# Welcome and Introduction

**NDN Tutorial – ACM ICN 2015**
September 30, 2015

Steve DiBenedetto
Colorado State University

Slides available at: http://bit.ly/1Lk1Tlx

# Outline

- Slides available at: http://bit.ly/1Lk1TIx

- Brief Review
  - Architecture Overview
  - NDN Platform

- Tutorial Outline
  - Synchronization
  - Storage Options
  - Security

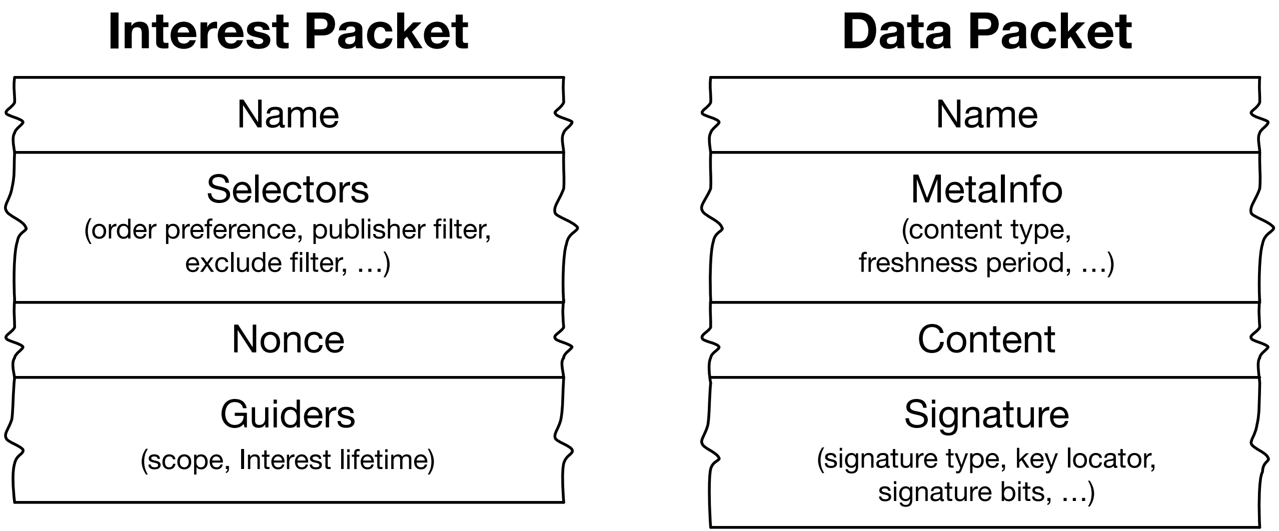
# Core Idea

Modern **communication** consists of
**requests for named data**

Today’s **networks** are based on
**host-to-host connections**

**NDN is a general-purpose network protocol built on requests for named data**

# Two Packet Types



**Interest Packet**

Name

Selectors
(order preference, publisher filter,
exclude filter, …)

Nonce

Guiders
(scope, Interest lifetime)

**Data Packet**

Name

MetaInfo
(content type,
freshness period, …)

Content

Signature
(signature type, key locator,
signature bits, …)

No addresses

**Publishers bind names to data; receivers verify**

# NDN Interest Forwarding

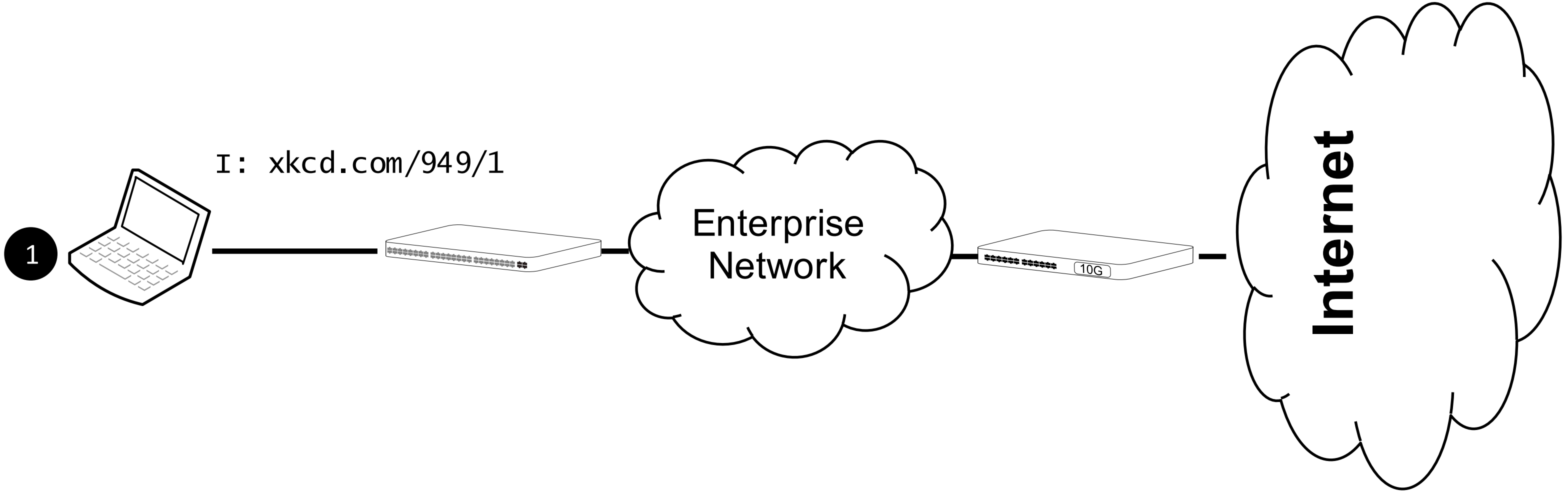
1. Do I have this data?

2. Is a request already pending?

3. Which next hop might lead to the source?

# NDN Forwarding Illustrated

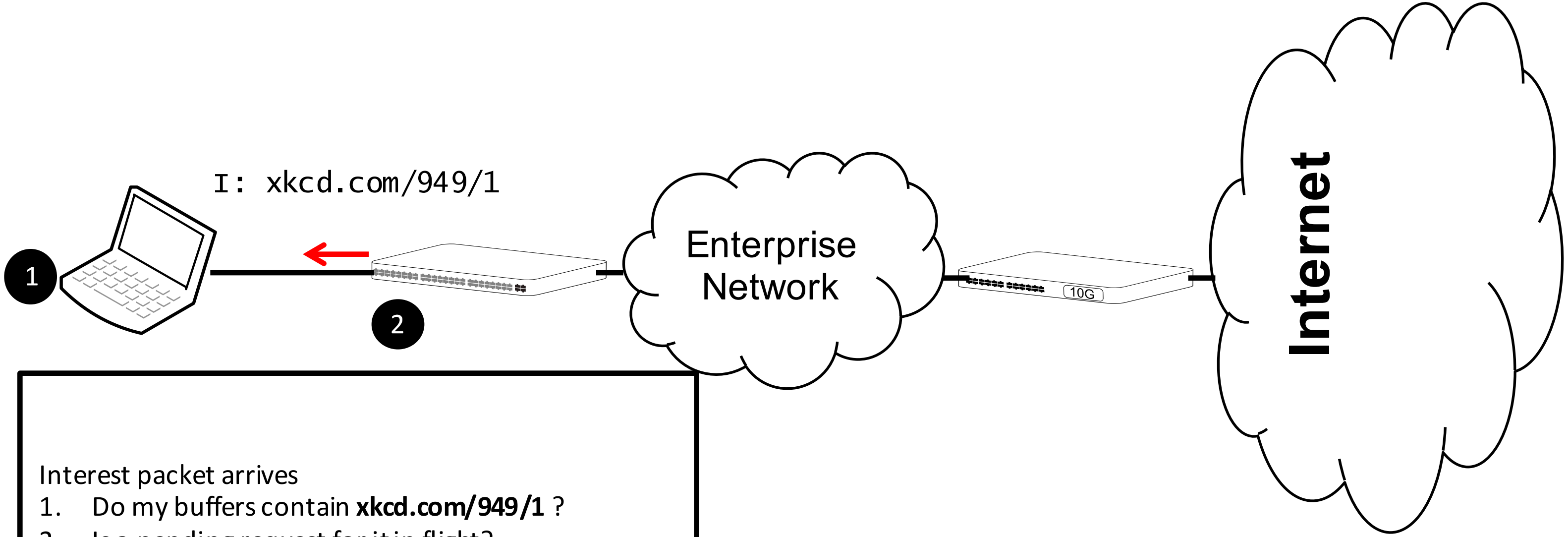
1 Emit Interest: xkcd.com/949/1



# NDN Forwarding Illustrated

1 Emit Interest: xkcd.com/949/1

2 Interest arrives at switch

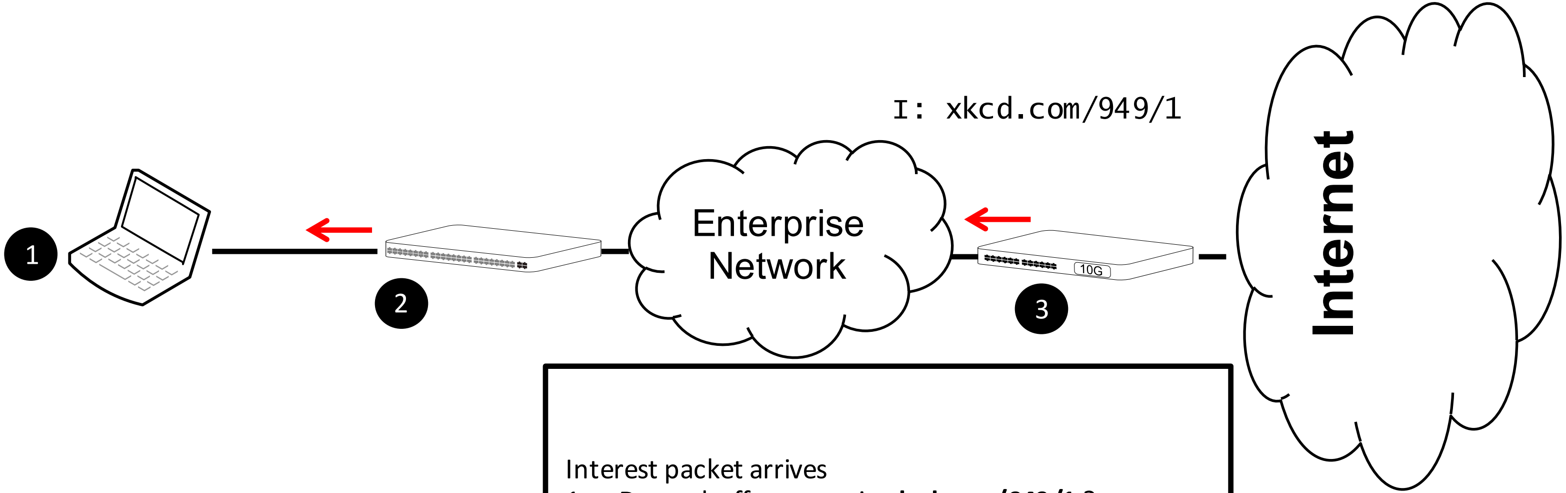


Interest packet arrives

1. Do my buffers contain **xkcd.com/949/1** ?
2. Is a pending request for it in flight?
3. Where should I forward the interest? Add arriving interface to the pending interest list.

# NDN Forwarding Illustrated

1. Emit Interest: xkcd.com/949/1
2. Interest arrives at switch
3. Interest arrives at gateway



Interest packet arrives

1. Do my buffers contain **xkcd.com/949/1** ?
2. Is a pending request for it in flight?
3. Where should I forward the interest? Add arriving interface to the pending interest list.

# 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**.”

- Zhang et al., “Named Data Networking,” CCR July 2014.

# NDN Platform

Core: NFD, the NDN Forwarding Daemon

Libraries: full featured implementations in a variety of languages

Applications: rich and growing software ecosystem

**NLSR**
**repo-ng**
**ndn-tlv-ping**
**ndn-traffic-generator**
**ndndump**
**Federated Wiki**
**ndn-bms**

**ndn-lighting**
**ndn-protocol**
**ndnfs**
**ChronoShare**
**NDNoT**
**ndnrjs**
**ndnrtc**

**Chronochat-js**
**Matryoshka**
**ndnstatus**
**NDNVideo**
**NDNFit**
**OpenPTrack-NDN**
**ndn-dissect**

**See:**
**https://github.com/named-data**

# NFD

NDN Forwarder, implementing the NDN network protocol and the latest packet format

Goals:

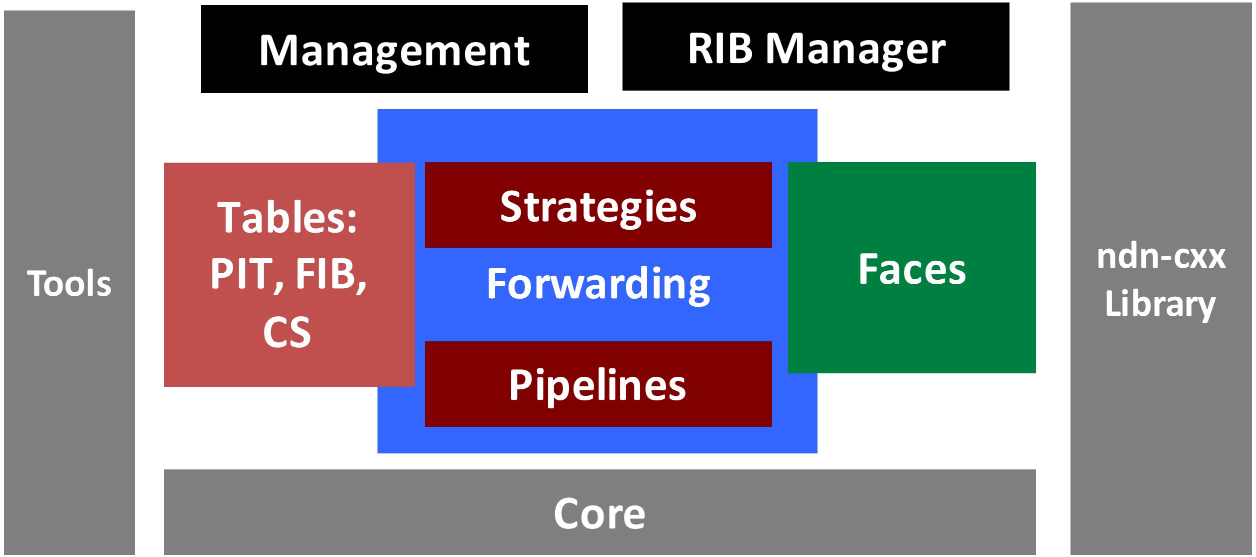
- Facilitate research and experimentation
- Provide free, open-source NDN implementation for the community

New features in the past year:

- NDNLPv2
- NACKs
- Links
- Permanent UDP Faces
- CS replacement policy interface
- Moved to C++11, lots of fixes and small improvements

https://github.com/named-data/nfd

# NFD's Major Pieces



Tools

Management

RIB Manager

Tables: PIT, FIB, CS

Strategies

Forwarding

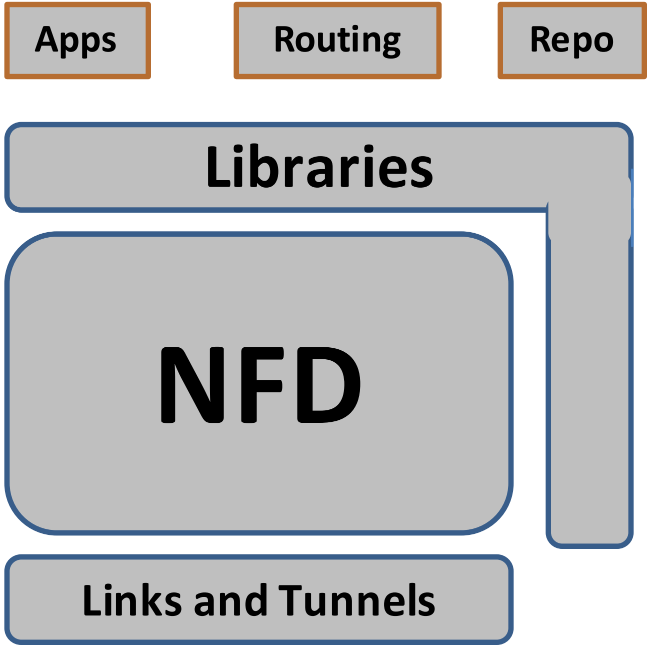
Pipelines

Faces

ndn-cxx Library

Core

# NDN Components



# Evolution of the Libraries

All libraries reflect fundamental architectural abstractions directly in objects

Name, Interest, Data, Face, KeyChain, Validator, …

Libraries:

C++, C, Python (2 & 3), JavaScript (browser & Node.js), and Java

https://github.com/named-data/ndn-cxx
https://github.com/named-data/ndn-cpp
https://github.com/named-data/pyndn2
https://github.com/named-data/ndn-js
https://github.com/named-data/jndn

# Outline

- Slides available at: http://bit.ly/1Lk1TIx

- Brief Review
  - Architecture Overview
  - NDN Platform

- Tutorial Outline
  - Synchronization
  - Storage Options
  - Security

# Today’s Agenda

Goal: Help guide research and application development beyond basics

Use a new chat application as a motivating example for “intermediate” NDN concepts:

Synchronization – Abstractions beyond Interest/Data exchanges

Storage Options – Alternatives to relying on in-network Content Stores

Trust & Verification – Specifying what content to trust

Also explore access control to fitness data (NDNFit project)

# Sync: Beyond Interests and Data

Sync : Interest/Data :: TCP : IP

- Provide higher-level abstraction for common functionality
- In IP: TCP provides features like reliability, ordering, etc.
- In NDN: Above features provided elsewhere, so we focus on improving knowledge about datasets/collections

Sync is an active NDN research topic

- ChronoSync (improved version coming)
- iSync

Today

- Discuss the sync's role and the current landscape (Hila Ben Abraham)
- Illustrate sync in action by starting to build our app (Jeff Thompson)

# Storage Options

Application In-Memory Storage (IMS): app regulated Data and Interest storage

- App can store Data packets in IMS
- IMS register prefix on app’s behalf, notify on cache miss
- Can also act as an in-memory PIT

Repository (repo): persistent, in-network, storage

- Stores Data packets on disk
- Interest/Data API for insertion and polling namespaces for new content

Content Store (CS): ephemeral, in-network, storage

- Useful for retransmission/recovery
- Essentially every node has (configurable) storage
- Entries stored in NFD’s memory

Today

- Discuss available & upcoming options over lunch (Jeff Burke)

# Trust

Trust is a cornerstone of the NDN architecture

Requiring every Data packet to be signed puts trust at the front of the developer and user's mind

- How do I identify the signer?
- How do I determine if I should trust someone?

Trust often follows structure (e.g. a particular organization)

- Structure can be reflected in content and key naming scheme

Today

- Demonstrate how construct and use trust schemas (Alex Afanasyev)
- Review Library trust API and add signing/verification to app (Jeff Thompson)

# Access Control

NDN encourages securing content over channels

Want to enforce access control on cached content anywhere in the network

Channel's security is gone once content arrives

Stock answer: just encrypt the content!

How do we make the content reusable? (Avoid per user encryption)

What is the granularity of control and how does it relate to naming?

How do we disseminate keys?

Today

Name-based access control in the context of NDNFit (Yingdi Yu)