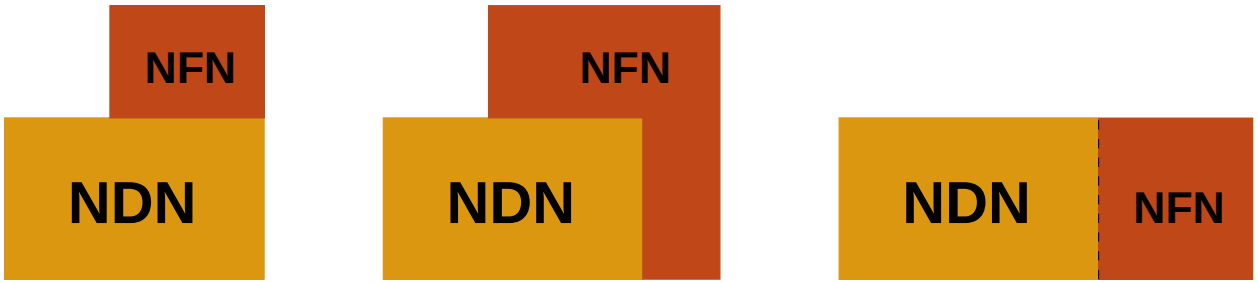
Named Functions

Vertical or horizontal named-data extension?

NDNcomm, Sep 4, 2014



UNI BASEL

Christian Tschudin, University of Basel

Abstract

Named Function Networking (NFN) extends the named-data approach: Interests typically carry more than one name, for example a function name and the names of the parameter data.

`resolve(`”/**some/data**”`)` $\Rightarrow$ `resolve(`”/**some/fct (/some/data)**”`)`

The question is whether NFN needs support from the NDN forwarder and the PIT, or whether NFN is just another application sitting on top of the named-data layer.

In this talk we sketch these two architectural paths and provide first insights based on simulation results.

From Named-Data to Named-Functions

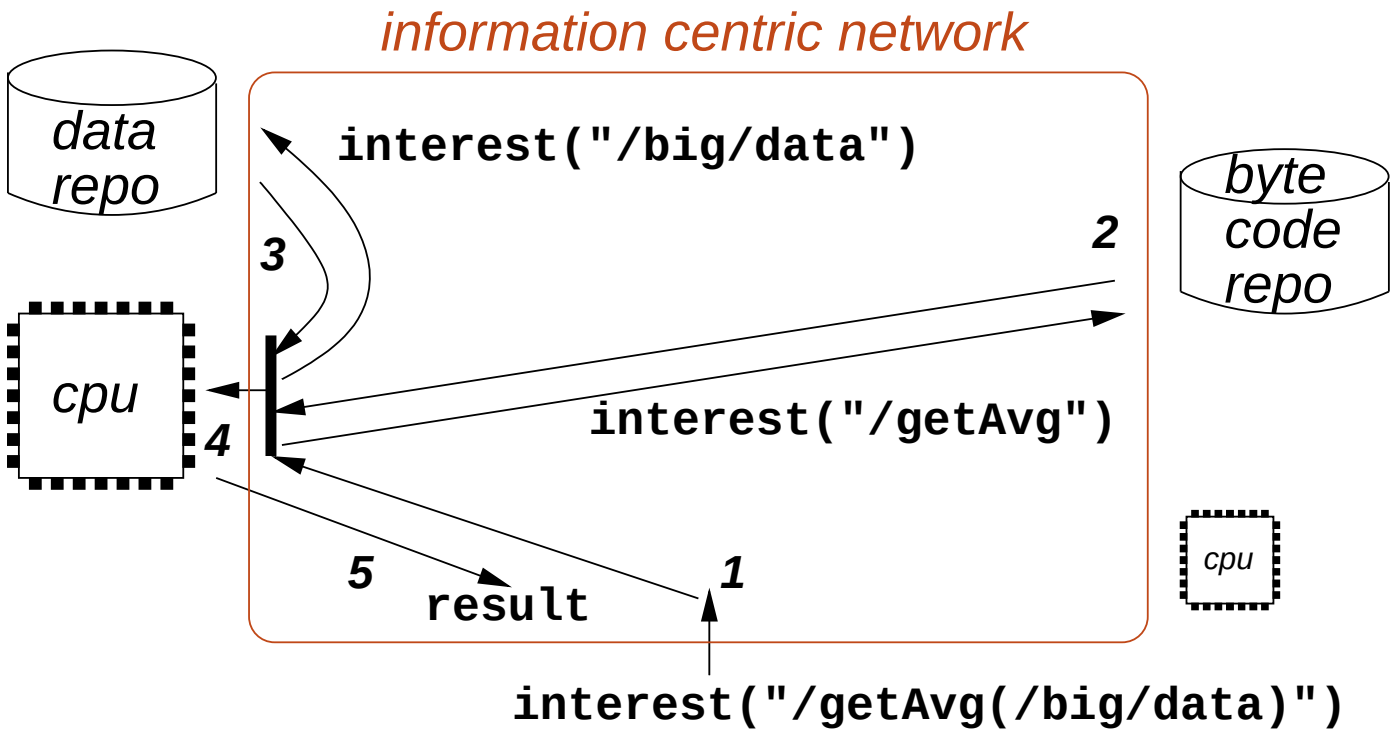
- Raw data are more prevalent, but clients want cooked data (highly variable on-demand profile)
- Examples:

```
/downScale( /this/video )
/getAverage( /sunShineHours/in/CA, 2014 )
/geoFence( /my/heart/rate, /my/gps/location, 10ft )
```

- The goal of Named Function Networking (NFN):
  – clients *name the desired computation result*, server-agnostically
  – network is in charge of finding execution places (servers)
  – network optimizes execution graph, caches the results

Three tasks: Locate data, fct and exec place; Run; Collect

REMOTE EVAL beats “download and process locally”: find a server close to the DB!



Network does **not** execute: NFN only orchestrates the computation by *juggling around names and triggering exec, later returning the collected result.*

Impact of NFN orchestration on the NDN Forwarder, PIT

- NFN intercepts requests, plans execution ($\sim$ database query plan)
- NFN needs richer routing information:

  pick compute servers based on performance, load, data-proximity, ISA support, trust ... $\rightarrow$ does not change forwarding per se, but
- NFN interfers with PIT/forwarding/caching mechanics:
  – forward a multi-name plan, not an interest, to select servers
  – trigger execution, wait for the server's result
  – collect result, return as "content-matching-the-plan"
  – cache results

Can we avoid modifying the forwarder, PIT data structure?

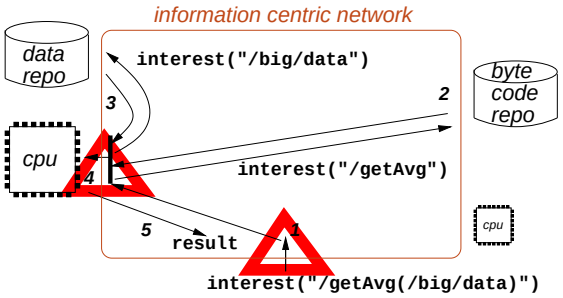
NFN-over-NDN

(or: a desirable feature becoming a disadvantage)

- NFN as an “app”: NFN logic placed at client and compute nodes
  - intercept queries, maintain own PIT timeouts,
  - map multiple names to interest’s name field, same for plans
  - maintain a separate routing system (cpu locations, ISA).

  In principle doable (might need modifs of NDN PIT timeouts).

- **But**: How to get data base locations? NDN hides it!
  - **conceptual mismatch when doing NFN-over-NDN** ↯
  - additional subtle problems: caching of results, effects of timeouts



Our current approach: embrace mix of plain NDN nodes and full NFN nodes (horizontal extension)

Other lessons (from simulations, for NDN-plus-NFN)

- Appropiate timeout values are crucial:

  A fixed PIT timeout leads to planning+exec to be placed close to client, instead of deep in the net.

- Computations can take a long time, fixed timeout not appropriate:

  NACKs should be generalized to “ETA – expected time of arrival” (with infinity meaning NACK), also needed for thunks.

- NACKs are mandatory for good performance of plan execution:

  Otherwise, NFN lacks “strategy triggers”, has to guess unavailability of resources.

Take home messages

1. Named functions as a “natural” extension of named-data
2. NFN-over-NDN problematic: must be able to learn location of data
   $\rightarrow$ mix plain-NDN, full-NFN nodes: resolution strategy matters
   $\rightarrow$ other mixes forseeable: full-NFN, plain-NDN, new-CCN-style
3. Need a NACK, ETA discussion (useful also beyond NFN)

See Jeff Burke’s talk tomorrow: “**Open mHealth**”, NFN for data filters