Statement: RESTful Information-Centric Networking

Dirk Kutscher
Hong Kong University of Science and Technology
Guangzhou, Guangdong, China
dku@ust.hk

David Oran
Network Systems Research & Design
Cambridge, MA, USA
daveoran@orandom.net

ABSTRACT

Web applications today utilize the Representation State Transfer (REST) architectural pattern, depending on HTTP, TLS, and either TCP or QUIC as the protocol substrate to build upon. Our vision is to achieve the key properties of REST using ICN protocols as an alternative. We argue that this is feasible given some of the recent advances in ICN protocol development and that the resulting suite is simpler and potentially having better performance and robustness properties. Our sketch of an ICN-based protocol framework addresses secure and efficient establishment and continuation of REST communication sessions, without giving up key ICN properties, such as consumer anonymity and flow balance.

1 INTRODUCTION

The Web today is based on an extended version of the Representational State Transfer (REST) [3] architecture pattern for client-server interaction. This simple model has been extended and applied to HTTP for web applications by supporting not only retrieval, but also creation, processing, and deletion of data. Real-world REST systems employ additional concepts and mechanisms such as security and privacy, support for application sessions, and have various optimizations to eliminate unnecessary round-trips.

Since nearly all web applications today are based on this RESTful client-server communication model, the question then occurs how such interactions can be achieved in ICN, i.e., secure and confidential RESTful access to web resources, with support for efficient handling of a sequence of interactions in a session-like context.

The applicability of ICN's Interest/Data interaction to modern web applications that provide a significant amount of data in requests headers for cookies and other request parameters has been assessed by [6], concluding that it is not immediately clear how to use ICN effectively for web communication. We have also argued in [5] that the basic Interest/Data exchange model of CCNx/NDN-style ICN is not sufficient and that certain use cases (e.g., sending resource representations or request parameters from a client to a server) should not be implemented by overloading the Interest message. In [9] we have discussed the specific problems extensively. In its default mode, ICN also lacks name privacy, which we consider essential for any real-world application of ICN to web services.[1]

In this short vision paper, we argue that an ICN-based RESTful programming model that overcomes these limitations is feasible given some of the recent advances in ICN protocol development and provide the outline of the corresponding protocol framework.

2 RESTFUL ICN

2.1 Goals

HTTP has been extended and partially redesigned over time, and provides its own idiosyncratic conventions and mechanisms, e.g., which request-relevant information to represent in the URI vs. message headers vs. message bodies. The goal of this work is *not* to simply map current HTTP mechanisms to ICN, but rather to provide an ICN-idiomatic platform for RESTful applications including an Information-Centric web.

Any ICN web platform will only be useful and relevant if it provides equivalent (or better) security and privacy properties as the state-of-art, i.e., HTTP3 [1] over QUIC [4] and TLS-1.3 [10], so our proposed framework provides a TLS-like security context for RESTful communication (sessions). Also, RESTful ICN should not compromise on existing ICN benefits such as consumer anonymity and consumer mobility.

2.2 Design Overview

RESTful ICN enables client/server communication with a series of request/response interactions in a session context. We employ the same conceptual model as Reflexive Forwarding (RF) [9], i.e., we allow for robust ICN-idiomatic client/server communication with client parameter passing, for both key exchange and actual RESTFul communication. We enable secure RESTful communication using standard ICN mechanisms such as Content Object encryption and signatures without forcing all interactions into TLS-like tunnels.

For a series of requests in a session we avoid setting up context state for every request and the corresponding protocol interactions. In order to provide both efficient repeated RESTful requests to the same server, we establish and maintain the shared "session" state by using the identifiers of the keys and associated security context negotiated by the setup phase. Through the parameter passing machinery of Reflexive Forwarding [9] we provide for clients to refer to previously created application state – analogously to how HTTP *cookies* operate.

By having both a secure referent state held on a particular server (through key-ids) and a referent to application state through parameters secured through those keys, we can provide the key features of today's session based RESTful protocols, which are:

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
ICN '22, September 19–21, 2022, Osaka, Japan
© 2022 Association for Computing Machinery.
ACM ISBN 978-1-4503-9257-0/22/09…$15.00
https://doi.org/10.1145/3517212.3558089

[1]however, various techniques have been developed to improve name privacy in ICN, such as the onion routing approach in [2].

- application state caching on clients to allow server agility;
- securing the application state exchanged through pair-wise session keys with a particular server;
- rapid setup of these keys using a key exchange protocol compliant with TLS 1.3; and
- efficient state evolution (minimizing round-trips and state representation overhead) and RESTful semantics for multiple interactions with the application through the same server.

We further argue that these properties are obtained with ICN protocol machinery that approximates the capabilities obtained by the combination of HTTP, TLS, and QUIC (or TCP). The current 3-layer approach operates through separate protocol implementations, with independent state machines for transport, secure key exchange and data encapsulation, and transactional request/response operations. In contrast, by combining these functions as part of the ICN request/response scheme with secure object encapsulation, we argue the resulting protocol suite is substantially simpler to implement with the result having the potential for better performance, higher robustness, and a smaller implementation attack surface.

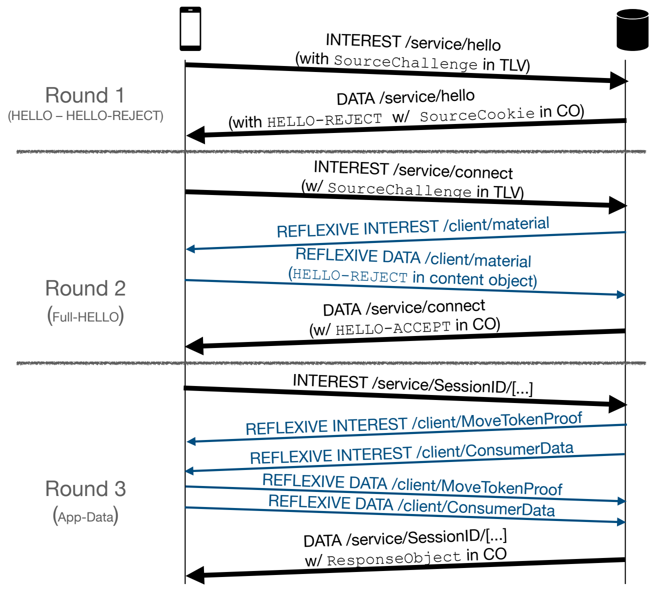
2.3 Security Contexts

CCNx key exchange [7] specifies a TLS-1.3-like key exchange protocol between two peers for establishing a shared, forward-secure key for secure and confidential communication. RESTful ICN adopts this protocol conceptually and integrates the different exchanges in a Reflexive Forwarding framework to avoid having to stuff all client parameters (key exchange and application data) into Interests, thus making the exchanges more ICN-idiomatic, at the cost of two additional round-trips. Note that we maintain the encryption part of CCNx key exchange and intentionally do not propose a new key exchange protocol – RESTFul ICN maps the key exchange onto Reflexive Forwarding, adding the application layer token transmission to give key exchange the ability to carry RESTful state as well. Therefore, important CCNx key exchange properties are retained such as protection against computational overload attacks and the ability to use different infrastructure for security and for service functions. One notable consequence is that session state and keying are coupled; if you need to revoke a key, you therefore terminate and restart the session.

There are three main variants: (*i*) the regular approach (without prior session establishment, depicted in figure 1), (*ii*) the One-RTT approach (with prior session token exchange), and (*iii*) Resumption and PSK mode (for re-creating a previous session). The regular approach would typically be used as an initial request, and the one-RTT approach (not shown here for brevity) would be used for subsequent requests in the same "security session" context.

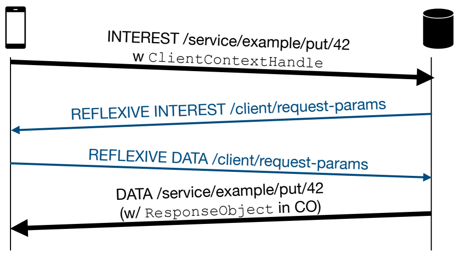
2.4 RESTful ICN Communication

One of use cases for CCNx key exchange [7] is to negotiate encryption keys for tunnel-like encrypted sessions (ESIC) [8], encapsulating every Interest and Data message. RESTful ICN does not use this approach – instead we use the obtained keys for securing the native communication in Reflexive-Forwarding-based interactions, i.e., the server encrypts Content Objects with the symmetric server key, and the client encrypts Reflexive-Forwarding Content Objects with the symmetric client key, as depicted in figure 2.



Round 1
(HELLO – HELLO-REJECT)
INTEREST /service/hello
(with SourceChallenge in TLV)
DATA /service/hello
(with HELLO-REJECT w/ SourceCookie in CO)
Round 2
(Full-HELLO)
INTEREST /service/connect
(w/ SourceChallenge in TLV)
REFLEXIVE INTEREST /client/material
REFLEXIVE DATA /client/material
(HELLO-REJECT in content object)
DATA /service/connect
(w/ HELLO-ACCEPT in CO)
Round 3
(App-Data)
INTEREST /service/SessionID/[...]
REFLEXIVE INTEREST /client/MoveTokenProof
REFLEXIVE INTEREST /client/ConsumerData
REFLEXIVE DATA /client/MoveTokenProof
REFLEXIVE DATA /client/ConsumerData
DATA /service/SessionID/[...]
w/ ResponseObject in CO

Figure 1: Initial Exchange for Session Setup



INTEREST /service/example/put/42
w ClientContextHandle
REFLEXIVE INTEREST /client/request-params
REFLEXIVE DATA /client/request-params
DATA /service/example/put/42
(w/ ResponseObject in CO)

Figure 2: RF-based REST request for Session Continuation

The ClientContentHandle Interest parameter contains the necessary SessionID and key-id for the security conext and an encrypted name for the application state representation that the client explicitly conveys to the server. The ResponseObject content object contains the actual response body (as per regular REST request semantics) and a set of optional elements such as SessionID when used in key exchange and an encrypted name for the new application state representation at the server that the client can use for subsequent requests to explicitly refer the server to its application state. When the security session terminates, or the server is no longer reachable, the client can establish a new session (using the key exchange procedure above) to interact with a new server instance and refer it to the application state.

Note that the server can decide to encrypt the request response (in the ResponseObject) using the per-session symmetric key or its public namespace key. The latter approach could be chosen if the request is referentially transparent, and the application semantics allow for caching and potentially sharing the request result.

Our next steps are to prototype this approach, demonstrate ease of developing and deploying applications, evaluate the security and privacy properties, explore potential performance optimizations to support low-latency web applications.

REFERENCES

[1] Mike Bishop. 2021. *Hypertext Transfer Protocol Version 3 (HTTP/3)*. Internet-Draft draft-ietf-quic-http-34. IETF Secretariat. https://www.ietf.org/archive/id/draft-ietf-quic-http-34.txt https://www.ietf.org/archive/id/draft-ietf-quic-http-34.txt.

[2] Steve DiBenedetto, Paolo Gasti, Gene Tsudik, and Ersin Uzun. 2012. ANDaNA: Anonymous Named Data Networking Application. *ArXiv* abs/1112.2205 (2012).

[3] Roy Thomas Fielding. 2000. *Architectural Styles and the Design of Network-based Software Architectures*. Ph. D. Dissertation. University of California, Irvine. http://www.ics.uci.edu/ fielding/pubs/dissertation/top.htm.

[4] J. Iyengar and M. Thomson. 2021. *QUIC: A UDP-Based Multiplexed and Secure Transport*. RFC 9000. RFC Editor.

[5] Michał Król, Karim Habak, David Oran, Dirk Kutscher, and Ioannis Psaras. 2018. RICE: Remote Method Invocation in ICN. In *Proceedings of the 5th ACM Conference on Information-Centric Networking* (Boston, Massachusetts) *(ICN ’18)*. Association for Computing Machinery, New York, NY, USA, 1–11. https://doi.org/10.1145/3267955.3267956

[6] Ilya Moiseenko, Mark Stapp, and David Oran. 2014. Communication Patterns for Web Interaction in Named Data Networking. In *Proceedings of the 1st ACM Conference on Information-Centric Networking* (Paris, France) *(ACM-ICN ’14)*. Association for Computing Machinery, New York, NY, USA, 87–96. https://doi.org/10.1145/2660129.2660152

[7] M. Mosko, Ersin Uzun, and Christopher A. Wood. 2017. *CCNx Key Exchange Protocol Version 1.0*. Internet-Draft draft-wood-icnrg-ccnxkeyexchange-02. IETF Secretariat. https://www.ietf.org/archive/id/draft-wood-icnrg-ccnxkeyexchange-02.txt https://www.ietf.org/archive/id/draft-wood-icnrg-ccnxkeyexchange-02.txt.

[8] Marc Mosko and Christopher Wood. 2017. *Encrypted Sessions In CCNx (ESIC)*. Internet-Draft draft-wood-icnrg-esic-01. IETF Secretariat. http://www.ietf.org/internet-drafts/draft-wood-icnrg-esic-01.txt http://www.ietf.org/internet-drafts/draft-wood-icnrg-esic-01.txt.

[9] David R. Oran and Dirk Kutscher. 2022. *Reflexive Forwarding for CCNx and NDN Protocols*. Internet-Draft draft-oran-icnrg-reflexive-forwarding-02. IETF Secretariat. https://www.ietf.org/archive/id/draft-oran-icnrg-reflexive-forwarding-02.txt https://www.ietf.org/archive/id/draft-oran-icnrg-reflexive-forwarding-02.txt.

[10] E. Rescorla. 2018. *The Transport Layer Security (TLS) Protocol Version 1.3*. RFC 8446. RFC Editor.