Control Center GUI Improvement

- Motivation and problem statement
  - NFD Control Center GUI needs to be improved to provide more convenient usage for both NDN developers and users
  - NDNCERT is integrated into NFD Control Center with very simple GUI design
  - NFD full management GUI can also be improved to provide much easier and more convenient way to check and manage local NFD
- Contribution to NDN
  - GUI improvement will simplify the usage of NFD management and NDN certificate management so that will make NDN closer to people
- Tasks
  - NDN certificate management
    - Redesign the list of identity and certificate display and move the operation button to right-click menu on each entry.
    - Redesign the certificate application window. Put all steps into one integrated window.

- **Qi, Alex**

NFD Control Center GUI Improvement

- NFD management
  - Redesign FIB and RIB display from list to tree.
  - Show the detailed Face information when checking the specific FIB or RIB entry.
  - Move management operations to the right-click menu on each entry.

- Required knowledge for participants
  - C++, Qt (or other GUI design framework)

- Expected outcome
  - Satisfy most of the hackathon attendees

- **Qi, Alex**

NDN Maps application

- Motivation and problem statement
  - The idea is to design and develop an outdoor map application that will use the NDN architecture:
    - The application will include a client (mobile device app) running on an android platform and a server that will act as a NDN gateway between the client and google maps servers (we will use their API)
- Contribution to NDN
  - Explore NDN features on an android environment
  - Explore the namespace design for outdoor maps
  - Create a new type of applications

- **Siham Khoussi (NIST)**

NDN Maps application

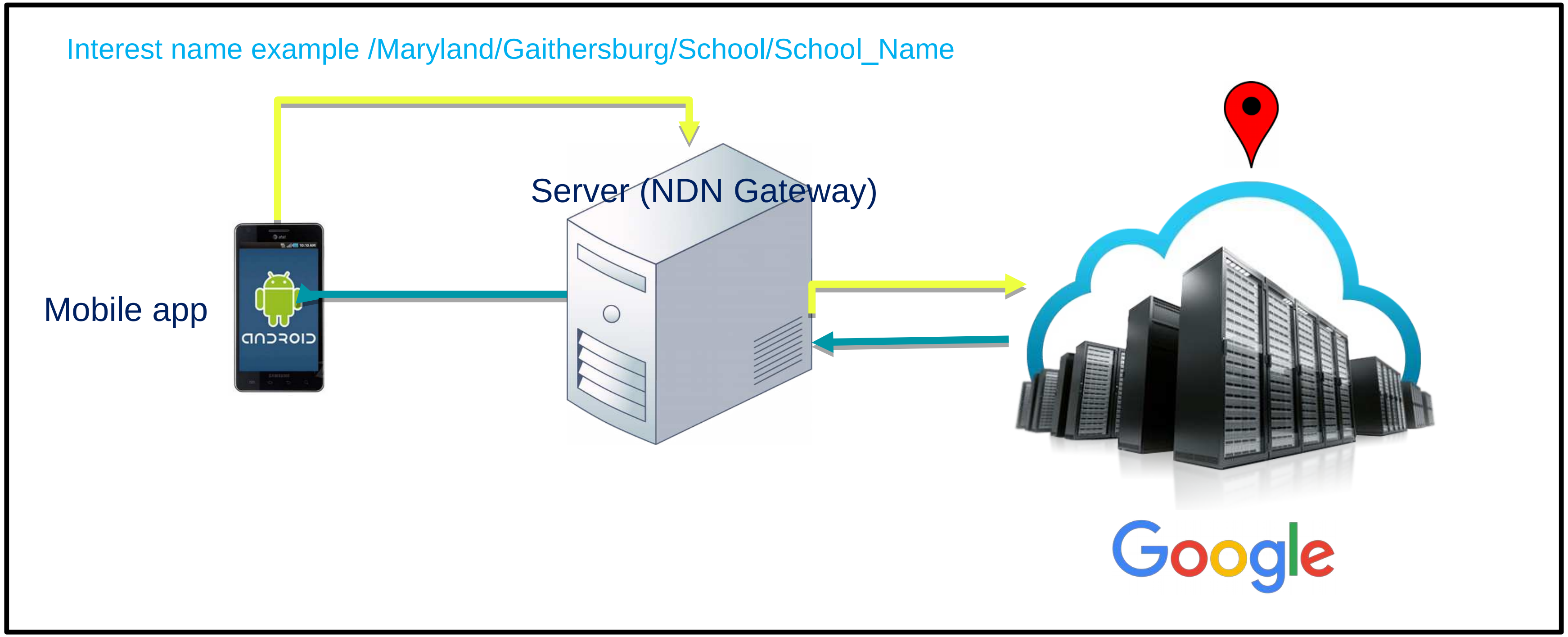

Interest name example /Maryland/Gaithersburg/School/School_Name
Server (NDN Gateway)
Mobile app
Google

Figure 1: Proposed architecture

- Siham Khoussi (NIST)

NDN Maps application

- Tasks
  - Come up with a suitable namespace for this application
  - Design a suitable application architecture (e.g. Figure 1)
  - Configure NDN on an android platform
  - Develop the client side (user’s mobile device application):
  - Develop the server side
- Required knowledge for participants
  - Be familiar with JAVA and android development
  - Be familiar with web app development (e.g. JavaScript)
  - Be familiar with the NDN forwarder
  - Be familiar with any of the NDN libraries
- Expected outcome
  - Demo: Get GPS locations, display maps and get directions to a desired location through NDN naming with the developed mobile application

- **Siham Khoussi (NIST)**

ndnSIM Mobile Simulation Package

- Motivation and problem statement
  - Implement new simulation scenarios in mobile environments for ndnSIM

- Open to suggestions. Starting point: vehicular environment of paper "Rapid Traffic Information Dissemination Using Named Data”

- Contribution to NDN
  - Write ndnSIM code that will be used by hundreds of users all around the world and enrich the NDN simulation environments available in ndnSIM

- Tasks
  - Design the simulation environment and scenario(s)
  - Design the actual implementation and decide if changes have to be made in ndnSIM or NFD (the goal is to make minimal or no modifications, if possible)
  - Implement the scenario(s) and any changes required

- Required knowledge for participants
  - 1) C++ , 2) ndnSIM, 3) Link-layer capabilities of NS-3 (NetDevice, models, etc.), 4) Potentially NFD

- Expected outcome
  - ndnSIM port that will include the implemented environment or ndnSIM scenario template, if no major changes in the ndnSIM core/NFD are required
  - The simulation scenario execution (using the visualizer if possible) will be demonstrated

- **Project Leader: Spyros Mastorakis**

Software-defined Named Data Networking

- Motivation and problem statement
  - SDN and ICN can benefit from each other
    - SDNs can benefit from the power of caching from NDNs.
    - NDNs can be adopted with little effort by already existing SDNs.
    - The greater management control and monitoring over the network of SDN could simplify NDN management.
  - Researchers have been using SDN-like framework to address NDN management issues, but …
    - SDN controller cannot directly be deployed due to lacking the devices' support of header parsing
  - P4 framework could allow programming of packet forwarding planes
    - Define parsers for custom packet headers
    - Allow SDN controller to manage user-defined switches

- **Chengyu Fan,** NDN group

Software-defined Named Data Networking

- Contribution to NDN
  - The implementation will be implemented as a standalone project.
  - NDN in P4 can be one method for future NDN deployment.
  - Researchers can extend this code for their own requirements.
- Tasks
  - Set up the testbed using VMs
  - Settle down implementation details.
  - Implement NDN.p4, and integrate it with minindn.
- Required knowledge for participants
  - C++/Python, SDN, P4, NDN
- Expected outcome
  - Controller sets up the forwarding path when using ndn-cat-chunks for demonstration

- **Chengyu Fan,** NDN group

NDNS Problem Statement

What is NDNS:

- DNS-Like Name Service for NDN
- Initially, serving Link Objects and Certificates
- However, compared with how we can set up DNS today, NDNS is not quite user-friendly in:
- Using command-line tools to set records
- Writing configuration file to configure serving zones

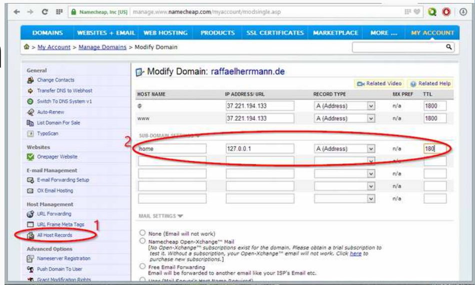
Build a WebUI for NDNS!

At the end, you should build a NDNS web application that has functions including:

create/delete a zone.

list zone information.

add/remove hosting zones(by changing configuration
restart ndns-deamon).



Modify Domain: raffaelherrmann.de

HOST NAME	IP ADDRESS/ URL	RECORD TYPE	MX PREF	TTL
@	37.221.194.133	A (Address)	n/a	1800
www	37.221.194.133	A (Address)	n/a	1800
home	127.0.0.1	A (Address)	n/a	180
			n/a	
			n/a	
			n/a	
			n/a	

None (Email will not work)

Namecheap Open-Xchange™ Mail
[No Open-Xchange™ subscriptions exist for the domain. Please obtain a trial subscription to test it. Without a subscription, your Open-Xchange™ email will not work. Click here to purchase new subscriptions.]

Free Email Forwarding
Email will be forwarded to another email like your ISP's Email etc.