ICNRG
Internet-Draft
Intended status: Experimental
Expires: April 4, 2019

M. Krol
University College London
K. Habak
Georgia Institute of Technology
D. Oran
Network Systems Research & Design
D. Kutscher
Huawei
I. Psaras
University College London

October 01, 2018

Remote Method Invocation in ICN
draft-kutscher-icnrg-rice-00

Abstract

Information Centric Networking has been proposed as a new network layer for the Internet, capable of encompassing the full range of networking facilities provided by the current IP architecture. In addition to the obvious content-fetching use cases which have been the subject of a large body of work, ICN has also shown promise as a substrate to effectively support remote computation, both pure functional programming (as exemplified by Named Function Networking) and more general remote invocation models such as RPC and web transactions. Providing a unified remote computation capability in ICN presents some unique challenges, among which are timer management, client authorization, and binding to state held by servers, while maintaining the advantages of ICN protocol designs like CCN and NDN. This document specifies a unified approach to remote method invocation in ICN that exploits the attractive ICN properties of name-based routing, receiver-driven flow and congestion control, flow balance, and object-oriented security while presenting a natural programming model to the application developer.

time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 4, 2019.

Table of Contents

1. Introduction

Much of today's network traffic consists of data sent for processing to the cloud and web-servers exchanging high volumes of dynamically generated content. While today's ICN networks can deal efficiently with static data delivery, they have difficulty handling service/function invocation [MOISEENKO2014]. In view of these limitations, multiple works have recently tried to extend ICN's capabilities to deal with dynamic content.

Notable among these efforts, Named Function Networking (NFN) [TSCHUDIN2014NAMED] and Named Function as a Service (NFaaS) [NFAAS] extend ICN's named data access model to a remote function invocation capability, enabling consumers to request the network to execute functions remotely. In NFN \cite{tschudin2014named}, for instance, function invocation corresponds to independent computational processes, evaluated as expressions in a functional programming model.

ICN provides several attractive benefits compared to remote invocation over current network protocol stacks (e.g. CORBA, RESTful HTTP [REST]. Name-based routing allows the network to optimise the placement of computations with automatic load distribution and failure resiliency. The built-in object-based security of ICN frees application designers from the need to craft custom solutions in the common cases where channel security alone is insufficient. Short-term caching brings latency benefits under transient error conditions and mobility events, while long-term caching can substantially reduce server load for referentially transparent computations.

There have been several approaches for integrating computation with ICN. However, when using them to realize real-world applications like web-style interactions, several additional aspects beyond the fundamental Named Function invocation concept need to be addressed:

- Consumer authentication and authorization: a producer should not blindly answer any consumer request. In basic ICN, this protection is provided by cryptographic data object integrity and encryption, i.e., only authorized consumers are able to decrypt a received data object. In a Named Function Networking environment, the computation may be an expensive operation, so just relying on encryption and performing computations without validating consumer authorization may critically impede scalability of the whole approach.

- Parameter passing: Remote function execution typically requires a set of input parameters/arguments. In dynamic web content creation for example, the volume of such parameters (in bytes) can easily surpass the volume of the actual returned data objects {{MOISEENKO2014}. Adding larger sets of parameters to Interest messages can introduce additional unsolicited traffic in ICN networks that could interfere with congestion control.

- Accommodating non-trivial computations: Unlike responding to an Interest message with a (possibly pre-generated) static Data message, constructing responses by performing general computations (that could in turn invoke further remote computations) may take relatively long. In CCN/NDN, forwarders keep Interest state for

matching received Data messages. The design and dimensioning of Pending Interest Tables (PITs) is typically based on the assumption that corresponding Data messages are received in a time frame that is based on typical network RTTs (e.g., order of 10s or 100s of milliseconds for wide-area networks). PIT state is, therefore, short-lived by design. As a result, Application Timescales can differ significantly from Network Timescales, which must be considered by a general purpose function invocation scheme.

RICE aims to overcome these three limitations by enhancing the ICN model with function-oriented capabilities while preserving the core architectural and protocol design elements of ICN networks. RICE is a general-purpose network-layer framework and can be applied to any named-function networking context. Its main features are:

- a secure, 4-way handshake for ICN networks in order to achieve shared secret derivation, consumer authentication and parameter passing.

- the concept of thunks [THUNKS] from the programming language literature to decouple method invocation from the return of results to enable long-running computations. The thunk is used to name the results for retrieval purposes.

The ultimate goal of the RICE framework is to enable in-network function execution with client authentication and non-trivial parameter passing, to support cases where computation takes longer than PIT expiry time. RICE achieves this goal by decoupling application processing time from PIT timers and network RTT. We argue that this is a necessary feature of any name-based remote function invocation scheme, where computation is accommodated in distributed compute spots in local or wide area networks at the core or edge of the network. The mechanisms we propose follow ICN principles and require minimal and short-lived additional network state.

More rationale, a comparison with related work, and an evaluation of the RICE approach is provided in [RICE2018].

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119, BCP 14 [RFC2119] and indicate requirement levels for compliant RICE implementations.

2. Terminology and Design Overview

RICE is as a general Remote Method Invocation (RMI) service - providing a robust and secure basis for a wide range of applications and in-network computation scenarios, including scenarios where method invocation requires a significant amount of input data and involves computations that take significantly longer than a network RTT to complete.

RICE is independent of any particular function execution environment. RICE provides all the required ICN protocol mechanisms and conventions for clients and servers and can serve as an underlying platform to support frameworks such as NFN and NFaaS (as well as any other in-network compute framework that utilises core ICN principles). In the description of RICE we use the following terminology:

Consumer: ICN protocol entity that is sending an Interest message

Producer: ICN protocol entity that is sending a Data message, replying to a received Interest message

Client: RICE protocol user that wants to request a remote method invocation

Server: RICE protocol user that is processing and answering the remote method invocation request (this may be a logical entity that is represented by more than one ICN protocol entity)

2.1. Design Goals

The RICE protocol is based on the following design goals:

Decouple application time scale from network time scale: RICE does not map remote method invocation directly to one Interest/Data exchange for the reasons discussed above. We want clients to request remote method invocation from a server without changing the network behavior with respect to Pending Interest management.

Support client authorization: A RICE server should be able to authorize clients before committing resources such as state, processing power etc. A server that blindly accepts any RMI request opens itself to computational and/or memory overload attacks. This also implies that the authorization mechanisms must be designed so that performing authorization itself does not overload servers and open a vulnerability to computational attacks.

Support non-trivial method invocation with arbitrarily complex parameter sets

Be robust and ICN-friendly to a mix of RICE and non-RICE traffic: we want RICE to coexist seamlessly in existing ICN networks, e.g., it should adhere to the same flow balance principles. The main consequence is that we do not transmit RMI parameters in Interest messages.

Support non-trivial, long-running computations with large amounts of result data

Support session-like interactions, where a client and a server use a sequence of exchanges:
RICE result data should be chunkable, and it should also be possible to retrieve result data that is generated over time, for example in multiple invocations in the same "session".

Allow ICN caching for referentially transparent method invocations:
ICN generally supports location-independent data access and opportunistic caching. In RICE, we want to support referentially transparent functions efficiently, i.e., function expressions that can be replaced with the result of the actual function execution. In other words, RICE is able to cache the result from such function invocations and enables the network to answer subsequent Interest messages from caches. Since not all functions are referentially transparent (some functions may depend on other data from the environment that is not specified in arguments), our framework distinguishes between functions that are referentially transparent and those that are not.

Adhere to ICN principles: RICE should not give up important ICN principles, such as flow balance, implicit support for consumer mobility, consumer anonymity (no source addresses). This last property is worth mentioning, because some ICN extensions/ applications rely on the fact that one end of an interaction would provide a globally routable "source" address to the other end to achieve "callback" behavior or generally enable bidirectional communication. Since such schemes would expose client identity information to the network (and to peers), we deem this approach an unacceptable deviation from ICN principles. Client identities should be exposed only to the application layer. Other forms of identifiers that help the network for maintaining reverse forwarding state to clients should be designed so that they do not expose clients' identities.

Be compatible with ICN extension mechanisms: Some recent proposed extensions make benign changes to ICN forwarding behavior (without

compromising general interoperability). One example would be fast forwarding information updates using technique such as Map-ME [MAPME]. RICE should be designed to work with such extensions. In this particular case, this means that RICE should support client and server mobility in a Map-ME-enhanced network.

Make minimal changes to ICN protocols and forwarder behavior: We want to allow for some changes to ICN forwarder behavior, but these should be limited and designed so that they do not complicate forwarder implementations or impair their performance.

The general guideline for achieving these goals is to prioritize robustness, security and scalability over absolute efficiency (with respect to number of handshakes and message exchanges), while still arriving at a design with reasonable efficiency.

3. Protocol

Remote Method Invocation (RMI) operations in RICE are split in two ICN interaction phases: 1) RMI Initiation (eventually triggering the remote method execution), and 2) the Result Retrieval. The RMI Initiation phase is designed to complete in Network Timescale (on the order of a few RTTs), whereas the remote method execution is decoupled from that and can take as long as required. The Result Retrieval phase can consist of several ICN Interest-Data exchanges (for chunked results or for computations that generate results iteratively).

3.1. Thunks

RMI Initiation and Result Retrieval are somehow decoupled from each other. The RMI Initiation generates a 'Thunk' name, i.e., a handler that the client can use later to retrieve the RMI result (or status) from the server. 'Thunk' thus represents the computation process at the server.

The RMI Initiation and the actual RMI invocation (and results) delivery do not necessarily have to take place on the same ICN node. For example, a RICE server could accept an RMI request, perform the Initiation and then delegate the actual computation to a backend server. For that, it would generate a Thunk name that the client can use to reach that backend server (or to obtain the corresponding result, generally speaking).

3.2. Naming

RICE relies on a generic naming scheme for remote method invocation. We distinguish between 'Method Names' and 'Thunk Names'. A function name identifies a method requested by a client that can be executed by any server able to perform the computations. In contrast, a thunk name identifies a specific method instance already being run on a specific node.

When a client initially requests the invocation of a referentially transparent method, the name in the Interest MUST unambiguously specify both the invoked method and the set of input parameters. The method part can be system-specific (\e.g., lambda-expressions in [TSCHUDIN2014NAMED] or expressed as a hierarchical name structure as in [NFAAS]}), but MUST unambiguously identify the method to invoke.

The input parameters can be represented directly when very small in size (e.g., username, number of iterations to perform), or as a hash when larger. (The choice of the actual hashing method can be left to the application.)

For referentially opaque methods, the result can be different even when using the same input parameters. Every invocation (either from the same client or from other clients) MUST lead to a new computation instance. Therefore, Interests for triggering referentially opaque method invocations MUST use a unique name for each invocation, while Interest retransmissions from the same client MUST use the same name.

This prevents the network from aggregating the corresponding Interests and limits cached responses to only answer retransmitted interests from a individual consumer. To achieve this, the client MUST include a name component that distinguishes its request from those generated by other clients. This component MUST be chosen by the client such that other nodes cannot predict the value. (Using predictable information (e.g., a MAC address) opens an additional attack surface and can leak client's sensitive information.)

The thunk name MUST unambiguously identify the server's forwarder, the instantiated method and the method's internal state (i.e., input parameters, chunk number). In that way, when the client uses the thunk name in consecutive requests, the Interest can be forwarded to the correct server and dispatched to the application possessing the associated result data. With thunk names, we do not need to distinguish between names for referentially transparent and opaque methods. They unambiguously identify a handler to a method execution instance, and it is up to the server to return the same or different handlers to multiple clients.

3.3. Handshake

RICE uses a 4-way handshake which serves the purposes of 1) deriving a shared secret, 2) authenticating and authorizing clients, and 3) providing input parameters to methods.

A client starts a 4-way handshake by sending an Interest message I1 towards a server. Similarly to the TCP SYN flag, the message contains an additional TLV Handshake field (this description assumes NDN [NDN] implementation of ICN).

The field contains an identifier that is chosen by the client and distinguishes among different handshakes. Upon receipt of the Interest, each forwarder creates a corresponding PIT entry as with regular Interests. Forwarders also inject a new temporary entry in the FIB formed from the identifier and pointing to the face on which the Interest was received. This FIB entry has a short expiry time after which it is deleted.

TODO: specify the timeout

When the server receives the Interest, it MUST respond with an I2 Interest message and forms the name from the received identifier. The Interest thus follows the previously established FIB entries towards the client. The forwarder also increases the expiry timer of I1's PIT entry.

At this point, the client MUST be ready to respond and send a D2 Data message. (Authorization information and parameters may be larger than what can be carried in a single D2 message. Therefore, multiple I2/D2 exchanges using the built-in chunking capabilities of the underlying ICN protocols can be employed.) When the server receives D2 it allocates resources for the computation for the client and sends back a confirmation in D1 Data message.

Interest I1 of the first handshake contains a method name that is delivered to a server advertising the corresponding prefix. At the end of the first handshake, in D1, the server returns its thunk name. This name can be then used by the client in I1 message of consecutive handshakes to assure future invocations reach the same method instance.

3.4. Shared Secret Derivation

Dirk: Not sure we need the following:

Common secret derivation plays an important role in many security protocols. Once a common secret is established, both parties can

encrypt the communication using symmetric cryptography. Common secret derivation (e.g., via Diffie-Hellman Key Exchange) requires sending data in both directions between the involved parties and is clumsy to implement in vanilla ICN networks where not all nodes have globally routable prefixes. Using one or multiple handshakes as described above allows use of any symmetric key scheme to secure the communication over arbitrary paths.

3.5. Client Authentication

Reliable and secure authentication requires multiple messages to be exchanged between the server and the client. Utilizing the handshake for client authentication, specially the exchanged I2/D2 messages, we achieve this goal and we decouple it from function invocation and input parameter passing.

In addition, by using the I2/D2 messages for client authentication and avoiding adding authentication information in the original interest (I2/D2), the RICE handshake mechanism becomes protected against any record-and-replay attack by a malicious party which can intercept the traffic. Once authenticated, the client creates a security token that can be included as a last component of the thunk name. In further communication such as commands (i.e., pause, stop) or referentially opaque methods, the token can be changed for each consecutive Interest message (i.e., based on the last received Data message).

It should be noted that RICE can rely on idiomatic ICN mechanisms for server authentication, i.e., validate the signatures on Data or signed Interest messages. The client could also encrypt input parameters using the public key of the server. A detailed specification will be provided by a future version of this document.

Following ICN principles, clients SHOULD NOT authenticate the server performing computations, but rather authenticate the returned result. If submitted input contains confidential data, it can be encrypted and shared using existing ICN access control techniques [ION2013TOWARD].

3.6. Input Parameters

RICE passes input parameters to remote methods "by reference". Since the server cannot access memory in the client, we use the I2/D2 Interest/Data Exchange as a "callback" from the server to the client to fetch the input parameters. This preserves the ICN principle that data is never pushed to the server if not requested. The server piggybacks on the PIT entry established by I1 to reach the client and pull the required data. If the input contains sensitive information,

the client can encrypt it using a common secret derived in previous handshakes as described above.

3.7. Dynamic Content Retrieval Using Thunks

In dynamic content generation or method invocation, the server may require a significant amount of time to create the requested data. In this section, we describe our use of thunks to allow clients to retrieve computation results.

The client starts by sending an Interest with a function name. Thunks allow multiple clients calling the same referentially transparent method with identical parameter sets to share one PIT entry and efficiently retrieve the data, while keeping entries separate for referentially opaque computations.

Upon receipt of the Interest, the server starts the requested computation and immediately responds with a thunk Data message. For the network, the thunk does not differ from a regular Data message that consumes the pending PIT entry. The payload of the message contains a 'thunk name' and an estimated completion time. The client waits for the time indicated in the server's response and issues a new Interest with the received thunk name.

Once the computation has finished, the function responds with a Data message containing the result. If the server mis-estimated the completion time and the data is not ready, it returns the same thunk target with an updated completion time estimate.

4. Security Considerations

By building on a well-studied ICN protocol framework, RICE shares the fundamental security advantages and difficulties of those protocols [MISRA2013], [CHOI2013THREAT], [ALSHEIKH2015FLOODING]. RICE employs native CCN/NDN machinery for cryptographic data integrity, origin authentication, confidentiality, and key management [MAHADEVAN2014]. Similarly, RICE shares the privacy problems and limitations of extant ICN protocols [GHALI2016]. In general, the additional threat that needs to be addressed is a computational or state-creation attack against a server by un-authorized clients. These threats request long-running computations, or flood the server with remote invocation requests that start useless computations [AIRTNT], [SPOC].

For the security consideration, we concentrate on the vulnerabilities associated with RICE, and provide a brief security analysis of the RICE machinery. The proposed 4-way handshake provides secure input parameter passing. It does so by fetching the parameters via the Interest-Data exchange "callback" from the server to the client.

Such callbacks could open a reflection attack via interest flooding [CHOI2013THREAT] if a globally routable name were used for the callback operation. However, the reverse-path mechanism and non-routable name RICE uses confines knowledge of the client's input parameter set to the server and on-path forwarders. The remaining vulnerability is the need to maintain forwarding state for the entire duration of the four-way handshake, rather than just a two-way exchange.

When a client sends I1 towards a producer, it creates additional state in FIB tables on all the intermediary forwarders. However, similarly to PIT entries, the created state is purged when its timer expires. If the timer is set to the same value as the PIT entries divided by 2, it does not expand the attack surface and existing Interest flood prevention techniques can be applied [ALSHEIKH2015FLOODING].

Upon reception of I1, the producer does not create any local state or allocate resources for the client. This protects our system from DoS attacks similar to the TCP SYN flood attack [RFC4987]. When the producer responds with the I2 message, that follows the trail created by I1. Such an approach assures that I2 is delivered only to the client initiating the session and eliminates the threat of using the producer as a spam bot. Following ICN principles, the data is effectively pulled by the producer from the consumer assuring that the producer does not receive large volume data that it did not request.

5. References

5.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

5.2. Informative References

[AIRTNT] Mustafa Al-Bassam et al, ., "Airtnt -- Fair Exchange Payment for Outsourced Secure Enclave Computations", 2018, <https://www.ee.ucl.ac.uk/~uceeips/files/airtnt-payments-v1.pdf>.

[ALSHEIKH2015FLOODING]
Al-Sheikh, S., WA[currency units]hlisch, M., and T. Schmidt, "Revisiting Countermeasures Against NDN Interest Flooding", Proceedings of the 2nd International Conference on Information-Centric Networking - ICN '15, DOI 10.1145/2810156.2812604, 2015.

[CHOI2013THREAT]
and , "Threat of DoS by interest flooding attack in content-centric networking", The International Conference on Information Networking 2013 (ICOIN), DOI 10.1109/icoin.2013.6496396, January 2013.

[GHALI2016]
Ghali, C., Tsudik, G., and C. Wood, "(The Futility of) Data Privacy in Content-Centric Networking", Proceedings of the 2016 ACM on Workshop on Privacy in the Electronic Society - WPES'16, DOI 10.1145/2994620.2994639, 2016.

[ION2013TOWARD]
Ion, M., Zhang, J., and E. Schooler, "Toward content-centric privacy in ICN", Proceedings of the 3rd ACM SIGCOMM workshop on Information-centric networking - ICN '13, DOI 10.1145/2491224.2491237, 2013.

[MAHADEVAN2014]
Mahadevan, P., Uzun, E., Sevilla, S., and J. Garcia-Luna-Aceves, "CCN-KRS", Proceedings of the 1st international conference on Information-centric networking - INC '14, DOI 10.1145/2660129.2660154, 2014.

[MAPME] Auge, J., Carofiglio, G., Grassi, G., Muscariello, L., Pau, G., and X. Zeng, "MAP-Me: Managing Anchor-Less Producer Mobility in Content-Centric Networks", IEEE Transactions on Network and Service Management Vol. 15, pp. 596-610, DOI 10.1109/tnsm.2018.2796720, June 2018.

[MISRA2013]
Misra, S., Tourani, R., and N. Majd, "Secure content delivery in information-centric networks", Proceedings of the 3rd ACM SIGCOMM workshop on Information-centric networking - ICN '13, DOI 10.1145/2491224.2491228, 2013.

[MOISEENKO2014]
Moiseenko, I., Stapp, M., and D. Oran, "Communication patterns for web interaction in named data networking", Proceedings of the 1st international conference on Information-centric networking - INC '14, DOI 10.1145/2660129.2660152, 2014.

[NDN] Zhang, L., Afanasyev, A., Burke, J., Jacobson, V., claffy, k., Crowley, P., Papadopoulos, C., Wang, L., and B. Zhang, "Named data networking", ACM SIGCOMM Computer Communication Review Vol. 44, pp. 66-73, DOI 10.1145/2656877.2656887, July 2014.

[NFAAS] KrA^3l, M. and I. Psaras, "NFaaS", Proceedings of the 4th ACM Conference on Information-Centric Networking - ICN '17, DOI 10.1145/3125719.3125727, 2017.

[REST] Fielding, R., "Architectural Styles and the Design of Network-based Software Architectures", 2000.

[RFC4987] Eddy, W., "TCP SYN Flooding Attacks and Common Mitigations", RFC 4987, DOI 10.17487/RFC4987, August 2007, <https://www.rfc-editor.org/info/rfc4987>.

[RICE2018]
Michael Krol, ., Karim Habak, ., Dave Oran, ., Dirk Kutscher, ., and . Ioannis Psaras, "RICE -- Remote Method Invocation in ICN", September 2018.

[SPOC] Michael Krol, . and . Ioannis Psaras, "SPOC -- Secure Payments for Outsourced Computations", 2018, <https://www.ee.ucl.ac.uk/~ipsaras/files/spoc-payments.pdf>.

[THUNKS] Ingerman, P., "Thunks: a way of compiling procedure statements with some comments on procedure declarations", Communications of the ACM Vol. 4, pp. 55-58, DOI 10.1145/366062.366084, January 1961.

[TSCHUDIN2014NAMED]
Tschudin, C. and M. Sifalakis, "Named functions and cached computations", 2014 IEEE 11th Consumer Communications and Networking Conference (CCNC), DOI 10.1109/ccnc.2014.6940518, January 2014.

Authors' Addresses

Michal Krol
University College London
Gower Street
London
United Kingdom

Email: m.krol@ucl.ac.uk

Karim Habak
Georgia Institute of Technology
North Ave NW
Atlanta GA 30332
USA

Email: karim.habak@gatech.edu

Dave Oran
Network Systems Research & Design
TBD
Cambridge
USA

Email: daveoran@orandom.net

Dirk Kutscher
Huawei
Riesstrasse 25
Muenchen D-80992
Germany

Email: ietf@dkutscher.net

Ioannis Psaras
University College London
Gower Street
London
United Kingdom

Email: i.psaras@ucl.ac.uk