RICE: Remote Method Invocation in ICN

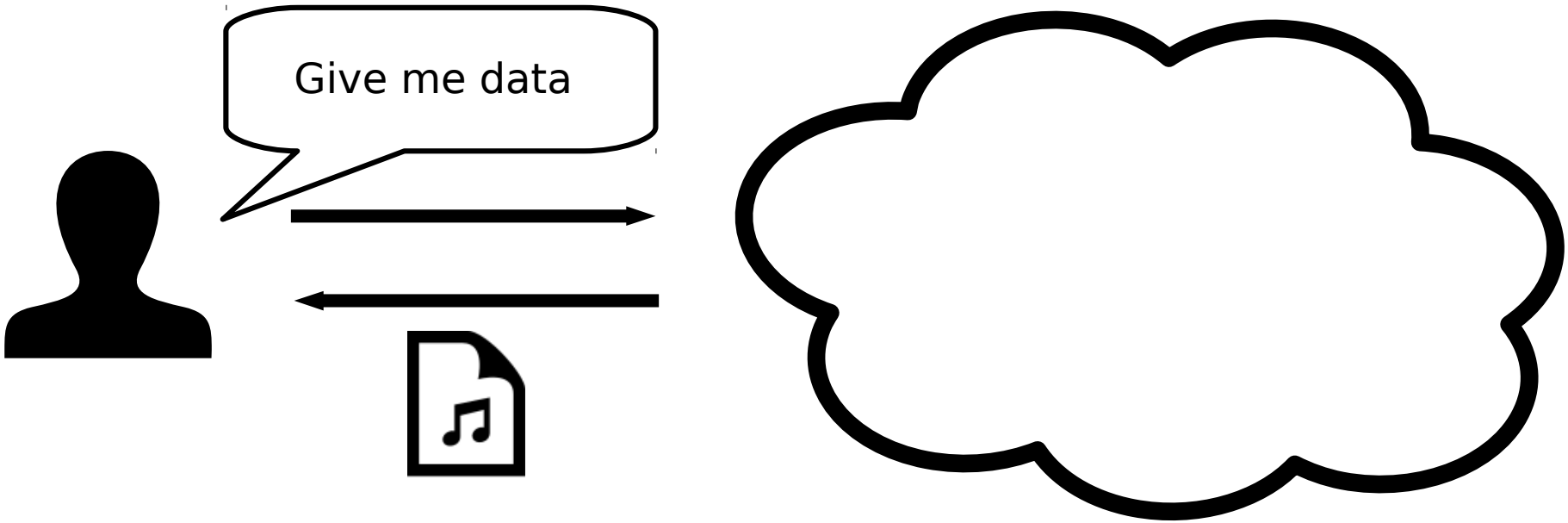
Michał Król, Karim Habak, David Oran, Dirk Kutscher, Ioannis Psaras

It’s NOT about

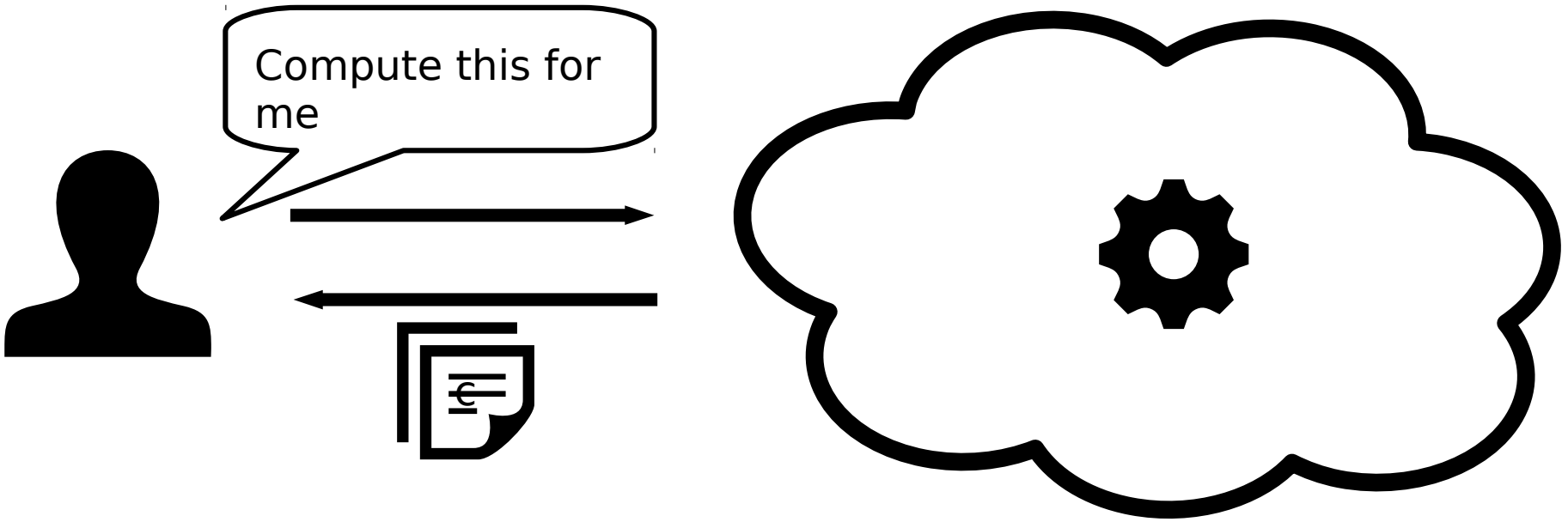
- ~~RPC, CORBA, Java RMI~~
- ~~Tightly-coupled system from (not so) long time ago~~


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

Static data retrieval



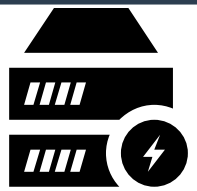
In-network computation



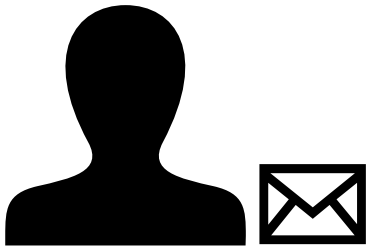
In-network computation

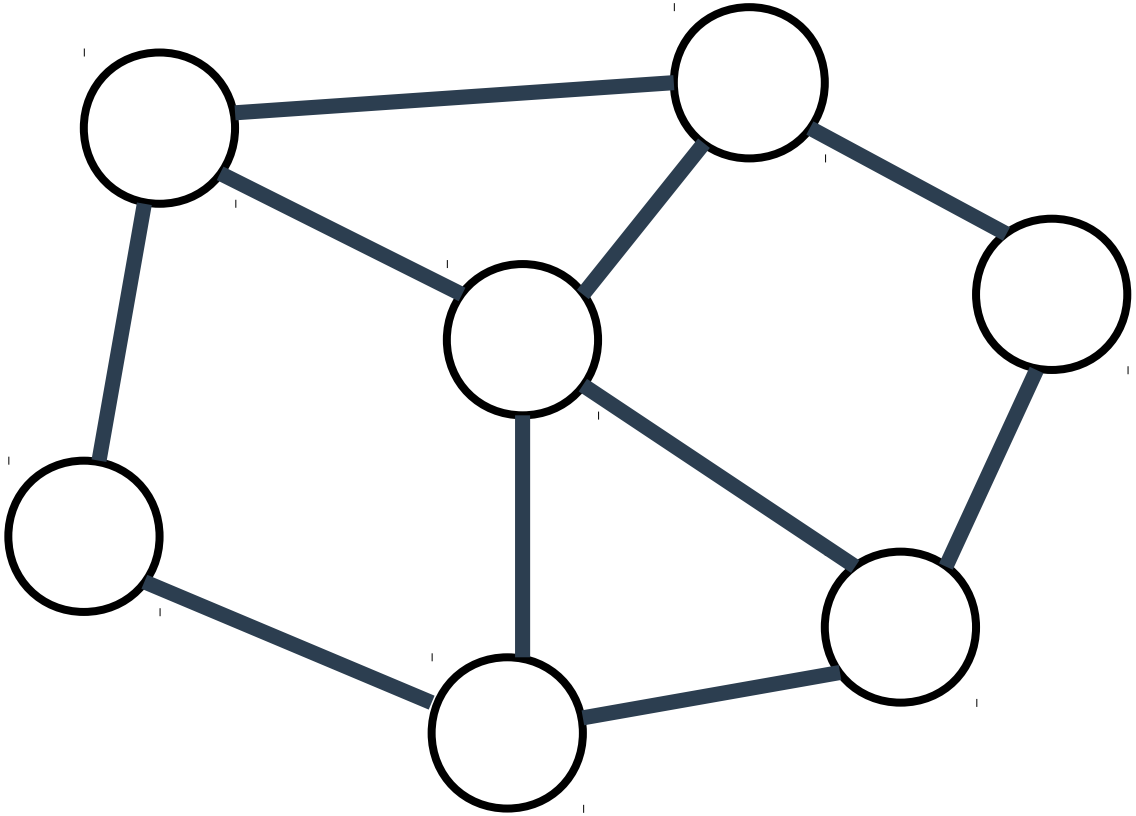
- **Multiple use-cases**
  - edge/fog computing, IoT, VR/AR
- **Multiple existing frameworks: NFN, NFaaS, CCNxServe, SCN, NextServe**
- **Migrate functions where they’re needed the most**
- **Populate FIB tables with routes to services**

Anycast

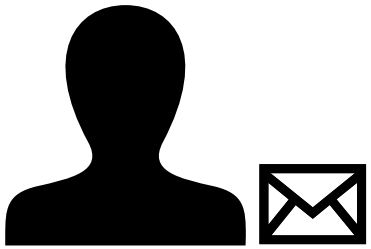


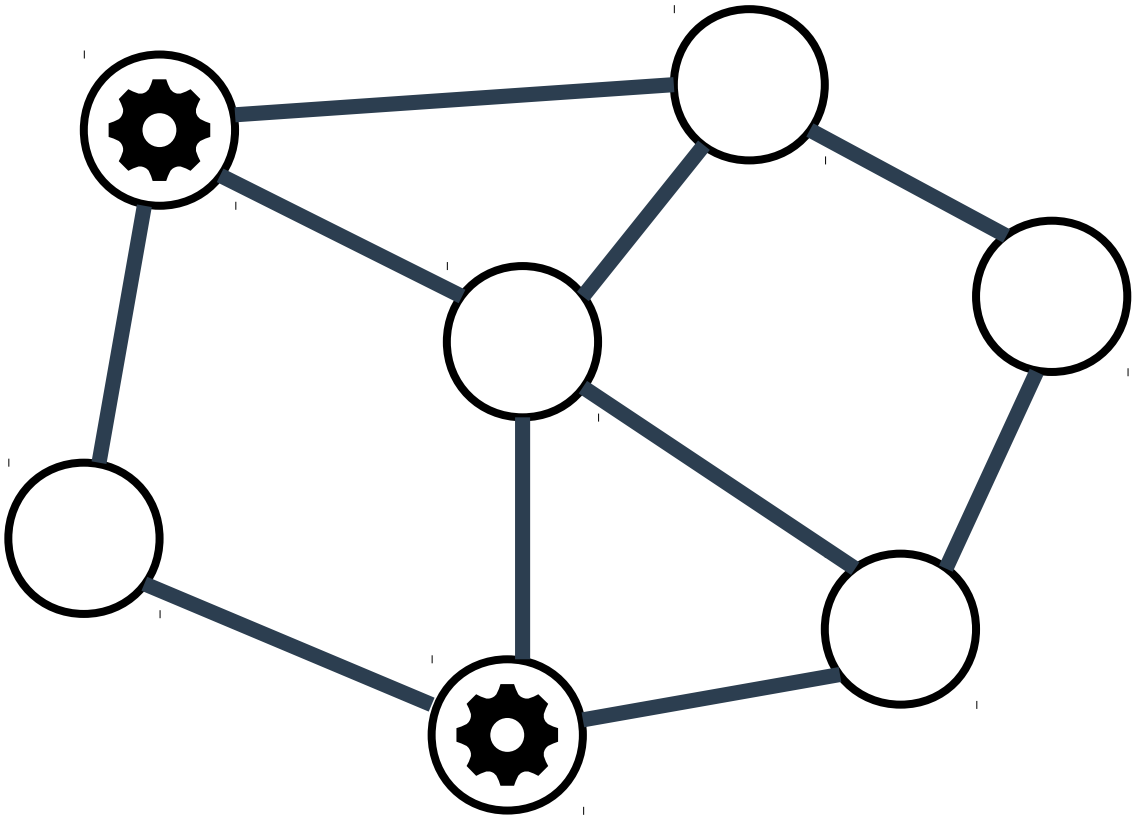
No need for a
DNS/SDN server



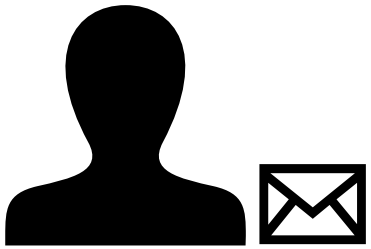


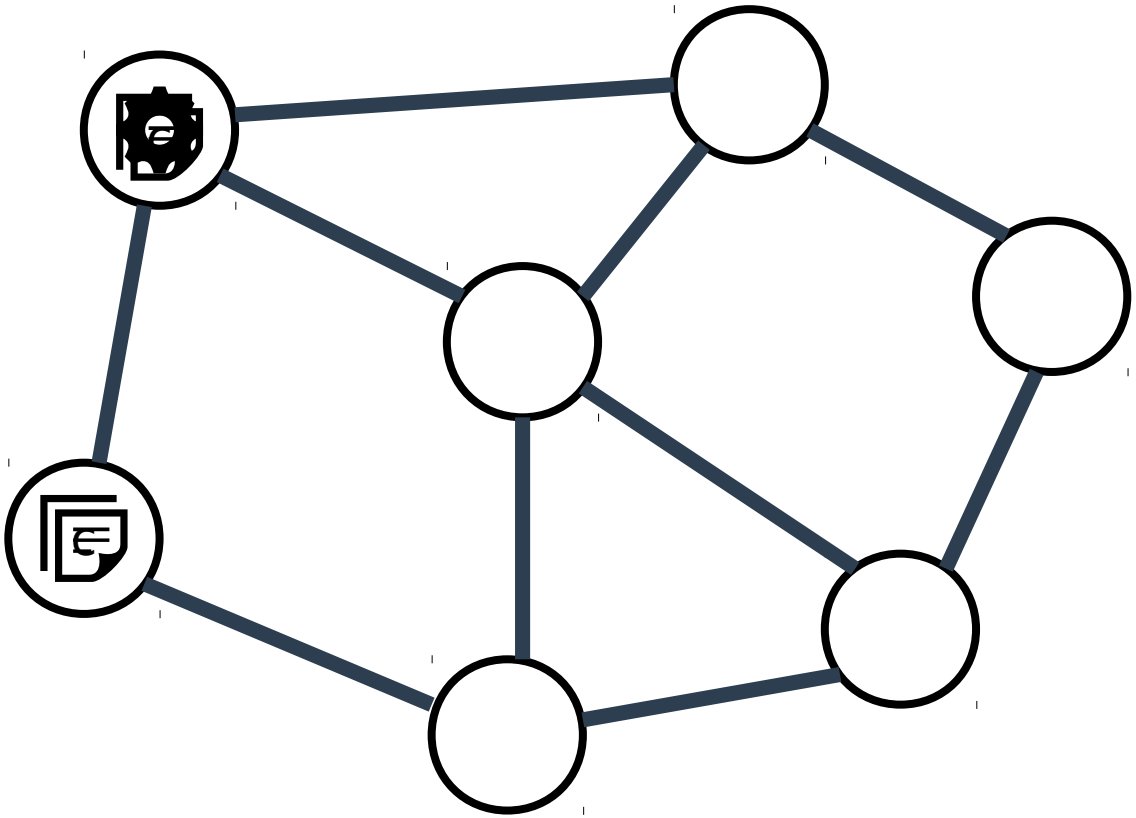
Load control





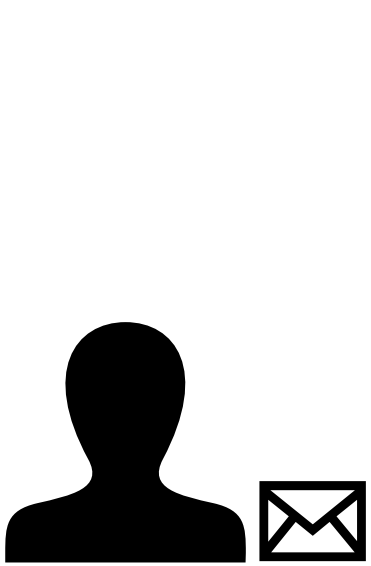
Result caching

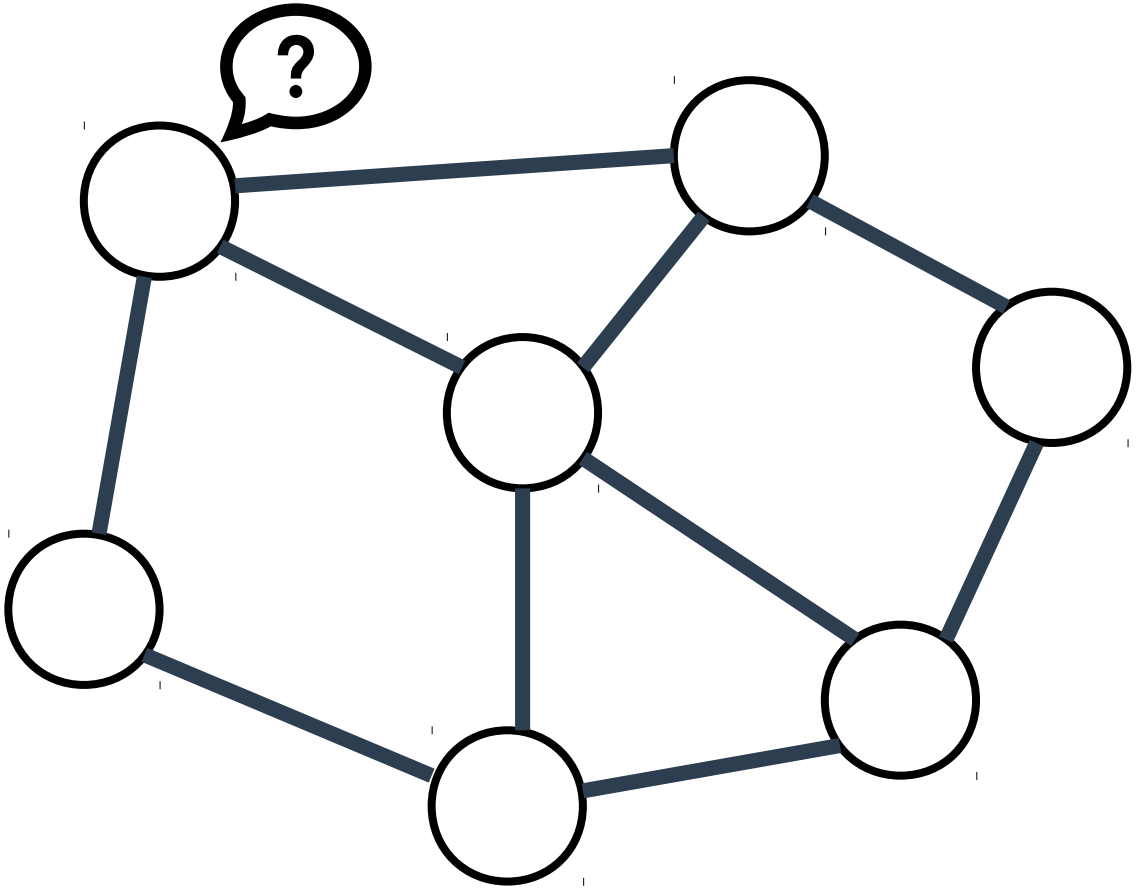




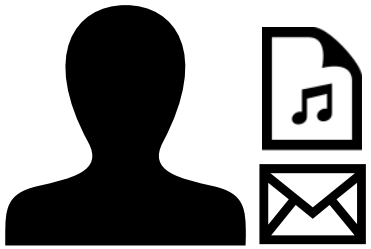
Issues

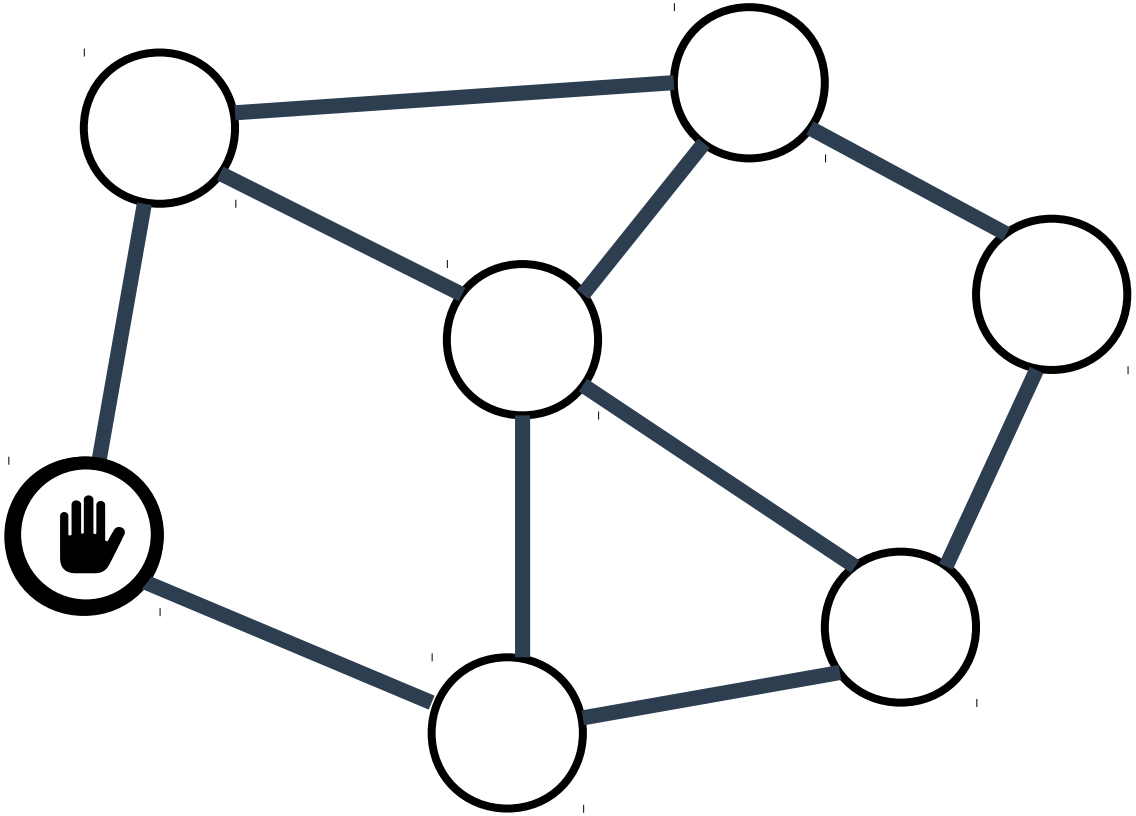
Client Authorization



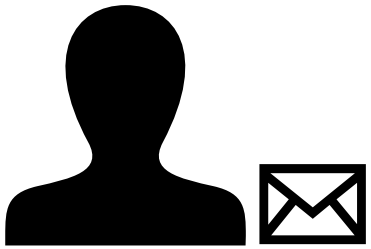


Large Parameter Passing

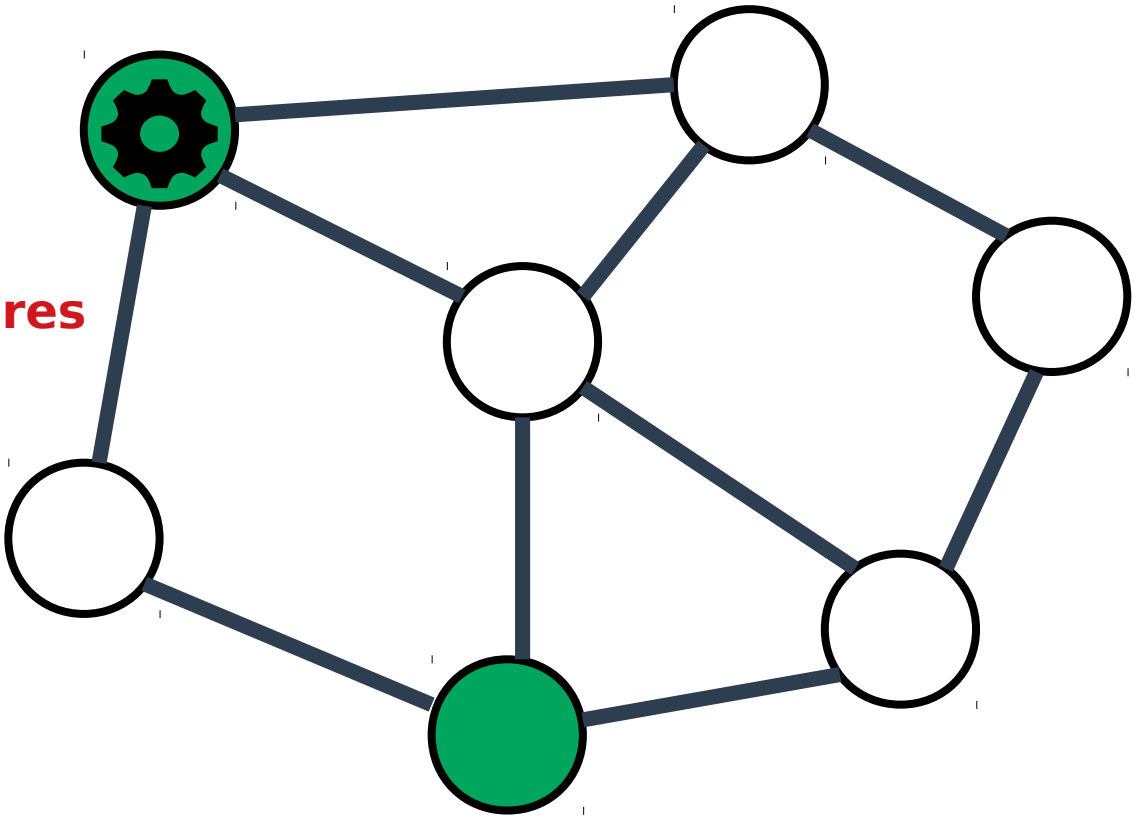




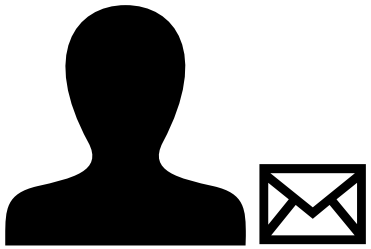
Accommodating non-trivial computations

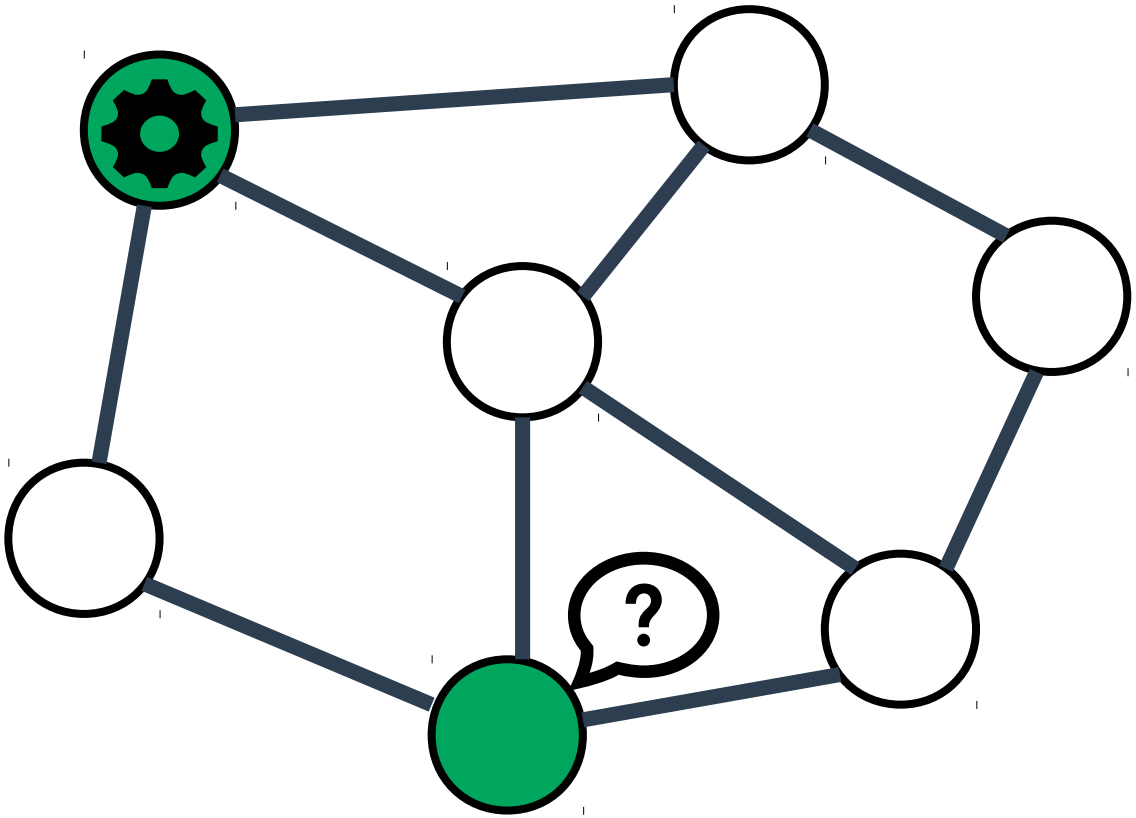


PIT expires



Accommodating non-trivial computations

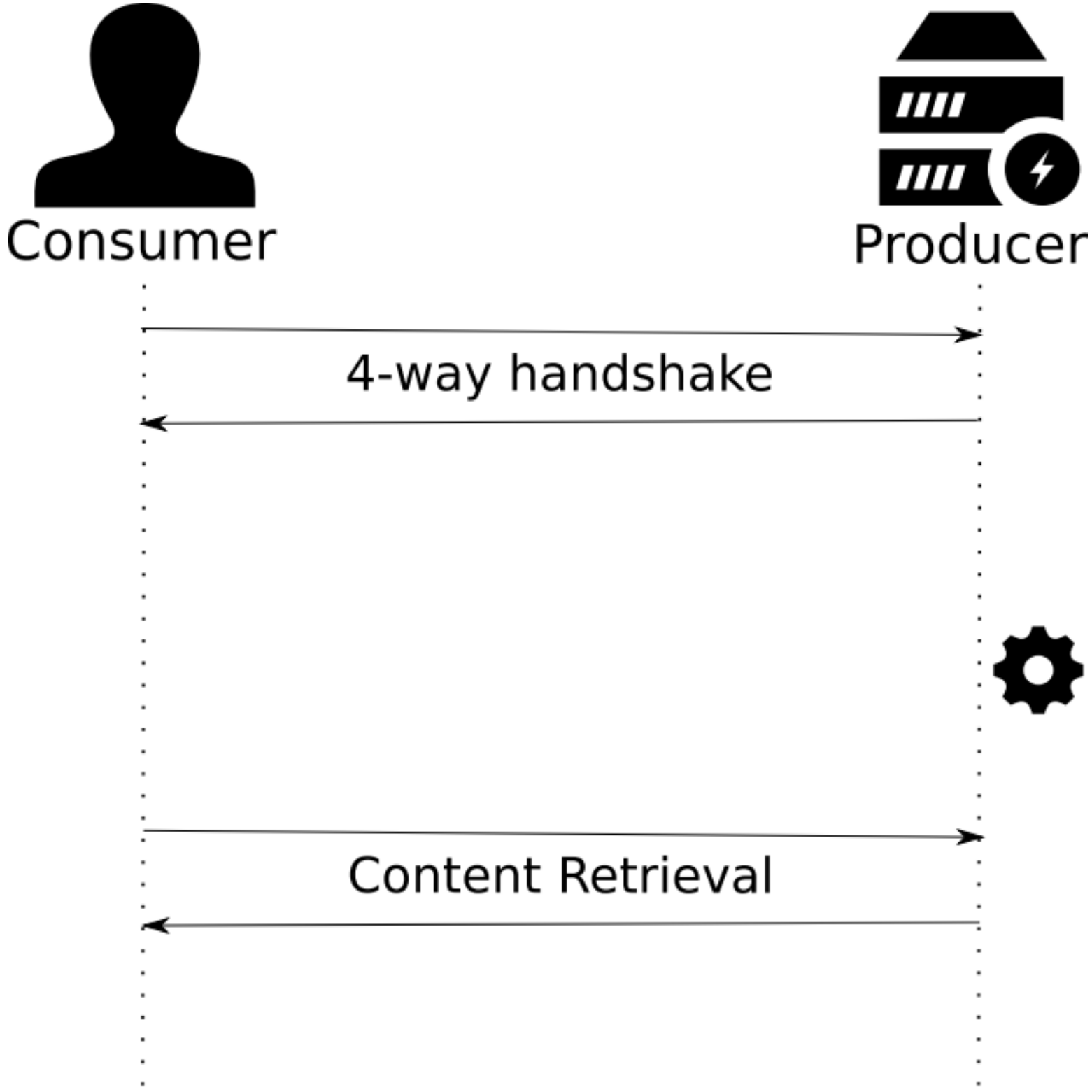




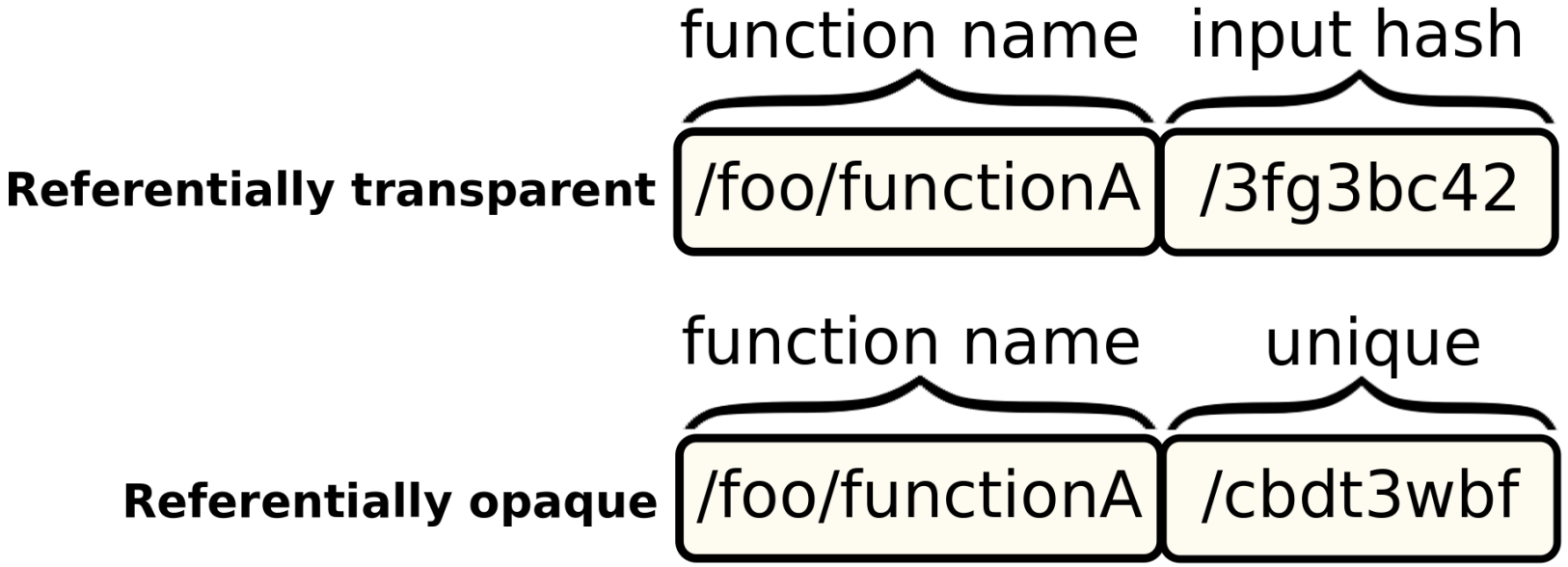
RICE Design

Design Goals

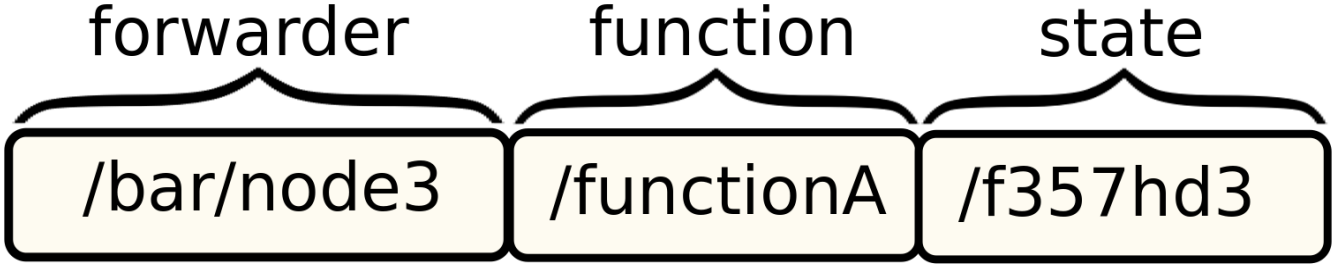
- **Consumer authentication and authorization**
- **Large parameter passing**
- **Accommodating non-trivial computations**
- **Allow result caching**
- **Adhere to ICN principles**
  - Pull model
  - Avoid revealing permanent client identifiers
  - Support client mobility
- **Make minimal changes to ICN protocols and forwarder behaviour**


Consumer
Producer
4-way handshake
Content Retrieval

Naming



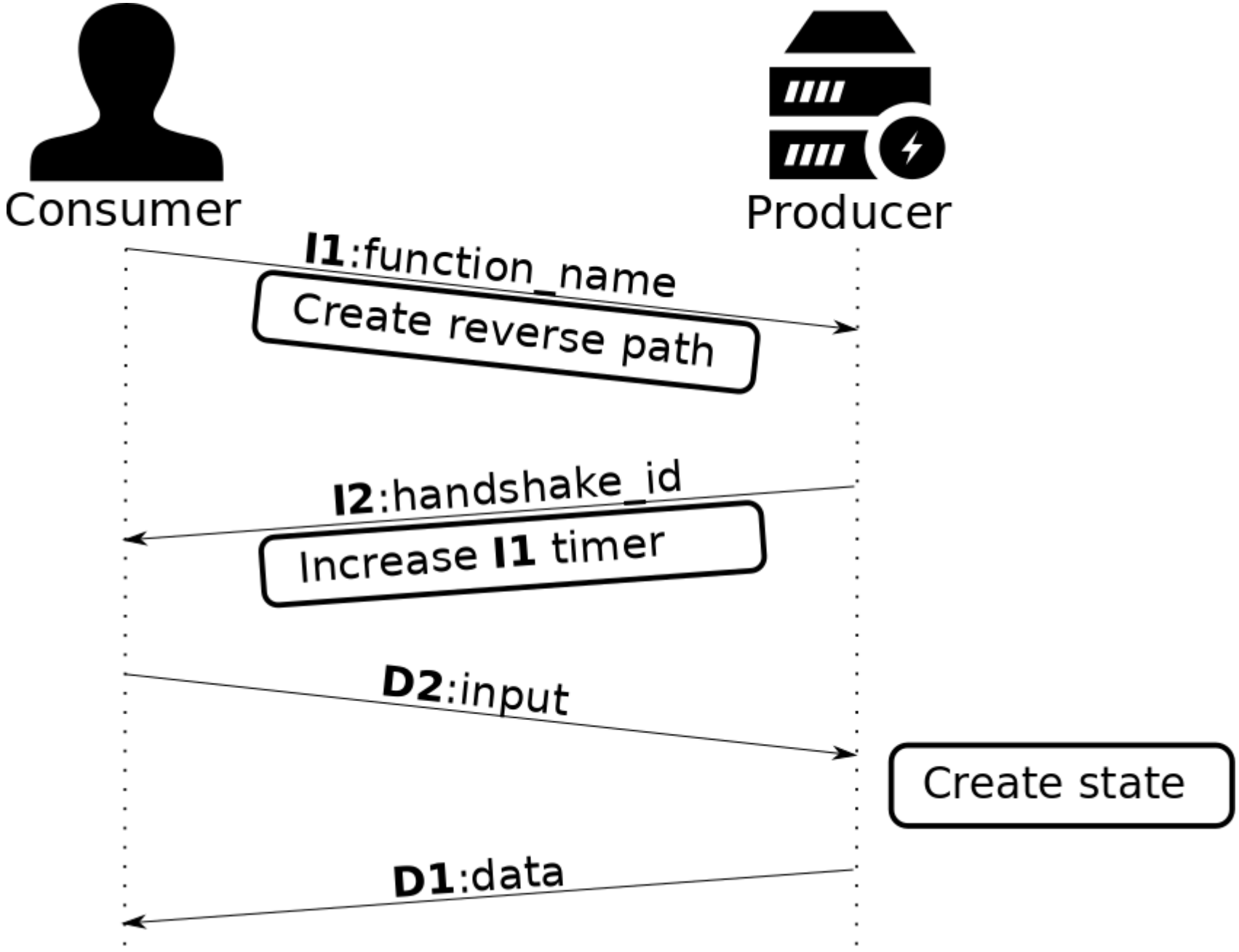
Thunk Names



4-way Handshake

- **Enable 2-way communication between producers and consumers**
- **Shared Secret Derivation**
- **Client Authentication**
- **Large Input Parameters Submission**

4-way Handshake



Dynamic Content Retrieval

Network and Application Timescale

- **PIT entries use timeouts**
- **When requesting static content Interest Satisfaction Time equals RTT**
- **Generating dynamic content adds a delay that is unknown to the network**
- **PIT entries can expiry before returning the results**

Network Timescale

- Fast recovery
- No assumption on execution time

- Huge overhead
- Challenging bandwidth allocation

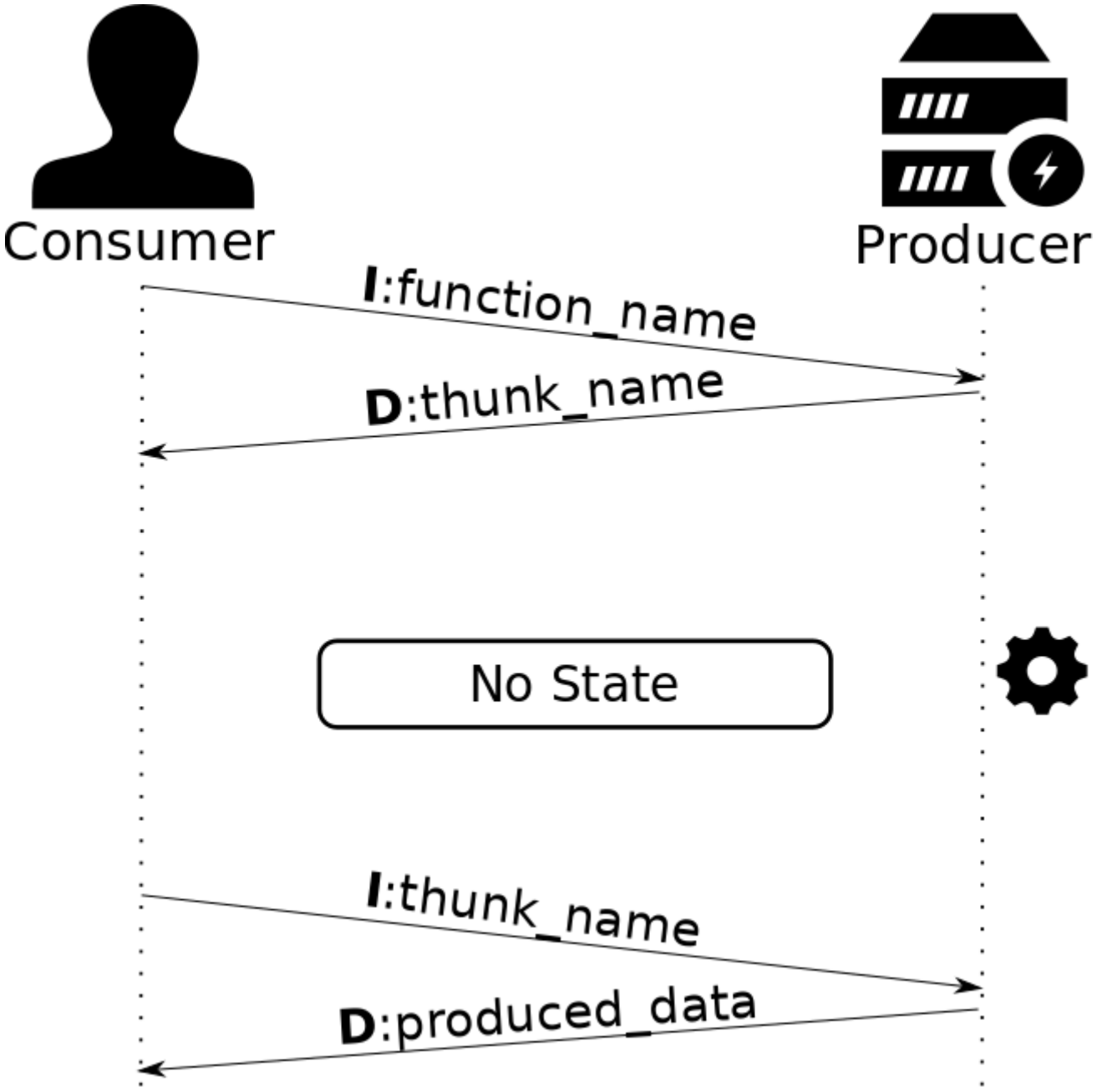
Application Timescale

- Low overhead
- Regular Bandwidth allocation

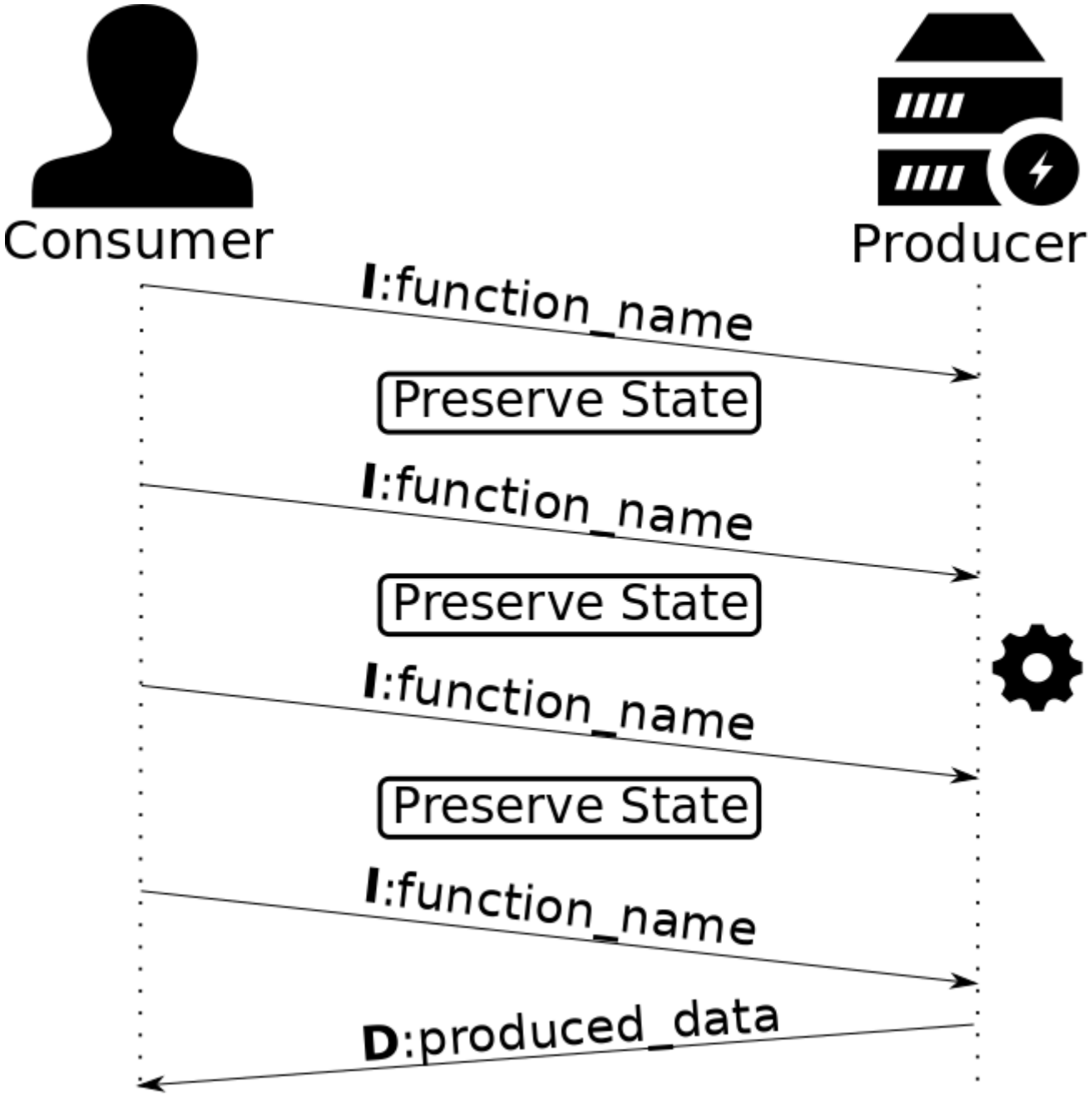
- Slow Recovery
- Requires a lot of knowledge

We want to decouple application timescale from network timescale

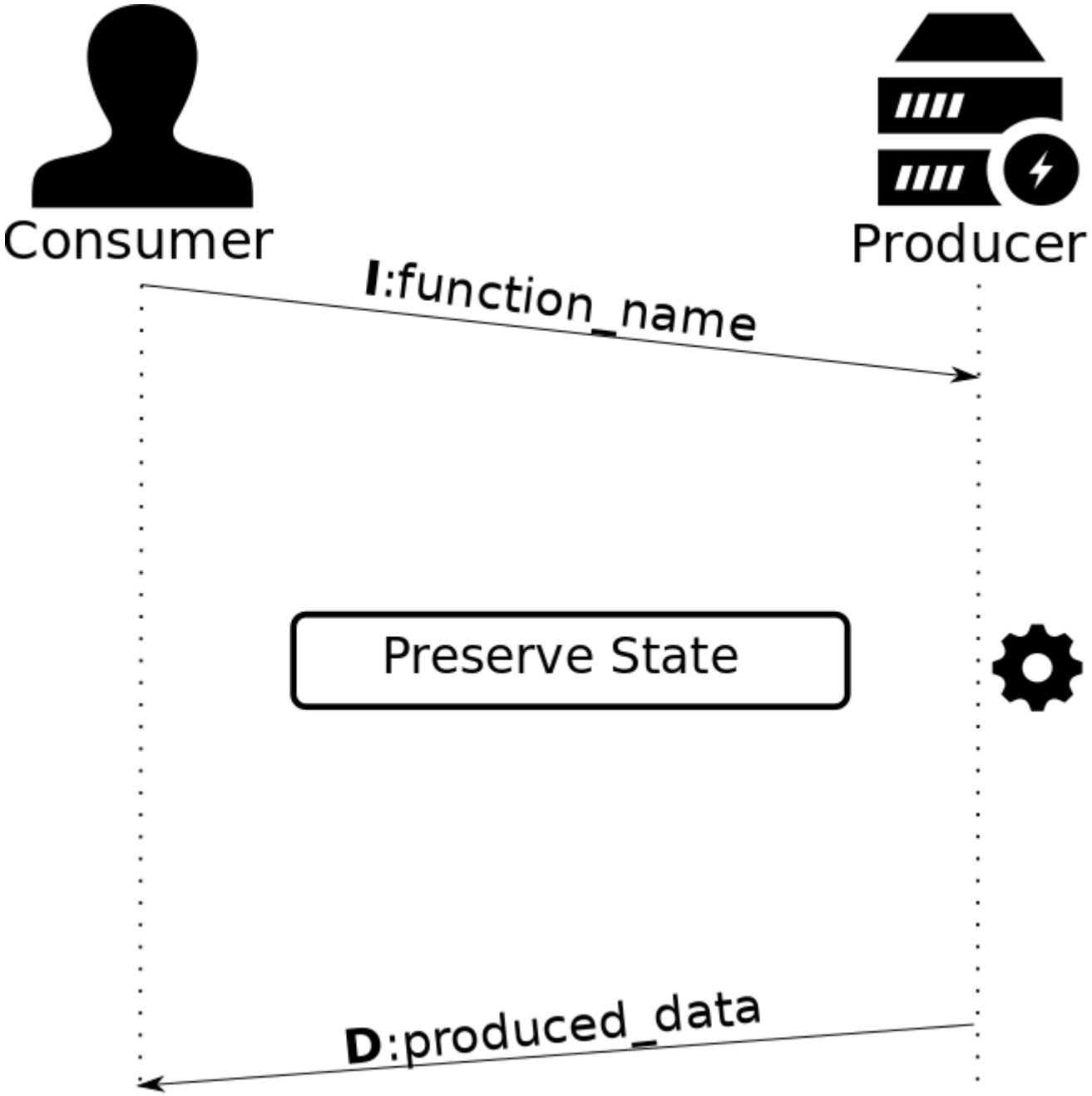
Thunks


Consumer
Producer
I:function_name
D:thunk_name
No State
I:thunk_name
D:produced_data

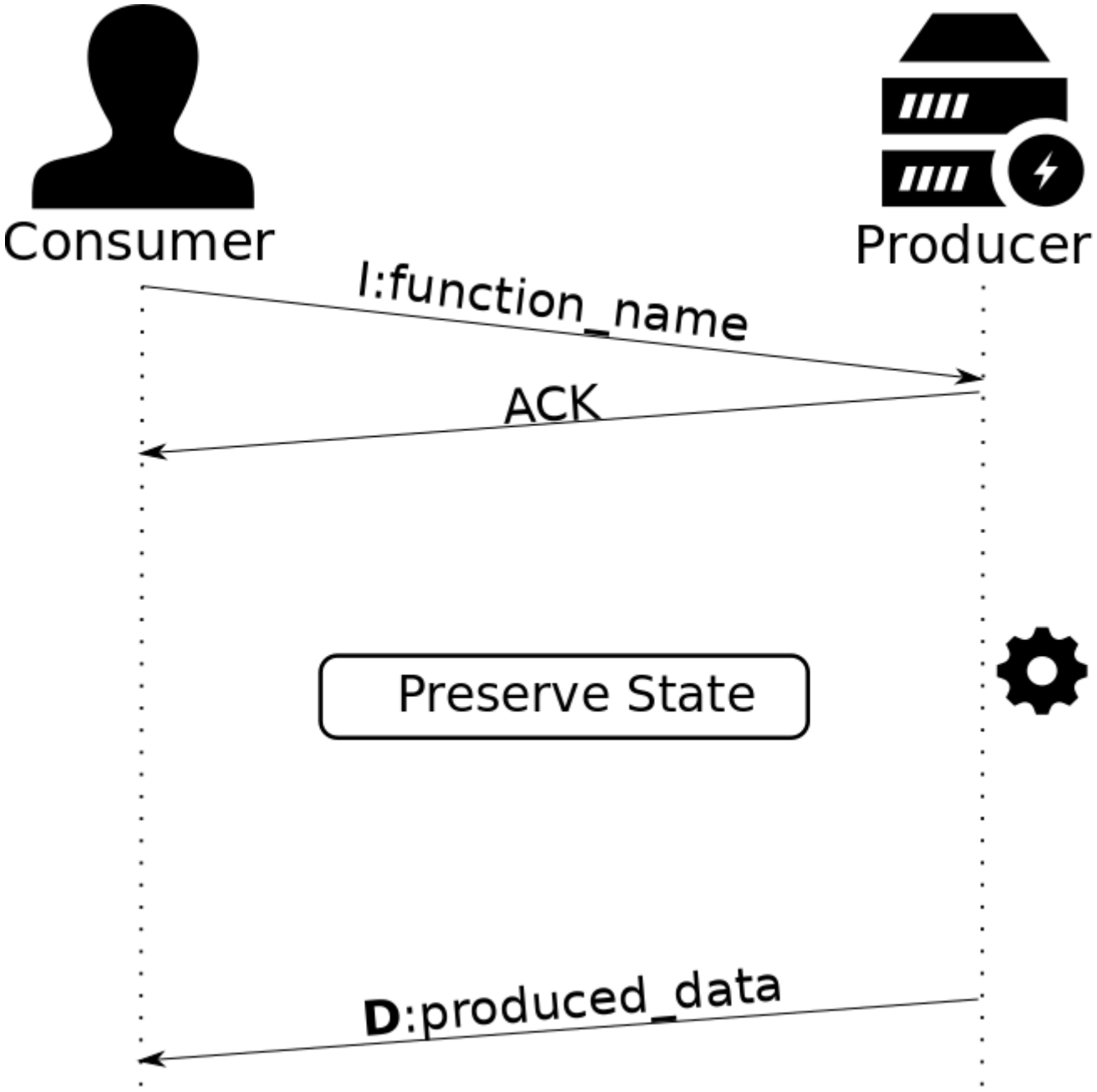
Network Timescale



Application Timescale

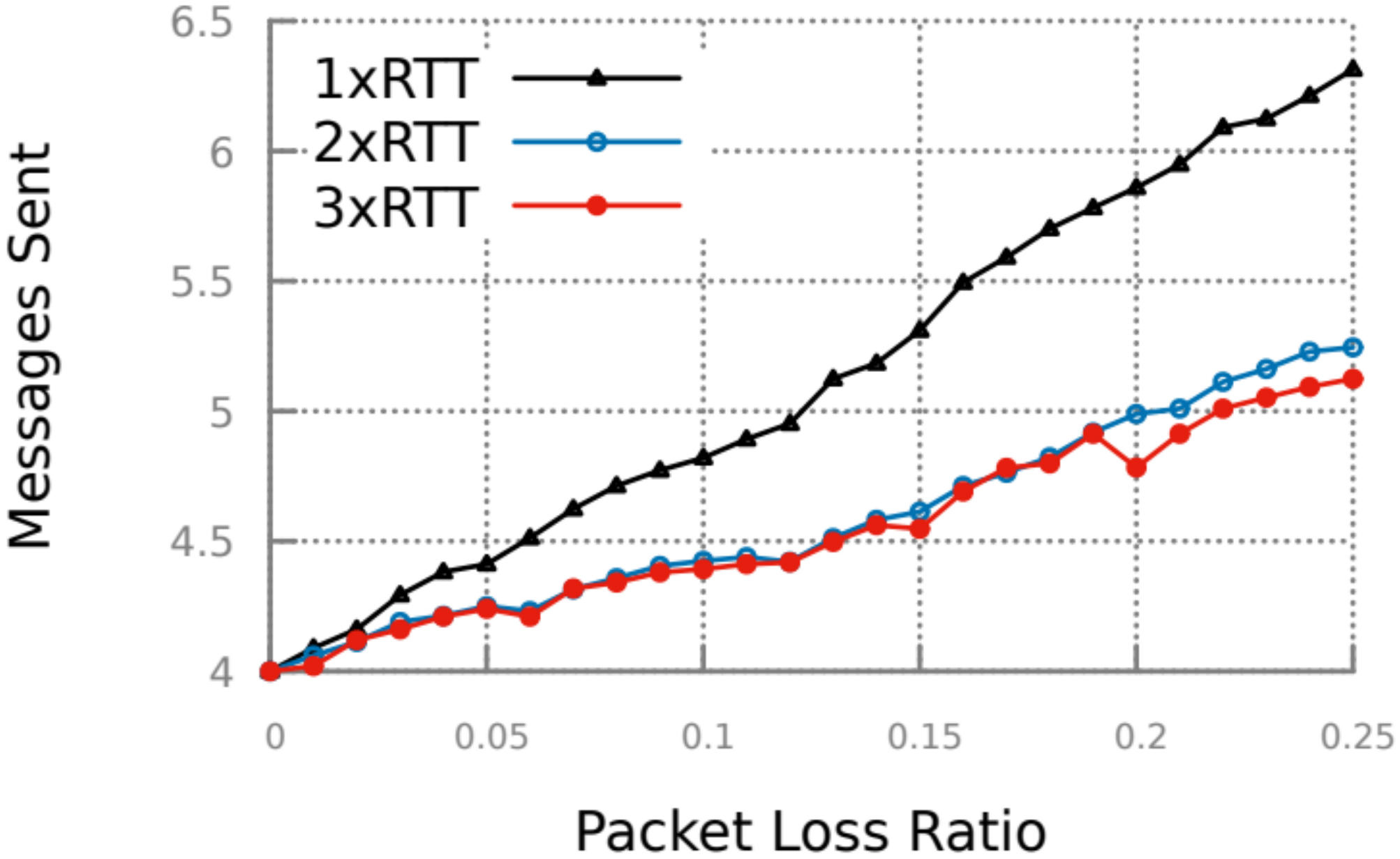


Acknowledgements

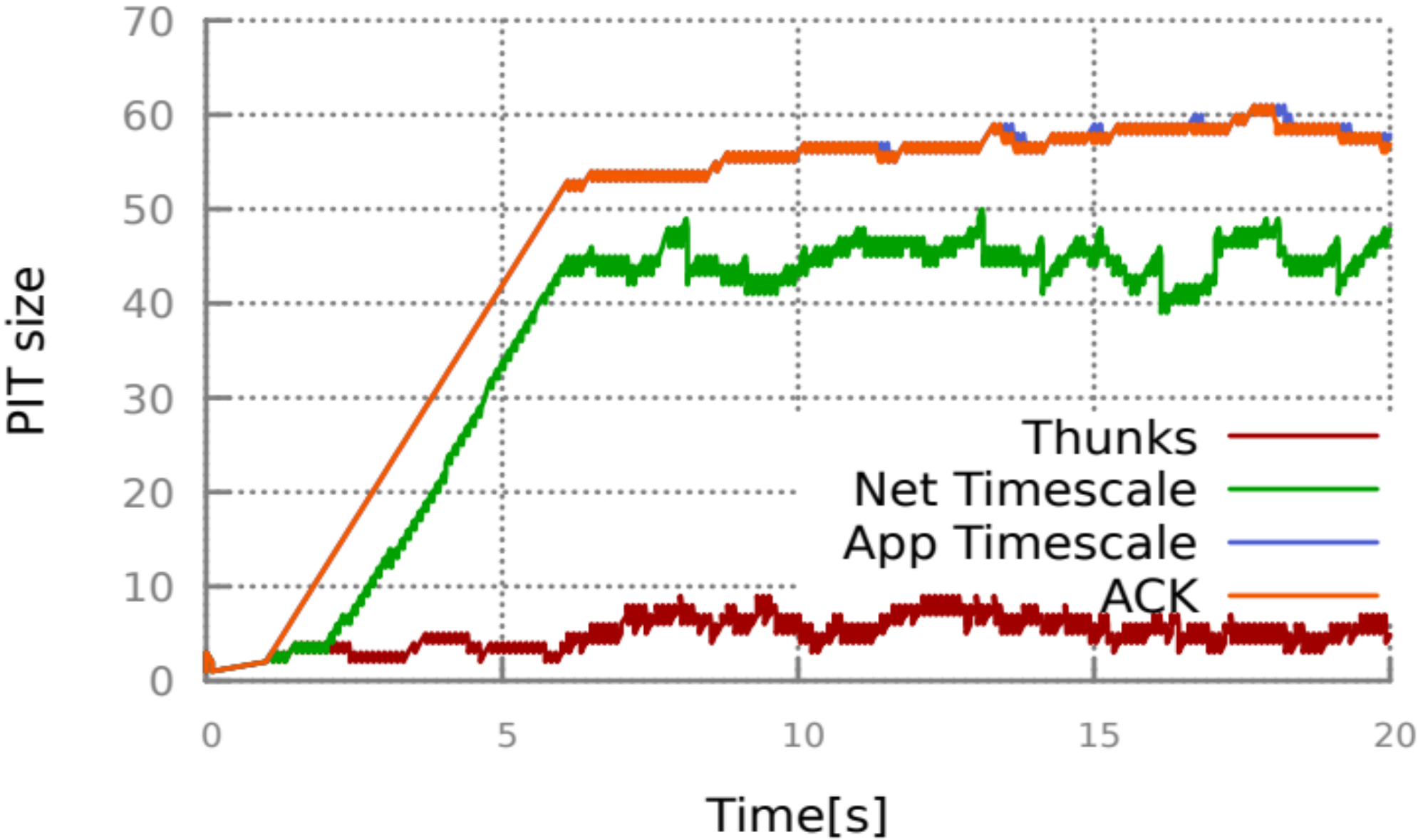


Results

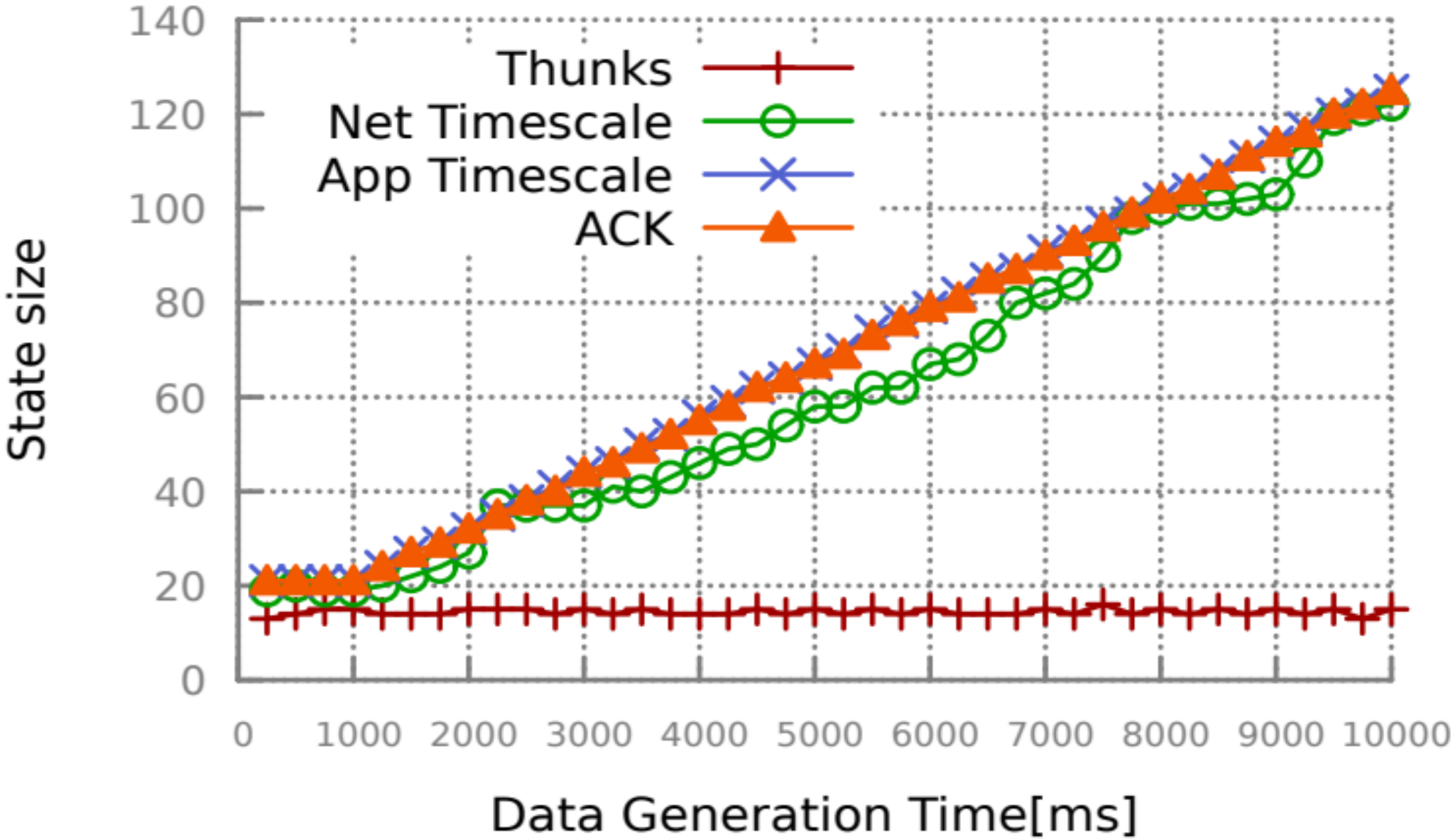
Handshake


1xRTT
2xRTT
3xRTT
Messages Sent
6.5
6
5.5
5
4.5
4
0
0.05
0.1
0.15
0.2
0.25
Packet Loss Ratio

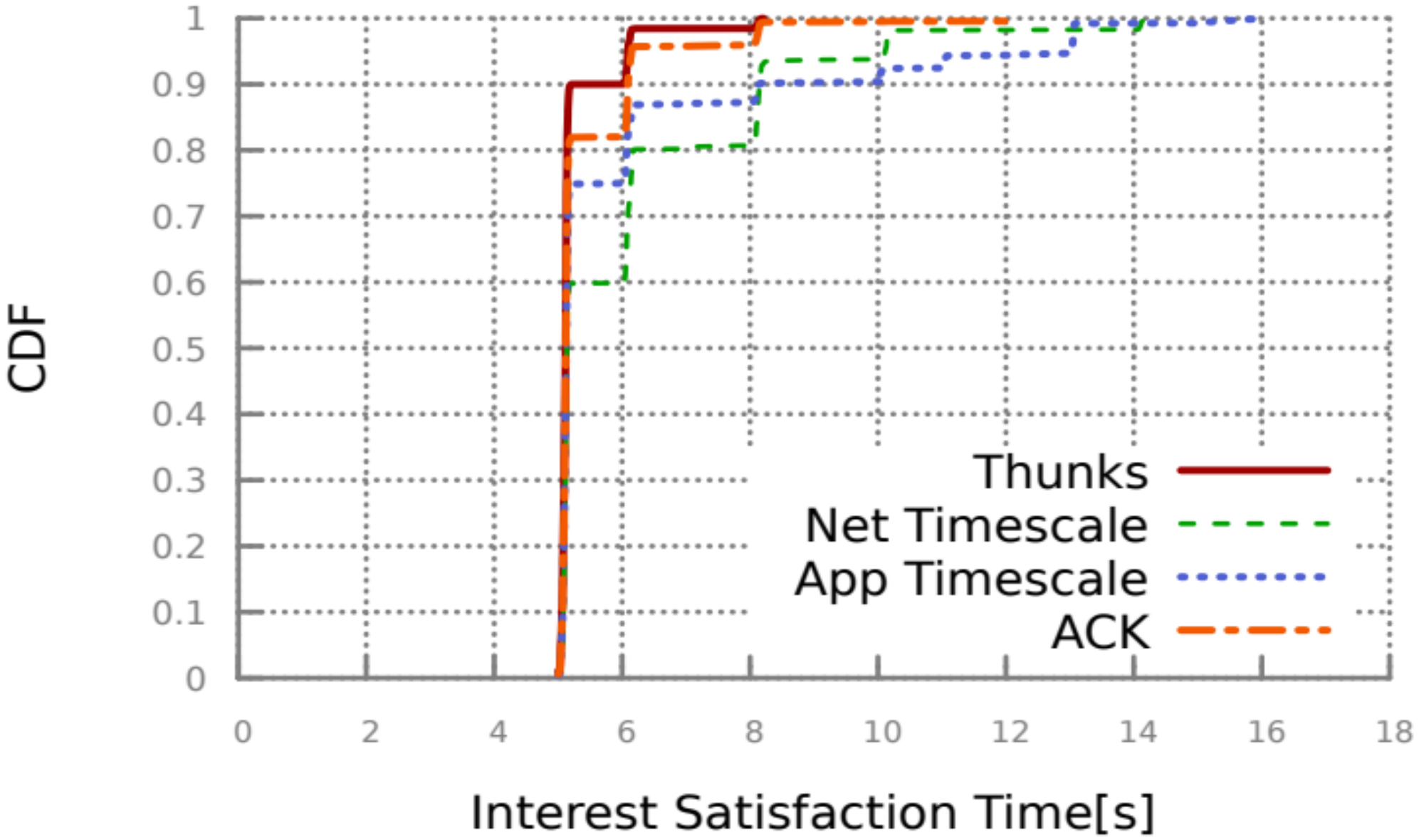
Thunks


70
60
50
40
30
20
10
0
PIT size
0
5
10
15
20
Time[s]
Thunks
Net Timescale
App Timescale
ACK

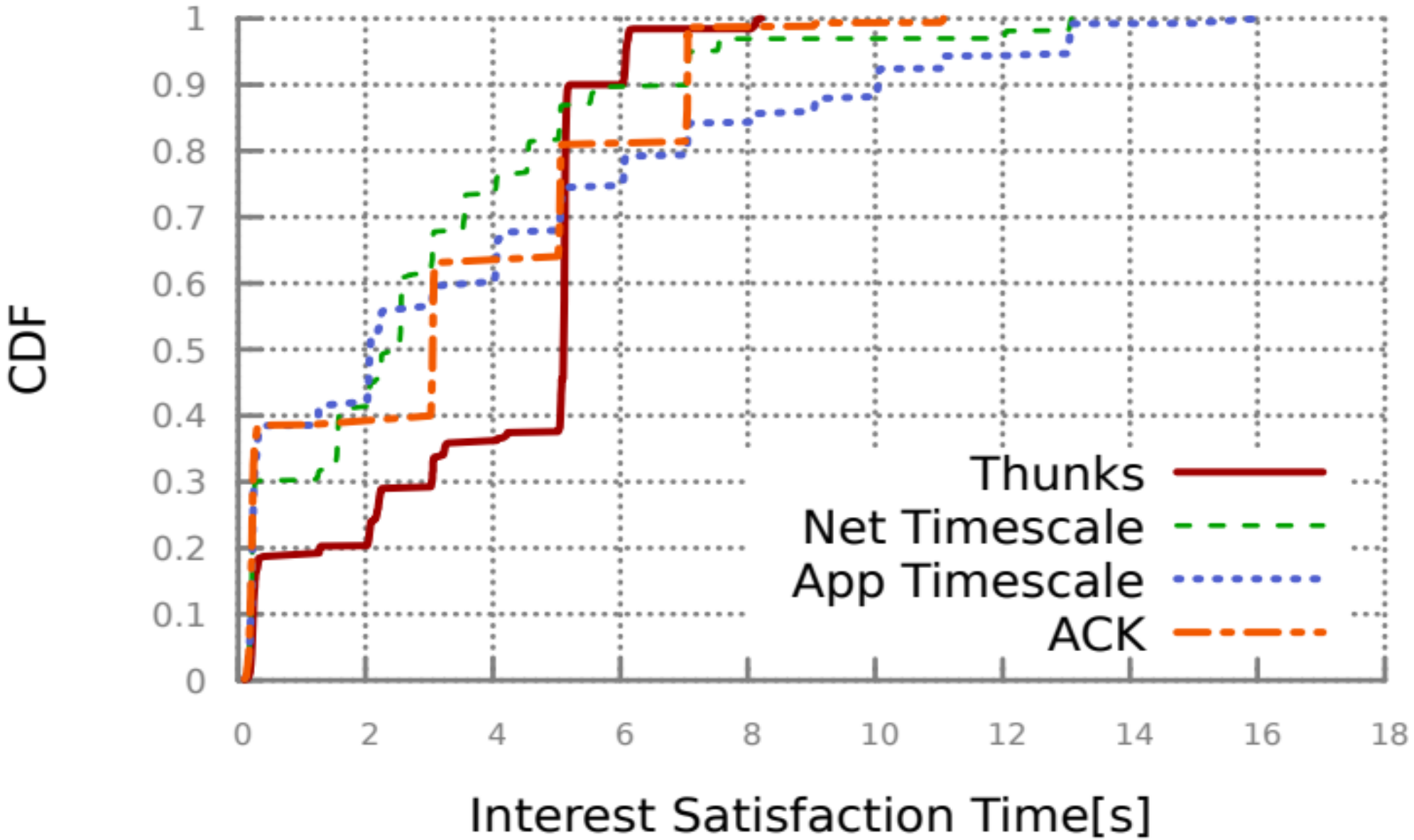
Thunks


Thunks
Net Timescale
App Timescale
ACK
State size
Data Generation Time[ms]
0
20
40
60
80
100
120
140
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000

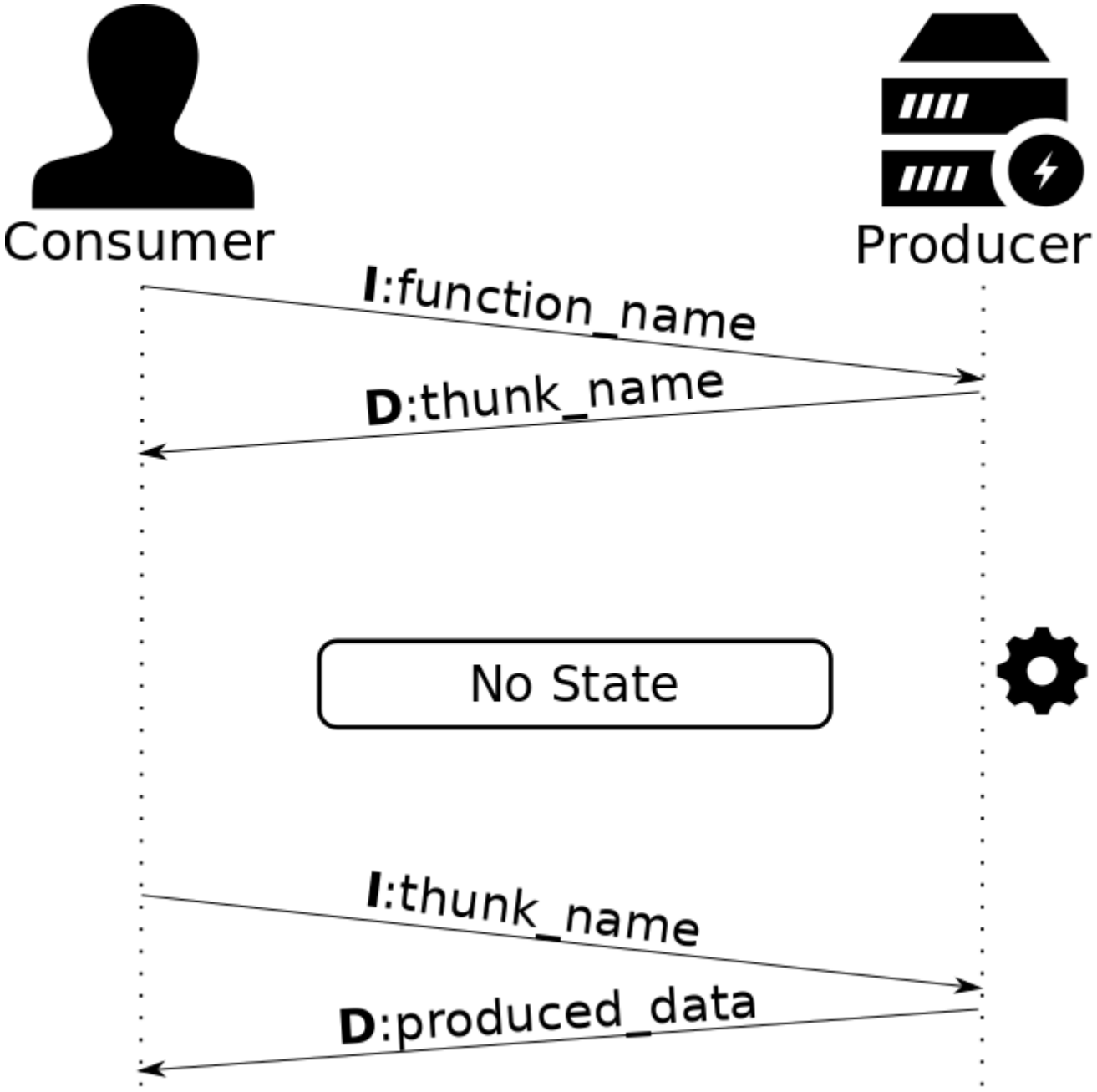
Referentially opaque function



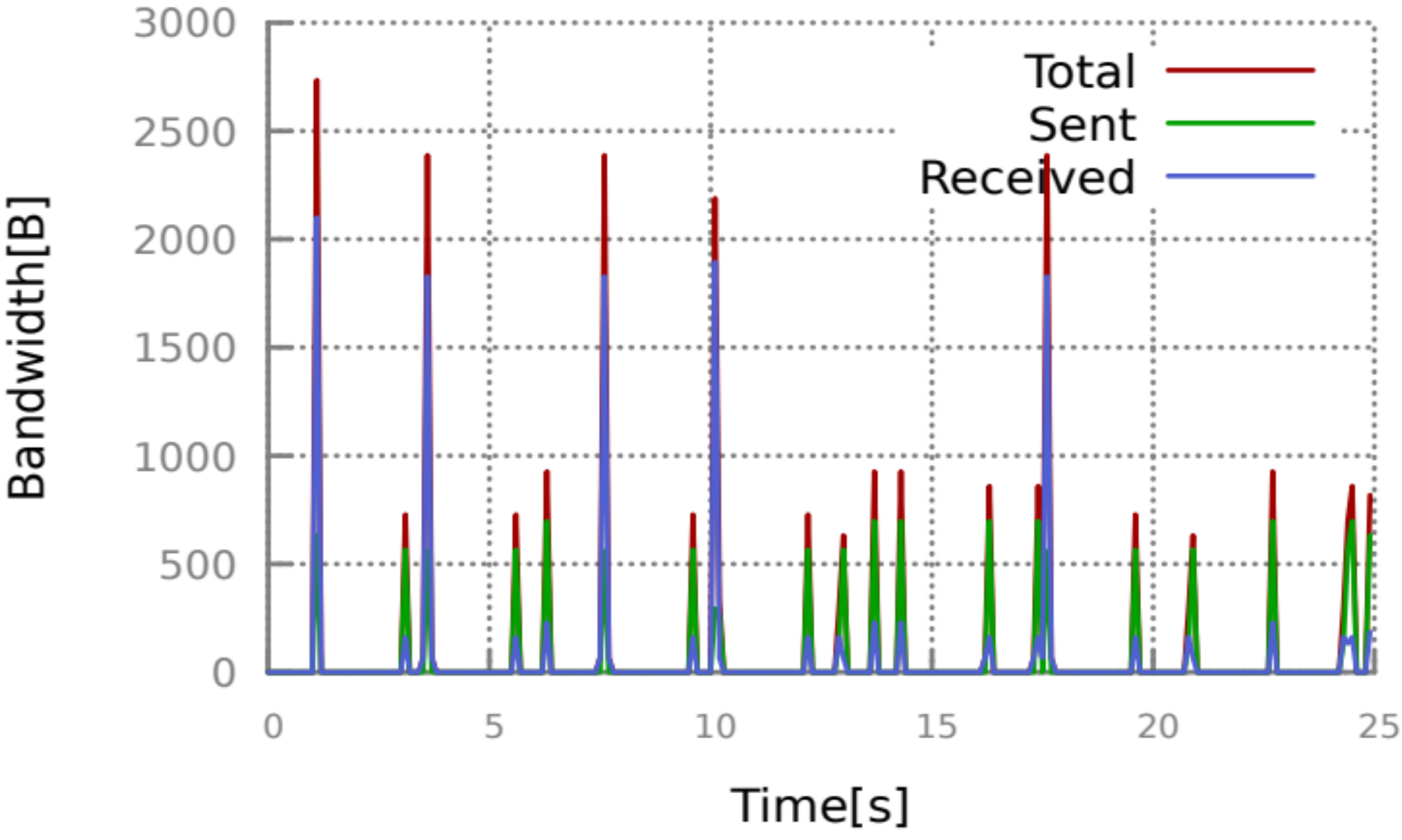
Referentially transparent function



Thunks


Consumer
Producer
I:function_name
D:thunk_name
No State
I:thunk_name
D:produced_data

Prototype


Total
Sent
Received
Bandwidth[B]
3000
2500
2000
1500
1000
500
0
0
5
10
15
20
25
Time[s]

Limitations

- **Thunks require accurate computation time estimation**
  - Overestimation increases the delay
  - Underestimation increases overhead
- **Referentially transparent functions can be cached under different names**
  - Can be solved by using forwarding hints

Conclusion and Future Work

- **Client authentication, large parameter passing, accommodating non-trivial computation using a 4-way handshake + thunks**
- **Generic API for function invocation**
- **RICE can be a basis for many NFN-inspired systems.**
- **Prototype and demo**

In the future

- **Integration with routing hints and NACKs**
- **Implement highly-scalable server**
- **Developing higher-layer abstractions on top of RICE.**
  - pushing data (for custodial transfer, re-publishing, storing)
  - support for more pervasive in-network computing
  - rethinking and re-engineering existing applications, especially web

Thank you