# Improved Content Addressability Through Relational Data Modeling and In-Network Processing Elements

Claudio Marxer
<claudio.marxer@unibas.ch>

Christian Tschudin
<christian.tschudin@unibas.ch>

Computer Networks Group · University of Basel · Switzerland

ACM ICN '17, Berlin · September 27, 2017

# Shortcomings of Hierarchical Content Organization

– **Sometimes, a document can be part of multiple collections**

```
/personal/joe/2017/NYmarathon/time-location.gpx
/personal/joe/marathons/NYmarathon/2017/time-location.gpx
/organizer/NYmarathon/2017/ranking/1/time-location.gpx
/organizer/NYmarathon/2017/joe/time-location.gpx
```

→ Different access patterns for a document require different ICN names.

# Shortcomings of Hierarchical Content Organization

- **Sometimes, a document can be part of multiple collections**

  `/personal/joe/2017/NYmarathon/time-location.gpx`
  `/personal/joe/marathons/NYmarathon/2017/time-location.gpx`
  `/organizer/NYmarathon/2017/ranking/1/time-location.gpx`
  `/organizer/NYmarathon/2017/joe/time-location.gpx`

  → Different access patterns for a document require different ICN names.

- **Unsupported access-pattern: Only tracks longer than 20 miles are relevant.**

  → Data Packet has implicit characteristics which can't be part of the ICN name

# Shortcomings of Hierarchical Content Organization

- **Sometimes, a document can be part of multiple collections**

  /personal/joe/2017/NYmarathon/time-location.gpx
  /personal/joe/marathons/NYmarathon/2017/time-location.gpx
  /organizer/NYmarathon/2017/ranking/1/time-location.gpx
  /organizer/NYmarathon/2017/joe/time-location.gpx

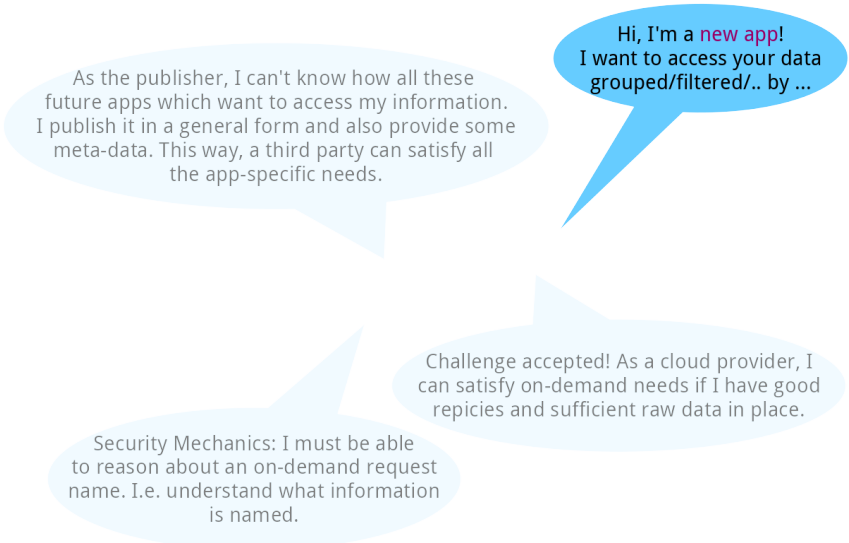
  → Different access patterns for a document require different ICN names.

- **Unsupported access-pattern: Only tracks longer than 20 miles are relevant.**

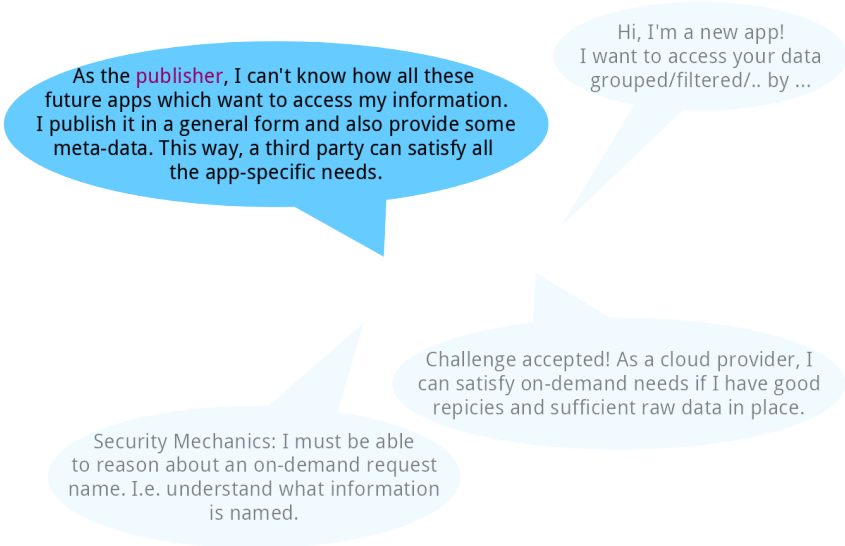
  → Data Packet has implicit characteristics which can't be part of the ICN name

- **Limited descriptiveness of an ICN name affects granularity of name-based security policies**
  - So, why not extensively descriptive names? Because: Overlong Names! Name Privacy!
  - Anyway, not all implicit information can be added to name (e.g. track length)

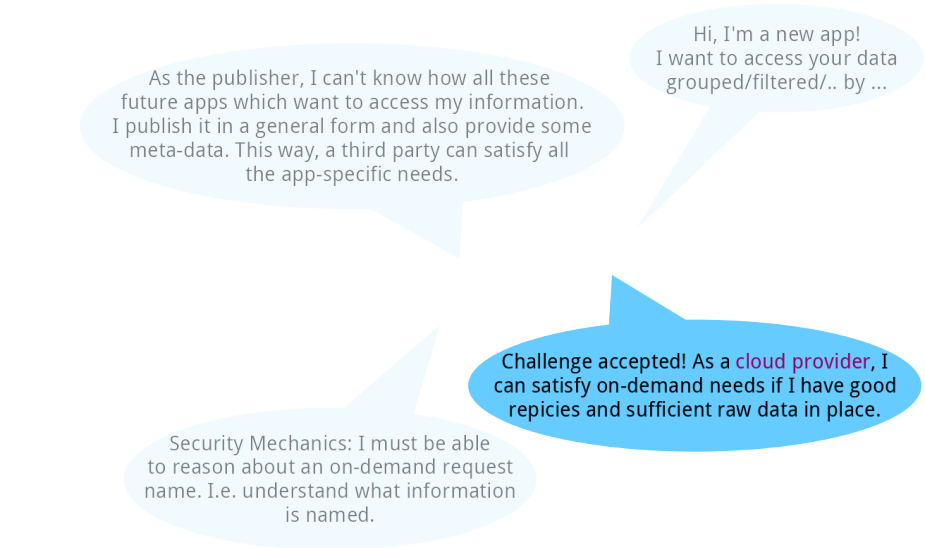
# Wish List: An Access and Naming Scheme Should...



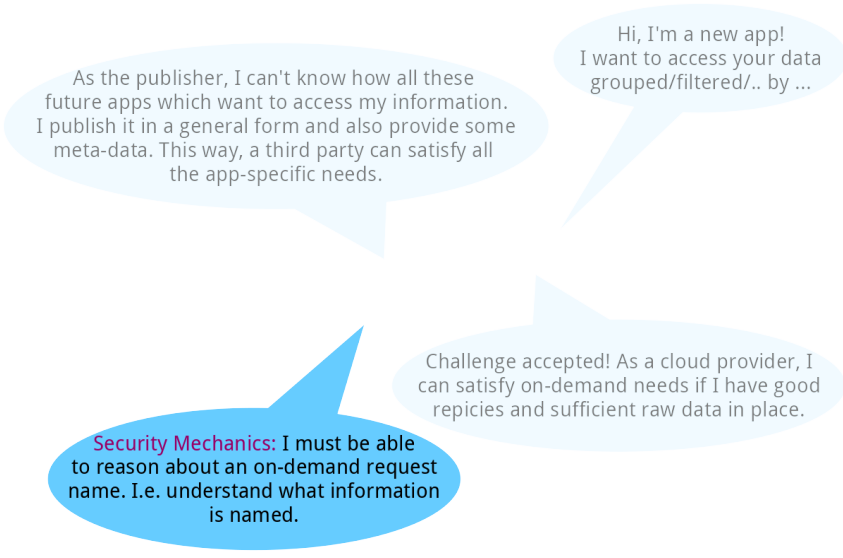
# Wish List: An Access and Naming Scheme Should...



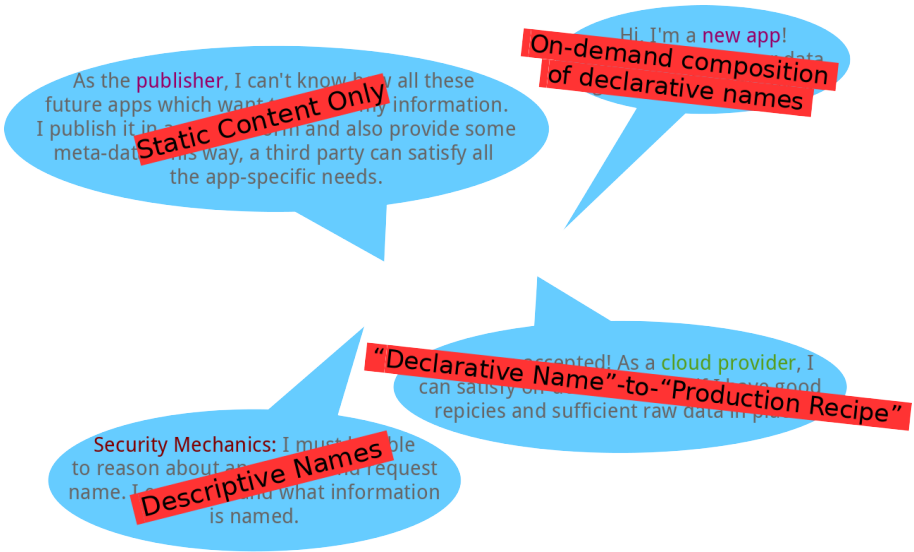
# Wish List: An Access and Naming Scheme Should...



# Wish List: An Access and Naming Scheme Should...



# Wish List: An Access and Naming Scheme Should...



# Outline

- Problem Statement: Shortcomings of Hierarchical Content Organization
- Wish List for a Access/Naming Scheme
- Background: In-Network Processing & Named-Function Networking
- ICN & Relational Data Modeling
  - Publisher Side: Data Organization
  - Consumer Side: Data Querying
- Comprehensive Example
- Conclusion

# Named-Function Networking: Consumer's View (Example 1)

**Classic ICN:** Distribution of named content (published)

```
lookup:  /joe/NYmarathon/track.gpx
```

**Named Function Networking (NFN):** Generation of named content (on-demand)

```
lookup:  /get/duration( /joe/NYmarathon/track.gpx )
```

# Named-Function Networking: Consumer's View (Example 1)

**Classic ICN:** Distribution of named content (published)

```
lookup:  /joe/NYmarathon/track.gpx
```

**Named Function Networking (NFN):** Generation of named content (on-demand)

```
lookup:  /get/duration( /joe/NYmarathon/track.gpx )
         ^^^^^^^^^^^^^  ^^^^^^^^^^^^^^^^^^^^^^^^
         named function      named content
```

# Named-Function Networking: Consumer's View (Example 1)

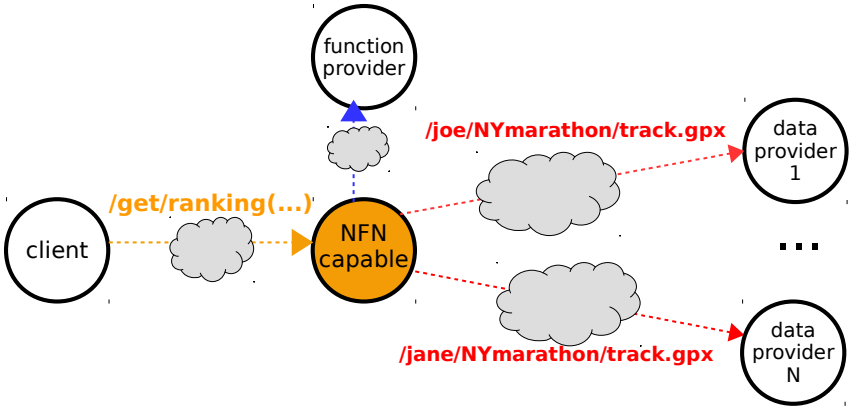
**Classic ICN:** Distribution of named content (published)

```
lookup:  /joe/NYmarathon/track.gpx
INTEREST[/joe/NYmarathon/track.gpx]
```

**Named Function Networking (NFN):** Generation of named content (on-demand)

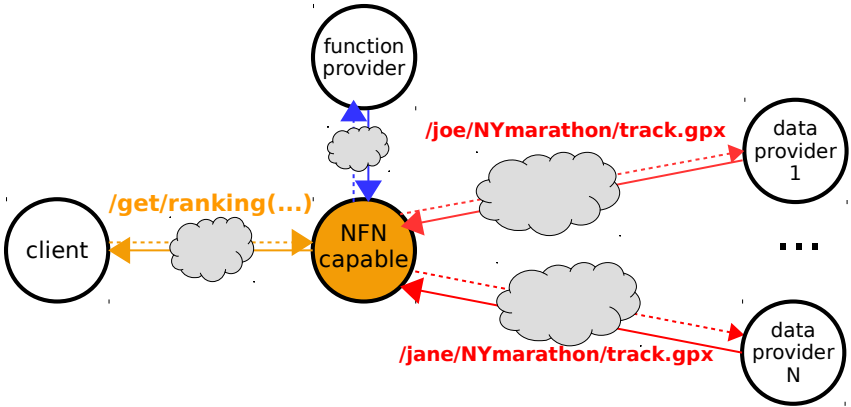
```
lookup:  /get/duration( /joe/NYmarathon/track.gpx )
         named function   named content
INTEREST[/get/duration/@x call 2 x |joe|NYmarathon|track.gpx/NFN]
INTEREST[/joe/NYmarathon/track.gpx/@x call 2 |get|duration x/NFN]
```

# Named-Function Networking: Network's View (Example 2)




Special NFN-capable nodes dissect the interest's NFN name and orchestrate the result derivation.

# Named-Function Networking: Network's View (Example 2)




Special NFN-capable nodes dissect the interest's NFN name and orchestrate the result derivation.

# Back to the Wish List...

**NFN Expression = Network Name of the Result = Production Recipe**

$\Longrightarrow$ *"On-demand Composition of Declarative Names"* (Apps)
High-level library of NFs allows to express domain-specific content needs.

$\Rightarrow$ *"Declarative Name"-to-"Production Recipe"* (Cloud Provider)
NFN Expression with NF bytecode is the "program" to produce the content.

$\Rightarrow$ *"Static Content Only!"* (Publisher)
NFN cares about dynamic/on-demand/derived content.

$\not\Rightarrow$ *"Descriptive Names"* (Security Mechanics)
Well, no per se clear what a NF does with the content.
But tomorrow: Schematized Access Control for Data Cubes and Trees

# Back to the Wish List...

**NFN Expression = Network Name of the Result = Production Recipe**

$\Rightarrow$ *"On-demand Composition of Declarative Names"* (Apps)

High-level library of NFs allows to express domain-specific content needs.

$\Longrightarrow$ *"Declarative Name"-to-"Production Recipe"* (Cloud Provider)

NFN Expression with NF bytecode is the "program" to produce the content.

$\Rightarrow$ *"Static Content Only!"* (Publisher)

NFN cares about dynamic/on-demand/derived content.

$\not\Rightarrow$ *"Descriptive Names"* (Security Mechanics)

Well, no per se clear what a NF does with the content.
But tomorrow: Schematized Access Control for Data Cubes and Trees

# Back to the Wish List...

**NFN Expression = Network Name of the Result = Production Recipe**

⇒ *"On-demand Composition of Declarative Names"* (Apps)

High-level library of NFs allows to express domain-specific content needs.

⇒ *"Declarative Name"-to-"Production Recipe"* (Cloud Provider)

NFN Expression with NF bytecode is the "program" to produce the content.

⟹ *"Static Content Only!"* (Publisher)

NFN cares about dynamic/on-demand/derived content.

$\not\Rightarrow$ *"Descriptive Names"* (Security Mechanics)

Well, no per se clear what a NF does with the content.
But tomorrow: Schematized Access Control for Data Cubes and Trees

# Back to the Wish List...

**NFN Expression = Network Name of the Result = Production Recipe**

⇒ *"On-demand Composition of Declarative Names"* (Apps)

High-level library of NFs allows to express domain-specific content needs.

⇒ *"Declarative Name"-to-"Production Recipe"* (Cloud Provider)

NFN Expression with NF bytecode is the "program" to produce the content.

⇒ *"Static Content Only!"* (Publisher)

NFN cares about dynamic/on-demand/derived content.

⇏ *"Descriptive Names"* (Security Mechanics)

Well, no per se clear what a NF does with the content.
But tomorrow: Schematized Access Control for Data Cubes and Trees

# Example: Relational Data Organization (1)

**Named Relations:** Structured Information

| PersID* | Name | Home |
|---|---|---|
| 1 | Alice | US |
| 2 | Bob | DE |
| 3 | ... | ... |

`/repo/people.table`

| EventID* | PersIDˆ | Name |
|---|---|---|
| 1 | 2 | NYmarathon |
| 2 | 1 | ParisMarathon |
| 3 | 1 | NYmarathon |

`/repo/events.table`

# Example: Relational Data Organization (1)

**Named Relations:** Structured Information

| PersID* | Name | Home |
|---|---|---|
| 1 | Alice | US |
| 2 | Bob | DE |
| 3 | ... | ... |

`/repo/people.table`

| EventID* | PersIDˆ | Name |
|---|---|---|
| 1 | 2 | NYmarathon |
| 2 | 1 | ParisMarathon |
| 3 | 1 | NYmarathon |

`/repo/events.table`

**Relation Schemata:** Meta-Data

```
PersID*:Int
Name:String
Home:String
```

`/ietf/relations/people.schema`

```
EventID*:Int
PersIDˆ:Int -> /ietf/relations/people.sche
Name:String
```

`/ietf/relations/events.schema`

# Example: Relational Data Organization (2)

**How to assign schemata to relations? Relation Type Schema (RTS):**

```
# published as /repo/rts
/repo/people.table       ->   /ietf/relations/people.schema
/repo/events.table       ->   /ietf/relations/events.schema
/repo/circuits/[^/]+     ->   /ietf/relations/circuits.schema
```

Typing: Mapping named relations (name patterns) to schemata documents.

# Example: Basic Query Operations

DB Research: Few basic set operations are sufficient to define a rich relational algebra.

→ **restrict:** `/named/fct/restrict( /repo/people.table, Home == 'US' )`
– **project:** `/named/fct/project( /repo/people.table, [PersID,Name] )`
– **join:**
```
/named/fct/join( /repo/events.table as 'event',
                 /repo/people.table as 'people' )
```

| PersID* | Name | Home |
|---|---|---|
| 1 | Alice | US |
| 2 | Bob | DE |

# Example: Basic Query Operations

DB Research: Few basic set operations are sufficient to define a rich relational algebra.

– **restrict:** `/named/fct/restrict( /repo/people.table, Home == 'US' )`
→ **project:** `/named/fct/project( /repo/people.table, [PersID,Name] )`
– **join:**
```
/named/fct/join( /repo/events.table as 'event',
                 /repo/people.table as 'people' )
```

| PersID* | Name | Home |
|---|---|---|
| 1 | Alice | US |
| 2 | Bob | DE |

# Example: Basic Query Operations

DB Research: Few basic set operations are sufficient to define a rich relational algebra.

- **restrict:** `/named/fct/restrict( /repo/people.table, Home == 'US' )`
- **project:** `/named/fct/project( /repo/people.table, [PersID,Name] )`

→ **join:**
```
/named/fct/join( /repo/events.table as 'event',
                 /repo/people.table as 'people' )
```

| event.EventID* | event.Name | people.PersID" | people.Name | people.Home |
|---|---|---|---|---|
| 1 | NY Marathon | 2 | Bob | DE |
| 2 | Paris Marathon | 1 | Alice | US |
| 3 | NY Marathon | 1 | Alice | US |

# Query Composition: Consumer's View

*"I need the list of countries from which at least one person attended the NY Marathon."*

**The app composes the list's network name:**

```
/named/fct/project(
  /named/fct/restrict(
    /named/fct/join(/repo/events as 'event', /repo/people as 'people'),
    event.Name == 'NY Marathon'
  ),
  [people.Home]
)
```

# Query Composition: Consumer's View

*"I need the list of countries from which at least one person attended the NY Marathon."*

**The app composes the list's network name:**

```
/named/fct/project(
  /named/fct/restrict(
    /named/fct/join(/repo/events as 'event', /repo/people as 'people'),
    event.Name == 'NY Marathon'
  ),
  [people.Home]
)
```

# Query Composition: Consumer's View

*"I need the list of countries from which at least one person attended the NY Marathon."*

**The app composes the list's network name:**

```
/named/fct/project(
  /named/fct/restrict(
    /named/fct/join(/repo/events as 'event', /repo/people as 'people'),
    event.Name == 'NY Marathon'
  ),
  [people.Home]
)
```

# Query Composition: Consumer's View

*"I need the list of countries from which at least one person attended the NY Marathon."*

**The app composes the list's network name:**

```
/named/fct/project(
  /named/fct/restrict(
    /named/fct/join(/repo/events as 'event', /repo/people as 'people'),
    event.Name == 'NY Marathon'
  ),
  [people.Home]
)
```

**Derivation Chain:** Primary Data → `join()` → `restrict()` → `project()` → Result

# Stepping Through the Derivation Chain...

*Step 1*

| EventID* | PersIDˆ | Name |
|---|---|---|
| 1 | 2 | NYmarathon |
| 2 | 1 | ParisMarathon |
| 3 | 1 | NYmarathon |

`/repo/events`

| PersID* | Name | Home |
|---|---|---|
| 1 | Alice | US |
| 2 | Bob | DE |

`/repo/people`

`/named/fct/join`

**Execution Step 1**

# Stepping Through the Derivation Chain...

*Step 1*

| EventID* | PersID^ | Name |
|---|---|---|
| 1 | 2 | NYmarathon |
| 2 | 1 | ParisMarathon |
| 3 | 1 | NYmarathon |

/repo/events

| PersID* | Name | Home |
|---|---|---|
| 1 | Alice | US |
| 2 | Bob | DE |

/repo/people

/named/fct/join

**Execution Step 1**

| event.EventID* | event.Name | people.PersID" | people.Name | people.Home |
|---|---|---|---|---|
| 1 | NY Marathon | 2 | Bob | DE |
| 2 | Paris Marathon | 1 | Alice | US |
| 3 | NY Marathon | 1 | Bob | DE |

```
/named/fct/join( /repo/events as 'event', /repo/people as 'people' )
```

# Stepping Through the Derivation Chain... *Step 2*

| event.EventID* | event.Name | people.PersID" | people.Name | people.Home |
|---|---|---|---|---|
| 1 | NY Marathon | 2 | Bob | DE |
| 2 | Paris Marathon | 1 | Alice | US |
| 3 | NY Marathon | 1 | Alice | US |

```
/named/fct/join( /repo/events as 'event', /repo/people as 'people' )
```

```
/named/fct/restrict
```

**Execution Step 2**

# Stepping Through the Derivation Chain... *Step 2*

| event.EventID* | event.Name | people.PersID" | people.Name | people.Home |
|---|---|---|---|---|
| 1 | NY Marathon | 2 | Bob | DE |
| 2 | Paris Marathon | 1 | Alice | US |
| 3 | NY Marathon | 1 | Alice | US |

```
/named/fct/join( /repo/events as 'event', /repo/people as 'people' )
```

```
/named/fct/restrict
```

**Execution Step 2**

| event.EventID* | event.Name | people.PersID" | people.Name | people.Home |
|---|---|---|---|---|
| 1 | NY Marathon | 2 | Bob | DE |
| 3 | NY Marathon | 1 | Alice | US |

```
/named/fct/restrict( /fct/join(/repo/events as 'event', /repo/people as 'people' ),
                     event.Name == 'NYmarathon' )
```

# Stepping Through the Derivation Chain... *Step 3*

| event.EventID* | event.Name | people.PersID” | people.Name | people.Home |
|---|---|---|---|---|
| 1 | NY Marathon | 2 | Bob | DE |
| 3 | NY Marathon | 1 | Alice | US |

```
/named/fct/restrict( /fct/join(/repo/events as 'event', /repo/people as 'people' ),
                     event.Name == 'NYmarathon' )
```

`/named/fct/project`

**Execution Step 3**

# Stepping Through the Derivation Chain... *Step 3*

| event.EventID* | event.Name | people.PersID" | people.Name | people.Home |
|---|---|---|---|---|
| 1 | NY Marathon | 2 | Bob | DE |
| 3 | NY Marathon | 1 | Alice | US |

```
/named/fct/restrict( /fct/join(/repo/events as 'event', /repo/people as 'people' ),
event.Name == 'NYmarathon' )
```

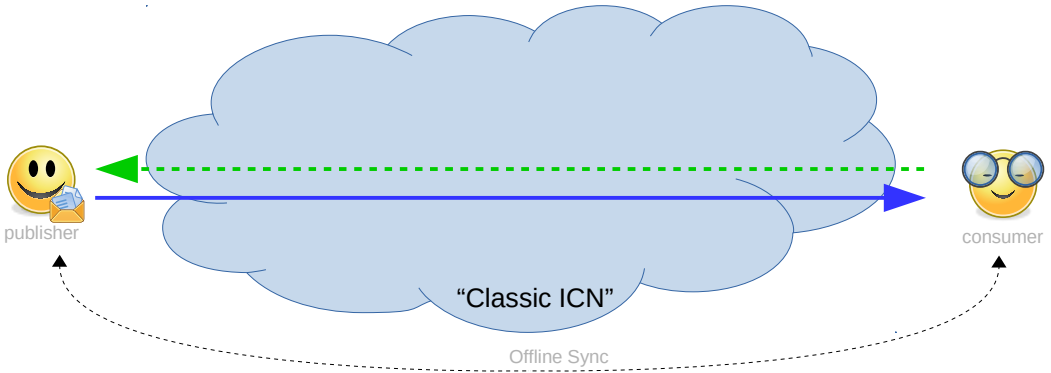
`/named/fct/project`

**Execution Step 3**

| people.Home |
|---|
| DE |
| US |

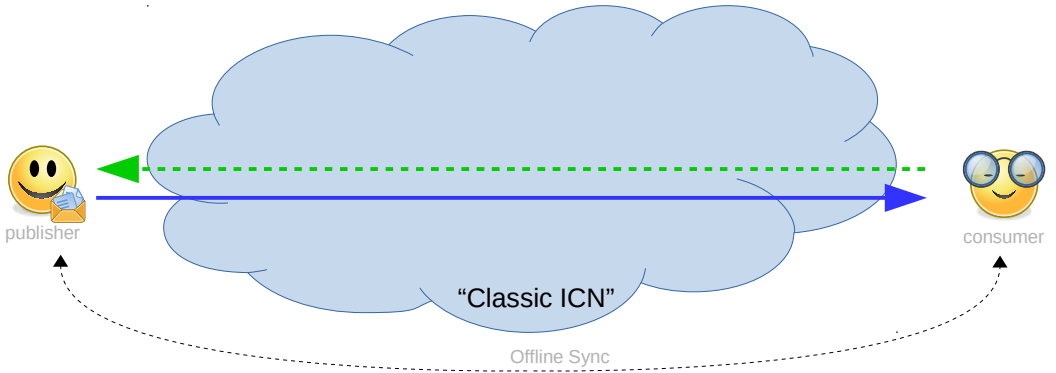
```
/named/fct/project( /fct/restrict( /fct/join(/repo/events as 'event', /repo/people
as 'people' ), event.Name == 'NYmarathon' ), [people.Home] )
```

# Conclusion



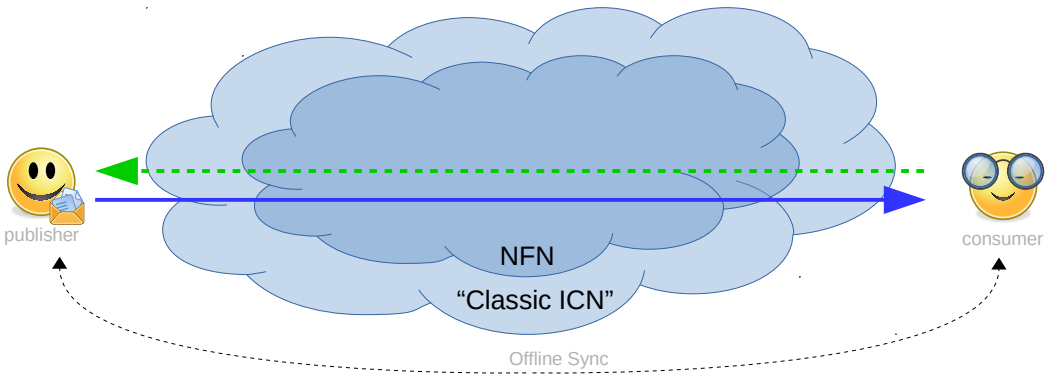
# Conclusion

**Limitations:** Static Content Only



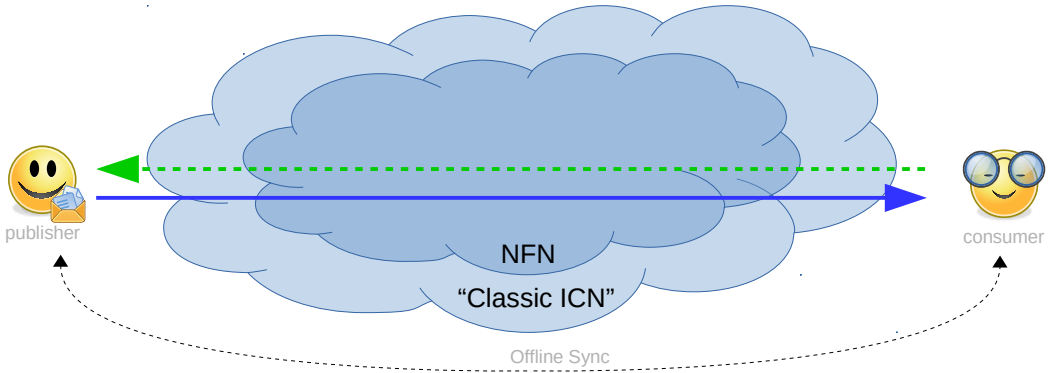

**Offline Sync:** Namespace and Data Format Conventions (empirically)

# Conclusion



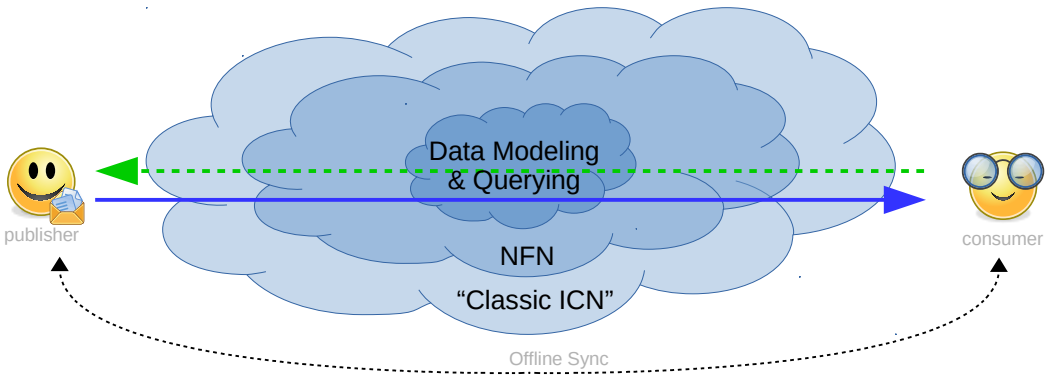
# Conclusion

**New:** Dynamic/On-Demand/Derived Content



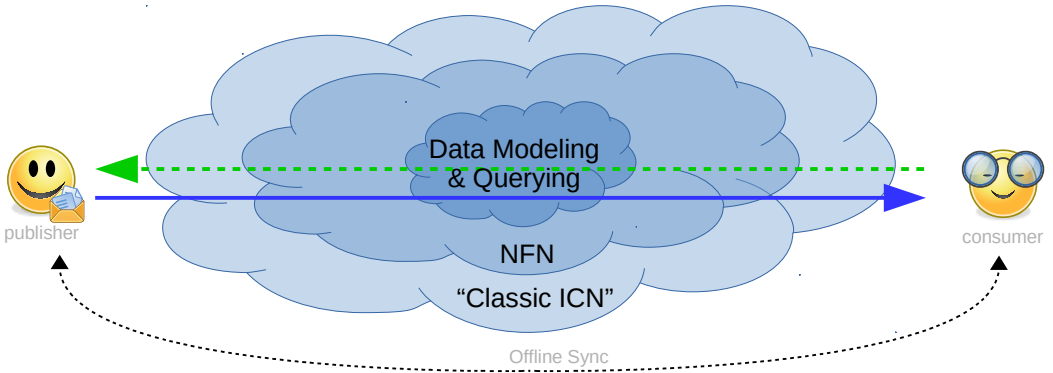

**Additional Offline Sync:** Named Function’s Semantics their I/O Formats

# Conclusion


Data Modeling
& Querying
NFN
“Classic ICN”
publisher
consumer
Offline Sync

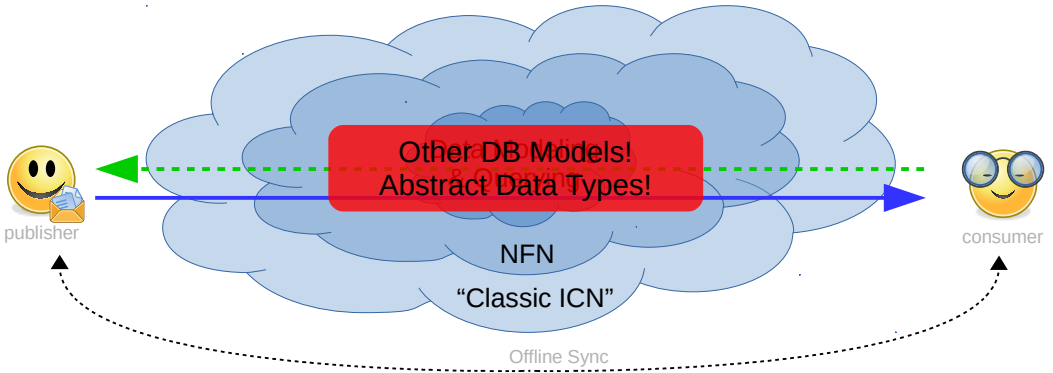
# Conclusion

**New:** Minimized Offline Synchronization




**Offline Sync:** Relation Type Schema Names, Basic Query Functions (join, restrict, project)

# Conclusion

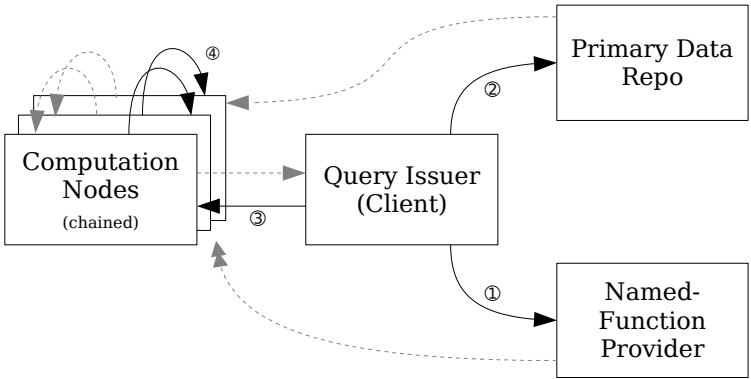
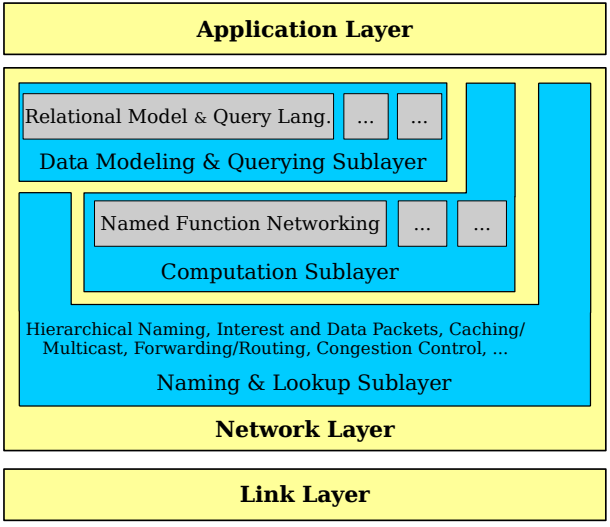



Relational Modeling just as an example: Select a proper data model for *your* application and put it inside the network.

# Q & A

# Additional Slides

# Trust Assumptions



# Network Model