RICE: Remote Method Invocation in ICN

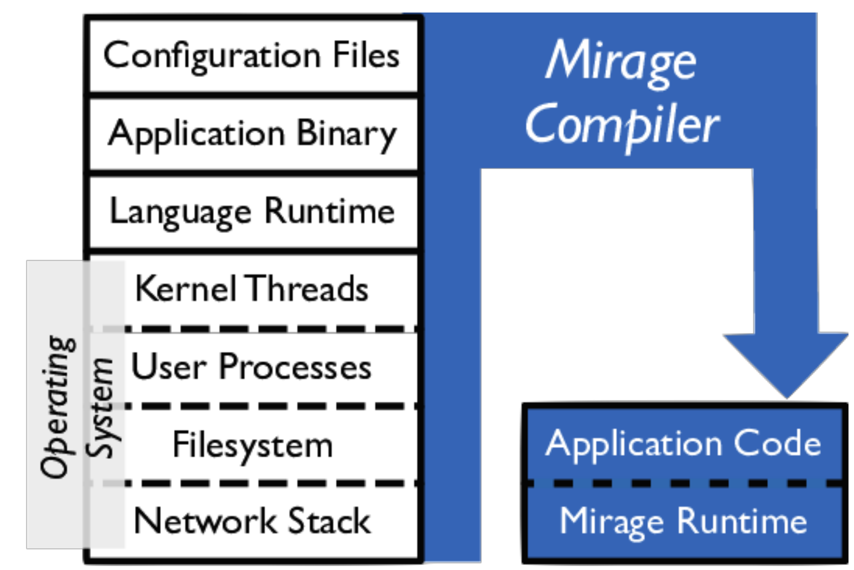
draft-kutscher-icnrg-rice-00

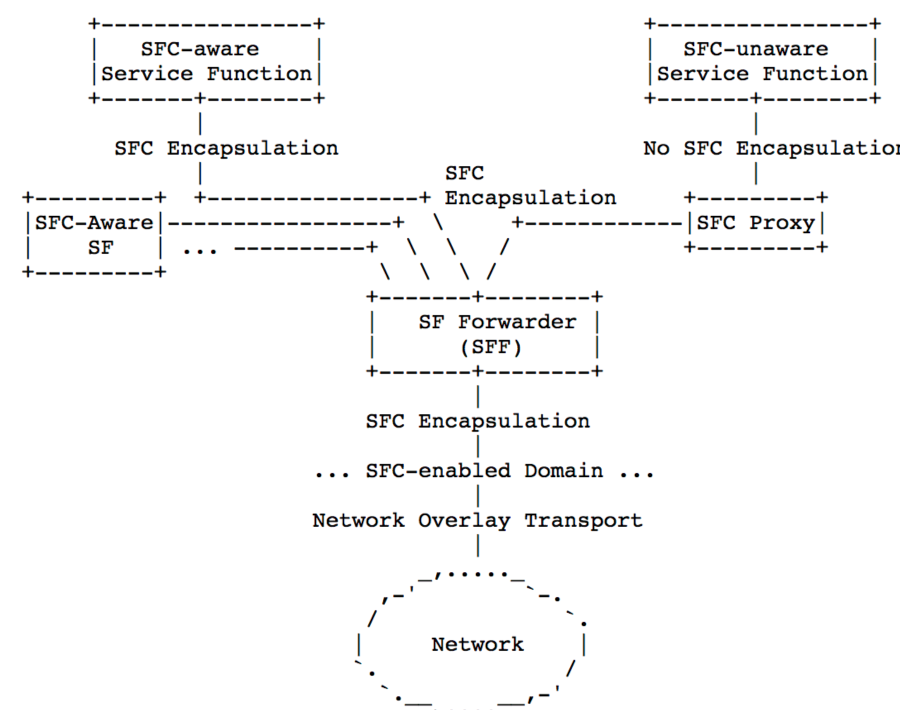
Michał Król, Karim Habak, Dave Oran, Dirk Kutscher, Yiannis Psaras


THERE ARE ONLY 2 HARD THINGS
IN COMPUTER SCIENCE:
0. Cache Invalidation
1. Naming Things
7. Asynchronous Callbacks
2. Off-by-one errors

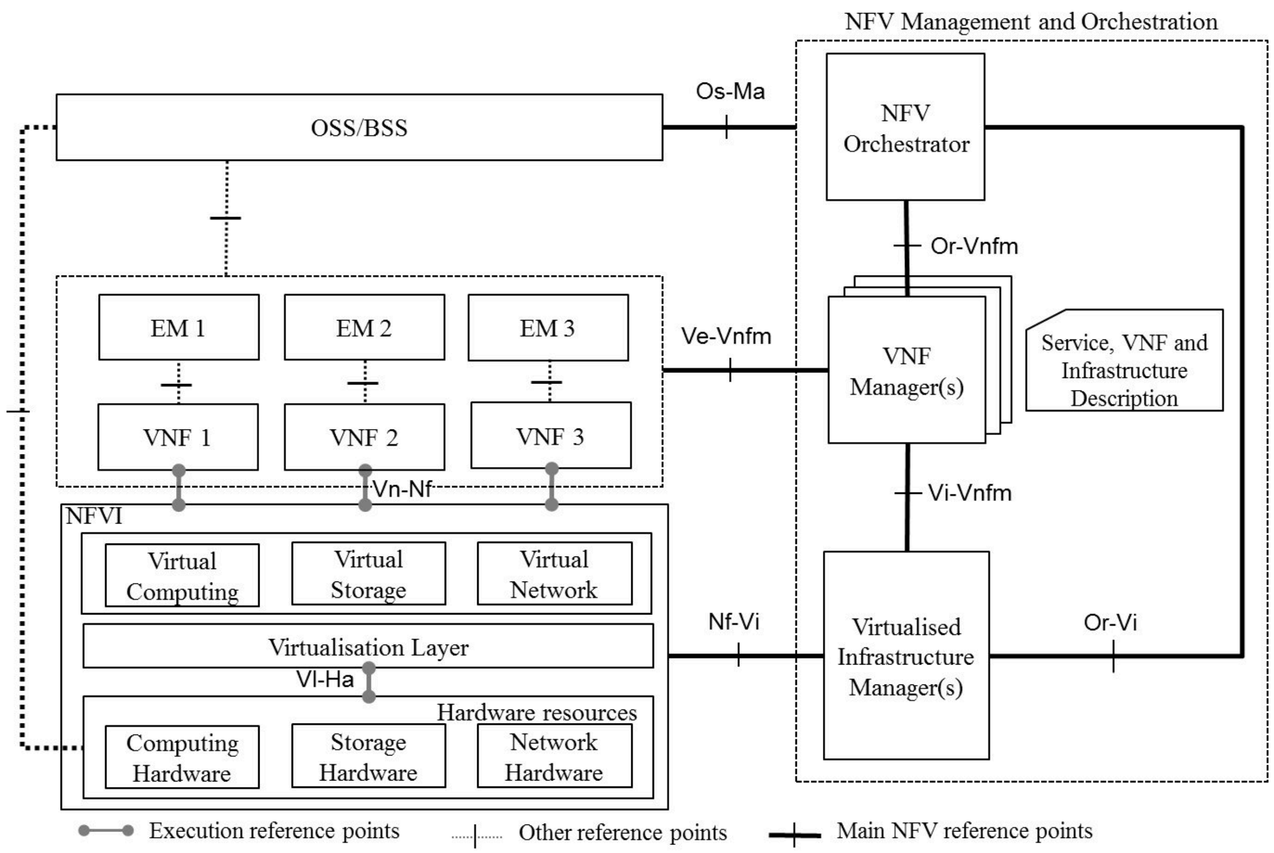
Cannot Leverage Computation in Networks Today

- **Significant advances in making computation available, affordable, programmable**
  - Virtualization: big leaps from host virtualisation to unikernels, lambda expression evaluation engines
  - Application layer frameworks for data processing, microservice architectures, virtualized network automation
- **Networking is lacking behind**
  - Connection-based communication and security model: cannot introduce computation without breaking security and introducing significant overhead
  - IP address-based communication: leads to static and difficult to manage networked computation (“service function chaining”) — not applicable to dynamic, mobile environments
  - No concept for computation on data plane: leads to complex orchestration and management frameworks





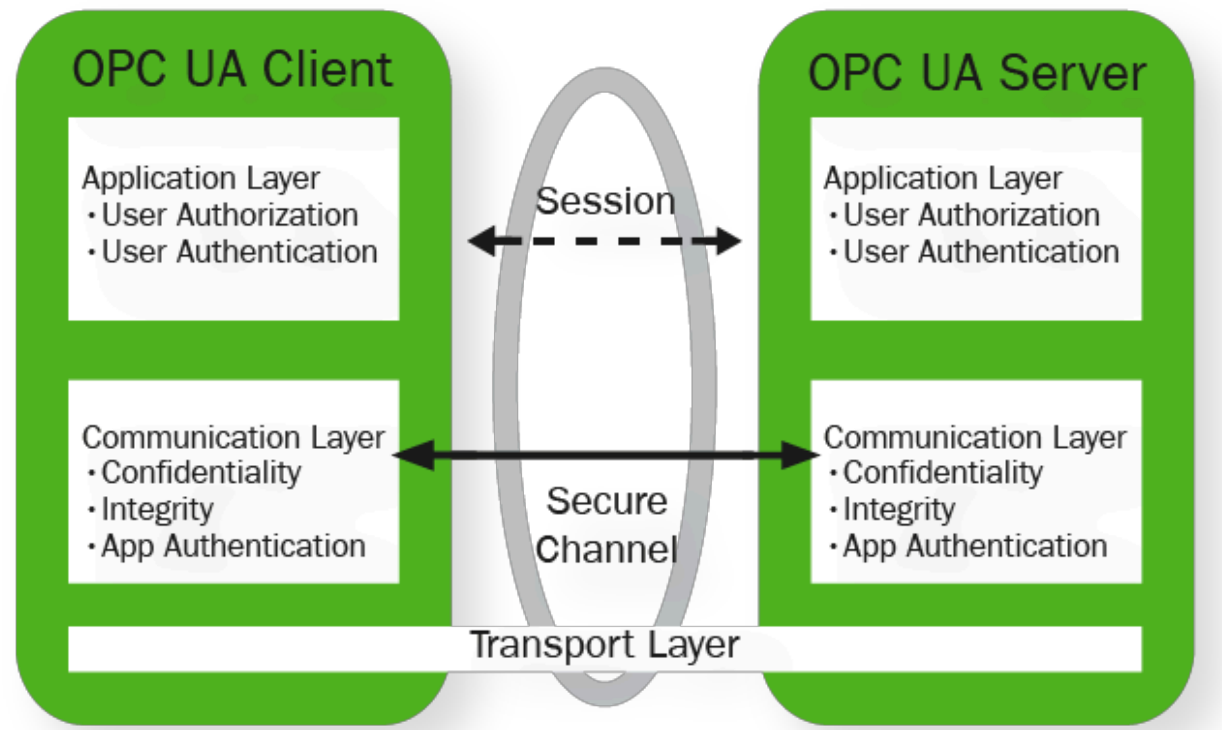
Different Perspectives on Compute & Networking



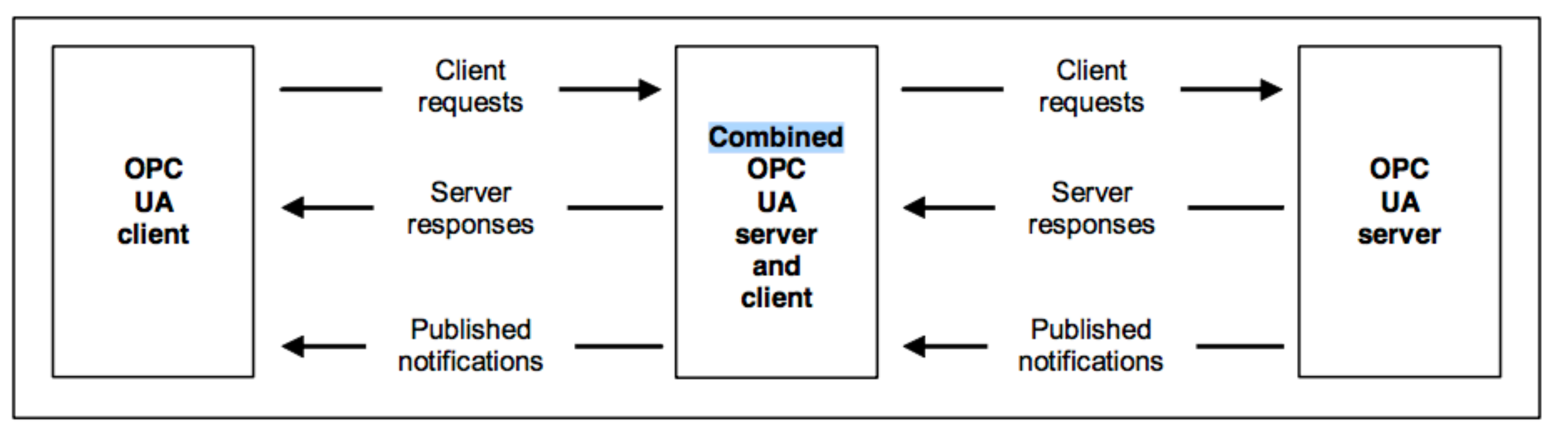
(Virtualized) Compute Servers in Networks

Networked Computations

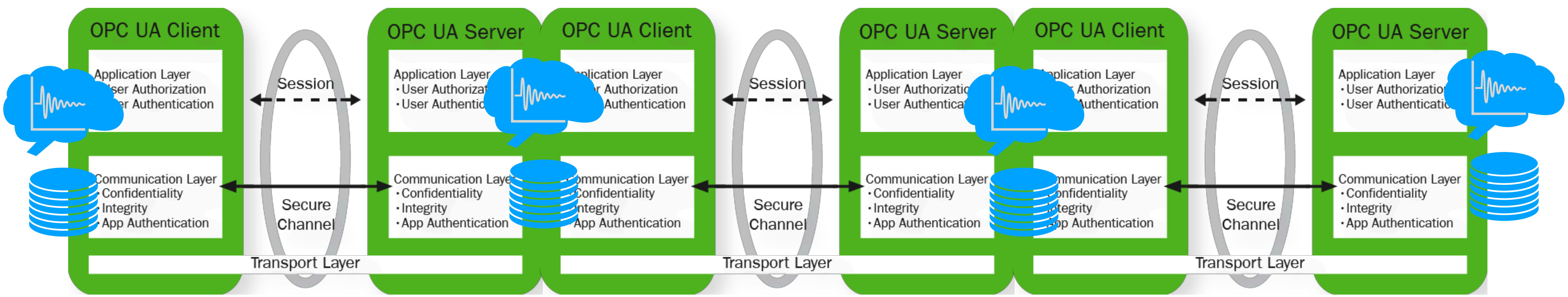
Data-oriented Communication



- Several application-layer frameworks
- Data-oriented communication (accessing named-data on a server)
- Communication inside TLS-secured connections
  - Data sharing difficult
  - Limited scalability
  - Potentially very inefficient
- Not designed for enterprise access control & communication policing
  - NAT & firewall traversal



In-Network Computing With Client-Server Protocols



- Overlays
  - Connection-based security
  - Client-server / broker-based
- Limited Scalability
  - Pub-sub distribution to many clients through single-server bottleneck
- Limited efficiency
  - Cannot share data directly
- Limited performance and robustness
  - Network cannot assist data dissemination

Adding a little computation to a data kiosk system is not exactly distributed computing.

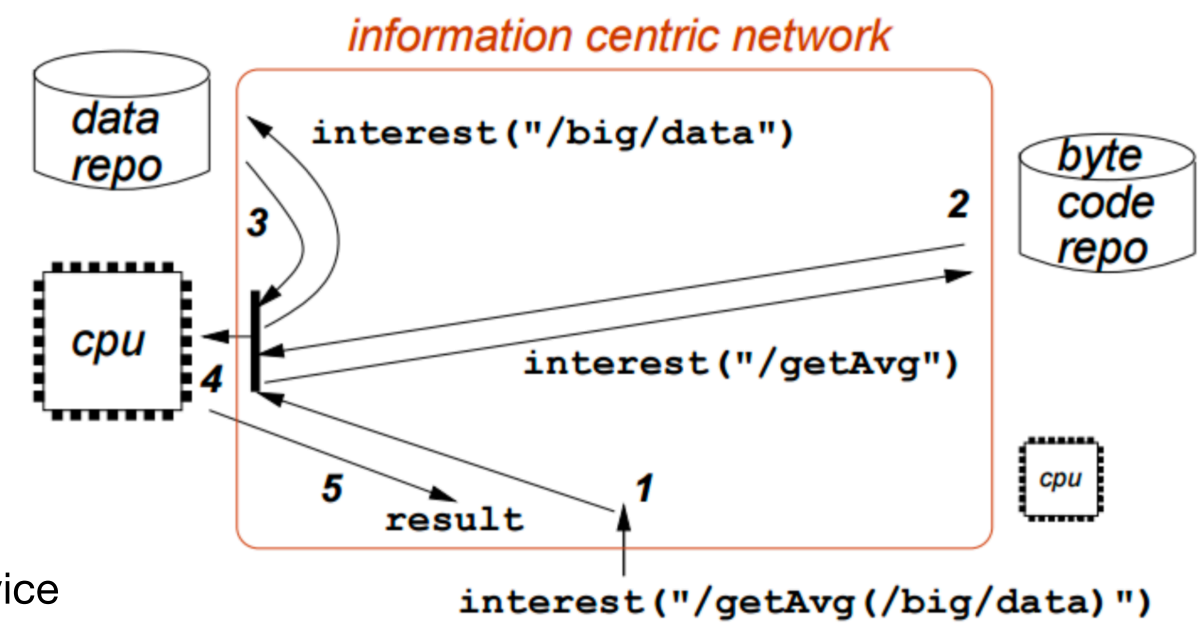
Computing in ICN Networks

Can move some functions from overlay (or app layer) to network layer

- Load balancing
    - Extend forwarder load-balancing for INTEREST forwarding to computation requests
    - Holistic view on load — server load *and* network load
- Failure resiliency
    - Routing state for multiple instances of a function in the network
    - Do fail-over implicitly through forwarding (and forwarding strategies)
- Result sharing
    - Caching computation results
- Pub-sub

Named Function Networking

- **ICN: Accessing named data in the network**
  - Securely
  - Both static and dynamic (e.g., live stream)
- **Challenge: How to achieve dynamic computation?**
  - With similar security properties?
  - ... And automatic function placement?
  - Think: edge computing, big data, stream processing, service chaining
- **Named Function Networking**
  - `/getAverage(/roomA/temp, /roomB/tmp)`
  - Apps specify desired results
  - Networks finds data and functions – and execution locations
  - Results can be cached just like regular ICN



Christian Tschudin; Named Function Networking Service chaining, big picture, division of labor boundaries; https://www.ietf.org/proceedings/interim-2015-icnrg-01/slides/slides-interim-2015-icnrg-1-13.pdf

Named Functions
(λ-reduction inside CCN)

http://www.named-function.net/

NFN for Data-Oriented Applications

Fine-granular access to Named Data structures in ICN

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] )`

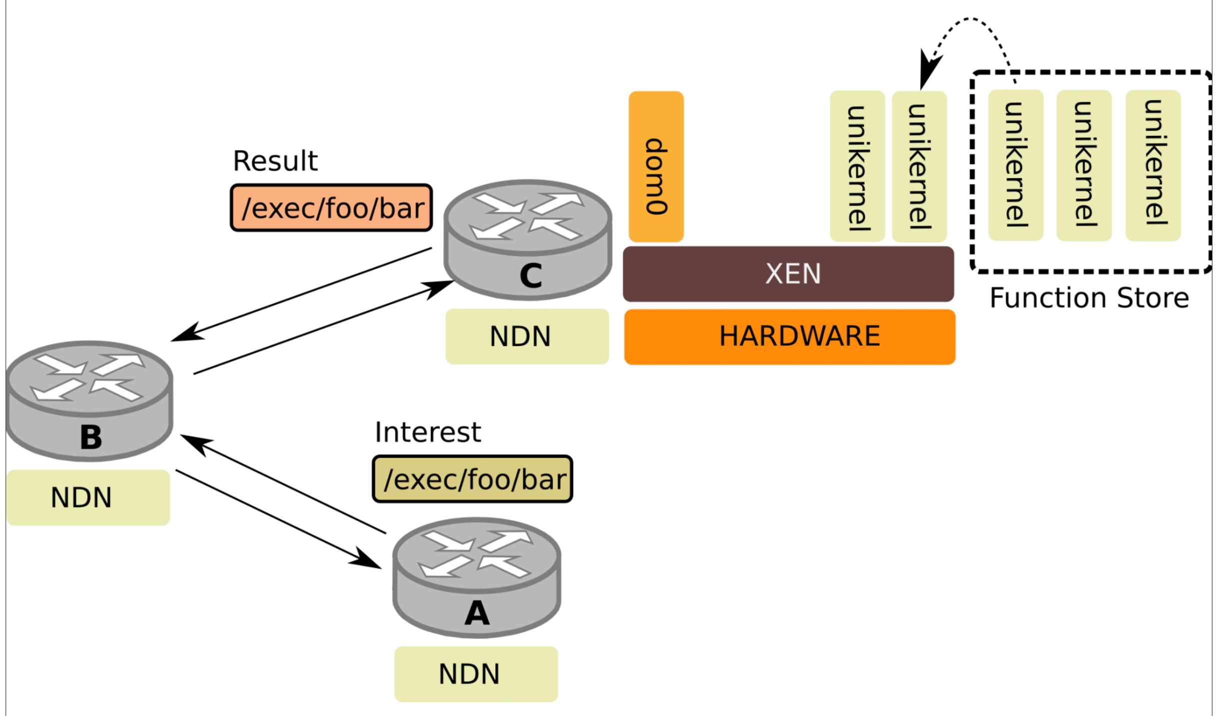
→ **join:**

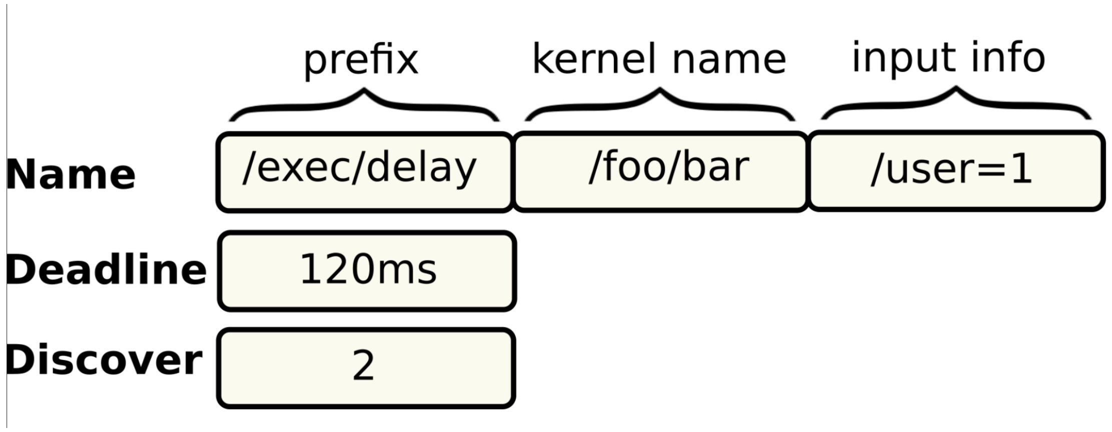
```
/named/fct/join( /repo/events.table as 'event',
                 /repo/people.table as 'people' )
```

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

Claudio Marxer, Christian Tschudin; **Improved Content Addressability Through Relational Data Modeling and In-Network Processing Elements**; ACM ICN 2017

Named Function as a Service (NFaaS)





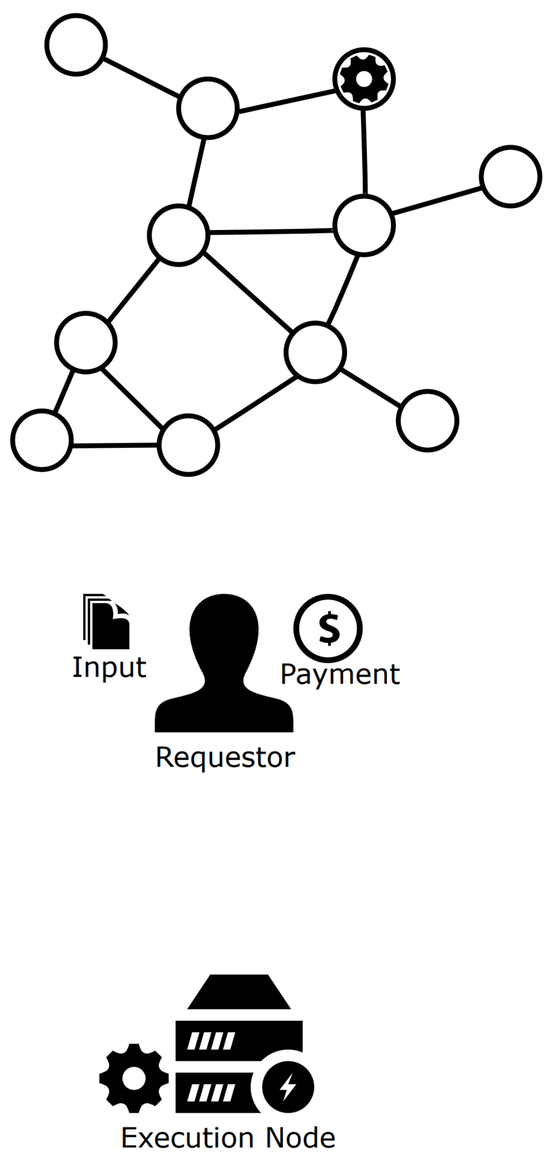
NFaaS

- Completely distributed
- Moving function where they're needed
- Functions as stateless unikernels
- Nodes run popularity contest
- Different forwarding strategies

Michal, Krol, Ioannis Psaras; **NFaaS: Named Function as a Service**; ACM ICN 2017

Decentralized Computations

Named Function as a Service



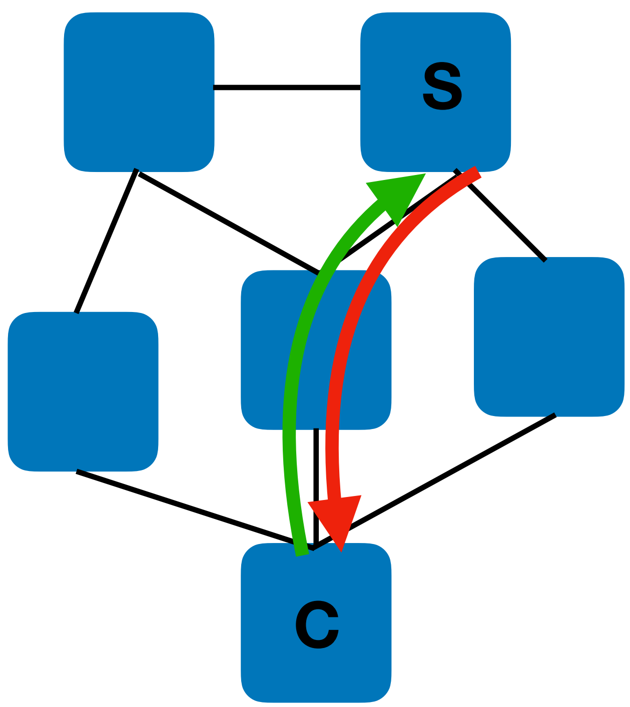
SPOC

- Automatic payments and result verification
- Based on Smart Contracts and Intel SGX
- No 3rd parties involved

- Secure against Rational Attacker
- Minimal computational overhead
- No calls privacy

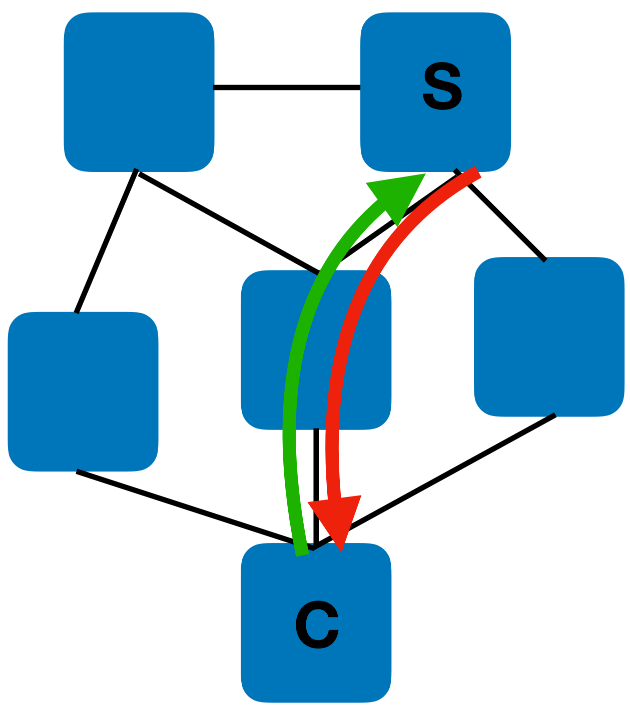
Michał Król , Ioannis Psaras; **Decentralized Computations**;
Presentation at IRTF Proposed DINRG Interim Meeting; February 2018

Robust Remote Method Invocation in ICN



- **ICN key properties**
  - No host addresses
  - Receiver-driven: Interest-Data Exchange
  - Flow balance: exactly one Data message per Interest
  - Path symmetry: Data follows reverse Interest path
  - Interest rate controls flow bandwidth, congestion etc.
  - No consumer identities needed
  - Consumer mobility through Interest soft state

Robust Remote Method Invocation in ICN

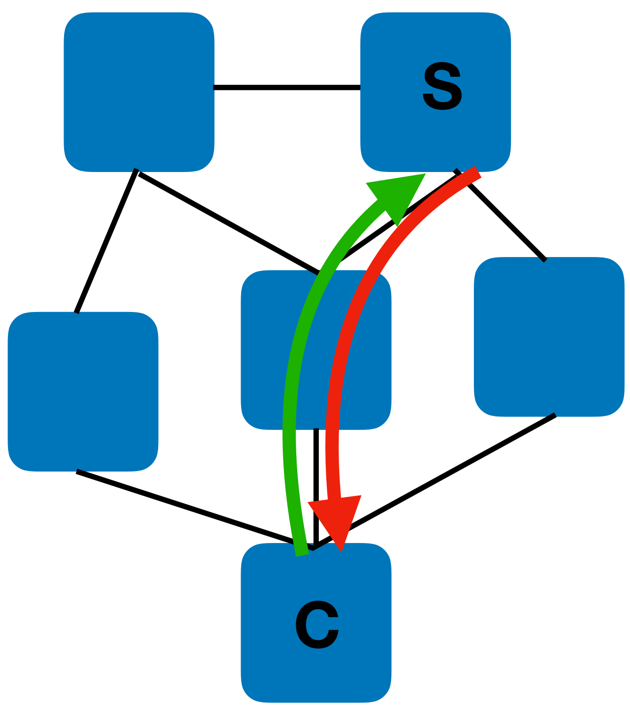


Interest: /s/rmi/funcA/

Data: <result object>

- **Naive Approach I**
  - Map method invocation to Interest-Data
  - Method Parameters in Interest message
  - Result data is the Named Data Object

Robust Remote Method Invocation in ICN

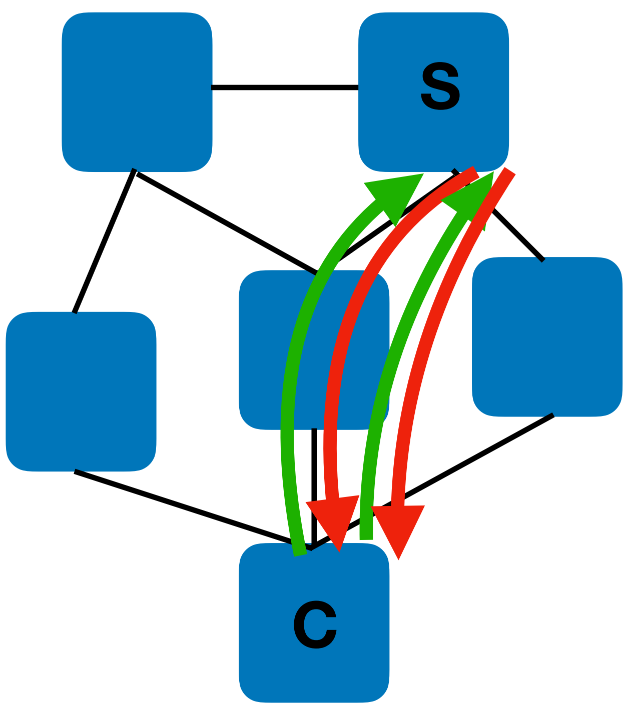


Interest: /s/rmi/funcA/

Data: <result object>

- **Issues**
    - Interest messages: large parameter sets in non-congestion-controlled messages
    - Non-trivial computations will take longer than Interest soft state in the network is available
    - Security: authenticating method invocations?
    - Robustness: computational overload attacks

Robust Remote Method Invocation in ICN



Interest: /s/rmi/funcA/

Data: ACK (handle XY)

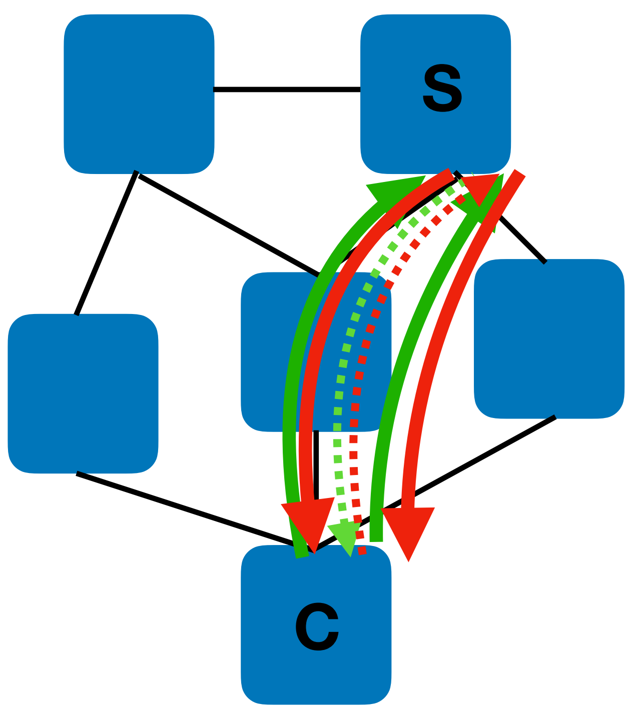
.
.
.

Interest: /s/rmi/funcA/result/XY

Data: <result object>

- **Naive Approach II**
  - Two Interest-Data Exchanges
    1. Initiating method invocation
    2. Collecting Results
- Problems
  - Don’t know when result is ready
  - Still same computational overload attack and Interest congestion problems

Robust Remote Method Invocation in ICN



Interest: /s/rmi/funcA/

Data: ACK (handle XY)

.

.

Interest: /c/funcA/XY/data-ready

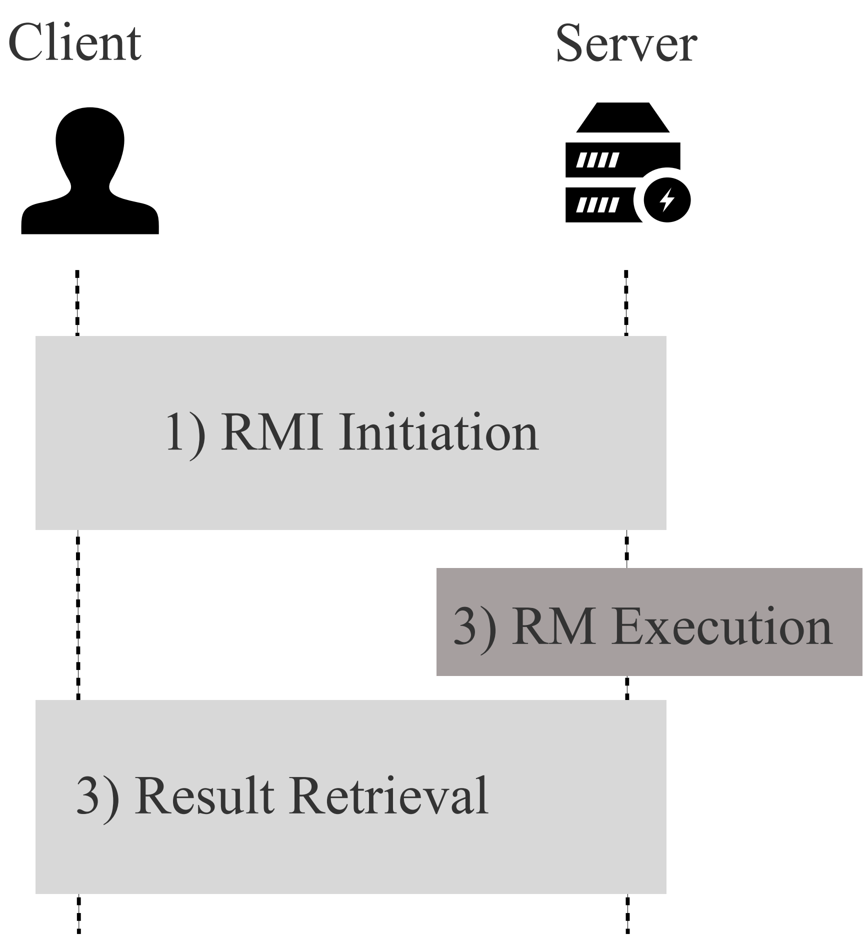
Data: ACK

Interest: /s/rmi/funcA/result/XY

Data: <result object>

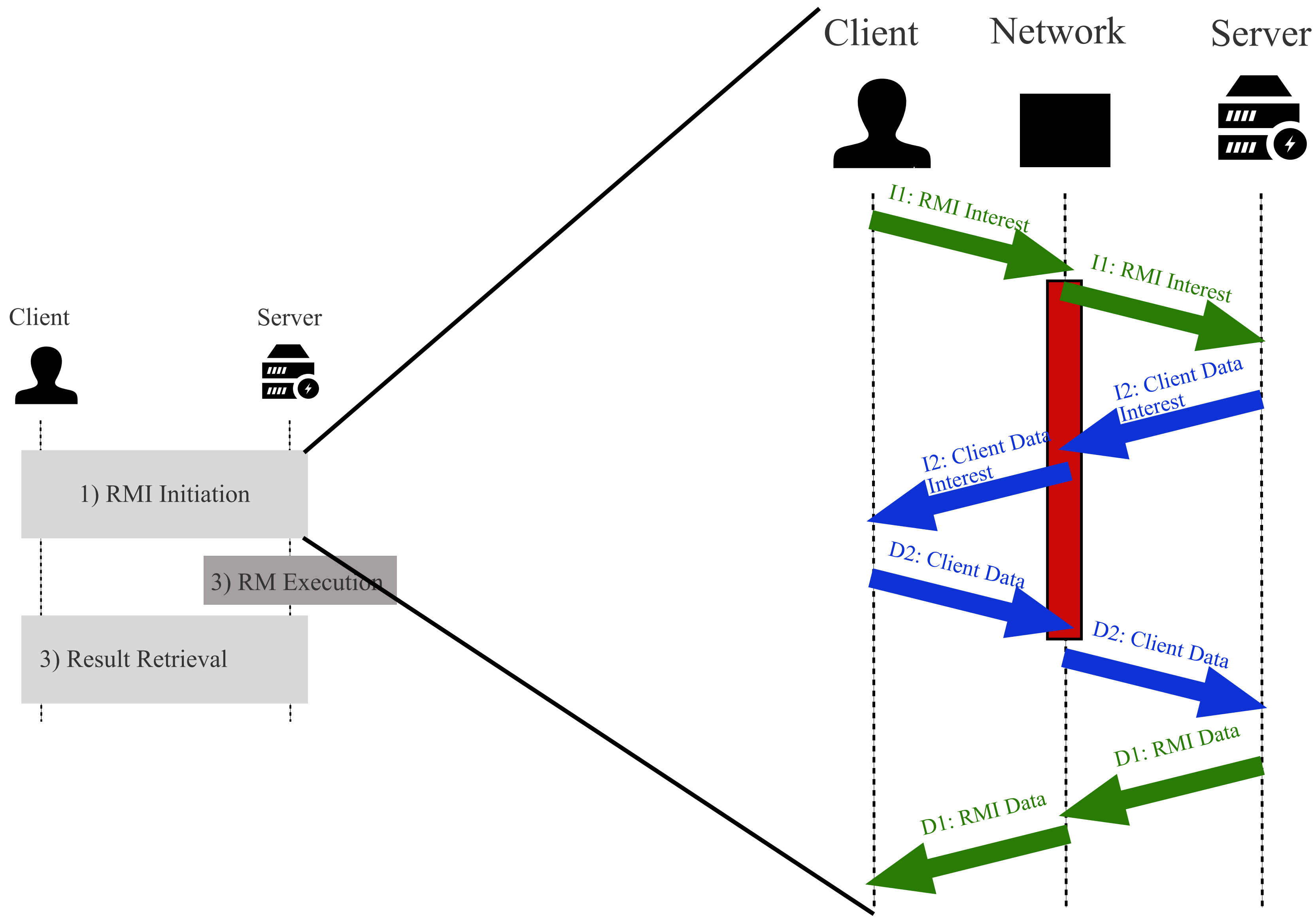
- **Naive Approach III**
  - Three Interest-Data Exchanges: Server notifies client when result is ready
    1. Initiating method invocation
    2. “Result ready” notification
    3. Client collects results
- Problems
  - Still same computational overload attack and Interest congestion problems
  - Client needs to be globally reachable and disclose its name
  - Introducing producer mobility requirements for clients

Robust Remote Method Invocation in ICN



- **Remote Method Invocation in ICN proposal**
  - Decoupling application (server) time from network time
    - State in network is ephemeral & soft — RTT timeframe…
    - Application/processing happens in different timeframes — should not be constrained by network
  - Idiomatic ICN parameter data retrieval
    - Server retrieves parameters (and authentication credentials) from client
    - Reducing surface for overload attacks

Robust Remote Method Invocation in ICN



I1 RMI Interest signals client handle to network.

Used to install ephemeral reverse forwarding state for I2 exchange (and to extend timers for I1 pending Interest state).

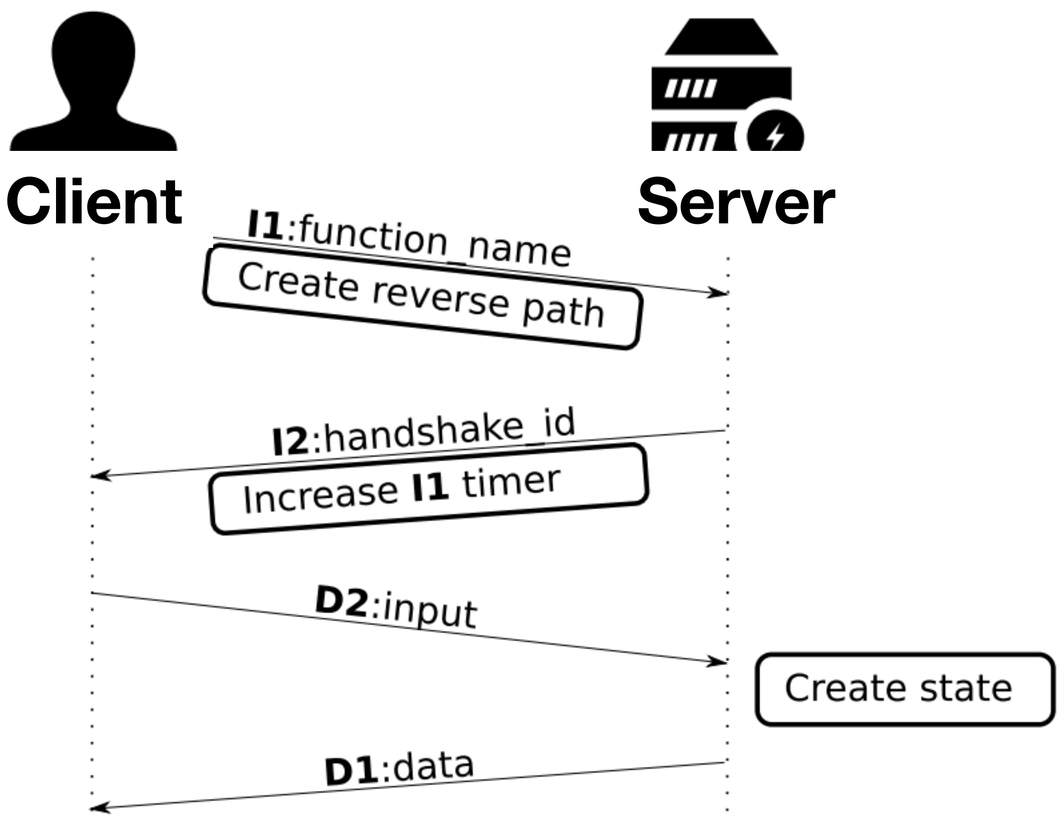
Server checks function name and requests client authentication and function parameters.

Server verifies client credentials and processes input parameters.

Server commits processing resources and returns a handle for the result data.

Note: I1/I2/D2 sequence could also be used to piggyback CCNxKey exchange (TLS-1.3 equivalent)

Additional Forwarder Behavior

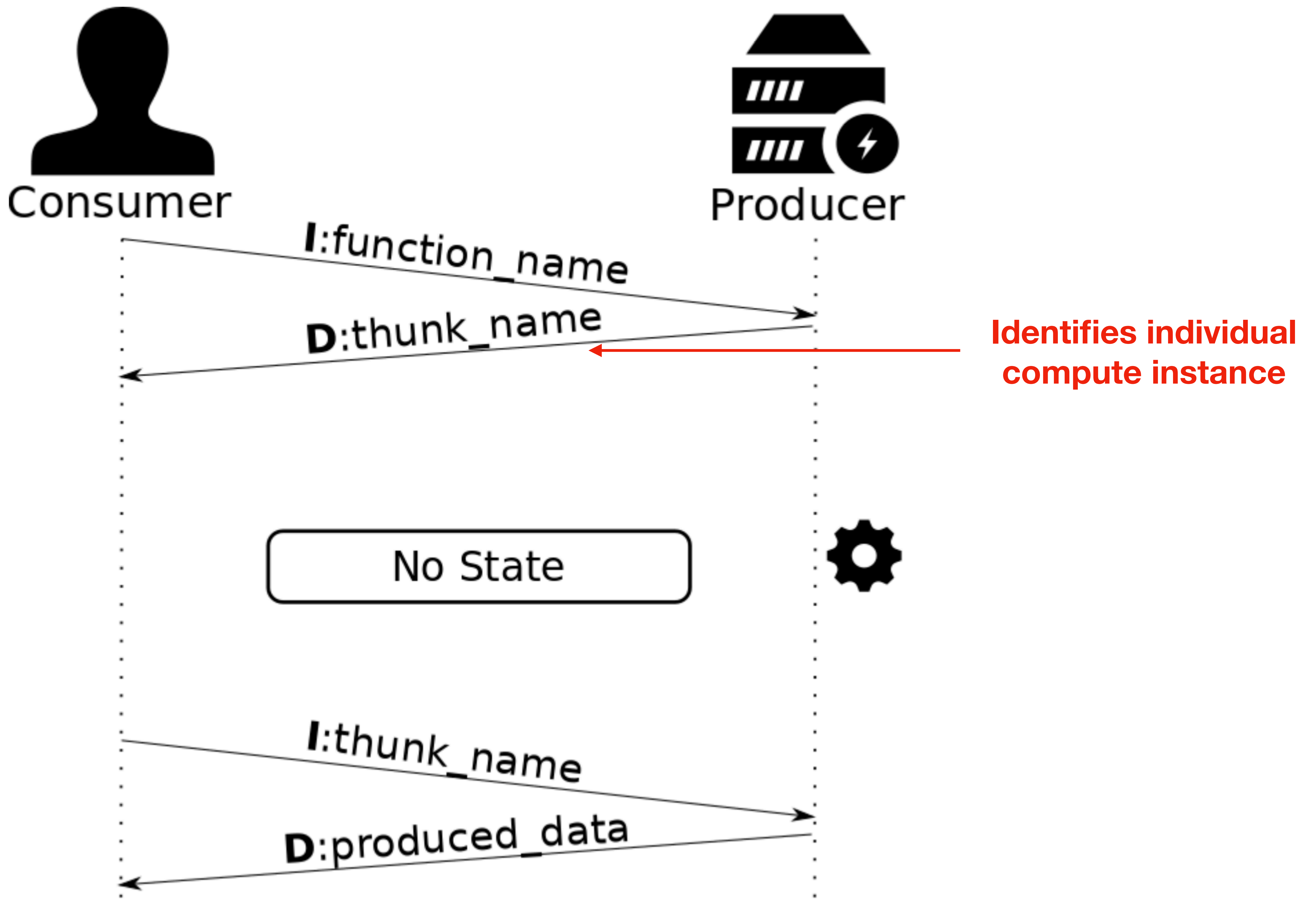


- **I1 Interest**
  - Signals invocation-specific client name (non-globally routable) to network and server
  - Creates emphemeral FIB-entry for client name for later I2 exchange
- **I2 Interest**
  - Follow path per ephemeral FIB entries
  - Extends timer for I1 PIT entries

Implementation Considerations

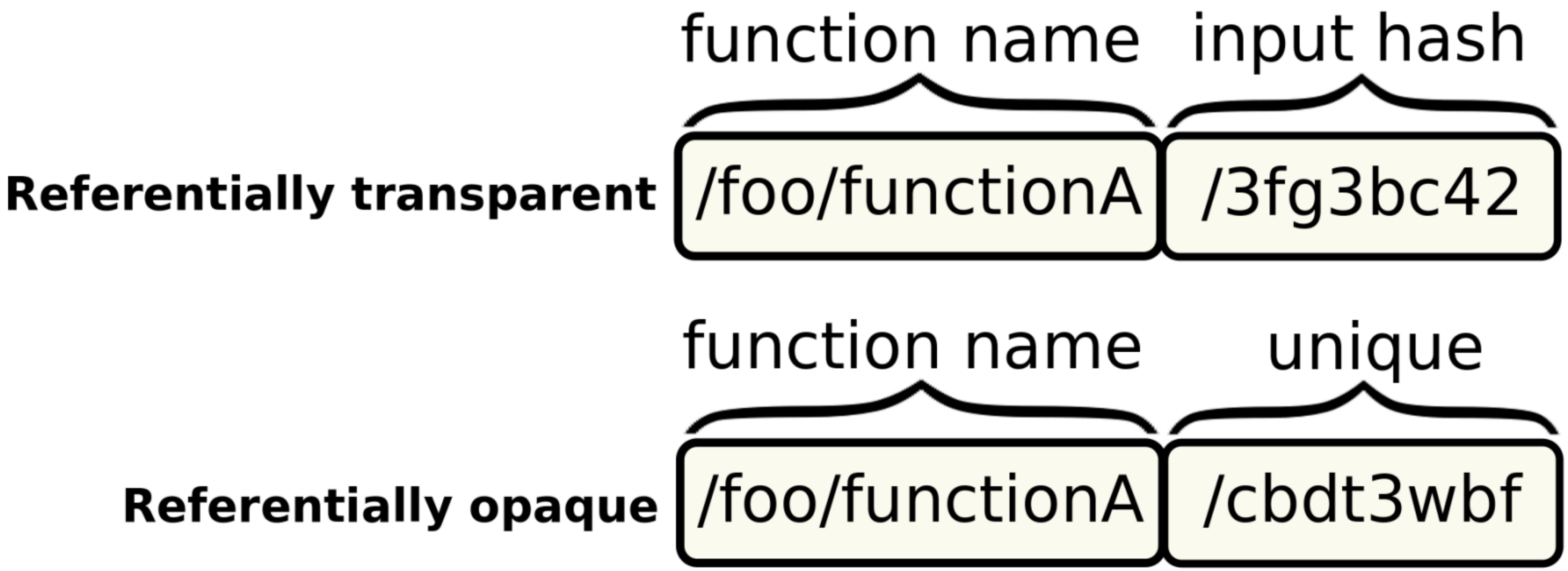
- Forwarding tables normally optimized for read access
  - RICE is modifying FIBs at line rate
- Different approaches
  - Chose appropriate access algorithms with good write performs and no read/write locks
  - Separate data structures for temporary RICE FIB entries
    - Could use name prefix convention to help forwarder making this lookup efficient

Thunks

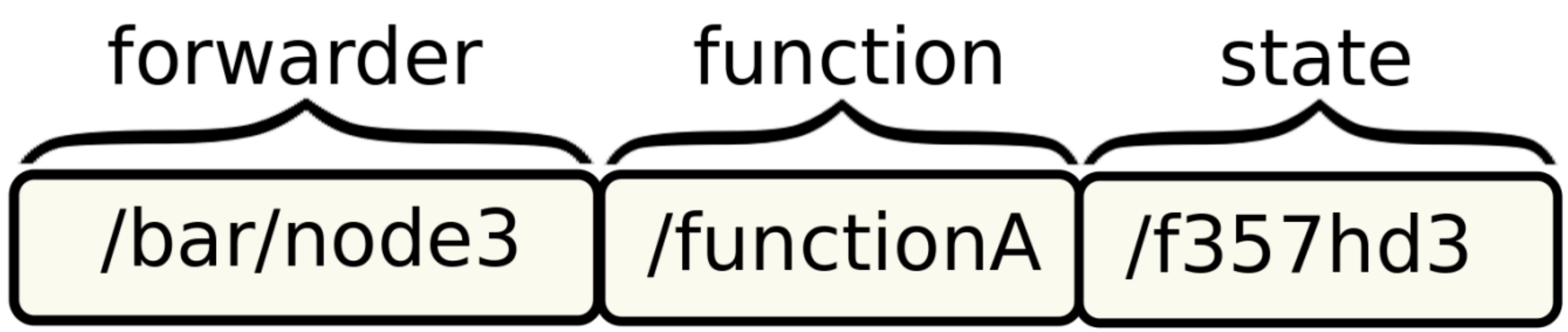


Consumer
Producer
I:function_name
D:thunk_name
Identifies individual compute instance
No State
I:thunk_name
D:produced_data

Naming

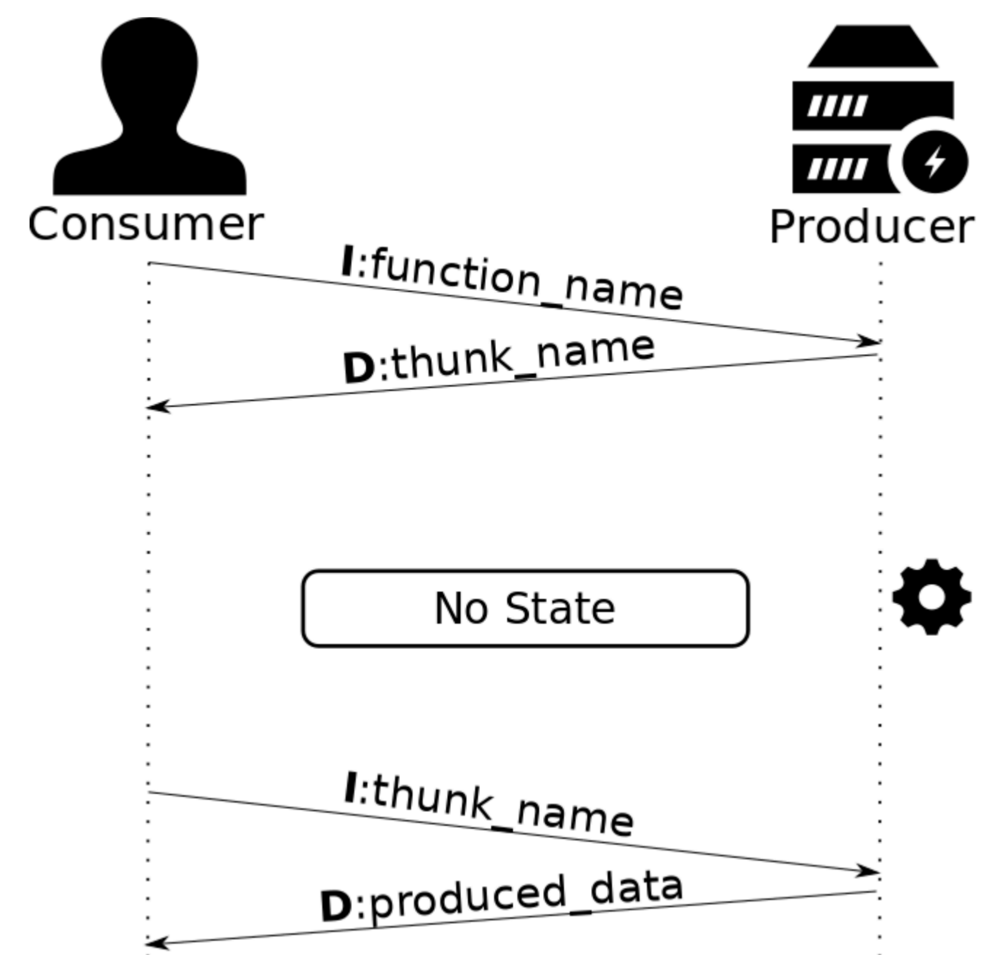


Thunk Names

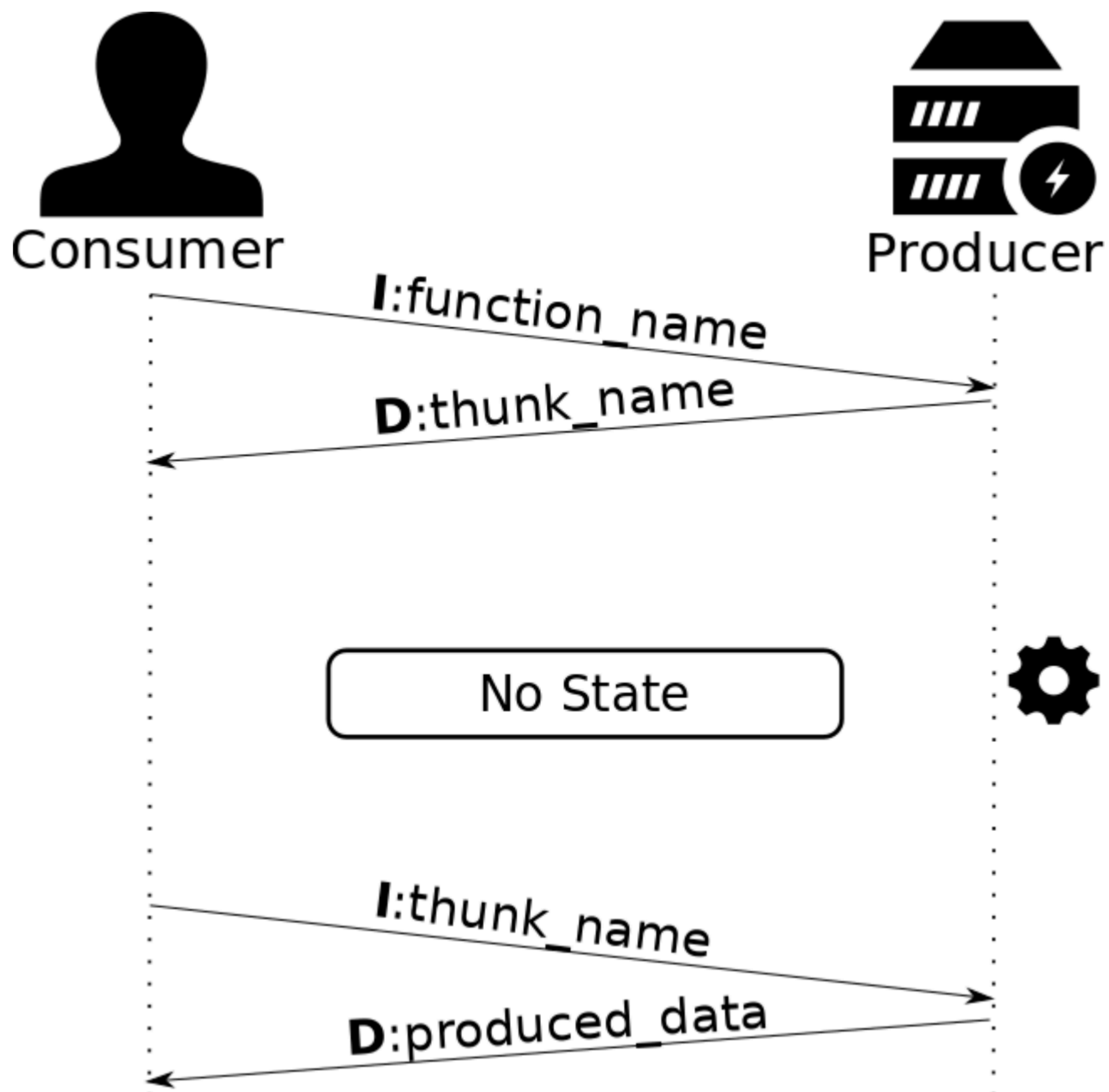


Thunks and Short/Long-Lasting Computations

- Duration of computations sometime unpredictable
- Server could estimate duration in I1 DATA message (when referring client to thunk)
- For short computation (or for returning pre-computed results) one could consider return the results directly in the I2 DATA message.
- Sometimes computations can take longer than expected
    - App-layer NACK messages?
        - Telling client to “call again later”
        - Have to avoid interference with caching
        - Currently not specified in draft



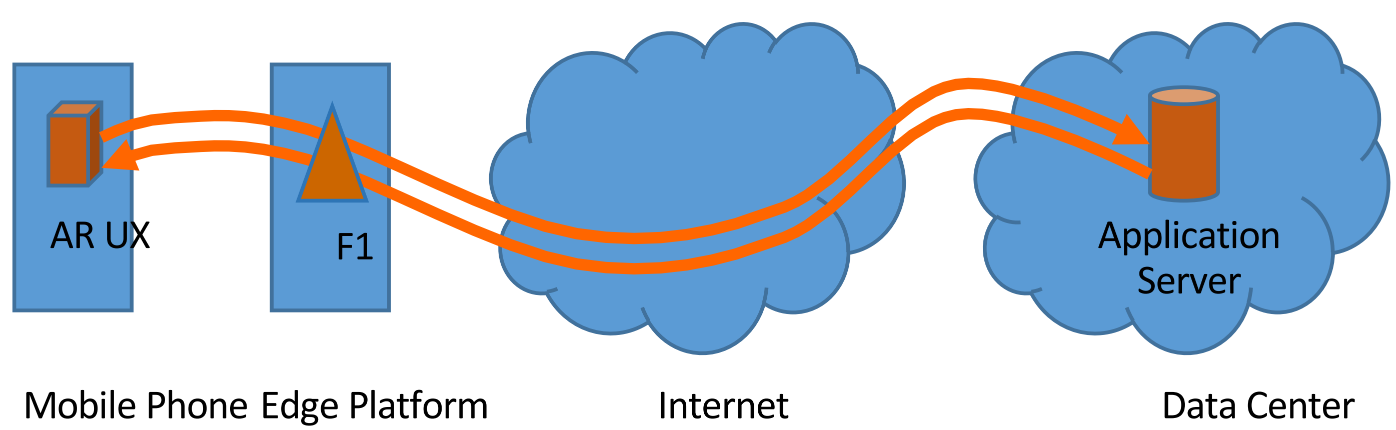
Referentially Transparent Functions and Caching



- INTEREST for function name results in DATA message that contains thunk name
  - Could be cacheable and re-used (if DATA message not encrypted)
  - Should not be done if authorization is required
- INTEREST for thunk name results in DATA message with computation result
  - Could be cacheable
  - However: thunk name specifies individual compute instance
- Forwarding hints for linking I1 function names to produced data?

RICE ICNRG Draft

- Captures protocol and node behavior specification parts from (longer) RICE paper (see ACM ICN-2018)
  - Intended as basis for interoperable RICE implementations (needs some more work)
- Intended as a basis for “everything NFN”
  - Also function chaining, distributed computing, distributed data structures
  - Cf. “Compute-First Networking” (COIN meeting on Friday)



Future Work

- Explore referentially transparent functions and caching more
- Understand interaction with other ICN extensions (e.g., MAP-ME, KITE)
- RICE as underlay for some existing distributed computing frameworks
- More experiments