

ICNRG
Internet-Draft
Intended status: Informational
Expires: August 22, 2013

D. Corujo
Instituto de Telecomunicacoes
K. Pentikousis
Huawei Technologies
I. Vidal
UC3M
February 18, 2013

ICN Management Considerations
draft-corujo-icn-mgmt-00

Abstract

This document aims to draw the attention of the ICNRG community to network management, an important but hitherto underdeveloped area of research in information-centric networking. We consider that the availability of modern management mechanisms for information-centric networks will foster their deployment in real-world environments. For example, we argue that there is a need for creating basic network management tools early on while ICN is still in the design and experimentation phases that can evolve over time. Perhaps ICN can borrow successful mechanisms from the host-centric paradigm and adapt them to the new network primitives. Alternatively, novel network management schemes can be designed based on ICN primitives. As a discussion starter, this document summarizes recently published approaches for ICN network management. In particular, this first version presents a management framework for named data networking and reviews previous work on NetInf management.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any 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 August 22, 2013.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. NDN Management Considerations	4
2.1. Towards a Management Framework for NDN	5
2.2. NDN Management Operations	7
2.2.1. Discovery Procedure	7
2.2.2. Management Data Exchange	9
2.3. Implementation Experience	10
3. NetInf Management Considerations	11
4. Acknowledgements	11
5. IANA Considerations	12
6. Security Considerations	12
7. Informative References	12
Authors' Addresses	13

1. Introduction

Information-centric networking (ICN) enables new ideas for naming and addressing, privacy, security, and trust, and should also lead us to think new ways for deploying, operating and managing networks in the future. By default, users, programs, information objects and hosts are in general untrustworthy and mobile in an information-centric network. This means that many of the assumptions in traditional network management, including all aspects of FCAPS (Fault, Configuration, Accounting, Performance, and Security) need to be rethought. However, despite the different instantiations of ICN architectures, and the plethora of novel research work built on top of them, little attention has been paid to management aspects so far. This includes both enabling "traditional" network management operations (which work well from small networks to large infrastructure networks), and supporting and optimizing intrinsic procedures of the ICN fabric.

This document aims to draw the attention of ICNMG to the importance of network management for real-world deployments. Today, network management is practically an add-on to host-centric deployments. We can do better as we move forward in ICN research considering the full range of deployments from home-office environments to challenged networks to tier-1 networks. To this end, we draft some first management considerations that, on the one hand, capitalize on ICN concepts for defining management procedures and, on the other, explore the possibilities for defining a common management framework irrespective of the ICN approach taken. We reckon that the later is a much more formidable task and we are looking forward to tackling it together with other members of ICNMG. We start this exercise in this first version based on published literature and in particular with a NDN approach.

We argue that addressing management at an early stage is not only important for real-world adoption and the successful future deployment of ICN, but also to deal with scenarios where management can simplify, enhance or optimize ICN network utilization and performance. The subject becomes particularly challenging, as disparate characteristics from different ICN approaches (e.g., in terms of namespace, granularity, routing, and so on) impact the definition and design of these management mechanisms. Section 2 below provides an initial assessment, proposal and evaluation of management mechanisms leveraging NDN intrinsic capabilities based on [NDN-MGMT], while Section 3 briefly summarizes earlier work on self-management for NetInf.

We plan to incrementally develop the draft and incorporate other ICN approaches (e.g., [PURSUIT] and [NetInf]) as well as address other

pertinent aspects as we receive feedback from the research group members.

2. NDN Management Considerations

The Named Data Networking [NDN] ICN architecture provides a new communication framework built on named data. Like other ICN counterparts, such as [NetInf], [PURSUIT] and [DONA], NDN intrinsically supports security, routing/forwarding, reliability, caching and even mobility, aiming at scalable and more efficient content-distribution than today's IP-based approaches. Fostered by an open-source implementation [CCNx], NDN has been at the heart of an active topic with several research contributions evaluating its deployment feasibility and performance in a number of scenarios [ICN-Scenarios].

NDN relies on a hierarchical, human-readable namespace to address named data objects, where the naming scheme is simultaneously used to both name information and to route it. It relies on content requesters sending an Interest packet with a Content Name, where the prefix can provide information for global and organizational routing, while the suffix indicates versioning and segmentation details. When a node receives an Interest packet asking for content which matches what is already available at the node, it responds with a matching Data packet carrying back the content.

Each NDN node comes with a set of supporting data structures which enable the coordination between the transmission of Interest packets with the reception of the corresponding Data packets. These structures include:

1. Content Store: maintains an indication of locally available content, according to name, and is used for Interest packet matching. If the content is available at the node, the Interest packet is consumed, and a Data packet with the respective content is sent towards the request origin.
2. Pending Interest Table (PIT): keeps track of Interest packets seen previously by the node, on their way to locate matching content. Interest packets in the PIT were not matched to content available in the node. Basically, PIT maintains a degree of state regarding Interest packets, mapping them to a corresponding egress network interface.
3. Forward Information Base (FIB): associates named data to potential holders of the content. A routing protocol can populate the FIB (although this is outside the scope of NDN) or

it can be populated through registration in a local NDN store.

NDN introduces the concept of a Strategy Layer, which can control Interest packet forwarding behavior. It basically determines which is the best interface (or set of interfaces) to send an Interest packet. The "strategy" component establishes a pre-configured algorithm for tackling Interest packet decisions, ranging from sending it sequentially on each interface until a Data packet is received, to evaluating which interfaces provide better performance (i.e., lower average RTT) in retrieving certain content (as discussed in [NDN]).

It is important to keep in mind that NDN replaces the commonly used term "interface" with the term "face", since packets can be forwarded over hardware network interfaces as well as between application interfaces, further acknowledging the information dissemination capabilities of ICN. This aspect is considered in [NDN] and [NDN-R], where programs can be associated to the NDN governing structures (like the FIB), defining configurations such as "sendToAll" and "sendToBest" with respect to managing the content reaching process. Corujo et al. [NDN-MGMT] exploit these concepts enabling management mechanisms to be deployed, and steer network operations and NDN operation, as described in the following section.

2.1. Towards a Management Framework for NDN

An important aspect supporting network management procedures is the interaction of network information residing at the network side with information about the network from the perspective of clients connected to it. The former includes, for instance, information stored in the network operator core about user profiles, associated policies, or data collected by the access network equipment, such as current and past traffic load levels, active flows, and maintenance information. Today, such information can be retrieved for management and operation support through dedicated signaling protocols (e.g., [RFC1157], [RFC6733]), or Operation Support Services (OSS) web services. The client point of view of the network includes information that, for example, a wireless terminal can provide, indicating wireless link quality, average return-trip times (RTT) or perceived Quality of Experience (QoE).

Both types of information can be capitalized upon allowing, for example, the network to coordinate network management procedures, considering as input information obtained from other network elements as well as from user nodes. One way to generate management information in network entities and at client nodes, as well as to consume and act upon it (i.e., using the management information exchange as a control channel) is to couple NDN nodes with Management

Agent (MA) entities.

Fig. 1 (redrawn here from [NDN-MGMT] for convenience) illustrates how a MA can be deployed in both network and client entities, interfacing with different operational aspects and protocol layers of an NDN node. By using NDN content reaching and disseminating mechanisms, management information can be consumed by the MA to steer not only the behavior of application processes and network interfaces, but also to interface with NDN supporting structures (i.e. Content Store, FIB, PIT). Effectively, different kinds of information can be conveyed to a network node responsible for managing the network (under different perspectives and processes), and resubmitted back towards client nodes, affecting the way applications interface with network interfaces and the NDN fabric.

NDN Fabric

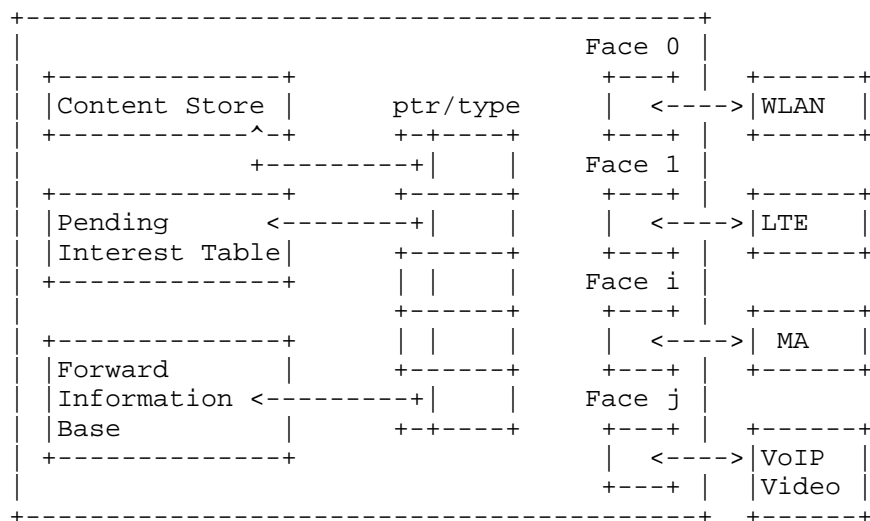


Figure 1. NDN Management Framework

MA can interface with the PIT and FIB structures, acting as a dynamic, application- and/or network-controlled interface to the strategy layer. This could also be used to direct how to forward NDN Interest and Data packets, in a configurable manner. Regarding network interfaces, the MA can interface with them not only to control (i.e., initiate wireless access scanning procedures), but also to collect information (i.e., an informational event regarding detected access points). Finally, the MA can also interface with application processes, drawing out information about the perceived QoS/QoE (e.g., lost packets or delay from a real-time video feed) and also to execute commands, such as selecting a better video codec when

the network commands the video flow to be accessed from a different wireless access interface.

Conversely, MA entities residing in network equipment can provide informational events as well, but related to network-side link layer characteristics (such as number of attached nodes or load), as well as accepting commands from the network (i.e., activate maintenance procedures). Management processes residing in the network core can leverage information collected from applications, client terminals and network equipment, to drive optimization procedures. Such optimization procedures can also tap into other entities, containing complementary information such as policies and subscription information, and use it to produce an overall network decision, which can then be forwarded to multiple client nodes, in a policy enforcing way.

An important consideration from the NDN architecture, is the hierarchical namespace, allowing nodes to request and convey management data, by simply using an appropriate prefix (e.g., `ccn://domain/management/ME`).

By leveraging the NDN information-centric dissemination mechanisms to convey management information and commands as content, these management extensions inherit the intrinsic capabilities of the NDN architecture, including security and reliability, which are fundamental for management procedures.

2.2. NDN Management Operations

In order to implement management operations, besides the interfacing capabilities of the MA entity mentioned in the previous section, a management framework needs other supporting mechanisms in order to provide the envisioned management capabilities, while maintaining the inherent NDN capabilities. Concretely, when nodes connect to the network, the management entities need to become aware of the management capabilities of the newly-connected node. In addition, an asynchronous information exchange capability needs to be provided, allowing not only the request of management information, but also the ability to push information towards a remote node (i.e., sending a command or an informational event).

2.2.1. Discovery Procedure

The discovery procedure is illustrated in Fig. 2 (redrawn from [NDN-MGMT]), and borrows for the procedures described in [NDN-VOIP]. The procedure starts with the newly connected User Equipment (UE) broadcasting an Interest packet (Fig. 2:1) perhaps with a well-known content name (e.g., `ccn://domain/management/mgmt-case/ME`) to its

local network.

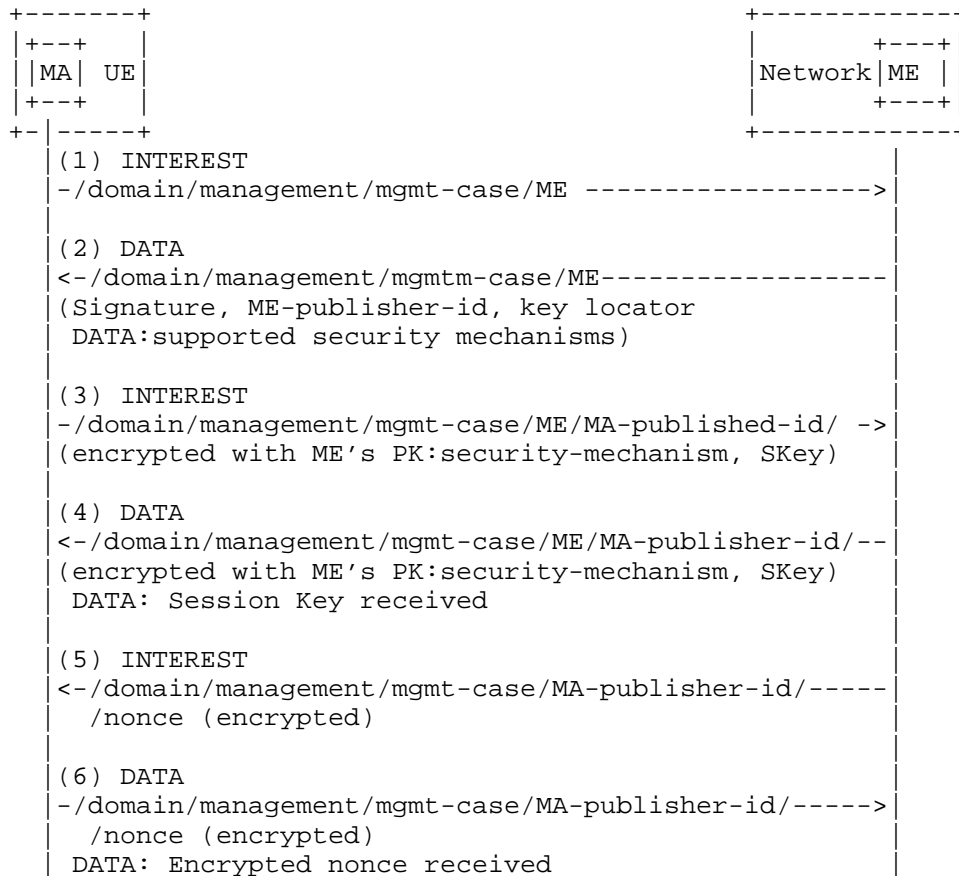


Figure 2. Secure Management Session Establishment

The "mgmt-case" part of the name can be used to select different aspects of management capabilities allowed by a Management Entity (ME) (i.e., a management decision point in the network). The ME then replies to this Interest with a Data packet (Fig. 2:2), providing its shorthand identifier (i.e., ME-publisher-key) and a key locator, indicating how to retrieve its public key (assuming it is authorized by another key trusted by the UE). In this way, the MA at the UE recognizes the ME as a valid signer (and provider) of management content.

A session key, K_s , is generated by the MA, considering an encryption algorithm from the ones indicated by the ME in the Data packet. The

MA then expresses its desire to receive (and reply to) Interests matching a specific NDN name associated with the management service (e.g., `ccn://domain/management/mgmt-case/ME/MA-publisher-id`), where MA-publisher-id uniquely and globally identifies the MA, through a cryptographic digest of its public key. After this, the MA sends an Interest packet (Fig. 2:3) to retrieve management Data from the ME containing the short-hand identifier of the MA (MA-publisher-id), the chosen encryption algorithm and session key (Ks), both encrypted with the public key of the ME. In this way, the confidentiality of the content exchanged between the ME and the MA is guaranteed. The ME responds with a Data packet (Fig. 2:4) signaling the reception of the session Key.

Before the actual exchange of management data begins, the ME generates a challenge (i.e., a nonce) which is sent via an Interest packet (Fig. 2:5) to the MA, indicating through a named data name that it requests the reception of the response to this challenge, sent by the MA using a Data packet (Fig. 2:6). This allows the ME, after verifying the signature of the Data packet, to verify that the encryption algorithm and the session key are valid for the MA, making it ready to exchange information for coordinating management procedures in the network.

2.2.2. Management Data Exchange

After the discovery and security establishment procedures have been finalized, the framework provides the capability for both the MA and the ME to securely obtain management content from one another.

In order to push unsolicited content, a dual Interest/Data procedure can maintain compatibility with the Interest and Data exchange/consumption of the NDN architecture. Fig. 3 (redrawn from Fig.2 of [NDN-MGMT]) illustrates the procedure which is initiated by the MA. In this case, the MA intends to push management information to the ME. It does so via an Interest packet manifesting its interest in receiving management content with a local sequence number. This sequencing allows the recovery of new content over cached content. If the ME is interested in retrieving content from the MA, it answers back with a Data packet, where it indicates that it is willing to receive management content. Then, the ME sends an Interest packet to retrieve the management data with the sequence number provided by the MA, which responds with a Data packet containing the information it wanted to push into the ME.

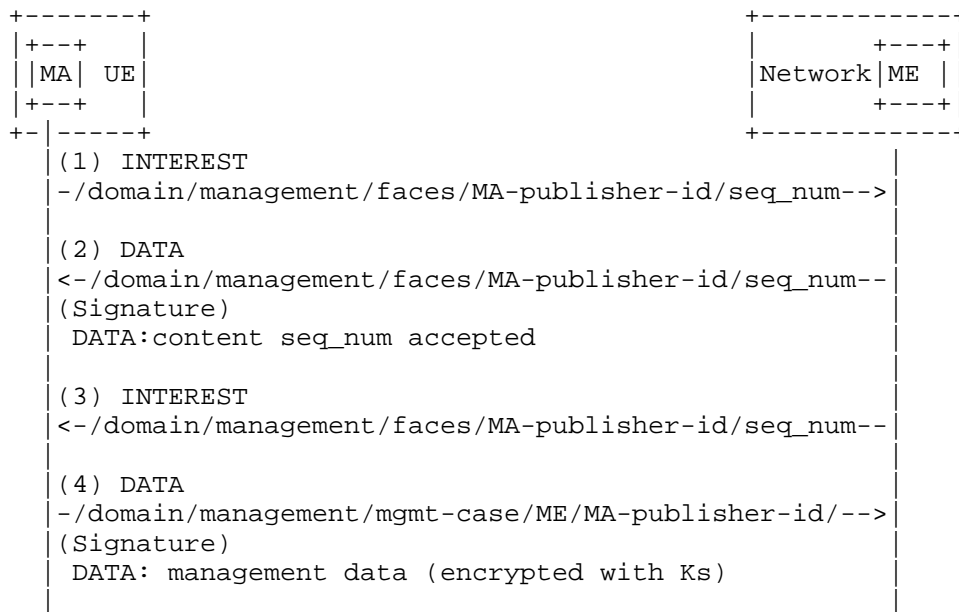


Figure 3. Content Management Push

2.3. Implementation Experience

As a proof-of-concept, a software prototype of the management framework was developed for [NDN-MGMT], using the CCNx Java API [CCNx]. At this early stage, it includes the implementation of an ME and an MA as NDN applications, supporting the NDN management operations outlined in Fig. 3. Thus, the ME and the MA can push unsolicited content to each other, related with management operations.

To validate this basic prototype, [NDN-MGMT] considered a specific use case supported by the framework, i.e., face management. This entails configuring and selecting an appropriate face in a UE to retrieve a given content. Based on the CCNx, an evaluation test-bed was deployed including an NDN UE (featuring an MA and a set of network interfaces), a content server and a network node (featuring an ME). These entities are interconnected by a set of NDN routers. The purpose of the evaluation scenario is to demonstrate feasibility for the protocol exchanges mentioned earlier. Note that the code has been tested in a small-scale environment where the ME is topology-aware and keeps track of conditions of the access networks that are available to the UE. Thus, the ME can provide the MA with management information reporting the appropriate face for content retrieval, or an alternative point of access that could be used to improve the

performance. The MA uses the management information to reconfigure the FIB (and possibly the network interfaces) in the UE, setting the appropriate face to forward subsequent Interests.

For validation purposes, a local application was also implemented at the NDN UE that works similarly to a ping utility, generating periodic Interests that match a given prefix (served by the content server), and computing the Round Trip Time of each Interest/Data exchange. The RTT values obtained by this application in [NDN-MGMT], indicate that the performance of the NDN management framework in the considered evaluation scenario is satisfactory, given the early stage of this work. Further development and testing is ongoing.

3. NetInf Management Considerations

Early-phase work in NetInf management [NetInfSelfX] discussed a two-fold problem. The first question that arises is whether it is possible by adopting a new set of network primitives and in-network storage to usher a new type of network management. In other words, can network management become information-centric while handling often host-centric data? The second question is whether an information-centric network is more suitable for self-management mechanisms than IP-based networks are. In particular with respect to the later, [NetInfSelfX] introduced some design considerations for adding self-management mechanisms in NetInf.

Of interest from this early work are two examples where network management can play a new role. First, network management can get involved in decisions about caching and (re)distribution of content, and not only whether an (inter)face is on or off, or what traffic limits should be enforced. Moreover, network policies can be distributed securely in the same way as other content in the network, removing the need for centralized management, and enabling improved recovery procedures. Second, network management can get involved in more intricate processes such as controlling multiaccess support, intermediating for content adaptation when deemed appropriate, and enabling richer tools for traffic engineering.

4. Acknowledgements

This document has benefited from comments and/or text provided by the following members of ICNRG:

Jaime Garcia-Reinoso (UC3M); Section 2.3

5. IANA Considerations

This memo includes no request to IANA.

6. Security Considerations

TBD

7. Informative References

- [CCNx] PARC, "CCNx Project", 2013, <<http://www.ccnx.org>>.
- [DONA] Koponen, T. et al., "A Data-Oriented (and Beyond) Network Architecture", SIGCOMM, ACM , 2007.
- [ICN-Scenarios]
 Pentikousis, K., Ohlman, B., Corujo, D., and G. Boggia,
 "ICN Baseline Scenarios", draft-pentikousis-icn-scenarios
 (work in progress), February 2013.
- [NDN] Jacobson, V., Smetters, D., Thornton, J., Plass, M.,
 Briggss, N., and R. Braynard, "Networking Named Content",
 CoNEXT 2009, Rome , Dec 2009.
- [NDN-MGMT]
 Corujo, D., Vidal, I., Garcia-Reinoso, J., and R. Aguiar,
 "A named data networking flexible framework for management
 communications", Communications Magazine, IEEE , vol.50,
 no.12, pp.36-43 , Dec 2012.
- [NDN-R] Zhang, L. et al., "Named Data Networking (NDN) Project",
 NDN Report ndn-0001, Tech Report, PARC , 2010,
 <<http://www.named-data.net/techreport/TR001ndn-proj.pdf>>.
- [NDN-VOIP]
 Jacobson, V., Smetters, D., Briggss, N., Plass, M.,
 Steward, P., and J. Thornton, "VoCCN: Voice Over Content-
 Centric Networks", ReARCH 2009, Rome , Dec 2009.
- [NetInf] Ahlgren, B. et al., "Design considerations for a network
 of information", CoNEXT, Re-Arch Workshop, ACM , 2008.
- [NetInfSelfX]
 Pentikousis, K. et al., "Self-Management for a Network of
 Information", IEEE ICC Workshops 2009 , June 2009.

- [PURSUIT] Fotiou, N. et al., "Developing Information Networking Further: From PSIRP to PURSUIT", BROADNETS, ICST , 2010.
- [RFC1157] Case, J., Fedor, M., Schoffstall, M., and J. Davin, "Simple Network Management Protocol (SNMP)", STD 15, RFC 1157, May 1990.
- [RFC3552] Rescorla, E. and B. Korver, "Guidelines for Writing RFC Text on Security Considerations", BCP 72, RFC 3552, July 2003.
- [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.
- [RFC6733] Fajardo, V., Arkko, J., Loughney, J., and G. Zorn, "Diameter Base Protocol", RFC 6733, October 2012.

Authors' Addresses

Daniel Corujo
Instituto de Telecomunicacoes
Campus Universitario de Santiago
Aveiro, P-3810-193 Aveiro
Portugal

Phone: +351 234 377 900
Email: dcorujo@av.it.pt

Kostas Pentikousis
Huawei Technologies
Carnotstrasse 4
10587 Berlin
Germany

Email: k.pentikousis@huawei.com

Ivan Vidal
UC3M
Av de la Universidad, 30
28911 Leganes, Madrid
Spain

Email: ividual@it.uc3m.es

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 14, 2013

D. Kutscher
NEC
S. Farrell
E. Davies
Trinity College Dublin
February 10, 2013

The NetInf Protocol
draft-kutscher-icnrg-netinf-proto-01

Abstract

This document defines a conceptual protocol and corresponding node requirements for NetInf nodes in a NetInf network. A NetInf network offers an information-centric paradigm that supports the creation, location, exchange and storage of Named Data Objects (NDOs). NetInf nodes can provide different services to other NetInf nodes, e.g., forwarding requests for information objects, delivering corresponding response messages, name resolution services etc. This (abstract) protocol is intended to be run over some "convergence layer" that handles transport issues. Two "wire" formats are defined, one that uses HTTP for message transfer and one layered on UDP.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any 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 August 14, 2013.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents

(<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document.

Table of Contents

1. Introduction	3
2. Principles and Assumptions	3
3. Convergence Layer Architecture	5
4. The NetInf Protocol - Overview	7
5. Protocol Details	9
5.1. GET/GET-RESP	9
5.2. PUBLISH/PUBLISH-RESP	11
5.3. SEARCH/SEARCH-RESP	13
6. Convergence Layer Specifications	14
6.1. HTTP CL	14
6.2. UDP CL	17
7. Security Considerations	19
8. Acknowledgments	19
9. References	20
9.1. Normative References	20
9.2. Informative References	21
Authors' Addresses	21

1. Introduction

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. [RFC2119]

Syntax definitions in this memo are specified according to ABNF [RFC5234].

There is an open-source implementation available that implements (most of) this. See <http://sourceforge.net/projects/netinf/> for code and <http://village.n4c.eu/getputform.html> for access to a test server.

2. Principles and Assumptions

A NetInf network provides an information-centric networking (ICN) environment in which units of data content can be identified and accessed using a URI-based naming scheme. NetInf nodes in a NetInf network support the creation, location, exchange and storage of these units of content. In order to support interoperable implementation of the NetInf design, [ref.netinf-db2] [ref.netinf-db3] the following assumptions are made here:

- o all nodes can take on all NetInf roles (but do not have to);
- o as necessary, nodes may access a Name Resolution System (NRS) and/or a (possibly name based) message routing infrastructure for NetInf messages; and
- o the NetInf protocol can be used directly to access content.

The NetInf protocol operates on Named Data Objects (see [ref.netinf-db2]) referred to as NDOs. An NDO is an ordered collection of octets associated with a name. The NetInf protocol is designed to cache, locate and transmit complete NDOs.

The NetInf protocol is specified so that NDOs can in principle be retrieved from nodes anywhere in the network to which messages can be routed. This routing is intended to be driven by the names of the NDOs, with the option to use an NRS, but this specification does not discuss how routing, nor calling to an NRS, is carried out. Routing will also depend on the underlying Convergence Layer protocol (see Section 3) in use at that node.

Nodes offering NetInf services may return locators in some cases. These locators designate network locations where an NDO might

potentially be available for retrieval, but locators may not be usable outside of some (possibly hard-to-characterise) domain, or for more than a limited period of time, due to mobility of nodes or time limited access through "pinholes" in middleboxes such as firewalls and Network Address Translators (NATs). Accordingly, a design goal is to enable preferential use of names, with locators mostly as hints to improve efficiency. For this reason one can argue that locators ought not be made available to applications using the NetInf protocol in a form that would allow them to try to use the locator outside the NetInf protocol. NDOs may have multiple locators in order to indicate specific interfaces or to reflect attachment to multiple addressing domains. Locators also typically map to a specific instance (copy) of an NDO residing at a given host.

Locators are an example of NDO associated data that may be stored in association with the data content of the NDO. Other types of such data include "metadata" relating to the data content of the NDO, information describing the history of the copy of the NDO in the node where it is stored and search terms that are applicable to the NDO content. The term "affiliated data" will be used to describe the overall set of data, other than the actual octets of the content, stored or transmitted in association with an NDO. This affiliated data could be described as metadata of the NDO but we will reserve that term for a subset of the affiliated data that is usually constructed by the publisher of the NDO describing the content data of the NDO that is sent out in tandem with the data content. The NetInf protocol allows this affiliated data to be transmitted, in whole or in part, in association with NetInf messages.

NDO names will often be based on hash-function output values, and since the preferred hash-function will change over time and may change depending on location, this implies that NDOs can also have more than one name. There may also be cases where truncated hash values are desired (e.g., in cases where packets must be kept below some small size and fitting an entire request into one packet is required), and in such cases collisions will occur, or can easily be generated by bad actors. There are also cases where it is desirable to use a name to refer to some "dynamic" NDO, whose octets change (e.g., perhaps the current weather report) and there are cryptographic methods for doing this. This all means that there is no strict 1:1 mapping between names and NDOs, however, we do expect that for most objects, for most ICN deployments, there will in practice be one NDO that is named by each name. That is, each name usually does refer to just one object, and this protocol is designed to work best for that case.

The following NetInf services are assumed to be implemented on nodes through the NetInf protocol:

- o caching of NDOs, both locally originated and acquired through network operations with the NetInf protocol;
- o requesting the fetching of an NDO using its name, possibly with the addition of a locator, from elsewhere in the network;
- o responding to NetInf protocol NDO fetch operations using a name referring to one of its locally known NDOs, which may have been locally generated or acquired from another NetInf node and cached here, by returning either or both of the data named in the operation or affiliated data including locator(s) referring to a node where that NDO is (assumed to be) available;
- o initiating a search for NDOs matching specified search criteria;
- o responding to search requests received by determining if any locally known NDOs meet the search criteria according to locally determined algorithms;
- o NDO publication via sending out the name and, optionally, either or both of the content data and some affiliated data, such as locators, to other nodes;
- o according to locally determined policy, the ability to accept or reject NDO publication requests that are delivered to the node, and to cache either or both of the objects and/or information about those that are accepted;
- o according to locally determined policy, after carrying out local processing, the ability to forward NetInf messages to other nodes or discard them;
- o managing the data affiliated with the NDO as well as the content data; and
- o local cache management, driven by local policy and (optionally) whatever cache directives are carried in NetInf messages.

3. Convergence Layer Architecture

The idea of the Convergence Layer (CL) is to provide a means to transport NetInf messages between pairs of nodes that offer NetInf services. Any protocol that allows NetInf messages to be passed without loss of information can be used as a NetInf Convergence Layer (NetInf-CL) protocol.

This document does not cover the bit-level specification of any CL

protocol. The individual CL protocols will provide their own specification regarding their bit-level format.

Different CLs can be used in the various regions forming a global NetInf network. Where a message has to pass through several intermediate NetInf-capable nodes from source to destination, the NetInf protocol layer at each node is responsible for selecting the appropriate link and CL to forward messages.

Each CL has to offer the following minimal set of capabilities:

- o unidirectional point-to-point transport of NetInf messages from source to destination,
- o preservation of message boundaries,
- o reliable transmission of message octets, and
- o in-order delivery of message octets to the destination node.

If an underlying protocol used by a particular CL cannot offer these capabilities natively, then the CL is responsible for synthesising these capabilities by appropriate means, e.g., use of retransmission or insertion of sequence numbers. However, this does not prevent a CL that uses a more capable underlying protocol from implementing additional capabilities, e.g., bidirectional connections that allow a single connection to send NDOs in both directions.

The CL itself does not specify the properties of the messages, how they are interpreted, and the way nodes should interact with them, as that is what is specified in the present document.

The CL architecture is inspired by, and similar to, the concept used in Delay-Tolerant Networking. [RFC4838][RFC5050].

However, in contrast to DTN-CLs, the NetInf-CL concept does not include the handling of fragments of an NDO "above" the CL. This is the main difference between the CL concept as used in DTNs and ICNs. Put another way, a DTN-CL may result in a bundle being fragmented, and those fragments are only re-assembled at the final bundle destination. In the case of an NetInf-CL, if an NDO is fragmented or chunked within the CL, then those fragments or chunks are reassembled at the next ICN node and that fragmentation or chunking is not visible to the ICN protocol. One can also consider that the DTN Bundle Protocol (BP)[RFC5050], which runs over a DTN-CL, can itself, with an appropriate extension such as the "BPQ" extension, [I-D.farrell-dtnrg-bpq] be an NetInf-CL. That is, a concrete instance of this protocol could use the BP with the BPQ extension as

an NetInf-CL.

4. The NetInf Protocol - Overview

This protocol assumes that NDOs are named using URIs, and in particular via the "ni" URI scheme [I-D.farrell-decade-ni] which **MUST** be supported. There are a set of extensions to the "ni" URI scheme [I-D.hallambaker-decade-ni-params] that **MAY** be supported by nodes. However, other URI forms **MAY** also be used in the NetInf protocol, in particular as locators, and nodes **SHOULD** support at least fetching of "http" URLs.

Nodes are assumed to be capable of discriminating between names and locators, based on the URI scheme or otherwise.

The most common operations for a NetInf node will be fetching (using a GET message) an NDO or responding to such queries. The response to the GET message will, if possible, contain the octets making up the specified NDO and **MAY** contain

- o one or more URIs (typically locators) that could subsequently be used to retrieve the octets of the NDO either via this NetInf protocol or by alternative, locator-specific, means, and/or
- o other affiliated data such as metadata relevant to the NDO.

There are some circumstances in which it **MAY** be appropriate for the response to the GET message to contain only one or more locators and, optionally, other affiliated data. Examples of this situation occur if the responding node is aware that the object content can be returned more effectively using an alternative protocol or from an alternative source because of bandwidth limitations on the links connecting the responding node.

In addition to GET, there is the analagous PUBLISH operation where one node sends URIs and/or NDO octets to another. There is also a SEARCH operation, where one node submits a search query and receives a set of URIs and optional meta-data in response.

GET, PUBLISH and SEARCH messages **MAY** be forwarded by any node that receives them if there is good reason and local policy indicates that this would not result in excessive usage of network resources.

If a request message is forwarded, then a response message **MUST NOT** be sent for that request while the overall "transaction" is still in progress. That is, a node that forwards a request does not answer that request itself until it gets an answer from elsewhere.

Response messages MUST be forwarded by routers to the node from which the corresponding request message was received. The routing mechanisms that are used to ensure responses are correctly forwarded in this way are not specified here.

Since this specification does not determine how message routing, nor use of an NRS is done, we do not otherwise specify how or when messages are to be forwarded.

Nodes that want to make a locally stored NDO available with a specific name can use the PUBLISH message to announce that data to the network. This message MAY "push" the octets of the NDO into other nodes' caches. (If those nodes are willing to take them.) The reasoning behind this is that in many circumstances pushing just a name or a locator will not be helpful because the node with the NDO may be located behind a middlebox that will not allow access to the data from "outside." Pushing the complete NDO to a node that is accessible from the originating node but is also accessible from outside the middlebox "interior," can allow global access, e.g., by caching the NDO on a server in the DMZ ("DeMilitarized Zone") of an enterprise network or in a server provided by a home user's ISP.(Internet Service Provider). The publisher MAY also push affiliated data for the NDO, including additional locators and content metadata that can be stored in a node's NDO cache. The caching node MAY choose to store just the affiliated data without the content data depending on local policy.

As in the case of routing messages generally, this specification does not determine the node(s) to which an NDO can be "pushed."

Finally, NetInf nodes can send a SEARCH message to other NetInf nodes. In response, a NetInf node can perform a local search (i.e., of its local cache) As a response, any of the NetInf nodes that receives the SEARCH message returns a set of "ni" URIs of objects matching the search query. It may also return other types of URI such as "http" URIs. Searching of a node's local cache is the main goal for the SEARCH operation, but if a set of nodes were to forward SEARCH messages, then a global search (e.g., a Google-like service) service could be offered.

NDOs together with any affiliated data are represented using MIME objects. [RFC2045]. Placing as much of the affiliated data linked to the NDO in a multipart MIME object along with the octets of the actual object allows for significant specification and code re-use. For example, we do not need to invent a new typing scheme nor any associated registration rules nor registries.

As an example we might have a MIME object of that is multipart/mixed

and contains image/jpeg and application/json body parts, with the named image in the former and loosely structured associated data in the latter. The "ni" scheme parameters draft discusses such examples. This means that the details of the verification of name-data integrity supported by the ni name scheme also depend on the MIME type(s) used.

MIME also simplifies the specification of schemes that make use of digital signatures, reusing techniques from existing systems including Secure MIME (S/MIME) [RFC5751] and the Cryptographic Message Syntax (CMS) [RFC5652].

Note that (as specified in [I-D.farrell-decade-ni]) two "ni" URIs refer to the same object when the digest algorithm and values are the same, and other fields within the URI (e.g., the authority) are not relevant. Two ni names are identical when they refer to the same object. This means that a comparison function for ni names MUST only compare the digest algorithms and values.

5. Protocol Details

We define the GET, PUBLISH and SEARCH messages in line with the above. GET and PUBLISH MUST be supported. SEARCH SHOULD be supported. Each message has an associated response.

This means that GET and PUBLISH MUST be implemented and SEARCH SHOULD be implemented. In terms of services, GET and PUBLISH SHOULD be operational but SEARCH MAY be turned off.

5.1. GET/GET-RESP

The GET message is used to request an NDO from the NetInf network. A node responding to the GET message would send a GET-RESP that is linked to the GET request using the msg-id from the GET message as the msg-id for corresponding GET-RESP messages if it has an instance of the requested NDO.

The "ni" form or URI MUST be supported. Other forms of URI MAY be supported.

The msg-id SHOULD be chosen so as to be highly unlikely to collide with any other msg-id and MUST NOT contain information that might be personally identifying, e.g., an IP address or username. A sufficiently long random string SHOULD be used for this.

The ext field is to handle future extensibility (e.g., for message authenticators) and allows for the inclusion of a sequence of type,

length value tuples. No extensions for GET messages are defined at this point in time.

```
get-req = GET msg-id URI [ ext ]  
get-resp = status msg-id [ 1*URI ] [ ext ] [ object ]  
  
ext = json-coded-string
```

Figure 1: GET/GET-RESP Message Format

Any node that receives a GET message and does not have an instance of the NDO referenced in the message **MUST** either

- o forward the message to another node, or
- o generate a GET response message with an appropriate status code and the msg-id from the GET message as the response msg-id.

If the message is forwarded, the node **SHOULD** maintain state that will allow it to generate the GET response message if a matching response message is not received for forwarding within a reasonable period of time after the GET message was forwarded.

If the node has an instance of the NDO, the response **MAY** contain zero or more URIs that **MUST** be either locators for the specified object or else alternative names for that object. If the receiving node has a copy of the relevant object in its cache it **SHOULD** include the object in the response. Possible reasons for not including the object would include situations where the GET message was received via a low-bandwidth interface but where the node "knows" that returning a locator will allow the requestor faster access to the object octets. Alternatively, the node may only be maintaining the affiliated data for the NDO and not the content data if it has not yet received the content data or has discarded it due to cache size limitations.

The object **MUST** be encoded as a MIME object. If there is affiliated data linked to the object this **MUST** also be encoded using MIME and integrated with the object in a multipart/mixed MIME object.

If the receiving node does not have a cached copy of the object it **MAY** choose to forward the message depending on local policy. Such forwarding could be based on name-based routing, on an NRS lookup or other mechanisms (e.g. a node might have a default route).

If an get-resp is received with an object that is not MIME encoded or of an unknown MIME type then that **MUST** be treated as an application/

octet-stream for the purposes of name-data integrity verification.

get-resp messages MAY include extensions as with all others.

5.2. PUBLISH/PUBLISH-RESP

The PUBLISH message allows a node to push the name, and optionally, alternative names, locators, a copy of the object octets and/or object meta-data. Ignoring extensions, only a status code is expected in return.

A msg-id MUST be included as in a GET message.

A URI containing a name MUST be included. The "ni" URI scheme SHOULD be used for this name.

The message MAY also contain additional URIs that represent either alternative names or locators where the identical object can be found and metadata relating to the published content. As mentioned in Section 4 it is the responsibility of the receiving node to discriminate between those URIs used as names and those used as locators.

The object octets MAY be included. This is intended to handle the case where the publishing node is not able to receive GET messages for objects. An implementation SHOULD test (or "know") its local network context sufficiently well to decide if the object octets ought to be included or not. Methods for checking this are out of scope of this specification.

A node receiving a PUBLISH message chooses what information from the message, if any, to cache according to local policy and availability of resources. It is RECOMMENDED that a node that receives a PUBLISH message containing the object octets verify that the digest in the name under which the content is published matches with the digest of the received data.

One way to "fill a cache" if the object octets are not included in the PUBLISH would be for the recipient of the PUBLISH to simply request the object octets using GET and cache those. (There is no point in sending a PUBLISH without the octets and without any locator.) This behaviour is, of course, an implementation issue.

In some cases it may make sense for a (contactable) node to only publish the name and metadata about the object. The idea here is that the metadata could help with routing or name resolution or search. Since we are representing both NDO octets and affiliated data such as the metadata as MIME objects, we need to tell the

receiver of the PUBLISH message whether or not that message contains the full object. We do this via the "full-ndo-flag" which, if present, indicates that the PUBLISH message contains enough data so the receiver of the PUBLISH message has sufficient data to provide a complete answer a subsequent GET message for that name, i.e., data content and affiliated data.

If a node receives a PUBLISH message for an NDO which already exists in its cache, the received information SHOULD be used to complete or update the node's cached information for the NDO:

- o If the object octets are included and the node currently does not have the octets cached, the data content MAY be added to the cache. Again it is RECOMMENDED that the received data has the correct digest as specified in the NDO name, and
- o Items in the affiliated data MAY be merged into cached affiliated data, including adding additional locators to the list of known locators for the NDO and merging any content metadata with previously received metadata. If there is a conflict, the choice of metadata to be stored is a matter of policy.

It is RECOMMENDED that a timestamp be recorded whenever the cached information for an NDO is updated and that this timestamp be stored in the affiliated data and the most recent timestamp returned with any subsequent GET or SEARCH request that references the NDO.

Extensions ("ext") MAY be included as in a GET request. One such HTTP CL-specific extension ("meta") is defined in Section 6.1 below.

```
pub-req = PUBLISH msg-id 1*URI [ ext ] [ [ full-ndo-flag ] object ]
pub-resp = status msg-id [ ext ]
```

Figure 2: PUBLISH/PUBLISH-RESP Message Format

The response to a PUBLISH message is a status code and the msg-id from the PUBLISH message and optional extensions.

A node receiving a PUBLISH message MAY choose to forward the message to other nodes whether or not it chooses to cache any information. If this node does not cache the information but does forward the PUBLISH message, it should postpone sending a response message until a reasonable period of time has elapsed during which no other responses to the PUBLISH message are received for forwarding. However, the node MAY send an extra response message, even if it forwards the PUBLISH message, if the sender of the PUBLISH message

would have expected the receiving node to cache the object (e.g., because of a contractual relationship) but it was unable to do so for some reason.

5.3. SEARCH/SEARCH-RESP

The SEARCH message allows the requestor to send a set of query tokens containing search keywords. The response is either a status code or a multipart MIME object containing a set of metadata body parts, each of which MUST include a name for an NDO that is considered to match the query keywords.

```
search-req = SEARCH msg-id [ 1*token ] [ ext ]  
search-resp = status msg-id [ results ] [ ext ]
```

Figure 3: SEARCH/SEARCH-RESP Message Format

In the case where the response contains results, these MUST take the form of an application/json MIME object containing an array of results. Each result MUST have a "name" field with a URI as the value of that field. Any other fields in array elements SHOULD contain metadata that is intended to allow the requestor to select which, if any, of the names offered to retrieve.

The URIs included in a search-resp SHOULD be names, but MAY be locators, to be distinguished by the requestor as in the case of GET responses.

The intent of the SEARCH message is to allow nodes to search one another's caches, but without requiring us to fix the details (ontology) for NDO content metadata. While this main intended use-case does not involve forwarding of SEARCH messages that is not precluded.

As with PUBLISH messages, if a SEARCH message is forwarded, the forwarding node postpones sending an empty SEARCH response until a reasonable time is elapsed to see if alternative node responds to the SEARCH.

If a SEARCH at a node identifies an NDO that is included in the results of a search, the tokens that were used for the search MAY be recorded in the affiliated data cached with the NDO. Each set of search tokens for which a "match" is obtained should be recorded separately resulting in an array of set of tokens. If the search mechanisms used provides a reliability measure, this MAY also be recorded and the measure may be used to limit the size of the search

tokens array by discarding (or never inserting) sets of tokens with low reliability scores.

SEARCH messages MAY include extensions as for other messages.

6. Convergence Layer Specifications

This section specifies two convergence layers that represent instantiations of the NetInf protocol. The first, based on HTTP, is intended for using NetInf in existing web infrastructures, whereas the second, based on UDP, provides an efficient datagram-based hop-by-hop message transport that can be used to query for GET requests sent to an NRS node or for multicasting such requests in a local network.

6.1. HTTP CL

The HTTP CL maps the NetInf protocol to HTTP, ensuring interoperability with existing web infrastructure (client and server implementations as well as installed proxies).

All NetInf protocol requests are mapped to HTTP POST requests [RFC2616]. The corresponding NetInf protocol responses are mapped to the HTTP response messages of such HTTP POST request messages.

The HTTP CL assumes that the client knows the address of the HTTP server to which it will send requests. Clients MAY use the authority part of an ni URI, if one is present to select the HTTP responder. NetInf HTTP responders MUST accept requests sent to the following paths:

/netinfproto/get for NetInf GET requests

/netinfproto/publish for NetInf PUB requests

/netinfproto/search for NetInf SEARCH requests

So for example a client would send an HTTP POST request containing a NetInf GET to `http://example.com/netinfproto/get`

NetInf HTTP responders SHOULD also make ni URIs available at the relevant well-known URL [RFC5785] for the ni URI.
[I-D.farrell-decade-ni]

NetInf protocol requests use HTML forms, and as specified in [W3C.REC-html401-19991224] the form data set for the HTTP POST messages is included in the body of the form. The content type of

the form data body depends on the actual NetInf protocol request type (see below). Receivers MUST accept both 'application/x-www-form-urlencoded' and 'multipart/form-data' for all requests.

The mapping of the fields from the abstract protocol is as shown in Figure 4.

Abstract Protocol Field	Form field	Comments (field type in form)
URI	urival, URI loc1,loc2	usually an ni URI (text) or locator
msg-id	msgid	a message identifier (text)
ext	ext	extension(s) (JSON encoded string)
full-ndo-flag	fullPut	true if object supplied (checkbox)
object	octets	object octets (file specification)
n/a	rform	response format required, can be "html" or "json" (radio)
token	tokens	one text field with all search keywords (text)

Figure 4: Form fields used in NetInf requests

Notes for Figure 4:

For GET messages: 'application/x-www-form-urlencoded' SHOULD be used as a content type.
 "URI" and "msgid" parameters are MANDATORY.
 "loc1" and "loc2" are OPTIONAL.
 "ext" may be used in future but no values currently defined.

For PUBLISH messages: 'application/x-www-form-urlencoded' SHOULD be used as a content type if the request does not contain an object. If the request contains an object (also see the description of the fullPut parameter below), 'multipart/

form-data' MUST be used as a content type.
"URI" and "msgid" are MANDATORY.
"loc1", "loc2", "ext", "rform" and "fullPut"
are OPTIONAL.
If "rform" is absent, the "json" value is
assumed.
If "fullPut" is absent, a "false" value is
assumed.
If "fullPut" is present and set to "true",
"octets" must be present.
If present, "octets" contains a file
specification and the object octets.
If present, "ext" may contain a "meta" item.
The value of "ext" MUST be a JSON object
string and the value of the "meta" item MUST
be a (subsidiary) object, e.g., the "ext"
string might be
{ "meta": { "mi1": 5, "mi2": { ... },
"mi3": "abcd". "mi4": [...] } }

For SEARCH messages: 'application/x-www-form-urlencoded' SHOULD be
used as a content type.
"msgid" and "tokens" are MANDATORY.
"rform" is OPTIONAL.
If "rform" is absent, the "json" value is
assumed.
"ext" may be used in future but no values
currently defined.

HTTP responses for each request can differ.

For GET, the a successful HTTP response (HTTP response code 2xx) MUST
contain either an application/json (if no object is returned) or else
a multipart/mixed with exactly two body parts, the first being of
content type 'application/json' and the second containing the object
octets, with whatever MIME type is appropriate.

The application/json component MUST consist of a JSON object that
SHOULD contain the following named fields:

NetInf	A string describing the version of the NetInf protocol in use (e.g., "V0.1a").
ni	The "canonicalized" form of the NDO as a URI in the ni scheme: "canonicalized" means that the URI has empty netloc and query string fields. For example: "ni:///sha-256-64;gf2yhPY9Mu0" or "nih:/ sha256-32;81fdb284;d".

msgid	The value of the msgid field in the GET message that resulted in this response.
ts	The timestamp of the last update of the cached information in the cache from which the NDO is being sent.
status	A code, taken from the HTTP 2xx response codes indicating what has been returned (200 if both affiliated data and content has been returned and 203 if only affiliated data is returned).
ct	The MIME content type of the NDO content data, if known. Empty string if not yet known.
loclist	Array of locator names (strings) from where the NDO might potentially be retrieved.
metadata	A JSON object containing any named items copied in from "meta" object(s) supplied by any PUBLISH messages received at the node that sent the response plus an entry named "publish" which contains a string indicating the class of node and software that generated the cache entry.
searches	A JSON array of objects each containing a set of strings representing search tokens and information about the search mechanism that resulted in a match with the NDO during a previous search.

For PUBLISH, the HTTP response will contain an application/json or text/html response, depending on the value of the rform form field. (If rform is missing json is the default.) The application/json structure is as for a GET response. The text/html document will provide a report of the successful publication of the NDO and whatever other relevant information from the affiliated information seems appropriate for inspection by a human user.

For SEARCH, the HTTP response will contain an application/json or text/html response, depending on the value of the rform form field. (If rform is missing json is the default.) The application/json structure is similar to the previous structures, but has a "results" object that contains an array of object details.

6.2. UDP CL

The UDP CL implements the NetInf protocol with a UDP datagram services, i.e., all NetInf messages are mapped to individual UDP

messages. The purpose is to provide a light-weight datagram-based CL that can be used to implement NetInf transport protocols on top and that can provide efficient communication for querying NRSs, and request broadcasting/multicasting. The UDP CL provides no hop-by-hop flow control, retransmission and fragmentation/re-assembly.

The UDP CL has two sending modes: 1) send to specified destination IP address and 2) send to the well-known IPv4 multicast address 225.4.5.6. For both unicast and multicast the UDP port number is 2345. All request and response messages are JSON objects, i.e., unordered sets of name/value pairs.

For UDP CL messages, the following JSON names for name/value pairs are defined (not all objects have to be present in all messages):

```
version  # the NetInf UDP CL protocol version -- currently
          # "NetInfUDP/1.0"

msgType  # the message type (e.g., GET)

uri      # the NI URI

msgId    # the message ID (must be unique per CL hop and
          # request/response pair)

locators # an array of locators

instance # an UDP CL speaker identifier (must be unique per IP host,
          # e.g., process ID and per process ID)
```

Figure 5: UDP CL JSON request structure

This version of the specification defines the GET request and the corresponding GET response only.

GET request A GET request provides the following objects:

```
version:  "NetInfUDP/1.0"

msgType:  "GET"

uri:      name of the requested NDO
```


msgId: message ID (see above)

GET reponse A GET response provides the following objects:

version: "NetInfUDP/1.0"

msgType: "GET-RESP"

uri: name of the requested NDO

msgId: message ID (see above)

locators: a list of locator strings

7. Security Considerations

For privacy preserving reasons requestors SHOULD attempt to limit the personally identifying information (PII) included with search requests. Including fine-grained search keywords can expose requestor PII. For this reason, we RECOMMEND that requestors include more coarse grained keywords and that responders include sufficient meta-data to allow the requestor to refine their search based on the meta-data in the response.

Similarly, search responders SHOULD consider whether or not they respond to all or some search requests as exposing one's cached content can also be a form of PII if the cached content is generated at the behest of the responder.

Name-data integrity validation details are TBD for some common MIME types.

Users need to be aware that the affiliated data is NOT protected by the name-data integrity as this applies only to the data content octets.

[[More TBD no doubt.]]

8. Acknowledgments

This work has been supported by the EU FP7 project SAIL (FP7-ICT-2009-5-257448).

Claudio Imbrenda and Christian Dannewitz contributed to early versions of this document whilst working at NEC and the University of Paderborn respectively.

Petteri Poeyhoenen and Janne Tuononen helped with interop testing and corresponding feedback.

9. References

9.1. Normative References

- [I-D.farrell-decade-ni]
Farrell, S., Kutscher, D., Dannewitz, C., Ohlman, B., Keraenen, A., and P. Hallam-Baker, "Naming Things with Hashes", draft-farrell-decade-ni-10 (work in progress), August 2012.
- [I-D.hallambaker-decade-ni-params]
Hallam-Baker, P., Stradling, R., Farrell, S., Kutscher, D., and B. Ohlman, "The Named Information (ni) URI Scheme: Optional Features", draft-hallambaker-decade-ni-params-03 (work in progress), June 2012.
- [RFC2045] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, November 1996.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.
- [RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.
- [RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, September 2009.
- [RFC5785] Nottingham, M. and E. Hammer-Lahav, "Defining Well-Known Uniform Resource Identifiers (URIs)", RFC 5785, April 2010.
- [W3C.REC-html401-19991224]
Hors, A., Raggett, D., and I. Jacobs, "HTML 4.01 Specification", World Wide Web Consortium Recommendation REC-html401-19991224, December 1999, <<http://www.w3.org/TR/1999/REC-html401-19991224>>.

9.2. Informative References

- [I-D.farrell-dtnrg-bpq]
Farrell, S., Lynch, A., Kutscher, D., and A. Lindgren,
"Bundle Protocol Query Extension Block",
draft-farrell-dtnrg-bpq-01 (work in progress), March 2012.
- [RFC4838] Cerf, V., Burleigh, S., Hooke, A., Torgerson, L., Durst,
R., Scott, K., Fall, K., and H. Weiss, "Delay-Tolerant
Networking Architecture", RFC 4838, April 2007.
- [RFC5050] Scott, K. and S. Burleigh, "Bundle Protocol
Specification", RFC 5050, November 2007.
- [RFC5751] Ramsdell, B. and S. Turner, "Secure/Multipurpose Internet
Mail Extensions (S/MIME) Version 3.2 Message
Specification", RFC 5751, January 2010.
- [ref.netinf-db2]
SAIL, "NetInf Content Delivery and Operations", SAIL
Project Deliverable D-3.2 , May 2012.
- [ref.netinf-db3]
SAIL, "Final NetInf Architecture", SAIL Project
Deliverable D-3.3 , January 2013.

Authors' Addresses

Dirk Kutscher
NEC
Kurfuersten-Anlage 36
Heidelberg,
Germany

Phone:
Email: kutscher@neclab.eu

Stephen Farrell
Trinity College Dublin
Dublin, 2
Ireland

Phone: +353-1-896-2354
Email: stephen.farrell@cs.tcd.ie

Elwyn Davies
Trinity College Dublin
Dublin, 2
Ireland

Phone: +44 1353 624 579
Fax:
Email: davieseb@scss.tcd.ie
URI:

INTERNET-DRAFT
Intended Status: Informational
Expires: August 21, 2013

Rong Wang
Chunfeng Yao
Zhefeng Yan
Huawei
February 17, 2013

Container Resolution System in ICN
draft-wang-icnrg-container-resolution-system-00

Abstract

We illustrate the architecture and mechanism of container resolution system, and to the protocol of resolving containers in ICN. The fundamental characters are: 1) Use Interest and Data packet as the communication primitives, 2) Every routing node has FIB, CS, PIT tables to do packet routing and caching. Current NDN/CCN architecture does not require a resolution system. The routing is totally based on prefixes of content names, which causes scalability and mobility issues described in [Cisco-Name]. This draft addresses the aforementioned issues.

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at
<http://www.ietf.org/lid-abstracts.html>

The list of Internet-Draft Shadow Directories can be accessed at
<http://www.ietf.org/shadow.html>

Copyright and License Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1	Container Name System	3
1.1	Introduction	3
1.2.	System Architecture	3
1.3.	System Components of CNS	4
2	Container Name Scheme	5
2.1	Content Name Composition	5
2.2.	Container name Registration	5
2.3	Unregister Container name	7
2.4.	Resolve Container name	7
2.5.	Packet flow in network	9
3	Implementation of CNS	11
4	Execution Engine Process Logic	13
5	IANA Considerations	14
6	Conclusions	14
7	References	14
7.1	Normative References	14
7.2	Informative References	14
	Authors' Addresses	16

1 Container Name System

1.1 Introduction This draft is the companion to [Huawei-name] and illustrates the container resolution system (hereinafter referred to as NRS) . The NRS assists the routing requirement of the container name, which is the extension of conventional Information Centric Networking (hereinafter referred to as ICN) naming.

1.2. System Architecture

An NRS is composed by a number of distributed container name systems (hereinafter referred to as CNS) deployed in an ICN network. CNS maintains the mapping from containers to its access containers. The concept of container, container name, container set , container assisted routing, resolvable container name, and access container are described in [Huawei-name]. The following figure illustrates the high level CNS architecture.

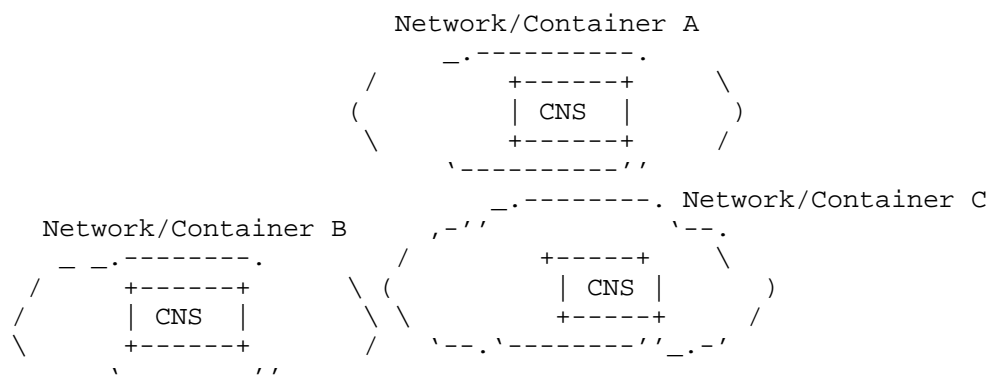


Figure 1

For each container name, there is an authoritative CNS (hereinafter referred to as A-CNS) acting as its final name resolution provider and usually residing within the container (network). One container can have an A-CNS for multiple containers. Other CNS can identify the A-CNS of its container via the container name, e.g., with one of the following methods:

- Use category tree hierarchy similar to current DNS, and configure an A-CNS for each container.

- Leverage container assisted ICN routing mechanism to identify the A-CNS of the container.

Since containers are part of the hierarchical tree or DAG of a content name, they have to know their predecessor and successor(s) in

the tree or DAG [Huawei-name]. Thus CNS within a container has the similar knowledge of its predecessor and successor(s). If a CNS of a container cannot locate the A-CNS of its container, it asks the help from its upper level CNS.

Registration and unregistration operation of a container has to be done on its A-CNS. Other CNS, for example the CNS for the local access container, proxies the registration and unregistration requests to the A-CNS. In other words, the final resolution result of a container name must come from its A-CNS. For any resolution request, if an intermediate CNS has the result cached, then it can reply directly, otherwise this CNS has to forward the request to its A-CNS, and caches the resolution result according to the predefined policy.

1.3. System Components of CNS

CNS has 3 components, DataBase (hereinafter referred to as DB), Execution Engine, and Network interfaces.

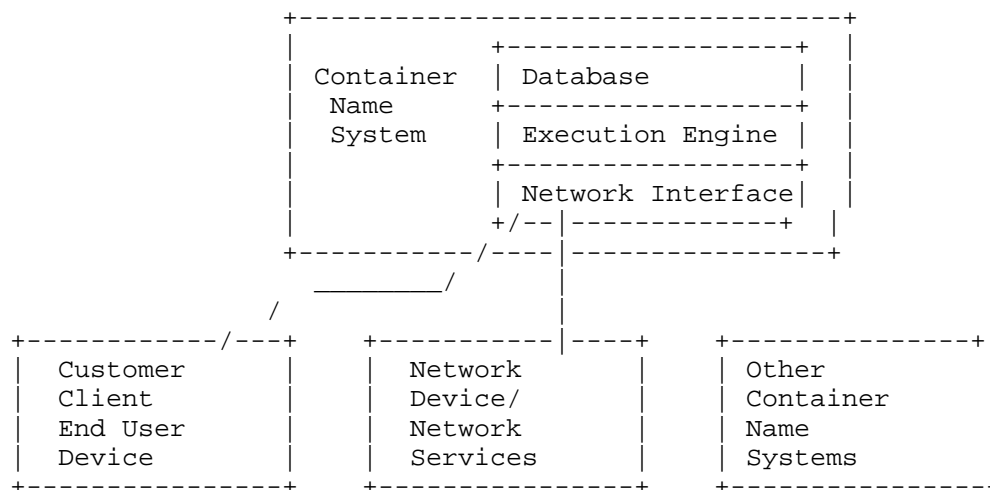


Figure 2

Database stores resolution related data. Implementation details can leverage current DB technologies or file systems.

Execution engine is responsible for processing resolution related logic. It analyzes packet from network interface, operates on DB, and sends operation result out of Network interfaces.

Network interfaces sends/receives ICN packets according to ICN protocols. It receives Interest packet from client, end device, network node, service or other CNS, and sends packet to the execution

engine for further processing. It forwards Data packet by following the instruction from execution engine.

2 Container Name Scheme

2.1 Content Name Composition

We propose to reserve a few keywords in ICN for the container name resolution operations. They are RESOLVE\REGISTER\UNREGISTER, representing 3 primitive operations: as inquiry, registration and unregistration, respectively. Keywords are configured automatically or manually during ICN joining into the network.

Once a CNS with resolution capability joins into ICN, it publishes aforementioned keywords(RESOLVE\ REGISTER \ UNREGISTER) according to the ICN rules. Any ICN node who receives the published packet will add table entries associated with those keywords to its FIB. All those entries point to the CNS.

We then prefix resolution keyword to the content ID required resolution to be the new content ID of ICN packet (Interest/Data). Since each node in ICN use LPM to lookup FIB for forwarding, any match of the keyword will route the resolution request to the aforementioned CNS.

We can also use container assisted resolution to route packet to a CNS. In this case, CNS publishes the name of the container that it resides in [super container] to ICN according to ICN rules. Any node in ICN will add routes accordingly. We can use the routing protocol described in draft [Huawei-name] to route interest to CNS by adding super container name as the suffix to aforementioned Content ID.

For example, CNS has published that it resides in the super container named RESOLVER to the ICN. The interest to inquiry "chinamobile/johndoe" can be composed as "RESOLVE/chinamobile/johndoe|RESOLVER".

2.2. Container name Registration

The registration operation utilizes ICN protocol's Interest packet to carry the request, and uses Data packet to carry the reply (resolution result).

In details, information of container A registration includes:
"replace (Yes/No){access provider Container B/Container set C with its properties (attributes)}", for example: "(replace=yes;){cn/gd/sz ; TTL =100 | hostsrv.com; resolvable=yes; TTL =5000; }". Where:
-Value of Replace field can be Yes or No (True/False?) If container A

exists in CNS,
 --If Replace value is No: Insert the new resolution result to the front of result list. If container B or container set C has already existed in CNS, then delete the old result.
 --If replace value is Yes: Directly replace the existing resolution result with the new one.
 --By default, "replace=no", even with the absence of REPLACE field.
 --Example: "Register /chinamobile/johndoe" -> "(replace=yes;){cn/gd/sz ; TTL =100 | hostsrv.com; resolvable=yes; TTL =5000; }" For now, original resolution result of "chinamobile/johndoe" will be invalid and will be replaced by "cn/gd/sz" and "hostsrv.com"
 -Resolvable value (Yes/No): Whether the resolution result (container B/container set C) can be resolved again. The operation of recursive resolution can be seen as the expansion of resolution results. The expansion resolution can be done when writing to Database, or when resolution system idling, or when receiving a resolution request.
 -(Optional) TTL (Unit Second): Time of the resolution result can be cached. The result will not be cached if this value is 0.

Registration information can be put into Selector field. A new property, RESOLVE is added to represent name solution (see Figure 3) to hold register and unregister information.

In the Data packet, it also carries the success or failure information of the registration operation. In this reply Data packet, the "signed info" field has to set as "no caching" (This is implementation dependent), this setting ensures upcoming Interest requests will not be replied by the intermediaries but the CNS.

Aforementioned content name register and unregister information , it can be put directly into content name field rather than in Selector field. The whole content name will be composed of content ID and content name register/unregister information. As illustrated in Figure 4.

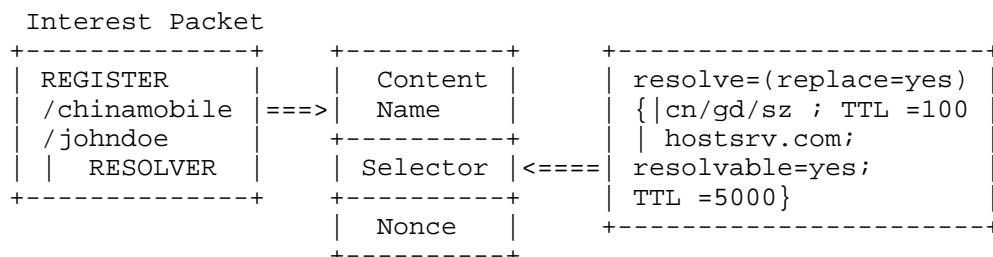


Figure 3

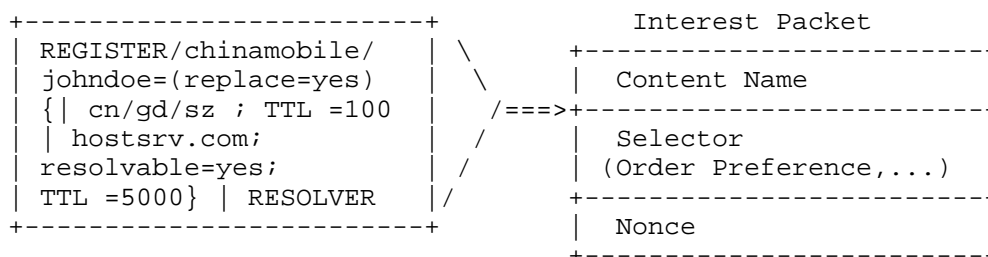


Figure 4

2.3 Unregister Container name

The purpose of this operation is to unregister the access container B/container set C for container A. In other words, container B/container set C is removed or voided from container A's resolution result list. This operation piggybacks on ICN protocol's Interest packet to carry the request, and leverages Data packet to carry the reply.

In details, the unregistration information of a container name looks like: "{access provider container B/container set C}". For example, "{cn/gd/sz ; | hostsrv.com;}". This information is in the new added properties in Selector field similar to that of registration operation.

-If container A does not exist in CNS, or there is no aforementioned container B/container set C as the resolution result of container A, then CNS replies failure status.

-If there are resolution results, remove them from CNS.

-If all resolution results are removed successfully, then reply "total success", otherwise, reply "partial success" with failed portions and reasons. The common reason is "non-exist".

-If resolution result of container A does not exist, then remove container A from CNS. The operation result will be carried back by Data Packet similar to that of registration. The Data Packet will be set as uncacheable.

2.4. Resolve Container name

The purpose of this operation is to obtain container B/container set C which provides access service for container A. This operation utilizes ICN protocol Interest Packet to carry the request, and Data packet to carry the resolution result.

In details, the carried packet has to include content name A. For example "chinamobile/johndoe"

The resolution result is carried back by Data packet in the Data field, for example, "airchina/cal314; resolvable=yes;ttl=5000; ||| cn/beijing; ttl=1000;||hostsrv.com ||| cn/gd ||| cn/beijing ||| us/ca", CNS will do the following:

-If content name A does not exist in the CNS, then replies "failed" or NULL.

-Return the resolution result of container A in order. If the result container name has the "resolvable=yes" property, then it can be further resolved. The resolution results can be inserted in the original result by following the method described in [Huawei-name]

-If the result has TTL property, the caching timer at Signed Info field of Data packet (implementation details may be different, for example, in CCN, it is FreshnessSecond, in NDN it is staledtime) will be set as the minimum of TTL (in previous example, it is 1000)

-The intermediaries ICN nodes can cache the resolution results in its Content Store (hereinafter referred to as CS). Cache Time can be processed according to ICN rules. Request packet is shown in Figure 5.



Data Packet		
Content Name	<===	RESOLVE/chinamobile/johndoe
Signature		airchina/cal314; resolvable=yes; ttl=5000
Signed Info	<===	cn/beijing; ttl=1000 hostsrv.com; resolvable=yes; ttl=1000
Data		cn/gd cn/beijing us/ca

Figure 5

2.5. Packet flow in network

Packet flow of a client/request node requesting container registration/unregistration is shown in Figure 6.

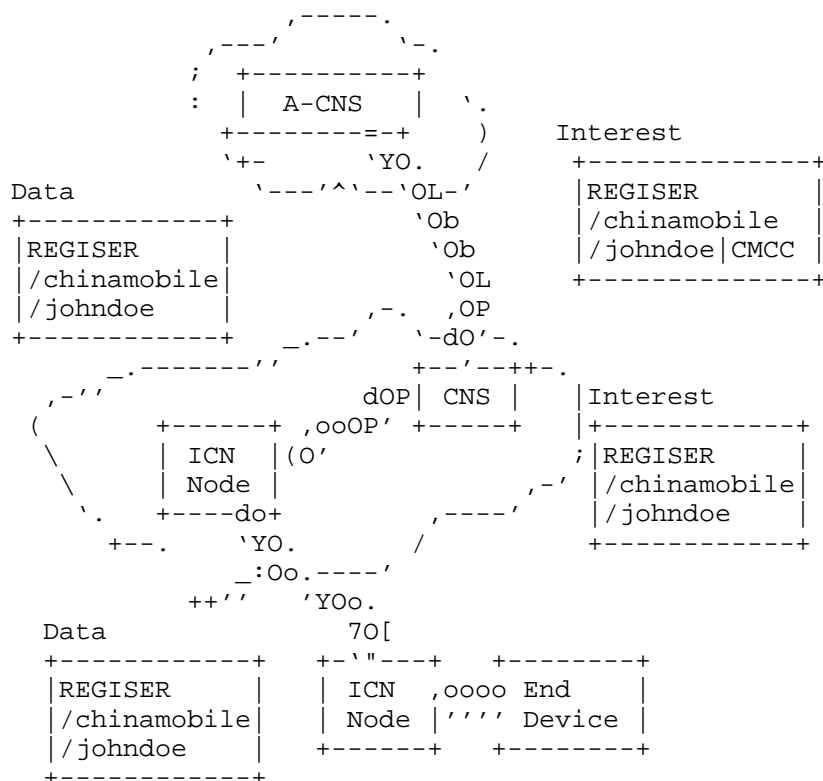


Figure 6 container registration/unregistration
 -Utilize Interest Packet to request name resolution to ICN.
 -Local CNS processes the Interest Packet first.

-If the local CNS is the aforementioned A-CNS of this container, then registration will be done. A Data packet with result will be sent back.

-If not, the Interest packet will be routed out to the next level CNS or A-CNS (for example, CMCC) of container A, per the preset rules,

until the registration/unregistration operation can be done on A-CNS. -The intermediaries CNS will forward the Data packet to the original requester once it receives the Data packet from next level CNS or A-CNS.

-Data packet of Register/Unregister does not allow to cache by intermediaries. This can be done by set "no-caching" in Data packet in ICN protocol.

Packet flow of a client/a node requesting a resolution of container is shown in Figure 7.

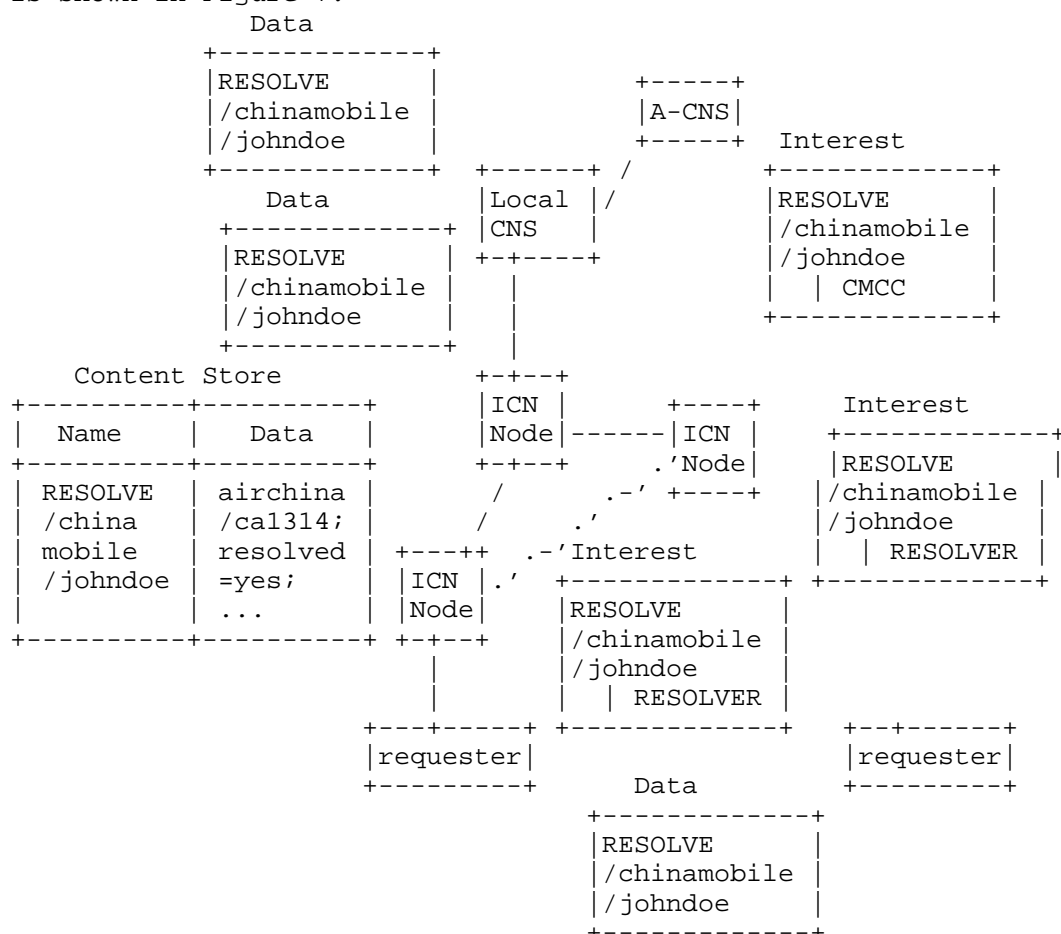


Figure 7 resolution of container

- Utilize Interest Packet to request name resolution to ICN.
- Local CNS processes the Interest Packet first.
- If local CNS can resolve the request, it returns a Data packet with resolution result.
- If it cannot be resolved, then appends the name of next level CNS or the name of A-CNS (for example, CMCC) of container A by following the preset rules. Route the Interest packet accordingly.
- After gets Data packet back from next level CNS or A-CNS, then it inserts the resolution result into its Database. It generates a new Data Packet.
- The intermediaries passed by Data packet will forward the Data packet as well as cache it in its CS table, according to the content forwarding rule.
- If the same request comes into this intermediary, then Data packet will be fetched from cache and return back to requester, according to the content forwarding rule.

3 Implementation of CNS

Aforementioned CNS is based on ICN node (support ICN protocol at lower level). It is responsible for name registration and resolution. It can be part of a resolution system together with other CNS'.

Any end device/network device/service can register its access container/container set in the CNS to enable other devices and services to inquiry them. Meanwhile, it has the capability to work seamlessly with other CNS.

As illustrated by Figure 8, John Doe applied a personal domain name "chinamobile/johndoe", China Mobile provides access service for him. Current access location is Shenzhen, Guangdong. Thus he registered the container as "cn/gd/sz"; the same user applied a third party (Hostsrv) to provide him with service. The name of the specific access container of the service can be obtained via resolution lookup (For example, Hostsrv provides 3 DCs for John Doe at "cn/gd", "cn/Beijing", and "us/ca"). Thus the container is registered as "hostsrv.com resolvable=yes". Once ICN receives Interest packet with name "chinamobile/johndoe", it sends resolution request to CNS. After two iterations of resolutions, NRS returns the resolution results.

	Interest
<pre> +-----+ Register /chinamobile /johndoe --> </pre>	<pre> +-----+ Content Name = {johndoe.com/blog /2012/June01 /main.html chinamobile/johndoe; </pre>

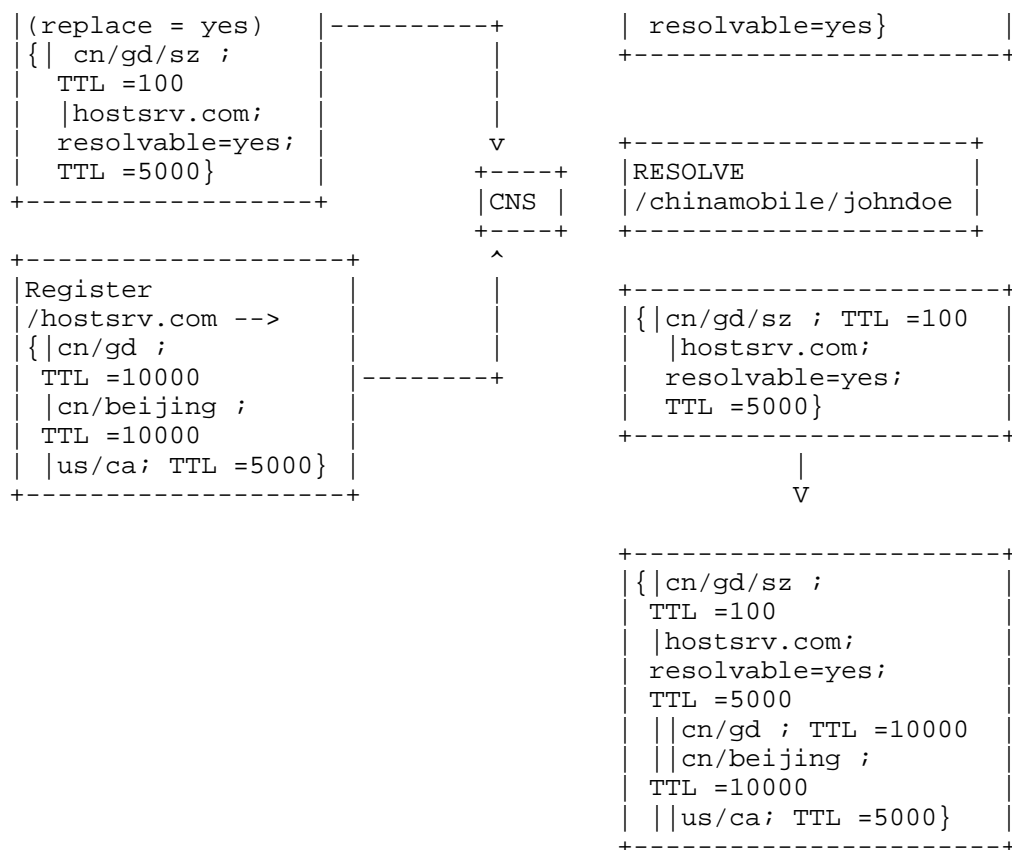
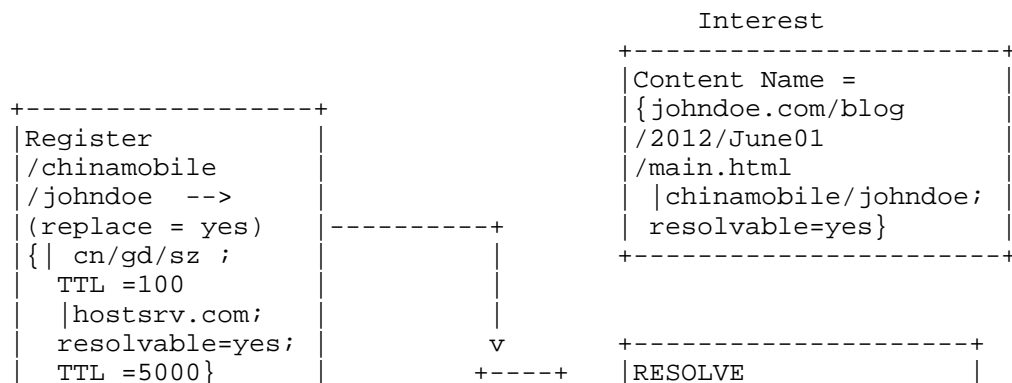


Figure 8

Another implementation method is to return the direct resolution result. If client wants to resolve the resolvable container, then it can send another resolution request. Figure 9 illustrates the procedure.



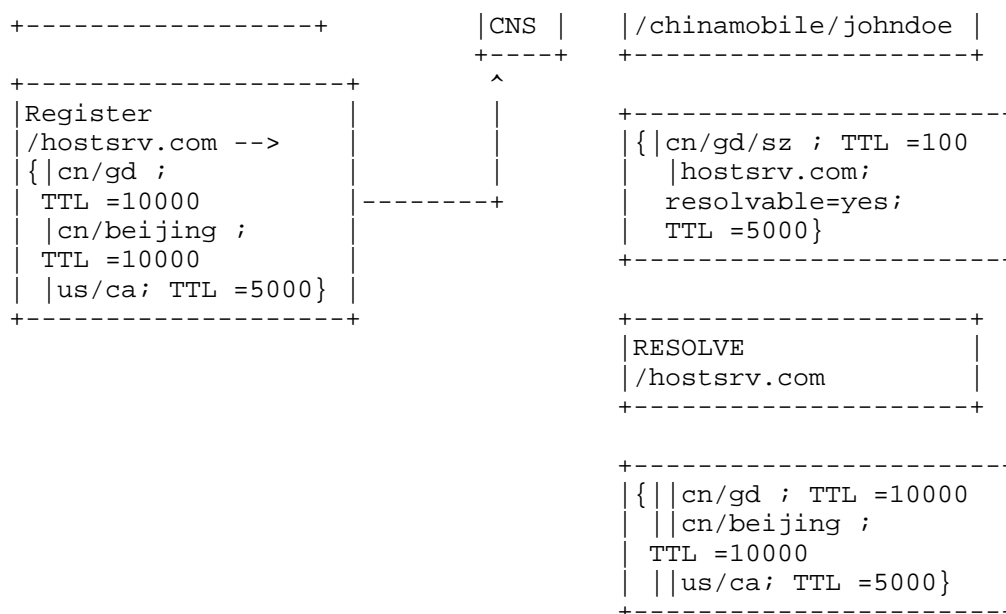


Figure 9

4 Execution Engine Process Logic

Detailed steps:

1. Is the receiving packet Interest packet or Data packet? If it is interest packet, go to step 2, otherwise, go to step 10.
2. Check operation type. If it is "register/unregister" type, then go to step 3. If it is "resolve" type, then go to step 8.
3. Use this node as A-CNS? Yes, go to step 4, otherwise go to step 9.
4. If it is "register" type, go to step 5, otherwise remove container names identified by Interest packet from the result list, go to step 12.
5. If container name about to register exists, go to step 6, otherwise, add new container name and associated resolution results to Table. Generate Data packet with successful status, go to step 7.
6.
 - i. If "replace=true", then replace the existing resolution results with the ones carried from interest packet, generate Data packet with successful status, go to step 7.
 - ii. If "replace=false" or no replace property, then insert resolution results from interest packet to the front of the container list. If container name in interest packet is the same as of old resolution results, remove original associated table entry. Generate Data packet with successful status, go to step 7.
7. (This optional step can be skipped and directly go to step 12). Traverse resolution results; index all resolvable containers. If a

resolvable container can't be resolved locally, then route Interest packet to the next level CNS or A-CNS of the container name. Wait for result carried back by Data packet. If the result is NULL, this container name cannot be resolved. Return failure status in Data packet, otherwise, create index to the next level resolution. Generate a complete Data packet with resolution results from all iterations, go to step 12.

8.

i. Does container name exist in the table? If not, then go to step

9. If yes, then traverse all resolution results, insert generated Data packet. Go to step 12 after all results have been checked.

ii. (Optional Step) If some of the results are resolvable ("resolvable=yes"), then resolve the container recursively by using this step. If the same result has been resolved (the same container name has been packaged into Data Packet), then ignore the result, and keep the traverse the next item (container name).

9. Append container name of next level CNS or A-CNS to the content ID in Interest Packet per preset policy. If there is a container name associated to the node, remove it. Generate a new Interest packet and route it out, and waiting for the returning Data packet.

10. If the reply Data packet is for register/unregister request, forward Data packet to the original requester, go to step 12. If the reply Data packet is for resolution request, go to 11.

11. Create table entry per the content ID (container name in this context) and resolution result within the Data packet, If the same container name exists, replace or remove it from the table per the preset rules. Generate Data packet based on local table results. If resolution result is NULL then no table entry is created. If this resolution is triggered by previous REGISTER resolution, go to step 7, otherwise, it is caused by RESOLVE resolution, go to step 6.

12. Call network interface module to forward Interest or Data packet.

5 IANA Considerations

No IANA consideration for this draft.

6 Conclusions

7 References

7.1 Normative References

7.2 Informative References

[Cisco-Name]

NDN and IP Routing, Can it Scale?

<http://trac.tools.ietf.org/group/irtf/trac/raw-attachment/wiki/icnrg/IRTF%20->

%20CCN%20And%20IP%20Routing%20-%202.pdf

[CCN] V. Jacobson et al. Networking named content. Proceedings of the 5th ACM International Conference on Emerging Networking Experiments and Technologies (CoNEXT 2009). NY: ACM; 2009; 1-12.

[ICN-name] Ali Ghodsi et al. Naming in Content-Oriented Architectures. In ACM ICN workshop, 2011.

[SCN] D. Smetters and V. Jacobson. Securing Network Content. Technical report, PARC, October 2009.

[Huawei-Name]
Draft-ietf-icnrg-icn-naming-and-routing-00.txt.

Authors' Addresses

Rong Wang
Huawei Technologies
Huawei Building,
BanTian, LongGang District
Shenzhen, Guangdong 518129
P.R.CHINA

EMail: WANG.RONG@huawei.com

Chunfeng Yao
Huawei Technologies
Huawei Building,
No.156, BeiQing Rd., Haidian District
Beijing, 100095
P.R. CHINA

EMail: chunfengyao@huawei.com

Zhefeng Yan
Huawei Technologies
Huawei Building,
BanTian, LongGang District
Shenzhen, Guangdong 518129
P.R.CHINA

EMail: yanzhefeng@huawei.com

INTERNET-DRAFT
Intended Status: Informational
Expires: August 21, 2013

Chunfeng Yao
Rong Wang
Yuanzhe Xuan
Zhefeng Yan
Huawei
February 17, 2013

Container Assisted Naming and Routing for ICN
draft-yao-icnrg-naming-routing-00

Abstract

In ICN (Information-centric network), everything is an identifiable object with a name, therefore the number of name prefixes is a few orders of magnitude higher than that in current BGP routing table. Towards scalable routing in ICN, we propose a name scheme, called container assisted naming. With this scheme, an object name consists of two components: a content name which uniquely specifies the object within certain scope, and one or more containers which define access relationships to the object. This document illustrates the concept of container and how it assists scalable routing in ICN.

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at
<http://www.ietf.org/lid-abstracts.html>

The list of Internet-Draft Shadow Directories can be accessed at
<http://www.ietf.org/shadow.html>

Copyright and License Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1	Design Principles	3
1.1	Naming in ICN	3
2	Container-assisted Naming	3
2.1	Container	3
2.2	Container Assisted Naming Formats	4
2.3	Resolving Container	5
3	Container Assisted Forwarding	6
3.1	Container Assisted Forwarding Procedure	6
3.1.1	Forwarding with full container resolution	6
3.1.2	Forwarding with iterative container resolution	6
3.2	Scalable Container Assisted Routing	7
3.2.1	Topology oriented containers	8
3.2.2	Non-topology oriented containers	8
4	Container Assisted Mobility	9
4.1	Container Assisted Terminal Mobility	9
4.2	Container assisted network mobility	10
5	IANA Considerations	11
6	Conclusions	11
7	References	11
7.1	Normative References	11
7.2	Informative References	11
	Authors' Addresses	13

1 Design Principles

1.1 Naming in ICN

Name plays a critical role in serving many fundamental functions in ICN: a unique name identifies a mutable or immutable content or information object; it is used to look up and access data in network cache; and it is used for routing and forwarding. Therefore, naming is the foundation for ICN architecture.

We follow several principles for defining a naming scheme in ICN:

-Unique: A name uniquely identifies an object or entity within some scope (e.g., within a domain or the entire Internet).

-Locatable: A name enables interested entities to locate the identified object in a network. For this purpose, the name is either routable to reach the object, or includes information to derive the routable location(s) of the object.

-Location-independent: identified object may be served by any node that has the object, which is independent of location or original source of the object.

-Scalable Routing: The number of potential prefixes in global content or information object namespaces is several orders of magnitude larger than that in IP address namespace [Cisco-Name]. Therefore, ICN name should support routing scalability.

2 Container-assisted Naming

2.1 Container

The container component in our naming scheme is appended to the original content name to assist routing. This naming scheme is not restricted to the hierarchical [NDN] or flat name [DONA].

A container is a space where content or information objects reside. A container in an ICN name can be one of the following:

- A content or object identifier prefix, e.g., Huawei.com/blog;
- An enterprise or organization name, e.g., Huawei.com, tsinghua.edu;
- A host or a device such as a mobile phone or a content storage device, e.g., chinamobile/johndoe/iphone;
- A mobile network, such as airplane, train, e.g., Airchina/cal314;
- A network domain, e.g., cn, cn/gd, cn/gd/sz.

Access Container: If container A is contained in container B, and

container B has the routing entry to container A, container B is defined as the access container for container A. One container can have multiple access containers, and it can provide access service to multiple other containers as while.

Base Container: The smallest container that can be used to assist routing for other container(s) is defined as the base container of the content. One content can have multiple base containers, such as multi-homing scenario.

2.2 Container Assisted Naming Formats

A container assisted name can be expressed in a Directed Acyclic Graph or a tree as Figure 1 shows.

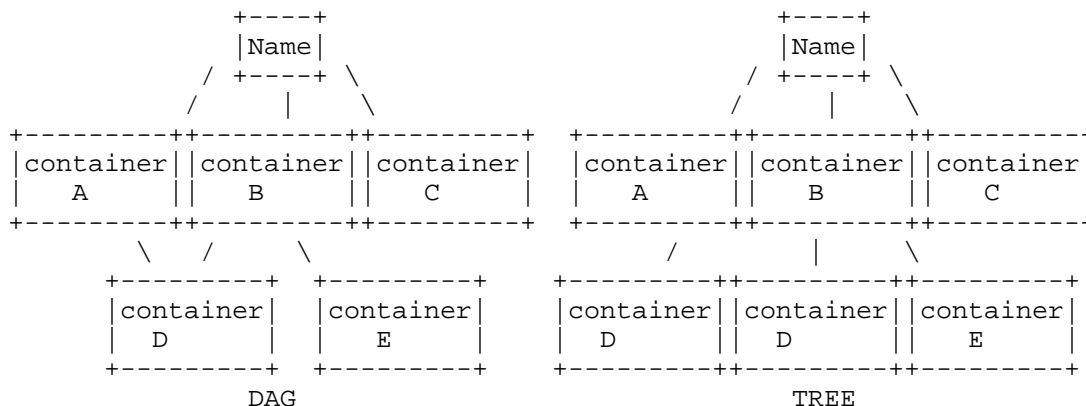


Figure 1.Container assisted naming

As illustrated in Figure 1, the content name can be hierarchical or flat, container A, container B, and container C are the base containers, container D is the access container for A, and container D and container E are the access containers for B.

As can be seen from figure 1, base containers A, B, C provide access to the content or information objects directly. These base containers may locate in or logically access to other containers, namely access containers. The base containers may be referred to as "first layer/depth containers" in the DAG or TREE, and their access containers may be referred to as "second layer/depth containers". Similarly, the access containers for the "second layer containers" may be called "third layer containers".

In an ICN request packet, we propose a new content name format by using depth-first traversal of the container access relationship tree (hereinafter referred to as CART). The name shown in Figure 1 can be represented as the following:

Name = {content name | container A || container D | container B || container D || container E | container C},

Where "|" is container separator. Each node of the tree is listed in the sequence of depth-first traversal, and separated by a separator "|". As the depth increases, the number of separators also increases. A container at higher layer provides access service for the ones at lower layer. The depth of a CART has no theoretical limitation.

2.3 Resolving Container

The location of a container can change from time to time, the same as its access containers. In order to avoid frequent changes in containers and guarantee the name persistency, we propose container resolution, which allows a container to dynamically register and query its access containers from a resolution system.

Container Resolution System is a distributed system composed of several container resolution servers deployed in the network to store the mapping relationship between a container and its access containers. The system supports dynamic register and query operations.

A container resolution request can be initiated by a content consumer or intermediate network node, when the request carries a resolvable container.

A container in a name or in a resolution result can carry an attribute "resolvable", if "resolvable = yes", this container can be further resolved in the resolution system to get its access container(s); if "resolvable = no", it is not resolvable; that is, it may be the deepest level container in the resolution tree. The default value is set as "resolvable = no".

A resolved container from resolution system can carry the following two attributes:

-- Cacheable: defines whether the resolution result can be cached in the network thus can be shared with other nodes or consumers. The default value is set as "cacheable = no".

-- Time To Live (TTL): defines the fresh time a resolved container result can live in the network. The default value is set as "TTL = 0", namely uncacheable.

If a resolved container from the resolution system is also resolvable, the container resolution process is iterated.

3 Container Assisted Forwarding

3.1 Container Assisted Forwarding Procedure

When a request is received, an ICN router first checks the content store for a matched content with the content name in the request packet. If a matched content is acquired, it is returned to the consumer or the preceding router.

If there is no matched content in content store, the router looks up its FIB for outbound faces. The forwarding decision is made based on content name first. If there is no match in the FIB, the decision is then made based on carried container(s) in the name. There can be two ways to assist forwarding with containers by the router.

3.1.1 Forwarding with full container resolution

The procedure of forwarding with full container resolution can be described as follows:

Firstly, the router checks whether there is a container carried in the request. If not, it forwards the request to the default routing entry or drops it.

When there is a container carried, the router checks whether the request includes resolvable container or not. If "yes", the router initiates a container resolution request to the resolution system. The result can be responded either by a server in the resolution system or an intermediary's cache where a previous resolution result resides. If the resolved container is also resolvable, the resolution process is iterated by the router with the result container(s).

When all resolvable containers are resolved, the complete set of container(s) including the carried ones in the name and the resolved ones from the resolution system are used to forward the request. The traversal order can be depth-first or can be breadth-first. In case one of these containers matches a FIB entry, the request is forwarded to the corresponding outbound interfaces. Otherwise, the request is forwarded to default outbound faces or dropped.

3.1.2 Forwarding with iterative container resolution

The procedure of forwarding with iterative container resolution can be described as follows:

Firstly, the router checks whether there is at least one container carried in the request. If there is not, the request is forwarded to default outbound interfaces or dropped.

For all carried containers, the router matches them with its FIB entries, if one container matches, the request is forwarded according to the matched outbound faces.

When the carried container(s) have no match in FIB, the router checks whether one of the carried containers is resolvable. If yes, it sends container resolution request to the resolution system to get the access container for the resolvable container(s).

After that, the router matches the resolved container(s) with FIB. When one container matches an FIB entry, it forwards accordingly. If there is no match, the container resolution can be iterated with other resolvable containers in the request.

When there is no more resolvable container in the request, the request can be forwarded to default outbound faces, or dropped.

3.2 Scalable Container Assisted Routing

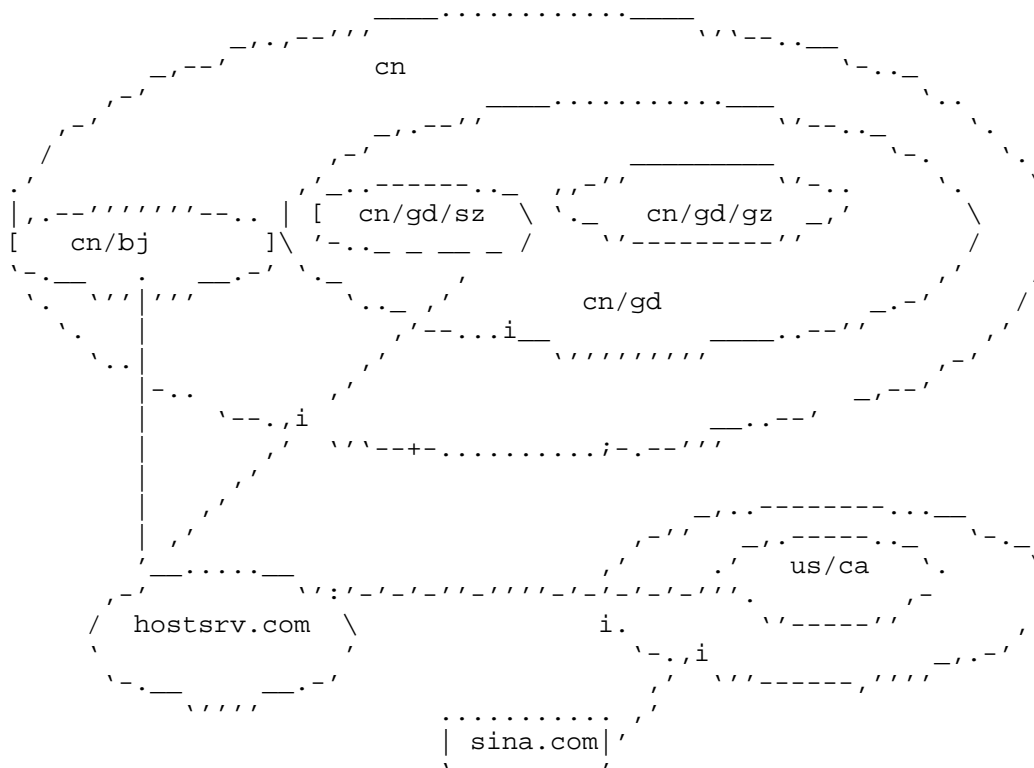


Figure 2. Solution to Routing Scalability

The scalability of name-based routing in ICN can be well addressed by

containers. We divide containers into two categories: topology oriented and non-topology oriented.

3.2.1 Topology oriented containers

Topology oriented containers aggregate naturally. For example, as illustrated in Figure 2, the whole network of China can be viewed as a national level container "cn", and province level containers such as "cn/gd" for Guangdong province and "cn/sd" for Shandong province are contained in "cn". Similarly, a city level container such as "cn/gd/sz" is contained in "cn/gd". Giant ISPs can also be treated as topology oriented containers. For example, China Telecom, as a top level container "ct", contains "ct/gd", which further contains "ct/gd/sz".

As LPM is used in FIB matching, a routing entry for a container may only propagate within the network domain of its access containers. For example, a routing entry for "cn/gd" does not need to propagate out of "cn". In this case, a core router in the network domain of "us" with only a routing entry "cn" can forward all the packets with "cn" as a container prefix. Obviously the size of FIB can be greatly reduced due to the recursive aggregation of topology oriented containers. Therefore, the routing entries created by topology oriented containers are the basis for FIB compression.

3.2.2 Non-topology oriented containers

According to the scale-free property of the Internet, we further divide non-topology oriented containers into popular ones and non-popular ones.

The majority of the non-topology oriented containers have non-popular content and low visiting volume, such as small companies and organizations, home networks, and personal digital devices. The large number of this type of containers is the major cause of the routing scalability problem. Since these containers do not have to propagate outside of their access containers, we can greatly reduce the size of FIBs in core routers with access containers. For example, the corporate network of a company "hostsrv.com" locates in three different places, "cn/gd", "cn/beijing", "us/ca", which can be seen as three topology oriented containers that provide access service to "hostsrv.com". The route to "hostsrv.com" exists only inside these three containers, and any outside router can use these three containers to assist packet routing in order to reach "hostsrv.com".

A small portion of non-topology oriented containers has several orders of magnitude higher visiting volume than others, such as large corporations (e.g., huawei.com) and large portal websites (e.g.,

sina.com). The small number of these frequently visited containers does not cause scalability problems for core routers.

To conclude, with the container assisted naming scheme and routing, the size of FIBs in core routers is determined by the number of "aggregated topology oriented containers" and "frequently visited non-topology oriented containers", which could be smaller than the size of routing table in current Internet's core routers.

4 Container Assisted Mobility

4.1 Container Assisted Terminal Mobility

By terminal we mean either consumer side terminal devices or data source side terminal devices or hosts. In the scenario where a data source terminal is static and a consumer terminal is moving, the consumer can resend requests to fetch the latest data packets after the movement.

In the scenario where the data source is moving, a carried container in data request packets can assist the forwarding during the data source movement.

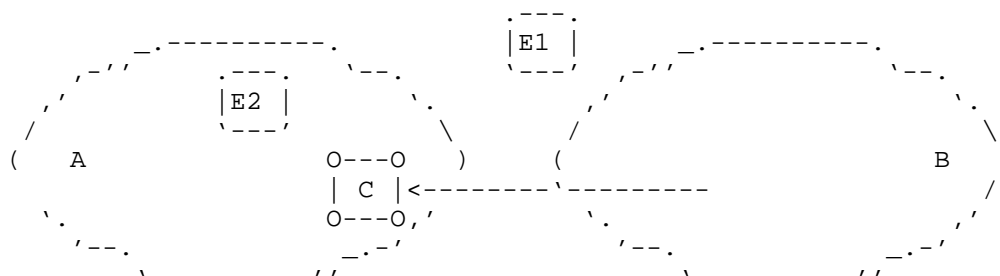


Figure 3. Data source terminal movement

As Figure 3 shows, if the data source in container C moves within its access container B, its routing update is contained within the access container, and there is no routing update out of container B.

If the data source moves out of its access container B to a new container A, it needs to register a new route in the new access container A, and updates the access container information in the resolution system. After this, its data consumers or on-path routers can acquire the latest base container (i.e., container A) with container resolution. The register process can refer to [Huawei-Resolution].

As shown in Figure 3, during the movement, if a data consumer E2,

which resides within the same access container (i.e., container A) as the data source in container C, sends a request to retrieve the data, the request packet can be forwarded straightly to the mobile source.

However, if a router E1 outside the access container A receives such request, it cannot find the route directly by the carried container in the name (i.e., container C). Therefore, the router sends resolution request to obtain the data source's access container. After gets the access container A, the router forwards the requests to A firstly, and then to destination C by content object name or carried base container(s).

4.2 Container assisted network mobility

A mobile container can provide access to a set of containers that moves together with it, for example, a train, airplane, or ship can provide access service to its passengers. This can be seen as network mobility.

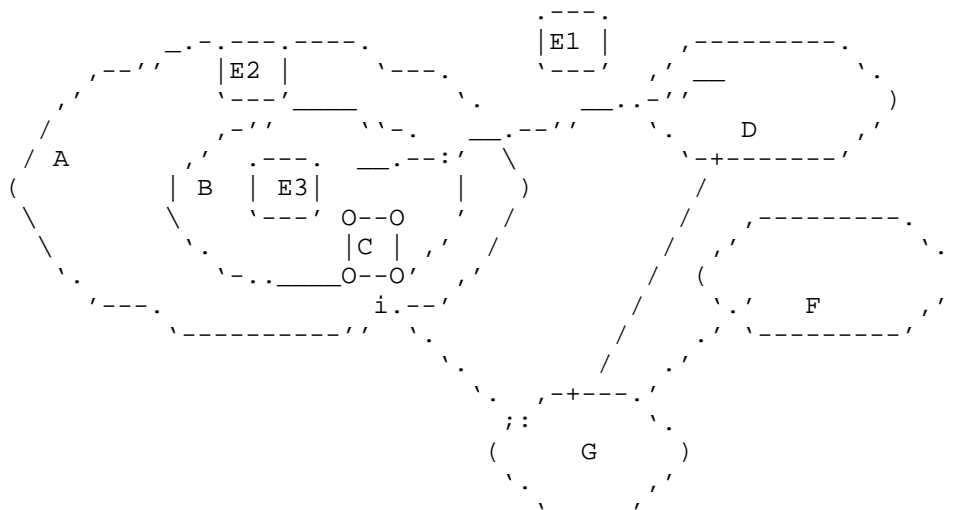


Figure 4. Network mobility

As Figure 4 shows, when container B moves, the processing is similar as the data source movement in previous case. If container B moves inside of its access container, i.e., container D, routing update only propagates within this container. If B moves out of its access container, it needs to register a new routing to its new access container, e.g., container A, and updates its new access container relationship in container resolution system. When container B moves across several containers, it only needs to update its access

containers (e.g., from D to A) in the resolution system, while all contained containers (e.g., container E3 and C) within itself do not need to do any update since their routing in the network (inside of container B) and their access container relationships do not changed. With this container assisted routing, the updating and querying frequency to the resolution system can be significantly reduced.

Consider the mobile network B as a high-speed train. Along with its inside containers (E3 and C), it moves from original access container D to a new access container A. During this movement, if a consumer within the same mobile network E3 requests for data in contained container C, the request is forwarded directly.

If a consumer in container E2, which is outside B, requests data in container C, the router in the access container A cannot forward the request directly by the content name. It then sends resolving request to the resolution system. With the resolved container B, the router can forward the request to the mobile network B, which in turn forwards to data source C with content name or carried container.

If a consumer in container E1, which is outside the access container of the mobile network B, requests data in contained container C, it needs two-level resolution: it obtains the mobile network B with the first level resolution, and the network's new access container A with the second. Routers can route the request with the second resolution result (i.e., container A), and then with the network container B, and finally to the destination container C with content object name in request or carried base container(s). Note that the number of resolution level is not restricted in our scheme, which is decided by access relationships among containers.

5 IANA Considerations

No IANA consideration for this draft.

6 Conclusions

7 References

7.1 Normative References

7.2 Informative References

[Cisco-Name]

NDN and IP Routing, Can it Scale?
<http://trac.tools.ietf.org/group/irtf/trac/raw-attachment/wiki/icnrg/IRTF%20->

%20CCN%20And%20IP%20Routing%20-%202.pdf

[CCN] V. Jacobson et al. Networking named content. Proceedings of the 5th ACM International Conference on Emerging Networking Experiments and Technologies (CoNEXT 2009). NY: ACM; 2009; 1-12.

[DONA] T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, and I. Stoica. A Data-Oriented (and Beyond) Network Architecture. In Proc. of SIGCOMM, 2007.

[Huawei-Resolution]
Draft-ietf-icnrg-icn-container-resolution-system-00.txt.

Authors' Addresses

Chunfeng Yao
Huawei Technologies
Huawei Building,
No.156, BeiQing Rd., Haidian District
Beijing, 100095
P.R. CHINA

EMail: chunfengyao@huawei.com

Rong Wang
Huawei Technologies
Huawei Building,
BanTian, LongGang District
Shenzhen, Guangdong 518129
P.R.CHINA

EMail: WANG.RONG@huawei.com

Yuanzhe Xuan
Huawei Technologies
Huawei Building,
BanTian, LongGang District
Shenzhen, Guangdong 518129
P.R.CHINA

EMail: xuanyuanzhe@huawei.com

Zhefeng Yan
Huawei Technologies
Huawei Building,
BanTian, LongGang District
Shenzhen, Guangdong 518129
P.R.CHINA

EMail: yanzhefeng@huawei.com

ICN Research Group
Internet-Draft
Intended status: Informational
Expires: February 21, 2014

X. Zhang
R. Ravindran
Huawei Technologies
H. Xie
Huawei & USTC
G. Wang
Huawei Technologies
August 20, 2013

PID: A Generic Naming Schema for Information-centric Network
draft-zhang-icnrg-pid-naming-scheme-03

Abstract

In Information-centric network (ICN), everything is an identifiable object or entity with a name, such as a named data chunk, a device, or a service end. Different from host-centric connectivity, ICN connects named entities using name-based routing and forwarding. At the same time, network entities, end devices, and applications have variant demands to verify the integrity and authenticity of these entities through names. This document proposes a generic naming schema called PID, which supports trust provenance, content lookup, routing, and inter-domain resolution for ICN. With PID schema, a name consists of three components: principal(s), identifier(s), and domain(s). In this draft, we only illustrate the principle and concept of PID and the functional role of each component, and leave encoding approaches as implementation options.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any 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 February 21, 2014.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Design Principles	3
1.1. Naming in ICN	3
1.2. Design Principles for Naming in ICN	4
2. PID Naming Schema	5
2.1. Naming Format	5
2.2. Routing Names	6
2.3. Cache Access	8
2.4. Dynamic Content Routing	9
2.5. Towards Generic Naming Schema	10
3. Security Considerations	11
4. IANA Considerations	11
5. Conclusions	11
6. Informative References	12
Authors' Addresses	12

1. Design Principles

1.1. Naming in ICN

In ICN design, a name has been required to serve for many purposes: ICN requires unique names to identify mutable or immutable content or information objects with which applications can send requests; in data caching, a name is used to look up and access a data chunk; in routing and forwarding, a name is used for reaching an information object; for security, a provenance between a name and its content required via cryptographic credentials. We summarize the following roles that a name may be desired from different perspectives in ICN:

- o R1 (unique): A name identifies a content object or network entity with uniqueness in some scope (e.g., within a domain or Internet).
- o R2 (locatable): A name enables interested entities to locate its identified content object in a network. For this purpose, the name is either routable to reach the location of the object, or includes information to derive the routable information of the object.
- o R3 (readable): A name enables a user or application to easily indicate the content of an object, even without knowing the content itself beforehand or before the content is generated. For this, the name may be required to be human-readable.
- o R4 (authenticable): A name has strong binding with the content object (either its publisher or owner, or the content itself), in order to provide content access authentication, to enable a receiver to verify its provenance, and to prevent denial-of-service attacks in an ICN [ICN-name].
- o R5 (trustable): A name includes information on how to derive the trust of a content object, e.g., by an end user who retrieves the content from ICN, which may or may not be from its owner or publisher. The trust can be built on mechanisms different trust management infrastructures.

There are many different naming schemes towards all or subset of the above roles. For example, flat names are used in DONA [1] and NetInf [2] for global uniqueness and authentication, but do not provide readability, routing, and trust-deriving information. PSIRP and PURSUIT [3] separate the namespaces for rendezvous and forwarding identifiers of a name, and both namespaces are flat. Standard ways to name objects with hash functions have been proposed in [4], where a named identifier (ni) schema is used to uniquely specify objects. This name schema focuses on the uniqueness and strong binding of

content objects and names, but not for routing. Hierarchical flat name is proposed in [5], where nested flat names are used for routing purposes. Hierarchical human-readable names are proposed in CCN and NDN [6], but they do not provide authentication and trust-deriving information. A generalized form of name is proposed in [7] to bind authentication with content names via digital signatures.

1.2. Design Principles for Naming in ICN

We follow several principles for defining a general naming schema in ICN.

- o A naming schema satisfies necessary but not more than necessary aforementioned roles: in our view, a single-component name cannot satisfy all roles at the same time.
- o A content name identifies a content object in persistent way, such that this name does not change with the mobility and multi-home of corresponding content object, device, or host. A client can always use this name to retrieve the content from network and verify the binding of the content object and the name.
- o A naming schema should give certain level of flexibility to support different networks, considering variant network architectures have been proposed, and in the future multiple architectures (or features of these architectures) and current Internet may co-exist. Ideally, a name can include any form of identifier, including flat, hierarchical, and human readable or non-readable. The identifier can be chosen by content owner or publisher with the uniqueness within certain domain or within an application-specific scope.
- o A network does not use persistent content name for routing directly; instead, a "routing name" (or routable address/location/label/tag) is network architecture dependent, which is usually routable within the network, such that a network node or client can reach the content within it. Usually, a routing name is the real location (or locator) of the content in the network.
- o Per-domain-based (globally or locally) naming resolution services (NRS) should be available, to map a persistent content name to routing names or locations within a domain. While per-domain NRS updates the routing labels for a content name, it creates a late-binding routing behavior. We note that a single content name can be mapped to multiple routing names. How to implement name resolution service is not included in this draft, e.g., [8] provides details of one implementation with content container.

2. PID Naming Schema

2.1. Naming Format

Based on these principles, we propose a P:I:D (or simply PID) naming schema for ICN. Each name is specified by three components of PID, where:

- o P is the principal to bind the object with the complete name for security purpose, towards different relationships, e.g., ownership, administration, or social relations. P is usually constructed by hashing the public key of the principal, or hashing the content object itself if it is static. We call the relationship between P and the named content object as "security binding".
- o I is an identifier of the object in variant forms and is referred by end user, applications, or other entities. It can be something chosen by the publisher or a network service, or administrative authorities. It can be hierarchical or flat, user-readable or non-readable, and usually location-independent. We call the relationship between I and the named content object as "application binding".
- o D is the domain that provides resolution from the identifier (I) to the real location(s) of the object by routers. For persistence purpose, D can be in any of the following forms:
 - * The locator of the target object if the locator is persistent;
 - * A resolution service name or location which maps the content identifier (I) to its real location, if the resolution service name is persistent;
 - * A resolution service name that maps the content identifier (I) to another resolution service name or location; that is, A is a meta-domain;
 - * Any combinations of above.

We call the relationship between D and the named content object as "network binding".

For example, D can be the domain name of the publisher's domain gateway, service, host that can resolve P:I, or a redirection gateway, service or host to preserve name persistence to deal with mobility or hosted services. D is the "fall back" used for name-resolution if P:I is not resolvable in the local cache of the

requesting domain. D is usually routable (either globally or locally), such that, when an application or network node first receives an interest with the content name, it can query a resolution service by routing with D and obtain the real location or locator of the named object. In case the resolution service is not static, a recursive name resolution may be performed, i.e., D points to a static resolution service, which in turn points to a dynamic resolution service, which points to the location of the object. If there is no D in a name, then a network node uses I to route to the location of the object if I is routable.

D can be in the same namespace of I, but in general it can be different. For example, in one case, D is the container of a set of objects which can locate and resolve objects [9].

For a published name that is in PID schema, a change of any field in P or I or D re-names the object, e.g., the object is re-signed by another entity, or its resolution service is changed, e.g., the publisher changes the host service of a web page.

We note that the domain concept in PID schema is more general than the administration domain in current Internet architecture. In PID, the relationship between a named object and its domain D is for location resolution and routing purpose. It can be the same as the administration domain of the content object, or a 3rd party resolution service provider, where the designated domain provides resolution service. In more general way, the domain of a name can have social-, admin-, owner-, host- relationships with the named object, which implies that the domain provides resolution service to locate a content object with its name. A domain can provide a DNS-like service that maps a content identifier to the location of the object or the resolution service. Different from current Internet's centralized DNS, a domain-based resolution can be more general with a distributed implementation. Furthermore, the meta-domain of a content object can be personal profile, e.g., as in social network service, an enterprise directory service, a cloud service provider, or a web hosting service. For example, to support the Example 2 of [10], the domain part of the content name can be simply the service name or location of the lookup database, which is more persistent than the mapping of a content identifier to location. Note that in [10], the lookup database is assumed to be static and already known by the network, which we believe is not scalable and flexible enough.

2.2. Routing Names

As aforementioned, PID schema differentiates content names and routing names, where the former is persistent to specify a content object, while the later is location-based for routing purpose.

Instead of a very specific format of routing names, PID supports variant forms of routable names (or routing labels), e.g., a network address or a locator. For a content name P:I:D, the D resolves P:I to one or more routing labels, and an application or network router can choose one to reach the content object or more for multicast. A routing label for a content object can be dynamic, and can be changed from domain to domain. For example, a single domain may by default set a gateway routing label to all the clients it is serving. The gateway then replaces it with some other labels. Through this way, the routing label can allow policy-based intra/inter-domain routing, late binding for mobility, and delay-tolerant content routing.

With a content name provided by a content requester, the network first returns the real location of the named object via resolution services specified by the domain information (D) in the name. This location information is then augmented in the head field of a PDU (e.g., an interest in CCN). The network then uses this location information to reach the object, retrieve the named content, and forward back to the requester. Resolving the location from name and augmenting the PDU can be transparent to applications.

In general, the resolution process works as follows. With a content name P:I:D, a client forwards a request to a network node (e.g., an access router). If not resolvable in the local cache, the router first routes to a naming resolution service (NRS) with D. With the input of P:I, the NRS returns the routing name (or routing label) of the content object, e.g., a location or a locator. We note that the format and semantics of a routing name can be domain specific, and may be only routable in one domain, e.g., it can be a flat location in DHT or a hierarchical node name in a network operator. Upon receiving this, the network node inserts this label in the head of the interest packet. The network then uses this routing label to reach the next hop, to retrieve the named content by using P:I at each hop, and to forward data back to the requester, e.g., following the PITs in CCN [6]. In case the routing name resolved from the NRS is another name resolution service named with D', the network node sends the request to this resolved NRS with D' in interest head, obtains the location of the target object, and then inserts the location into interest head to obtain the content object. This process happens recursively until the location of the named object can be reached. With a single name, an NRS may return multiple entries of the locations of object. A network node can use one or multiple of them to retrieve the object, according to its local policy or configuration.

In another case, where a separate locator address space is not managed, a per-hop forwarding can be adopted, where a content router tries to resolve the content name identifier (I or P:I) locally in

its cache. If it is un-resolvable, the router uses I:D or just D to route to domain D. In the latter case once the interest reaches D, the request I:D can be used to route to location(s) of the content object.

Therefore, logically, a data PDU could be of form <P:I:D, <Routing Label>, C, Sign_P(I:D,C), Metadata >, where C is the content payload, Sign_P is a signature generated from the private key corresponding to P on C and persistent content name, and the metadata includes other meta attribute information. With this hybrid naming approach, PID schema achieves the benefits of both pure self-certified names and hierarchical names. Specifically, similar to hierarchical human-readable name, the P:I part of the schema can achieve global uniqueness and readability (if needed). With D, the schema achieves location persistence without including the real location of the content object in its name. With the P part, the schema can achieve strong binding between content object and its name for security and data integrity. Note that trust management is usually built on some external mechanism out of the naming schema.

In a special case, the D of a content name P:I:D could also serve as a routing label; That is, D can serve dual purposes: a resolution/redirection point, and a routing label as well. For example, D can directly resolve to a container (server). This avoids one RTT to obtain the routing labels of the content name.

While D can serve the same purpose of routing label that is proposed in [10], PID schema has two improvements:

- o PID has better persistence property since it separates routing labels from content names, while in [10], a content ID includes both routing labels and identifier. When the routing label of a content object is changed, e.g., the host service is changed, or a new host service is added, the content ID has to be changed, which breaks the name persistency.
- o PID has stronger security binding of names and content objects via principal field.

Note: We focus on the logical semantics of fields in a naming in this document. In implementation, variant formats of PID can be options. For example, I:D can be in a single component, which acts as a resolvable identifier.

2.3. Cache Access

With a content name of P:I:D, a router can use the full name to index and look up cached content chunks and pending interests, e.g., in

content store (CS) and pending interest table (PIT) in [6]. Optionally, a router can only use P and I for the same purposes. This achieves location independence in data storage and forwarding, e.g., when a content chunk with P and I can satisfy any request of P:I:D with all possible Ds. That is, two content objects with same P and I are considered as the same and thus only one is cached at anytime, even though they may have different Ds.

2.4. Dynamic Content Routing

The PID schema lends itself to allow consuming and producing applications to choose naming semantic that meets requirements in terms of reliability, security, or performance metrics. The naming format follows a P:I:D format, where I identifies the named entity with a local or global scope, and D is the authority which could resolve the entity's location(s), and P securely binds the content object to I. For content routing I:D is the relevant portion. As I could be a hierarchical or flat name, several options for content routing are possible. In one case separate ICN domains can be built that are optimized to deal with either flat or hierarchical names, where name-resolution service allows the request to be directed to the appropriate domain criterion determined by the publisher, consumer or based on certain routing policies. In another case, a content routing domain can be built where the name-resolution infrastructure is enabled to deal with both flat and hierarchical names, where irrespective of the type of naming, a separate locator space exists to resolve the content name to its location(s).

If the combination of I:D is hierarchical, the content routing can follow the resolution mechanism similar to CCN. To resolve an interest, either I itself could be routable if it is globally unique, or the combination of I:D should be routable, which shall be interpretable by the name resolution service handling hierarchical names. Such ICN domains can leverage longest prefix match to take advantage of name-prefix aggregation mitigating routing scalability issue.

If I is flat, then the resolution through D should return a routing label(s), which can be appended to the interest packet for intra- and inter-domain name based routing on a fast path, or the name resolution can be handled by the global name resolution infrastructure through inter-domain cooperation on a slow path.

There are several considerations for dynamic name based routing. Based on the particular naming constructions, e.g., hierarchical, flat, or hybrid, each of which achieves the same objectives respectively with different mechanisms.

2.5. Towards Generic Naming Schema

In a general case, one object may have several names. Different names are assigned by different domains and served for different purposes. Logically, for a single object (e.g., a content object, a device, an application, or a service), it can have multiple identifiers. For example, a mobile device may have identifiers of an IMEI number, a phone number, an IP address, a human readable name (e.g., Alice's iPhone), and an organizational device id (e.g., if the device belongs to a company). A user generated content can have a user chosen ID, a URL, and a tinyURL. All these identifiers can have a single principal. Therefore the name of the object can be `P:(I1:...:In):D`, where `Ix` is an identifier, `D` is a domain that provides name resolution service, and `P` is the principal.

In a very general case, each identifier can be associated with different principals, and multiple locators can be used for a single content object, e.g., for load balance and duplication purposes. For example, Abel's iPhone has different public keys for different names it may use for different network services, one for Abel's personal use, and another from his company. Therefore, the relationship between the object, the identifiers, and the principals is a multi-element set.

As one object may have many persistent domains (e.g., a content is stored at different host services or CDNs), and one object may also have many IDs, in this generic schema, both domain and identifier may be a multi-element set, and content routers and consumers can select variant elements for content routing and forwarding (based on locally defined policy).

Note that there can be mapping relationships between multiple names of a single object. For example, an object may have a hierarchical identifier within its local domain owned by an enterprise, but has a flat identifier (hash of its content) with a DHT service. There can be a mapping service to link these two names towards the same object.

In general, mapping function among different names of a single object can be used to build flexible relationships between names, such as:

- o An identifier can be derived from another identifier, which forms nested or tunneled names.
- o A principal can be signed by another principal, to build trust between different principals, such as for ownership, administration, and social relationships.

- o A domain name can point to another domain name for the same object.

The PID schema can support these levels of flexibility. However, we consider these are extensions of core naming schema.

3. Security Considerations

As traditional, the integrity of a content object is maintained by a signature included in each data chunk. If the principal (P) of the object name is the public key or hash of the public key of its publisher, this key can be used to verify the integrity and authenticity of the object. When P is the hash of the content itself, the signature itself is built with P. Therefore, PID provides a strong binding between a name and its content object.

When P is (the hash of) a public key, it can be the trust derivation information of the object, e.g., an end user can use it to decide whether to trust the content object or not, based on a trust management infrastructure such as PKI or name-based trust [11]. However, PID schema is independent from any trust management infrastructure. The trust of a content object is derived from the trust of the principal. Either a network node or an end user can verify the trust of a content object. The trust management infrastructure is out of the scope of PID schema.

Similar to [6], the public key of a principal can be regular ICN data, also with the name format of P:I:D. For the name of a certified public key, its I can be some domain- or realm-based name, D can be the name (if static) of the certificate directory service of a CA, or a domain that resolves the location of a public key certificate, and the P is the hash the CA's public key.

4. IANA Considerations

This document makes no specific request of IANA.

5. Conclusions

In this draft, we propose PID, a naming schema for ICN. With this schema, an object name includes a principal P, an identifier I, and a domain D. The principal P acts for security binding, e.g., to verify if the object is bounded with its name, and to derive the trust of the object with possible trust management mechanisms. The identifier I identifies the object within certain scope, and can be used for

application binding such as caching access. The D refers to a name resolution service that can resolve the real time location of the object, directly or recursively. While this draft lays out the basic design principles and workflows of PID, we leave its encoding and implementation options to other documentations, such as [9].

6. Informative References

- [1] I. Koponen et al., "A Data-Oriented (and Beyond) Network Architecture.", Proc. of ACM SIGCOMM 2007.
- [2] C. Dannewitz et al., "Secure Naming for a Network of Information.", IEEE INFOCOM Computer Communications Workshops 2010.
- [3] PURSUIT, "<http://www.fp7-pursuit.eu>".
- [4] S. Farrell et al., "Naming Things with Hashes.", <http://datatracker.ietf.org/doc/draft-farrell-decade-ni/> 2012.
- [5] A. Ghodsi et al., "Naming in Content-Oriented Architectures.", Proc. of ACM ICN Workshop 2011.
- [6] V. Jacobson et al., "Networking named content.", Proc. of ACM CoNEXT 2009.
- [7] D. Smetters and V. Jacobson, "Securing Network Content.", PARC Technical Report 2009.
- [8] R. Wang et al., "Container Resolution System in ICN.", <http://datatracker.ietf.org/doc/draft-wang-icnrg-container-resolution-system/> 2013.
- [9] C. Yao et al., "Container Assisted Naming and Routing for ICN.", <http://www.ietf.org/internet-drafts/draft-yao-icnrg-naming-routing-00.txt>. 2013.
- [10] A. Narayanan and D. Oran, "NDN and IP Routing, Can it Scale?", <http://trac.tools.ietf.org/group/irtf/trac/raw-attachment/wiki/icnrg/IRTF%20-%20CCN%20And%20IP%20Routing%20-%202011.pdf> 2011.
- [11] X. Zhang et al., "Towards name-based trust and security for content-centric network.", Proc. of IEEE ICNP 2011.

Authors' Addresses

Xinwen Zhang
Huawei Technologies
2330 Central Expressway
Santa Clara, CA 95050
USA

Phone:
Email: xinwenzhang@gmail.com

Ravi Ravindran
Huawei Technologies
2330 Central Expressway
Santa Clara, CA 95050
USA

Phone:
Email: ravi.ravindran@huawei.com

Haiyong Xie
Huawei & USTC
2330 Central Expressway
Santa Clara, CA 95050
USA

Phone:
Email: haiyong.xie@huawei.com

Guoqiang Wang
Huawei Technologies
2330 Central Expressway
Santa Clara, CA 95050
USA

Phone:
Email: gq.wang@huawei.com

