March 16, 2014
PARC

CCNx 1.0 Naming: Transforming Network Addresses to Application Value

Marc Mosko[1]*, Ignacio Solis[1], Priya Mahadevan[1], Ersin Uzun[1]

Abstract
CCN advances networking to fit the modern information centric world we live in. Networking is no longer machine-to-machine communications to share resources; it is a complex mesh of content and interconnections beyond servers and hosts. A CCN advantage is that it unifies content addressing from the network layer up through applications. New CCN protocols, from forwarding algorithms to flow control to content manifests to application data access, unify to a common framework. Like Metcalfe's Law, CCN will bring dramatic gains in information value by encompassing a much large share of the protocol space. This document describes the challenges and potential of CCN to shatter the current protocol boundaries through a common information naming framework from the network to applications.

Keywords
Content Centric Networks

[1] *Palo Alto Research Center*
***Corresponding author**: marc.mosko@parc.com

Contents

Introduction 1
1 Names 1
1.1 Publishers and Routing 2
1.2 Name Resolution Service 2
2 CCN Enabled Services 3
2.1 Notional File System 3
2.2 Notional Database 3
2.3 Notional Document Management System 4
3 Security 4
4 Conclusion 5

Introduction

CCN allows the network to operate on content using names much closer to how applications handle information. A CCN name includes a part for network forwarding, a part for application-specific naming, and parts for various named-based protocols involved with the transport of the data.

Bob Metcalfe's Law states that "The value of a network is proportional to the square of the connected users of the network." We posit a corollary that is "the value of information in a network is proportional to the square of the protocols that have access to it." CCN is the unifying framework that allows protocols at all levels of the stack to address and operate on information.

This paper describes CCN names and how we use them today at the OSI Layer 3 (network) and OSI Layer 4 (Transport) levels. We also describe how systems of tomorrow will use CCN names at higher layers, the Session and Application, to realize the dramatic improvement in information value from the CCN framework, for example from integrating the information namespace with the security and trust namespace, so all three become interwoven rather than today's jumble of content, IP addresses, DNS names, and trust anchors.

1. Names

A CCN name is an absolute URI without an authority. Each URI path segment has a label and a value. The label identifies the purpose of the name component, such as a general name component used for routing, or a specialized component used to sequence numbers, timestamps, or content chunk numbers. There are also application-specific labels.

For example, in Fig. 1, the prefix `/parc` corresponds to a globally-routable prefix. This prefix, on a local network or enterprise network, could be of any scope allowed by the network. This prefix, on a globally routable packet, would need to be from an assigned

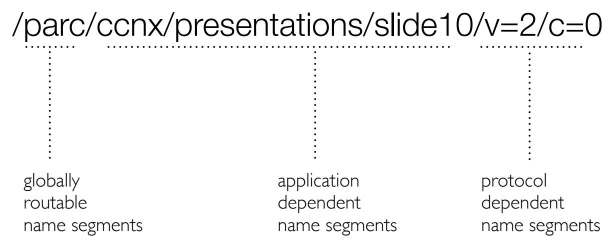
/parc/ccnx/presentations/slide10/v=2/c=0
globally routable name segments
application dependent name segments
protocol dependent name segments

Figure 1. Example CCN Name

pool and advertised by a global routing protocol. In this example, there is only one globally routable name segment, but there could be more. The next part of the name, `/ccnx/presentations/slide10`, is an application-supplied name. It identifies the content, similar to how an application would identify a file in a file system. The final part, `/v=2/c=0` are name-based protocol suffixes. There is a versioning protocol that applies the `v=2` version number to `slide10`, and there is a chunking protocol that divides the content into storage chunks and appends the `c=0` suffix.

As seen from the previous example, the hierarchical URI name format allows for name encapsulation, similar to how today's protocol headers operate on packet data. After the globally routable prefix, the next name segments narrow down the content to a specific slide, then name-based protocols narrow down the content to a specific version, then to a specific chunk. Name path segments encapsulate prior path segments, refining the name.

1.1 Publishers and Routing

A Content Publisher advertises its authoritative namespace via routing [1]. The routing protocol applies policies and security mechanisms to ensure that a publisher only advertises authorized namespaces. Service providers may impose additional limits such as sections of the network from which a publisher may advertise its service.

A home user, for example, is likely to have a peering relationship with their ISP. They might have a personal namespace, or more likely a namespace subordinate to their ISP. The personal namespace routes traffic to the home user, such that the home user could receive Interest messages.

A business is likely to have a well-known namespace, similar to how it has a domain name today. It may very well have two or more well-known names, where one name points towards "cloud" content hosted off-site and the other name points to local names such that users at the business site can receive Interest messages. The local name may be tied to the business' ISP.

A large business or network service provider would have well-known names that are advertised throughout the network. These may function like Anycast services.

[1] There are some experimental, small-network applications that use foraging and and epidemic information flow models rather than authoritative routing. These experimental networks are outside the scope of this document.

1.2 Name Resolution Service

From a network operation perspective, CCN does not require any Name Resolution Service (NRS). CCN can move Interests and Content Objects through the network with only the names and a routing protocol, just as IP can move packets without DNS. In CCN, routing does not require a NRS, forwarding does not require a NRS, low-level signature authentication does not require a NRS.

CCN does not enforce any naming policies such as number of name segments allowed or the length of each name segment. The specification of a network forwarder does not depend on the names conforming to specific rules. A practical Internet-scale routing protocol, however, may impose specific name policies, such as a maximum length for routable name path segments, value restrictions, or terms of use.

CCN offers one built-in method of indirection. There is a specific type of CCN object called a Link, which translates one name to another name. CCN links are signed Content Objects, so there is a transfer of trust explicit in a link chain. Links are commonly used to translate from a friendly name to a specific instance name. For example, the name `/parc/index.html` could link to a versioned and chunked content object named `/parc.com/index.html/v20140303/c0`.

An important aspect of CCN names is that they identify collections. For example, the name `/parc/ccnx/presentations` names a collection of presentations about CCNx published by PARC. As names encapsulate other names, the collections become more refined. Therefore, it is not necessary to lookup a network name for each individual piece of content. Google, for example, indexes about 10^{12} URIs. CCN does not need to store 10^{12} names in the forwarding table. We may need a number more like the number of DNS domains, on the order of 10^8 to 10^9. Beyond the routable name components, an application controls its own naming, so it does not need to consult an external name resolution service.

2. CCN Enabled Services

Network services in CCN will use CCN naming for information and services. The CCN names, as noted above, combine routing information, application-specific information, and name-based protocol information. Some services may have well-known names. For example, an App Store application for cooking recipes could have a name prefix `/topchef/app` embedded in the application. As a user queries the application, the application can generate well-formed names via its own independent calculations and does not need to consult an external service to form the names. Those names are usable from the application all the way down to the network forwarder.

Specific services and applications will have their own policies around how they form the application-dependent parts of the CCN name. For example, one application may name things like

(1) `/ccnpix/bylocation/california/sanfrancisco/bydate/2014/03/02/index`.

The `/ccnpix` service determined that it will organize data based on location and date. Other applications could organize their application-dependent part of the name based on how they choose to interpret them.

A search engine would have its own name format for user queries. It would resolve user queries to other names. For example, if a user searches for "San Francisco Trolly", the search engine might return results with `ccnpix` names, like the one above.

Identity management is an important element in any distributed system. Systems like LDAP, Radius, and single-sign-on systems, such as Passport or MDSSO, might still be used in a CCN environment. CCN identifies users, such as in an access control system, by public keys. These public keys may evolve over time and can be versioned. They can be associated with a friendly CCN name either by being published under a CCN from which software constructs friendly name, or through a trust management infrastructure that allows users to associate names with keys, as in a Simple Distributed Security Infrastructure (SDSI) [**?**]. For example, a system might publish its root key as `/ccnxpix/key`, which all applications can generate the name, and that Content Object could link to the specific key name, such as `/ccnpix/keys/0x4888338992/v3/c0`.

In the remainder of this section, we describe several notional CCN services. For each service, we describe how such a service could form and leverage CCN names.

2.1 Notional File System

We could imagine a CCN-based filesystem, for example CCNfs [2], made up of distributed file system servers. Such a file system would have an idea of the file system name space and enforce Access Control Lists (ACLs). CCN, of itself, does not solve the hard problems of distributed systems. It does make some of those problems more tractable by removing the barriers between network naming and application naming.

CCNfs, for example, could use an external name resolution service to identify the names of the quorum servers in the distributed file system, and then use a direct mapping from an individual client's name to each quorum server's name. Or, a system could use preconfigured name derived from a client's configured identity, such as being in the `parc.com` domain. In the first case, a client might request `/parc/ccnfs/servers`, which lists the specific quorum server names. This object could be returned dynamically or be published content in a repository. In the second case, a client could just begin issuing requests for `/parc/ccnfs/quorum1/...` and so forth, then let the dynamic routing protocol steer the request to the nearest server that advertises the `quorum1` service.

An application like CCNfs could use common naming conventions, such that keys and ACLs exist in well-known names in the file system hierarchy. An external name resolution service is not required to identify those individual items – an application (or rather the CCN stack) applies well-known rules to derive the names. For example, the current CCN code stack uses a form of Group Based Access Control where special security Content Objects exist at different levels of the naming tree [**?**]. These content objects allow nodes with the proper group keys to publish content objects that other group members will see as belonging to the group, and thus properly authorized.

2.2 Notional Database

PARC has experimented with CCN-based databases. In these systems, CCN names correspond to database rows and columns. We have also looked some at key/value nosql databases built on top of CCN. Each of these cases requires specifying a client API, and a set of network protocols built on top of the CCN core protocols and names. The use of CCN names gives visibility through the stack,

[2]For example, the Coda file system (`http://www.coda.cs.cmu.edu/ljpaper/lj.html`) could be implemented over CCN instead of IP

and implies there are no longer arbitrary disjunctions between naming the data (e.g. a select statement) and the routing of that statement to specific end hosts.

Such systems do not require a name resolution service, though such a service may be beneficial depending on the application deployment. A Key/Value nosql database, for example, could be based on service providers advertising a well-known name. Clients use the well-known name. In other deployments, a service provider may desire a layer of indirection, so the well-known name of the key/value database service directs clients to specific instances of the service using a name resolution service.

2.3 Notional Document Management System

An Enterprise document management system organizes an Enterprise's information corpus in to a system where users can find data, form associations, and manage information archival. These systems map very well to a CCN framework. The document management system does not need to track each individual CCN object name. It only needs to track the CCN name prefix of each document, not every individual CCN name of each version of each chunk of each document. Existing CCN name-based protocols derive those names from the routable prefix and application-dependent name.

A client – upon saving content to a file system or database – would update the document management system with the name prefix of the new document. The client only needs to make one update to the database, even if the document spanned over many individual CCN content objects. This is because CCN names identify collections of content objects, and the system can automatically generate those individual names using well-known CCN name protocols and a given name prefix.

In some systems, the document metadata is also stored as CCN content objects in the same information namespace as the documents themselves. An application would save a document under a name, for example `/parc/csl/names.pdf`, and then save the associated metadata to a derived name, such as `/parc/csl/names.pdf/meta/DublinCore`, where this object contains a set of Dublin Core [?] metadata tags. A document management system, upon seeing the document and its metadata inserted into CCN, could index the data so a user could find it via a dynamic query or via pre-published faceted search on a well-known namespace.

3. Security

This section describes the intrinsic mechanisms of CCN to support security and trust mechanisms in-network [?]. Those intrinsic properties allow the network to deliver the right content to each user and protect against content poisoning and malicious data injection.

One of the main tenets of CCN is to name content, instead of communication end-points. This also stipulates in-network content caching, by routers. To secure each content, the CCN design requires it to be (cryptographically) signed by its producer. This way, potentially globally addressable and routable content can be authenticated by anyone, which allows CCN to decouple trust in content from trust in entities that store and disseminate it (e.g., in-network caches). CCN entities that request content are called *consumers* and a consumer is expected to verify content signatures in order to assert:

- *Integrity* – a valid signature (computed over a content hash) guarantees that signed content is intact;
- *Origin Authentication* – since a signature is bound to the public key of the signer, anyone can verify whether content originates with its claimed producer;
- *Correctness* – since a signature binds content name to its payload, a consumer can securely determine whether delivered content corresponds to what was requested;

CCN's one key design objective is efficient and scalable distribution of content. This is facilitated by routers caching content. Whenever an CCN router receives an interest for a name that matches some content in its cache, it satisfies the interest with that content. However, the match of only the name does not mean the delivered content is guaranteed to be authentic (i.e., there might be many, potentially malicious, content objects sharing the same name). A consumer would always detect it if a fake content is received since it is required to verify signatures of all returned content and assumed to have the necessary application-specific trust context to decide which public keys to trust. However, this would still lead to a fundamental problem of network delivering consumers content that they would discard (a.k.a., content or cache poisoning attacks).

In order to address the above mentioned problem, it is necessary to recognize the obvious, i.e., that network-layer trust management/enforcement is required to prevent content poisoning attacks. Since content is the basic unit of network-layer "currency" in CCN, trust in content (and not in its producers or consumers) is the central issue at the network layer. Moreover, trust-related complexity (activities, state maintenance, etc.) must be minimized at the network layer. Specifically, as part of establishing validity of content, a router should not: fetch public key certificates, perform expiration and revocation checking of certificates, maintain its own collection of certificates, or be aware of trust semantics of various applications. On a related note, a router should verify at most one signature per content. This upper-bounds the heavier part of content-related cryptographic overhead; the other part is computing a content hash. Ideally, a router would not perform any signature verification at all, which is possible for most, yet not all, by the use self-certifying names (i.e., interest messages that specify the cryptographic digest of the content that is being requested) in CCN.

Our approach to network-layer trust is based on one simple rule, that we denote as Interest-Key Binding (IKB):

IKB: An interest must reflect the trust context of the consumer in a form enforceable at the network layer.

Recall that CCN interest format includes two optional fields: the KeyId of the publisher and the digest of the requested content object. Our approach suggests to have at least one of them in each interest packet.

A CCN public key is usually distributed via a special type of content in the form of a certificate signed by the issuing CA. Each certificate contains a list of all name prefixes that it is authorized to sign/verify. The name of the certificate-issuing (content-signing) CA and the name of the key contained in a certificate (content) are not required to have any common prefix. This is part of CCN's philosophy of leaving trust management up to the application, e.g., signed content C can be verified with public key PK with C and PK having no common prefix requirement. For instance, content containing the public key certificate `/cnn/usa/web/key` could be issued and verified by the key `/verisign/key`. Of course, an application is free to impose restrictions on the names of the keys and the content it is authorized to sign.

If an Interest message specifies a content digest or a keyId, CCN routers make sure a content object matches the criteria before responding to the interest with it. Thus, using the Interest Key Binding principle enables trust enforcement at the network layer and mitigate content poisoning attacks in CCN.

4. Conclusion

The CCN concept of using the same addressing namespace throughout the network stack advances the state of networking in a practical system with a codebase and deployment history. New CCN protocols and services, from forwarding algorithms to flow control to content manifests to application data access, unify to a common framework. We describe several notional services built on top of CCN, such as a file system, and database system, and a document management system. In some cases, what was once part of an application server computation now become named objects residing in the data namespace, such as ACL content objects or document metadata. Dynamic services, such as quorum servers, transaction services, or document indexing, can leverage the integrated network to application namespaces.