Design Principles for Named Data Networking

Work-in-progress by the NDN Project Team

ICNRG Interim Meeting, Buenos Aires
April 3, 2016

Purpose

- Clarify overarching (and evolving) perspective that is driving NDN architectural design, application experimentation, and other research. Motivated by discussion on different decisions on the protocol format.

- Respond to community requests for more background on design choices, and opportunities for constructive feedback.

- Sketch distinctions with other design choices in ICN research.

http://named-data.net/project/ndn-design-principles/

Approach

- Work in progress to explain how our protocol decisions are made.
  - Feedback is welcome.
- Flexible definition of principles, for now.
  - Principles, goals, strategies.
- Perhaps more principles to come, of different types.
  - These are not quite application design principles or guidelines; they focus on protocol decisions.
- Note: Largely by others. Some of my comments / highlights in blue.

Summary

- Universality
- Data-Centricity and Data Immutability
- Securing Data Directly
- Hierarchical Naming
- In-Network Name Discovery
- Hop-by-Hop Flow Balance

Universality

NDN should be a common network protocol for all applications and network environments.

- Applications and network environments that NDN should support include but not limited to:
  - today's infrastructure-based communication (Web, Youtube, real-time conferencing, etc.)
  - ad hoc with and without infrastructure communication (IoT applications, wireless mesh networks, vehicle-to-vehicle networking, vehicle-to-vehicle-to-infrastructure, etc.)
  - DTN-style communication, communication over intermittent and disruptive links (first responder environments), application using unidirectional links (e.g., satellite)
  - future application environments that we do not know at this time (as what happened to IP)

Universality

NDN should be a common network protocol for all applications and network environments.

- Therefore, NDN protocol and NDN packet format should support wide range of applications, from constrained (IoT) environments to big data science applications:
  - NDN packet format should be flexible and extensible.
  - NDN protocol and packet format should support protocol evolution without flag days: no fixed parts or fixed-length fields in the header.
  - The core network protocol operations should not depend on clock synchronization.

Universality

NDN should be a common network protocol for all applications and network environments.

- Motived by success of IP.
- Simplest design choice.
- Hard to draw useful boundaries given rapidly evolving landscape.

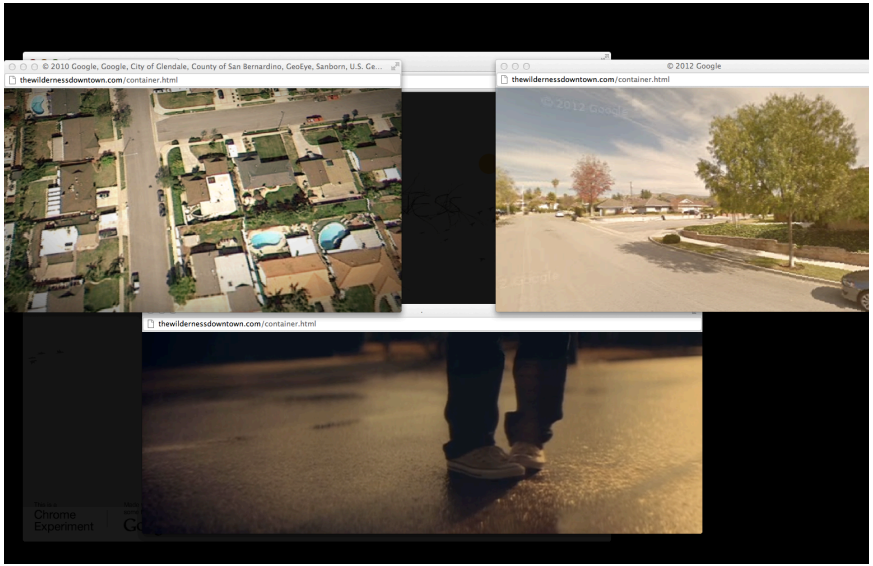
- Consider IoT - related reading:

  W. Shang, A. Bannis, T. Liang, Z. Wang, Y. Yu, A. Afanasyev, J. Thompson, J. Burke, B. Zhang, L. Zhang. "**Named Data Networking of Things**." *1st IEEE Intl. Conf. on Internet-of-Things Design and Implementation (IoTDI)*, April 4-8, 2016, Berlin, Germany.

Universality

Application example

- IoT + composable media
  = mobile augmented reality.



The Wilderness Downtown

Gandhi, Vineet, and Rémi Ronfard. "A Computational Framework for Vertical Video Editing." *Eurographics Workshop on Intelligent Cinematography and Editing*. 2015.

Data-Centricity and Data Immutability

NDN should fetch uniquely named, immutable "data packets" requested using "interest packets".

- NDN protocol and packet format should include only elements directly related to data, i.e., universally required, needed, and meaningful in all communication environments.

- Other elements needed in specific environments (e.g., in today's infrastructure-based Internet) should go to the network adaptation layer(s).

Data-Centricity and Data Immutability

NDN should fetch uniquely named, immutable "data packets" requested using "interest packets".

- Data packet immutability allows disambiguation of coordination in distributed system that may not be always connected.

- Applications can make changes to the communicated content by creating new versions of immutable data packets.

Data-Centricity and Data Immutability

NDN should fetch uniquely named, immutable "data packets" requested using "interest packets".

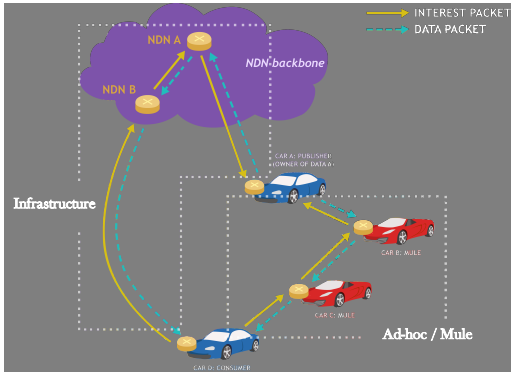
- Suggested reading:

  Helland, Pat. "Immutability changes everything." *Queue* 13.9 (2015): 40.

Data-Centricity and Data Immutability

Application example

- Networks of diverse and intermittent links.
- Community-based mesh networks
- Vehicular networking.



Securing Data Directly

Security should be the property of data packets, staying the same whether the packets are in motion or at rest.

- Directly secured and uniquely named data removes the requirement for direct channels between communicating ends. [And dependence on channel security.]

- It enables asynchronous production and consumption of named and secured data, e.g., using in-network caches and managed repos.

Securing Data Directly

Security should be the property of data packets, staying the same whether the packets are in motion or at rest.

- Consumers should be able to validate individual data packets. Ideally, each packet should be verifiable on its own.

- As an engineering optimization, packets can be made verifiable in the context of others, provided that the context can be inferred from the data packet itself (its name or information in signature field).

Securing Data Directly

Security should be the property of data packets, staying the same whether the packets are in motion or at rest.

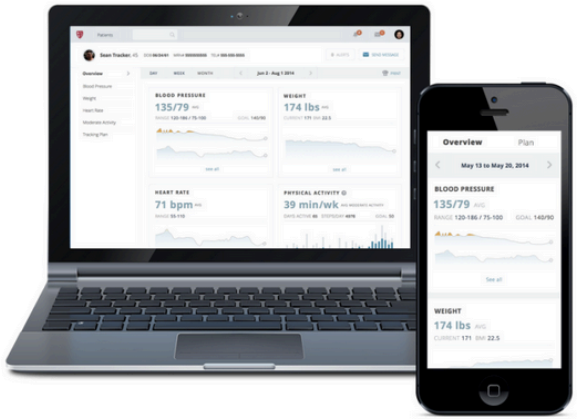
- Listserv: Why not explicitly discuss privacy (or confidentiality) as a principle here?

- Defer discussion to privacy part of the afternoon.

Securing data directly

Application example

- Life-long health data owned by the individual
  – Open mHealth.

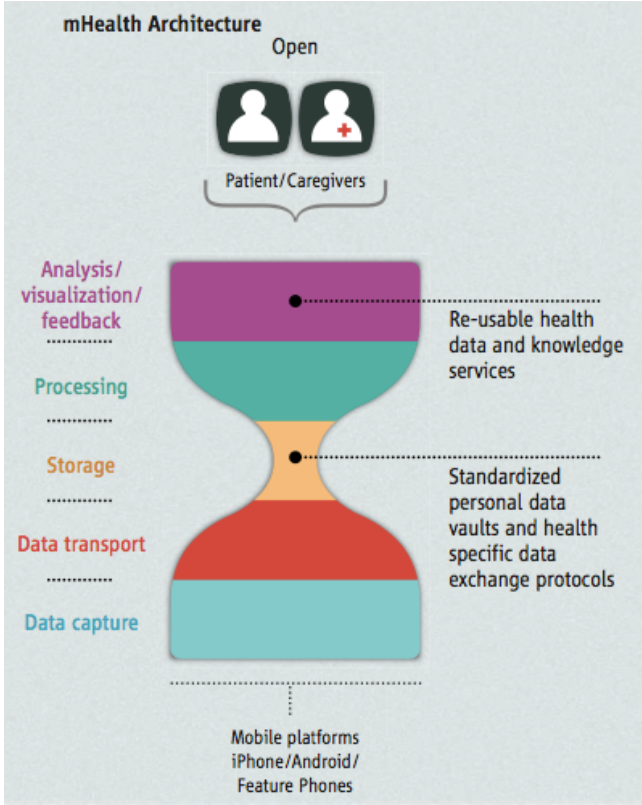


Linq

TRACK TOGETHER

Patients use the apps and devices they choose to track health measures with their doctor.

Data views designed for clinical insights give doctors the information they care most about.



Hierarchical Naming

Packets should carry hierarchical names to enable demultiplexing and provide structured context.

- Reasonable model for many applications.
- Name hierarchy provides context to implement and enforce various security models, i.e., giving structured restrictions on which keys can sign which data.
- Hierarchical names allow “flat” naming models, if needed/desired by applications.

Hierarchical Naming

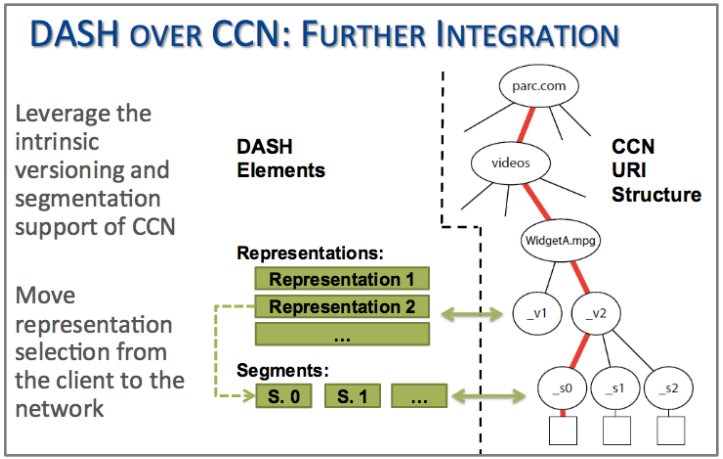
“Empirically, a large proportion of the complex systems we observe in nature exhibit hierarchic structure. On theoretical grounds we could expect complex systems to be hierarchies in a world in which complexity had to evolve from simplicity. In their dynamics, hierarchies have a property, near-decomposability, that greatly simplifies their behavior.”

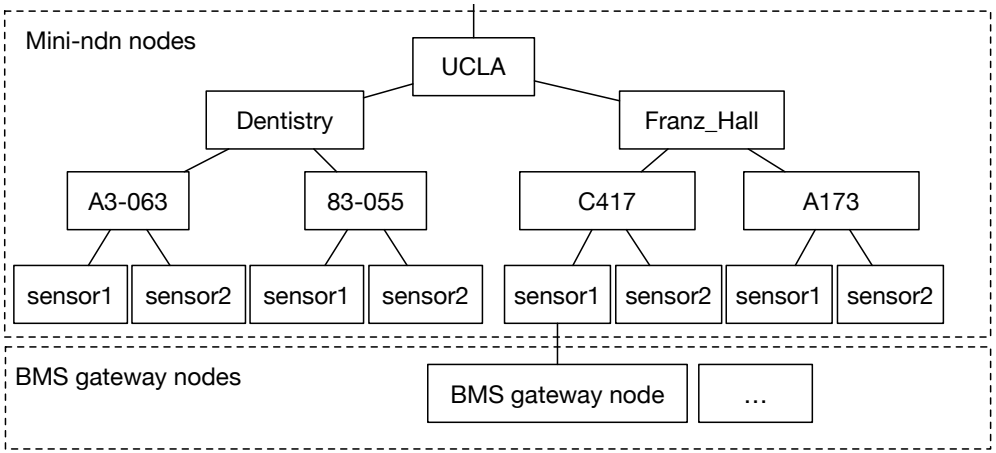
“The Architecture of Complexity” Herbert A. Simon, Proceedings of the American Philosophical Society, Vol. 106, No. 6., Dec. 12, 1962, pp. 467-482.

Hierarchical Naming

Application example

- Media, cyberphysical systems, etc.





DASH over CCN, Lederer, ICNRG / IETF 87

In-Network Name Discovery

Interests should be able use incomplete names to retrieve data packets.

- A consumer may not know the complete network-level name for data, as some parts of the name cannot be guessed, computed, or inferred beforehand.

- Particularly true for dynamic data.

- Once initial data is received, naming conventions can help determine complete names of other related data
  - majority of interests will carry complete names
  - in-network name discovery expected to be used to bootstrap communication

In-Network Name Discovery

Interests should be able use incomplete names to retrieve data packets.

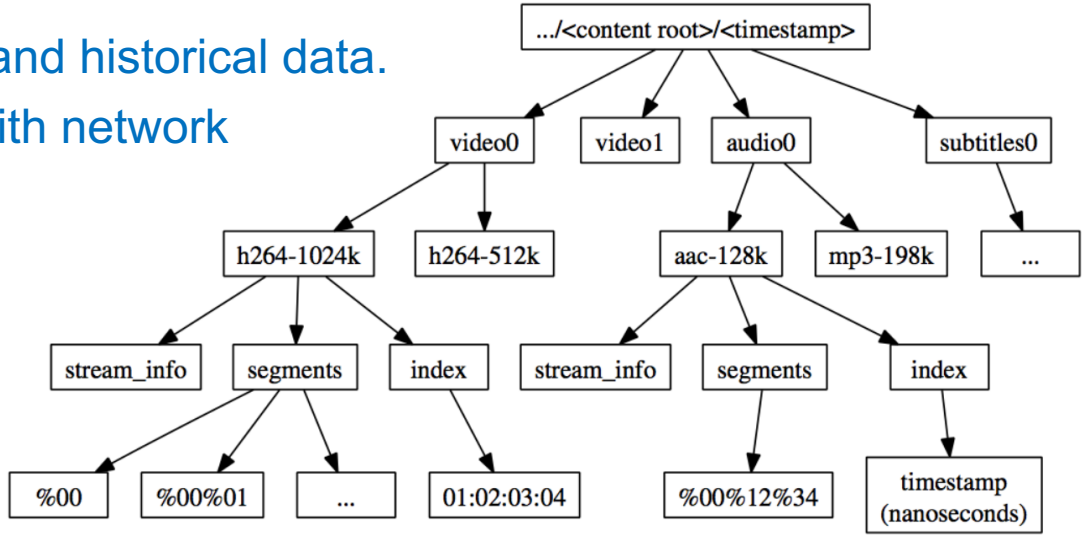
- Support dynamic production of immutable data.

- Multi-producer, multi-consumer scenario – new application scenarios with less emphasis on service infrastructure to achieve multiple party communication.

In-network Name Discovery

Application example

- Streaming sensor data, smart homes, video, versioned content.
- Same producers for real-time and historical data.
- Balance application benefits with network performance.



Hop-by-Hop Flow Balance

Over each link, one interest packet should bring back no more than one data packet.

- Hop-by-hop flow balancing enables each node to control load over its links.
- By deciding to sending interest over a link, router commits bandwidth for the returned data.
- By limiting the number of interests sent, each router and client node in the network control how much data it will receive.

Hop-by-Hop Flow Balance

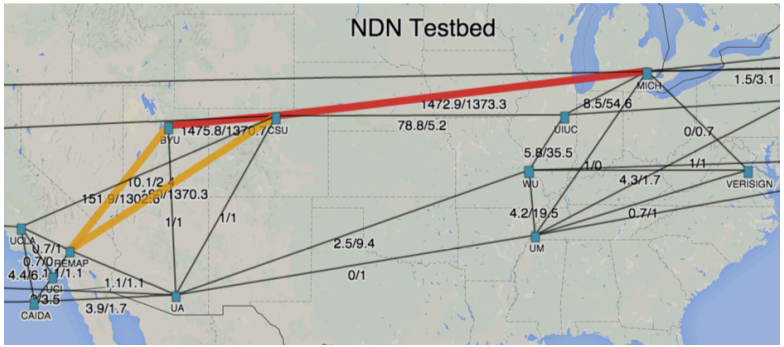
Over each link, one interest packet should bring back no more than one data packet.

- In a multi-input, multi-output mesh network, end-to-end congestion control cannot simultaneously achieve both
  1. no congestion over any link, and
  2. max. possible utilization of all links.

Hop-by-hop flow balance

Application example

- Real-time video –
  current work on NDN-RTC.



Next

- Continued iteration on principles, justifications, examples based on external feedback and ongoing research.

- Please continue to send comments to ndn-interest list.

Thank you!

NDN Project Team

ICNRG Interim Meeting, Buenos Aires
April 3, 2016