NFN – Recent Work on Expression Forwarding

Christopher Scherb · **Claudio Marxer** · **Christian Tschudin**

University of Basel, Switzerland

ICNRG Interim Meeting @ IETF 104 Prague, 24 Mar 2019

NFN Mindset

ICN generalization: Deliver cooked results instead of raw data

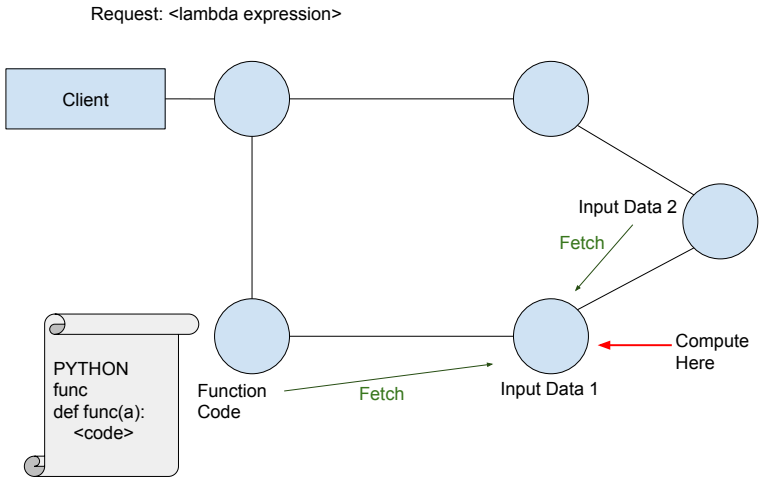
Idea:

- User defines computation / workflow
- Network finds execution location
- Similar to Serverless Computing, but offers workflow orchestration
- Implemented using Interest / Named Data Object Transport Layer

PiCN - pure Python Implementation of NFN

- code available at: https://github.com/cn-uofbasel/PiCN
- includes an ICN Forwarder
- executes Python code under network control
  - decomposes complex expressions “in the network”
  - optimizes by migrating either code and/or data (more about this in this talk)
- since recently also for native Intel x86 code
- previous Scala (JVM) support could be added again

Example


Request: <lambda expression>
Client
Input Data 2
Fetch
Compute
Here
PYTHON
func
def func(a):
<code>
Function
Code
Fetch
Input Data 1

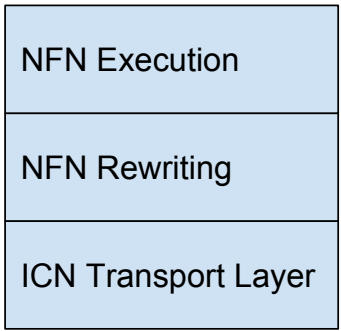
PiCN (cont.)

- Software contains:
  - NDN and NFN Forwarder
  - client and command line tools
  - repo
- modular, easy to extend
- simple simulation and debugging system

Qutline

- Basic NFN Principles (what we want to do)
- Layering Overview
- Core of NFN: **expression rewriting**
- Example: expression rewriting based on **FIB information**
- Example: expression rewriting based on **mobility patterns**
- Example: expression rewriting based on **pricing**
- Orchestration of **"Plans"** and creating **"Templates"**
- More research topics: result authenticity, compute privacy

Architecture / Core Components



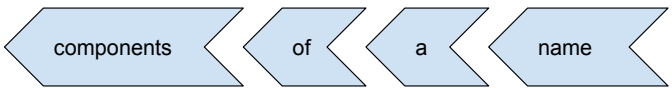
Execute Function Code

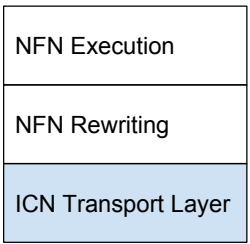
Decide where to compute

Forward Messages

CCN/NDN Layer

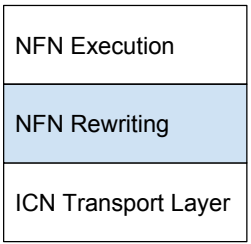
- Basic CCNx/NDN Forwarding
- **Hierarchical** Names
- Forwarding based on **Longest Prefix Matching**
- **Name** in Interest Message contains reference to single **Named Data Object**



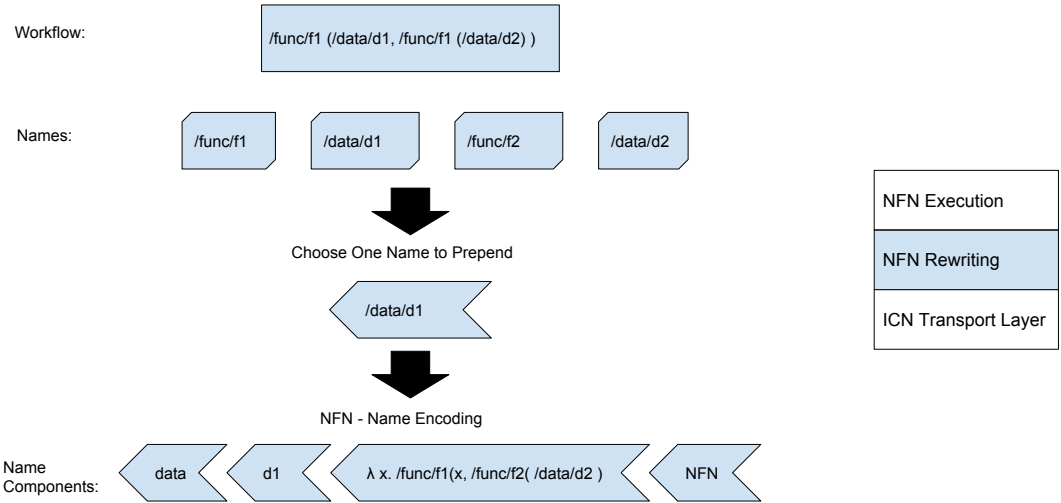


NFN Rewriting 1

- Workflow - consisting of **Function Calls** on **Input Data**
- Encoding the Workflow in CCN/NDN **Names**
- Using λ-**calculus** to describe workflow
- Name in Interest Message contains references to multiple **Named Data Object**
- Magic: **Longest Prefix Matching** and **Name Rewriting**

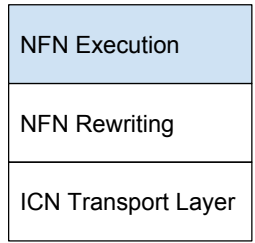


NFN Rewriting 2: Name Encoding Example

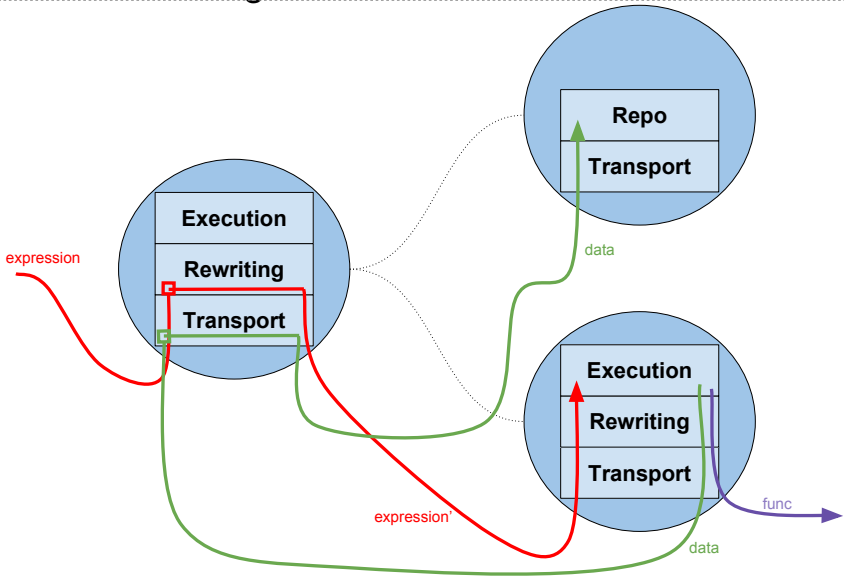


NFN Execution

- Store **Function Code** in **Named Data Objects**
- Any **NFN node** can execute **Function Code** by fetching it over ICN
- Reqeuesting missing **Named Data Objects** and **Function Code**
- Requires Safe Execution Environment (Sandboxing)



NFN Interest Handling



Rewriting Descisions

Interesting Part:

> Which Name to **prepend** in front of the computation (**Rewriting**)

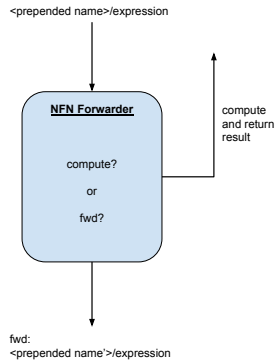
- Because: Influnces where to **forward** a computation
- Defined by a Rewriting Strategy

> Should a node forward a computation or execute it locally?

- Determines where to compute a result.
- A node can split a computation into subcomputations $\Rightarrow$ parallel/distributed execution

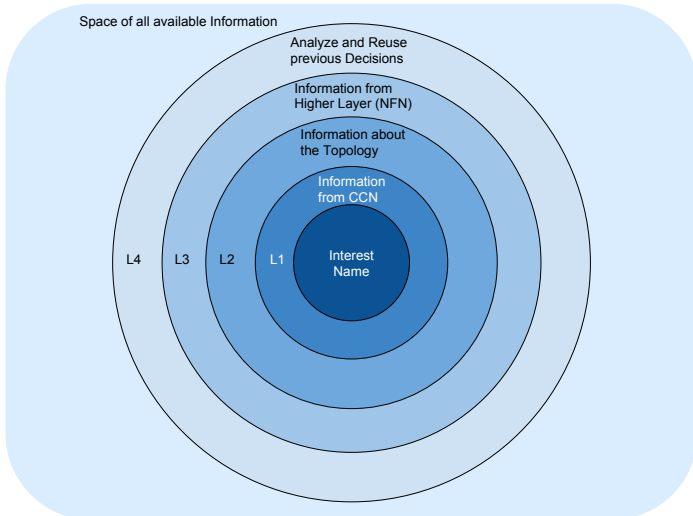
Rewriting Strategy

- Simple Rewriting Strategy - inspired by Hadoop
- Goal: Reduce **load** on links
- Forward a **computation request** to the **input data** (prepend input data)
- Start computation if the prepend data are available on the node.
- Transport **function code** to **input data**



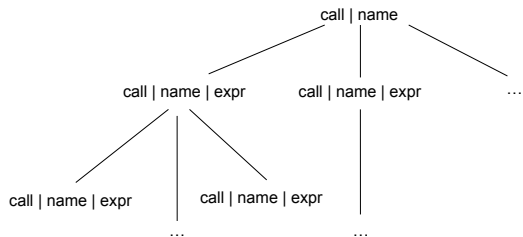
Advanced Expression Rewriting for NFN

- Add additionlal information to improve the **rewriting descision**
- Scenario **dependent** and **independent**

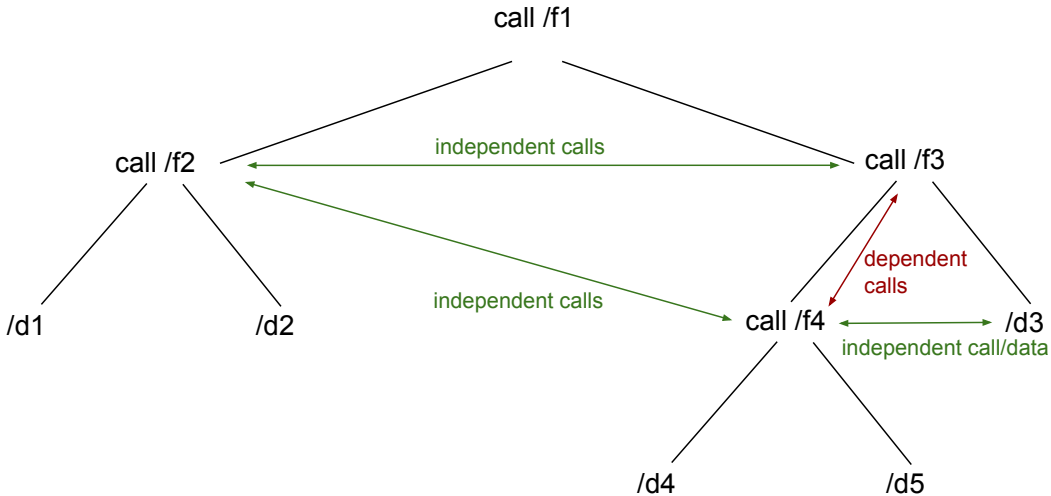


NFN-Expression Rewriting based on **FIB** Information - 1

- Use the **FIB** to decide which name should be prepended
- Create **AST** and search for independent subcomputations
- Split the computation if subcomputations are forwarded to different nodes
- Good for Map Reduce and Parallel Execution

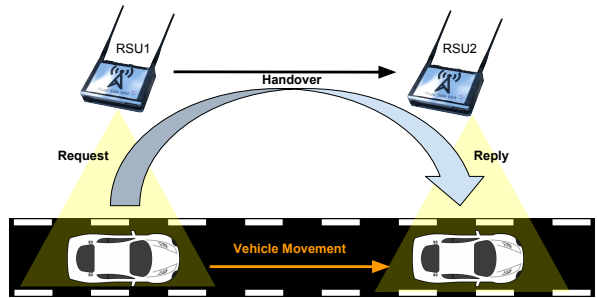


NFN-Expression Rewriting based on **FIB** Information - 2



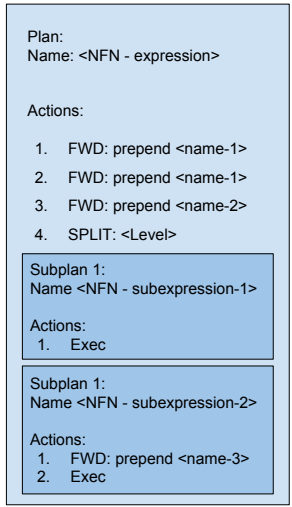
NFN-Expression Rewriting based on Mobility Patterns

- Use information about **mobility patterns**
- e.g. Node is **Edge Computing Node**
- e.g. **Neighbor Node** has no computational capabilities
- Execute even if **prepended data** are **not** on the node

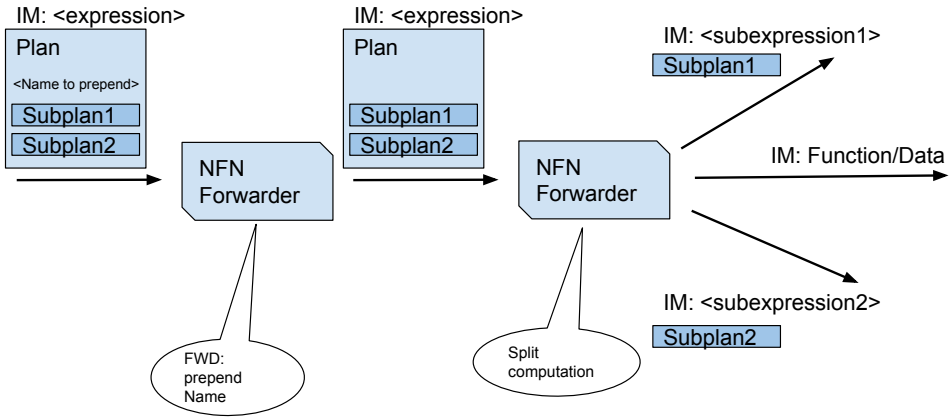


NFN-Expression Rewriting based on the Price - 1

- Request **Meta Information** about **file size**, **bandwidth** or **load**
- Compute a **Plan** for the **cheapest** execution possibility
- Storing **Plans** in **Named Data Objects**
- **Caching** and **Reusing** of Plans (for subcomputations)

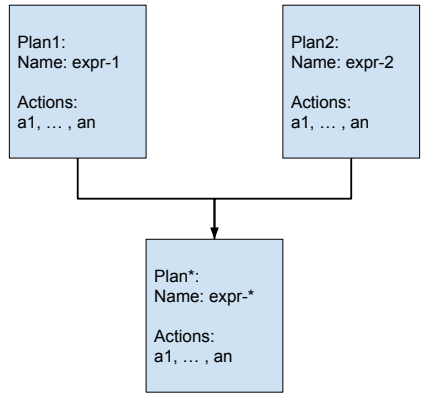


NFN-Expression Rewriting based on the Price - 2



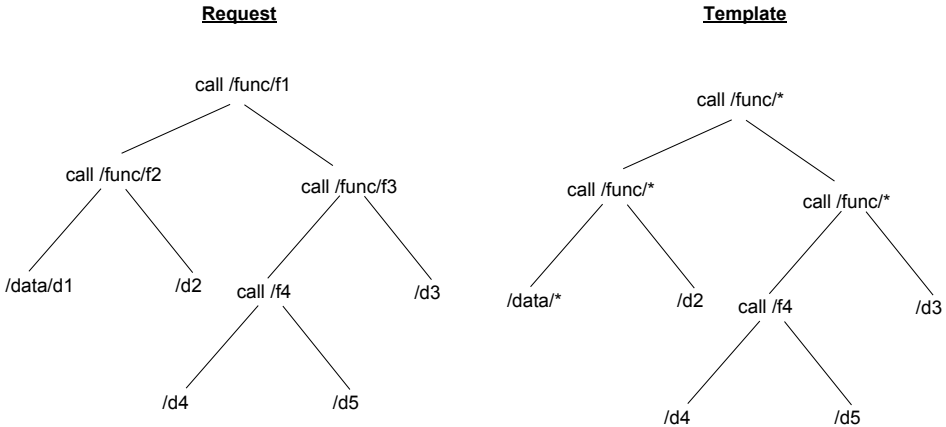
NFN-Expression Rewriting based on Templates - 1

- Create **Templates** based on the **Planning Process**
- Idea: Instead of creating a new **Rewriting Strategy**, let the network learn from previous situations (or simulation)
- Compare **AST** structure and names within the **AST**
- Introduce **Wildcard** Names into the **Plans**
- Based on Observation and Statistics:
  - $< prefix > /name\text{-}1 \ldots < prefix > /name\text{-}n \rightarrow$ Action a
  - if $n > x \rightarrow$ reduce $< prefix > /*$
- In future with Machine Learning?



NFN-Expression Rewriting based on Templates - 2

- A Templete is a Tuple: $< AST*, Action >$
- A request is matched against the AST containing the Wildcards



Result Authenticity

- Data are secured by **signatures**, what about **results**
- Question: How can you know the result is correct?
- Idea: Signed by execution node.
- Add Signatures of the Input Data and the Function Code
- Item: No Proof, but enables users to find out which nodes are fooling

Data Security

- Question: How to deal with sensible input data?
- Intel SGX may be a way
- Hormomorphic Computing (increases runtime)
- Only real solution: Do it local

Software

- PiCN
- https://github.com/cn-uofbasel/PiCN
- ICN and NFN Forwarder, Client Tools, Repository
- Modular, easy to extend
- NFN Rewriting Strategies as Plugins
- Simple Simulation System

References

- https://github.com/cn-uofbasel/PiCN
- An information centric network for computing the distribution of computations (M. Sifalakis, B. Kohler, C. Scherb, C. Tschudin)
- Access-controlled in-network processing of named data (C. Marxer, C. Scherb, C. Tschudin)
- Resolution Strategies for Networking the IoT at the Edge via Named Functions (C. Scherb, D. Grewe, M. Wagner, C. Tschudin)
- Execution State Management in Named Function Networking (C. Scherb, B. Faludi, C. Tschudin)
- A Packet Rewriting Core for Information Centric Networking (C. Scherb, M. Sifalakis, C. Tschudin)
- Smart Execution Strategy Selection for Multi Tier Execution in Named Function Networking (C. Scherb, C. Tschudin)

Question