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

Give me data
In-network computation

Compute this for me
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
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
Naming

Referentially transparent

```
function name: /foo/functionA
input hash: /3fg3bc42
```

Referentially opaque

```
function name: /foo/functionA
unique: /cbdt3wbf
```

Thunk Names

```
forwarder: /bar/node3
function: /functionA
state: /f357hd3
```
4-way Handshake

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

Consumer

Producer

\[ I_1: \text{function}_\text{name} \]
Create reverse path

\[ I_2: \text{handshake}_\text{id} \]
Increase \( I_1 \) timer

\[ D_2: \text{input} \]
Create state

\[ D_1: \text{data} \]
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 expire before returning the results
<table>
<thead>
<tr>
<th>Network Timescale</th>
<th>Application Timescale</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fast recovery</td>
<td>Low overhead</td>
</tr>
<tr>
<td>No assumption on execution time</td>
<td>Regular Bandwidth allocation</td>
</tr>
<tr>
<td>Huge overhead</td>
<td>Slow Recovery</td>
</tr>
<tr>
<td>Challenging bandwidth allocation</td>
<td>Requires a lot of knowledge</td>
</tr>
</tbody>
</table>

We want to decouple application timescale from network timescale
Network Timescale

Consumer

Producer

I:function_name
Preserve State
I:function_name
Preserve State
I:function_name
Preserve State
I:function_name
D:produced_data
Acknowledgements
Results
Handshake

![Graph showing the relationship between packet loss ratio and messages sent for 1xRTT, 2xRTT, and 3xRTT.]
Thunks
Thunks

![Graph showing state size over data generation time]

- Thunks
- Net Timescale
- App Timescale
- ACK

Data Generation Time [ms]

State size
Referentially opaque function
Referentially transparent function
Thunks

Consumer  ------  Producer

I: function_name  
D: thunk_name

No State

I: thunk_name  
D: produced_data
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