March 16, 2014
PARC

CCNx 1.0 Tutorial

Priya Mahadevan[1]

Abstract
The Internet's current architecture was designed and created about 40 years ago primarily to enable resource sharing. Its architecture was designed as a communication architecture allowing two machines to have a conversation. Over the past 40 years, we have seen significant changes in the way we use the Internet. Rather than use the Internet merely as a traditional communication network, we are now using it largely as for sharing and distributing information. As a result, there have been significant efforts to re-architect the Internet and evolve its infrastructure to support efficient dissemination of content. One such effort is Information Centric Networking (ICN) where the key idea is to name and address content directly rather than the end-points. These architectures allow a network device to obtain named data from a content producer, regardless of the content producer's physical or network location, or from any other device that has cached the same piece of content. In this paper, we describe in detail PARC's Content Centric Networking (CCN) architecture, an example ICN architecture.

Keywords
Content Centric Networks

[1] *Palo Alto Research Center*
***Email address**: priya.mahadevan@parc.com

Contents

1. Vision

The current Internet's architecture was designed and created in the 1960s and 1970s primarily to enable resource sharing. It was designed as a communication architecture to enable two machines, one using a resource and the other providing access to the resource, to have a conversation. Since those days, we have seen significant changes such as digitalization of all media, proliferation of social networking services, and online access and sharing of a wide variety of information. Rather than use the Internet merely as a traditional communication network, we are now using it largely as a distribution network.

Our network architecture needs to be generalized in order to effectively support these content distribution uses. Content-Centric Networking (CCN) [1] is an evolving *distribution* network architecture. The distribution framework is a strict superset of the communication one. CCN generalizes IP by relaxing restrictions such as addressing only named end-points. By enabling content addressing and content-based security, CCN enables more efficient and secure solutions to the problems the network is currently asked to solve.

The three key changes required to generalize from a communication model to a distribution model involve:

- Naming: In a communication model, all you need to name is the end-points of communication. In a distribution model, you name the goods that are being distributed. Today, users really want fast, efficient and secure access to content. They are rarely concerned with where the content is located. Thus, rather than name the end-points, CCN allows naming the content itself.

- Memory: In a communication model, memory (required in routers for statistical multiplexing) is invisible. In a distribution model, memory is something to transact with directly. Since the content is named, it can be obtained equally well from a memory as from a wire. By making memory explicit in the network architecture, CCN enables effective use of arbitrary quantities of memory throughout the network to trade storage for

Copyright © 2013-2014 Palo Alto Research Center, Inc.,
All Rights Reserved.

bandwidth, without elaborate and expensive configurations.

- Security: In a communications model, the security concerns of the network are limited to securing individual connections, the virtual "pipes" between end-points. In a distribution model, where content might be obtained from a variety of intermediaries other than the original source, we must focus on securing the content itself. This technique provides a more solid security foundation focused on what the users care about protecting. CCN requires cryptographic signing and verification of content as the basis for securing infrastructure, and supports encryption for restricting access to content.

2. Motivation

In the past decade, the Internet has witnessed an explosion of content. Furthermore, certain news articles, videos, etc. are much more popular than others. Traditionally, downloading a video from say www.cnn.com involves a user directly establishing a connection with one of cnn.com's servers and then proceeding to download the video. Highly popular content such as breaking news articles get repeated user requests, which are all handled by one of cnn.com's centralized servers, leading to congestion and bottlenecks near the servers. Content Delivery Networks (CDNs) such as Akamai and Limelight mitigated this problem by placing caches at strategic locations in the network. They are essentially a large overlay infrastructure comprised of a large number of caches, and serve contracted data. In a CDN, a client could access a copy of the data stored closer to the client, as opposed to all clients accessing the same central servers. However, deployment considerations force them to place these content caches at the peering point edges. Carriers are averse to having third parties such as Akamai and Limelight install their content caches in the carriers' Point of Presence (PoPs). Thus, these content caches are not integrated into the network - rather they connect to the network like an external application. Another disadvantage of CDNs are that these services are specific to contracted applications that are specifically modified to use them.

The proliferation of video in the past few years and its demanding Constant Bit Rate (CBR) communication patterns have further limited the benefits of using CDNs. Content caches at peering edges do not reduce traffic on the network backbones; they only reduce peering traffic. Traffic demands on the network backbone are growing due to growing video traffic and carriers cannot easily upgrade their backbones to handle this traffic surge.

More recently, Distributed Hash Tables (DHTs) are being used to store and access content. DHTs can scale to a large number of nodes, and they form a basic infrastructure over which a content distribution service can be built (for example see Coral Content Distribution Network - www.coralcdn.org and other peer-to-peer file sharing services built over DHTs). User requests for content are directed by the service to a relatively "close" node, thus aiming to minimize user-perceived latency. However, DHTs mask away key details of the underlying network topology; thus a node "close" to the user chosen by the service is not necessarily a node topologically close to the user. Further, efficient access to content in this model is critically dependent on correct placement of content in the caches. Content placement is challenging as it requires details about the underlying topology, constant changing network conditions, currently congested links, as well as predicting which content will be requested by the users. Moreover, getting the right placement strategy so that every client retrieves content from a topologically close node with minimal latency is almost impossible to achieve. Again, deployment considerations have ensured that storage associated with such services are typically placed at the network edges.

Some of the limitations of CDNs as well as DHT based content services are that they only offer storage at specific locations in the network and this storage is not integrated into the network. By separating the storage from the network and its demands, we have to resort to massive over-provisioning of the network if we need to meet any content access guarantees. For efficient distribution of content, we need to move towards a model where storage is an integral part of the network. Further, storage should also not be limited to specific points in the network topology. In an ideal scenario, memory would be present everywhere in the network and this memory fills itself up based on user demands.

CCN provides a network architecture to support the above requirements. CCN's policy of making memory explicit allows memory to be placed anywhere in the network including at routers inside a PoP. The data goes where there is demand and fills the caches automatically. Thus, in CCN, we no longer have to come up with strategies for content placement in the caches. This capability also limits the amount of configuration that needs to be performed in CCN. Further, most of the time data is retrieved from the topologically closest source, thus making efficient use of network bandwidth.

3. CCN Overview

CCN is built around secure name-based packet forwarding. While CCN is a new architecture for efficient, secure distribution of content, it is compatible with today's Internet. A CCN network is built around CCN nodes that perform name-based forwarding of packets between content consumers and content providers. CCN nodes also transparently cache content as it is being transferred, and respond with cached content when possible.

The CCN architecture was guided by the following design principles:

- Location independence: Content may come from any node that has the content, independent of location or source identity.

- Content immutability: CCN treats all published content

Copyright © 2013-2014 Palo Alto Research Center, Inc.,
All Rights Reserved.

as immutable. Changes introduce new versions, rather than altering existing objects.

- Flow balance: Every content request is satisfied by at most one content reply. No content is sent without a request.
- Extensibility: Only those mechanisms that support efficient and secure distribution of content are included in the core CCN architecture. Higher level services can be built on top of CCN.
- Cryptographic security: The integrity of content relies on cryptographic signing by the provider and verification by the consumer. Content itself is secured, not the connection or the underlying topology. Further, network links and storage used for the content transfers are not assumed to be secure.

Similar to IP, CCN may be deployed over any layer-2 technology. CCN can also be layered over IP. CCN can be layered over a higher protocol such as HTTP, should it be an expedient means to avoid firewalls. IP can also be deployed over CCN.

Deploying CCN in the existing Internet will initially layer all CCN packets on IP. Explicit tunneling between CCN nodes will be used to bridge segments of the network where no CCN nodes are placed. This feature allows incremental deployment of CCN in the Internet. Eventually, we envision regions in the Internet that primarily use CCN, with residual IP traffic possibly layered on native CCN nodes.

In the rest of this document, we describe the CCN architecture in greater detail. We describe how CCN operates in Section 4, CCN naming conventions in Section 5, and the role of a CCN node in Section 6. Transport and flow balance in CCN is addressed in Section 7. We discuss management and configuration in CCN in Section 8, CCN's strategy layer in Section 9, and security in Section 10. Next, we describe how CCN can be used for streaming (Section 11) and publishing (Section 12). Persistent storage of content in CCN is achieved through repositories that we discuss in Section 13. We detail examples of systems using CCN such as content distribution and mobile peering in Section 14.

4. Basic Operation

Communication in CCN is via two packet types namely the *Interest* packet and *Content Object* packet.

A consumer requests content (expresses an interest) by sending an Interest packet. Any CCN node that receives the Interest and has named data that satisfies the Interest responds with a Content packet (also known as a Content Object). Content satisfies an Interest if the Name in the Interest packet matches the Name in the Content Object packet. A hop limit is typically included in the Interest packet. Alternatively, an optional Nonce can be included in the Interest. The Nonce is a large random number used to detect and suppress duplicate Interests. If a CCN node receives an Interest, and does not already have a copy of the requested Content, it may forward the Interest towards a source for the content. The CCN node has forwarding tables that determine which direction to send the Interest.

A provider receiving an Interest for which it has matching named content replies with a Content packet. Besides the requested data, the Content packet contains the full name, a signature for the packet, information that identifies the signer, and supporting details.

A Content packet may optionally be verified at any intermediate router, but this check only ensures that the Signature was created by the key given by the SignedInfo, and includes the Name, the SignedInfo, and the Data. We address the issue of trust in the key itself through other techniques (Section 10). Any intermediate node can optionally choose to cache the Content Object and it can respond with a cached copy of the Content Object the next time it receives an Interest packet with the same name.

5. CCN Names

It is important to note that CCN provides a naming framework, and not a naming architecture or naming policy. Each CCN name has one or more components, and each component is a sequence of bytes. While there are no architectural limits to the number of components, the size of the components, or the contents of the components, there might be implementation limits.

CCN names need not be human readable. For names that are intended to be human readable, the current convention is to use UTF-8 encoding. A CCN name is similar to an URI, where 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 or timestamps. There are also application specific labels. An example CCN name for content might look like:
/parc.com/CCN/documents/CCN-Introduction.pdf

In the above example, the first segment can be a globally routable name under the control of some organization and can be derived from a DNS name assigned to that organization. Other conventions can also be used. The remaining segments in the name are application specific. Protocol specific segments such as version number or chunk numbers can also be included in the name.

CCN name matching is based on exact equality for segment. Exact matching for two names requires a match for all corresponding segments. Prefix matching requires that all segments in the prefix be equal to corresponding segments in the name. Since name matching examines entire segments, simple hash techniques can provide efficient name matching.

Copyright © 2013-2014 Palo Alto Research Center, Inc.,
All Rights Reserved.

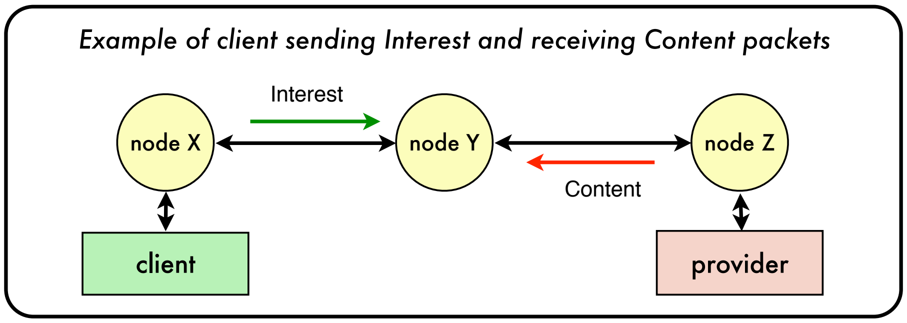
Example of client sending Interest and receiving Content packets
Interest
node X
node Y
node Z
Content
client
provider

Figure 1. Example of client sending Interest and receiving Content packets

6. CCN Node

In the CCN architecture, a CCN node model is comparable to an IP node model described in RFC 791. The basic operation of a CCN node is similar to an IP node. CCN nodes receive and send packets over faces. A face is a connection point to an application, or another CCN node, or some other kind of channel. A face may have attributes that indicate expected latency and bandwidth, broadcast or multicast capability, or other useful features.

A CCN node accepts Interest packets and either sends them out on an outgoing face, or replies directly with a matching Content packet. If the node has forwarded an Interest packet then it should normally receive a responding Content packet which it will send to the original requesting face. A CCN node has three main data structures: Content Store (buffer memory), Forwarding Information Base and Pending Interest Table.

The Content Store (CS) holds a table of previously seen (and optionally cached) Content packets indexed by the Name field of the packet. Besides providing communication buffering, the CS serves as a content cache. The CS is the same as the buffer memory of an IP router but has a different replacement policy. Since each IP packet belongs to a single point-to-point conversation, it has no further value after it is forwarded. Unlike IP, CCN packets are self-identifying and self-authenticating and can potentially be useful to many users. Thus, in order to optimize the use of network bandwidth, and reduce user-perceived latency, CCN nodes store the Content packets in their CS.

In CCN, only Interest packets are routed. Matching Content packets follow the reverse-path of the corresponding Interest packet. The Pending Interest Table (PIT) is used to keep track of Interests forwarded upstream by that CCN node toward the content source so that Content packets later received can be sent back to their requestor(s). The PIT holds a set of entries, each entry containing a previously seen Interest packet and a list of the faces that received the interests. There may be additional information, such as timeout values, that affect the entries.

The Forwarding Information Base (FIB) is used to forward Interest packets towards potential data sources and is analogous to the FIB in IP routers. The FIB holds a set of entries, each entry containing a name prefix and a list of the faces that might provide content for that prefix. There may be additional control information that affects forwarding.

We next describe the sequence of actions that are taken in order by CCN nodes when they receive an Interest or Content packet.

When an Interest packet P with content name N is received on face F,

- *Check for duplicate Interest packets:*
  if there are recently seen Interests with the same name or P has reached its hop-limit,
  then P is discarded (the definition of "recently seen" is implementation dependent)

- *Check for existing data:*
  if there is a Content Object in the CS whose name exactly matches N
  then send that Content packet as a reply, and P is discarded

- *Check for duplicate Interests waiting for replies:*
  else if N exactly matches the name in an entry of the PIT
  then F is added to the face list for that entry and P is discarded

- *Forward the Interest towards a potential provider:*
  else if there are any entries in the FIB with names that are prefixes of N,
  then the face list L is taken from the entry with the longest matching prefix, and if L is not empty, then
  - a new entry is made in the PIT for P and F, and is forwarded along one or more of the faces in L (possibly in parallel)

When a Content packet D with content name N is received on face F,

- *Discard duplicate replies:*
  if there is an exact match for N in the CS
  then D is discarded (as D is a duplicate)

Copyright © 2013-2014 Palo Alto Research Center, Inc.,
All Rights Reserved.

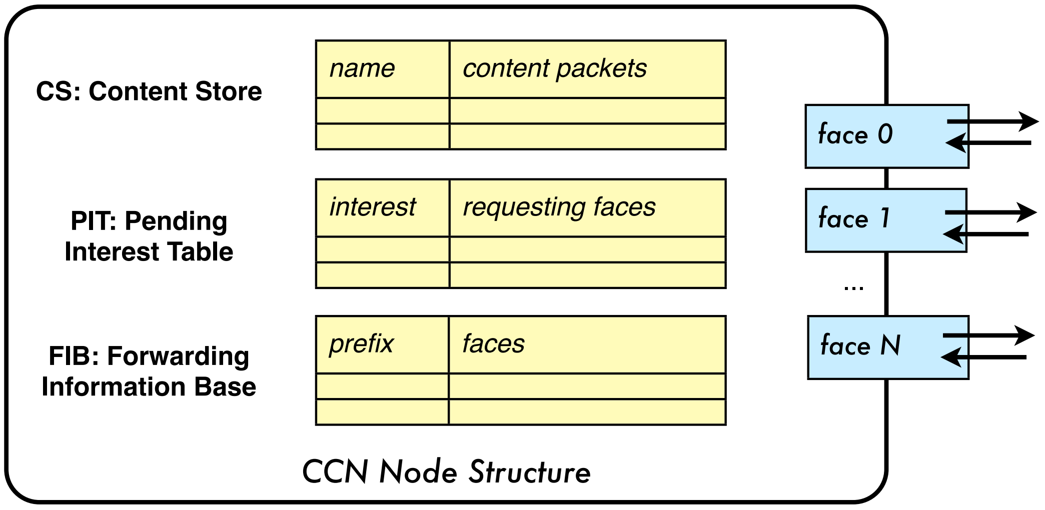

Figure 2. ccnd structure

- *Discard unsolicited replies:*
  else if there is no match in the PIT
  then D is discarded (D is unsolicited)
- *Forward replies to requestor:*
  else, the matching entry in the PIT has a face list L, and D is forwarded to every face in L, and the PIT entry is discarded
- *Store in the CS:*
  Optionally choose to cache D in the CS

A CCN node may forward Interest packets to one or more connected CCN nodes. Usually the best approach is to forward the Interest over the face that is most likely to respond first, based on statistics of past responses. However, sometimes forwarding packets to other faces, or even to all faces, is useful to detect malfunctioning links or to refresh statistics associated with the links. CCN supports other strategies as well and we discuss these in Section 9.

Since only Interest packets are forwarded (routed) in CCN and Content packets take the reverse of the path taken by the Interest packets, Content packets can be stored at all the intermediate CCN nodes between the content consumer and content producer. Typically, entries in the CS are removed using eviction polices such as Least Recently Used (LRU) or Least Frequently Used (LFU), though other policies can be used as well. Popular content will typically stay in the CS and seldom gets replaced, and thus Interest for popular content most likely will get served from the CS of the closest CCN node, thus reducing network bandwidth usage. We discuss large size storage in Section 13.

The Content retrieval mechanism in CCN ensures that popular Content packets will typically be on the path between content consumer and producer. Since storage is everywhere in the network and more importantly well integrated in the network, CCN does not have to deal with the usual storage management and content placement concerns that are inherent to pure storage-based topology-agnostic systems like DHTs.

7. Transport and Flow Balance

As stated previously, CCN can run over any layer-2 technology or above. The CCN protocol requires very little functionality from the underlying packet transport. CCN simply assumes stateless, unreliable, unordered, best-effort delivery of packets. Thus, Interest and/or Content packets might be lost or corrupted in transit, or the requested Content might be unavailable. In CCN, the return path for the Content packet is the reverse of the path for the Interest packet. Intermediate CCN nodes might fail as well, resulting in Content packets not reaching the consumer (the application that sent the initial Interest packet).

Reliability in CCN is achieved by retransmitting Interest packets that have not yet been satisfied. The onus is on the content consumer to retransmit unsatisfied Interest packets. The consumer can incorporate a timeout for every Interest packet it sends. Upon a timeout, an unsatisfied Interest packet can be retransmitted.

CCN enforces flow balance, so that one Interest packet results in at most one Content packet. As in TCP, it is not necessary, or even desirable, to wait for a reply before sending out the next Interest, so many Interests may be in flight simultaneously. The Interest packets in CCN are analogous to window advertisements in TCP. Unlike TCP, however, lost packets do not stall the pipeline, as CCN packets are independently named and are not part of a specific conversation.

CCN maintains flow balance to enable efficient communication over links with varying latencies and bandwidths and nodes with varying capabilities. Flow balance at every node allows for simple and effective techniques to avoid congestion. The underlying transport may receive duplicate Interest and

Copyright © 2013-2014 Palo Alto Research Center, Inc.,
All Rights Reserved.

Content packets. All duplicate Content packets are dropped by CCN nodes as described in Section 4. Duplicate Interest packets received over different paths are also detected and dropped through the use of a hop limit.

CCN transport can forward an Interest on multiple faces. This feature is especially useful in dynamically determining the best face to use based on varying network and traffic conditions. We discuss this feature in greater detail in (Section 9).

8. Management and Monitoring

CCN provides building blocks over which one can implement a variety of network protocols and services such as a routing protocol's transport, and network monitoring and management protocols. CCN's information-oriented guided-diffusion flooding model is an ideal model over which routing protocols can be built. Control messages such as route updates in a routing protocol can be disseminated as CCN Content packets. Since CCN Content packets contain Name, SignedInfo and Signature, a node receiving a route update can validate that the update information has been sent by a trusted routing peer.

Similar to supporting a routing protocol's transport, CCN can also be used to build network management monitoring protocols such as SNMP. SNMP queries map to CCN Interest packets and SNMP responses map to CCN Content packets. CCN's security model of digitally signing Content packets can be used to determine if the SNMP response is from a valid host.

By serving content from CCN nodes topologically close to the content consumer, CCN lowers the data plane operational costs for carriers. Additionally, CCN also lowers control plane operational costs for carriers as CCN nodes can be easily deployed in the network. Unlike IP nodes, CCN nodes are self configuring and impose little set-up overhead on network operators. Operators only need to configure higher level policies on CCN nodes.

9. Strategy Layer

CCN's performance depends on its ability to appropriately forward Interest packets in the network - either to the closest content cache or ultimately to the content source when necessary. As described in Section 7, CCN packets do not loop. Combined with the fact that CCN can send the same Interest packet on multiple faces, one interesting choice in CCN is determining the right face (or faces) to forward the Interest packets. The strategy layer performs this role in CCN.

Traffic patterns and network conditions change over time. For instance, routers may fail, links may go up or down, links may get congested, or routing contracts between peering ISPs may dictate which links should be used based on the time of day. Thus, while the FIB in a CCN node may contain multiple faces for a content prefix, only a subset of them may be an optimal choice at any given time. The strategy layer governs sampling of bandwidth and response time at periodic intervals in order to determine the best faces in real time. If all faces are used for 1% of the traffic (for example) then the extra load will be minimized, but changes in response time (including for broken links) will be recognized and used adaptively.

The strategy layer can also increase the bandwidth by noting when the use of two faces outperforms the use of a single face for multiple segments. For example, there might be two available paths from a requesting node to a nearby cache. Then, sending different Interest packets over the 2 faces (and thus using both faces simultaneously) might yield better performance than using just a single face for forwarding all the Interest packets. This decision is dynamic since the choice may shift when using one of those faces for other traffic.

The strategy layer is also used to propagate Interests to appropriate CCN nodes so that prefixes newly registered by publishers can be found and forwarded appropriately. The strategy might be as simple as flooding, or selectively forwarding only to nodes that are specially constructed to hold large FIB tables, etc.

10. Security and Trust

CCN supports content-based security, rather than connection-based security. In CCN, every Content packet is authenticated with digital signatures and private content is encrypted. This key attribute is what enables CCN to retrieve content from any CCN node without contacting the Content's source. Since the name and the content are cryptographically bound together, it is not necessary to trust the nodes or the connection.

Signature verification provides a means for securely associating a name, a publisher, and content together. With appropriate verification only valid content will pass to the client. As a side benefit, verification provides error detection for Content packets with a high degree of reliability.

Every Content packet has the following fields:

- Signature
- Name: the full CCN name for the content
- SignedInfo: key hash, key locator, and other details
- Data: the binary representation of the content

Every Content packet is signed with a signing key such as a public key. The signature is computed over the entire packet content exclusive of the Signature field, and so includes the Name, the SignedInfo, and the Data. The exact method for computing the signature depends on the cryptographic algorithm associated with the signing key.

The SignedInfo includes a hash of the key, a key locator, a timestamp, and some additional details not within the scope of this document. A key locator may either include the public key directly, or it make contain a name used to find the key. Signature verification proves that the content can be trusted, provided that the key can be trusted. CCN does not mandate policy decisions about trust. The level of trust for the key

Copyright © 2013-2014 Palo Alto Research Center, Inc.,
All Rights Reserved.

depends on a trust model that is chosen by the client. The information included in the Content packet must support a reasonable set of models, but is not dependent on them. In practice, we expect several trust models, including a choice of public key infrastructure.

By providing a mechanism for transporting authenticated data, CCN makes key distribution itself more reliable as keys can be distributed as CCN Content Objects. Privacy can be incorporated in CCN and can be layered on top of CCN.

11. CCN and Streaming

Today, the majority of traffic on the Internet uses TCP or UDP for streaming data from one host to another. In CCN, applications that need to use streaming leverage CCN's naming scheme and the fact that flow control is in-built.

We illustrate how streaming works in CCN with a simple example. Consider two nodes, Node$_A$ and Node$_B$, where Node$_B$ expects a sequence of packets from Node$_A$. The prefix for the content from Node$_A$ is /A, and the prefix for content from Node$_B$ is /B. The network is configured such that interests in /A are routed to Node$_A$, and interests in /B to Node$_B$, although nothing prohibits other nodes from advertising those prefixes. For generality, name the stream S to allow for additional streams. Then a schema for the messages exchanged is:

Node$_B$ expresses an interest in /A/S

Node$_B$ receives a Content packet from Node$_A$ with name /A/S/V/N/H
where V is the stream version, N is the first segment number (typically 0), and H is the hash of the data. The response is just one packet of data satisfying the named Interest and the public key locator for $Node_A$, which Node$_B$ can use to verify that Node$_A$ produced the response.

To continue the stream:

Node$_B$ expresses an interest in /A/S/V/N+1
where V is the version number from the first response, and N+1 is the successor to the previous segment number N.

Node$_B$ receives a Content packet from Node$_A$ with name /A/S/V/N+1/H
which includes the data, and, when it is known, the last segment number that can be used to determine the end of the stream.

We can easily compose a bi-directional stream from a pair of unidirectional streams. The name S can be used for both directions. The name pattern above simply swaps Node$_A$ and /A with Node$_B$ and /B. Any stream can be read by any number of nodes, usually at no extra cost to the stream originator or to any receiver, since recent packets will be cached in the CCN nodes along the paths.

Analogous to TCP, multiple interests may be outstanding, so pipelining can occur. The order of arrival of responses is arbitrary, so client applications need to limit the number of outstanding interests. Client applications also determine values for timeout and retransmission.

12. Publishing

A publisher can publish its content under a given name prefix. For example, all content published by PARC can be under the /parc name prefix. When an Interest is delivered to the publisher, a Content packet can be generated in response. The publisher signs each Content packet with the publisher's private key so it can be verified using the publisher's known public key.

In CCN, content that changes over time is represented as having multiple versions. It is possible to represent new versions that are minimally changed from old versions as a set of changes along with a reference to the older version, and this approach can provide significant bandwidth savings. However, such a convention would be layered on top of the core CCN protocol. CCN just provides a naming scheme that supports different versions.

Since CCN is a form of packet switching, large content must be represented as multiple chunks. This feature allows for partial caching and delivery of partial results before the entire content can be generated.

There is no mechanism in the core CCN protocol for deleting or revoking access to content. It is not always possible to find all of the copies in an extended network. Rather, when publishing content, one gives it a time-to-live, so that cached copies expire. Thus, only the originating sources for the content need to have their copies deleted to eventually ensure that the content is no longer in the network (although clients cannot be guaranteed to not have copies).

13. Repositories

The CCN architecture makes it easy to incorporate large quantities of persistent storage directly in the network, in addition to the buffer memory used for Content and Interest packets in the forwarding operations of a CCN node (Section 6). We give the name *repository* to any component that implements persistent storage of Content packets. A repository implemented in software for general purpose computers would typically use the filesystem as its stable backing store (as our early prototype software did), while a repository in the network would be implemented in hardware on a specialized line card (or equivalent) and use SSD for storage. The benefit of a separate repository memory is that it can easily have a capacity vastly larger than the buffer memory in the forwarding data plane, while remaining an integrated component of a router that is local and tightly coupled to forwarding. This design enables large-scale, transparent content caching to operate throughout the network.

To function as a cache, a repository connects to its CCN node via a type of face with a special role with respect to forwarding strategy. When the FIB lookup result includes the

Copyright © 2013-2014 Palo Alto Research Center, Inc.,
All Rights Reserved.

repository face, the strategy layer will send the unsatisfied Interest on that repository face first. The repository can very quickly determine definitively whether or not it has a Content packet to satisfy the Interest, since the repository is local (on the same hardware) and contains a store of packets. If the repository finds a matching Content packet, it returns it immediately and the packet is handled normally. If the repository does not find a matching Content packet, it immediately returns a NACK (a special type of Content packet) which will cause the CCN node to immediately forward the unsatisfied Interest on another face, without storing the NACK response in the CS. This small variation on the normal forwarding rules for repository faces allows the CCN node to check its local cache memory and get a definitive answer instantly, without any timeouts.

Given a finite memory, the repository must implement some replacement policy, but it can use a simple, fixed policy such as Least Recently Used (LRU) to find content to evict when memory is full. With a small amount of configuration, however, a repository can operate as a specialized cache that handles only certain content or can vary replacement policy or available memory based on the content. In these cases, the repository handles different name prefixes differently. If the network operator is renting memory in the network to a CDN like Akamai (or directly to a content producer), then it might configure routers so that the repositories handle only the name prefix of the customer. It may be that some storage space is allocated in this way to different customers explicitly, while some is left available for caching of arbitrary content responsive only to dynamic demand.

A repository may serve a role similar to a network file server as a persistent content store to which applications may explicitly write data. For example, routing and security data (such as encryption keys used for access control) will often need to be preserved locally. In the CCN architecture, the distinction between storage and communication is minimized, and it is appropriate to use Content packets with content-based security for a variety of storage purposes.

For general content storage, as in a distributed file system, there may be applications that need to merely discover what content exists in order to do things like present a file browsing interface to human users. The repository can implement a name enumeration protocol that allows discovery of names delivering the content. The example CCN repository can enumerate a prefix as if it were a directory name, although directories need not actually exist as such. The Interest sent to the repository can include markers that indicate the level of detail for the enumeration. The content for the enumeration results is dynamically generated to reflect the contents of the repository.

Given that CCN facilitates location independence and highly distributed operation, one can have distributed replicas that hold the same collection of content. Replication and consistency protocols can be implemented on top of the core CCN protocol.

14. Examples

In this Section, we discuss sample applications that leverage CCN's capabilities.

14.1 Content Distribution Example

Content distribution is one example application that benefits greatly when deployed in a CCN network. In the following example, there are 2 repositories, R1 and R2. There are 5 clients (content consumers): A, B, C, D, E. Each repository and client also contains a CCN node, and the network connectivity is shown in Figure 3. Both repositories contain identical content (for example named /X).

We assume that each client A through E asks in order for content named /X. The arrow $--->$ represents expressing an interest in /X, and the arrow $==>$ represents a content reply. We use "R store" to indicate that the content comes from the persistent store of the repository, and "R cache" to indicate that the content comes from the CCN node cache. Then, one possible message log is:

A $-->$ R1 store $==>$ A
B $-->$ R1 cache $==>$ B
C $-->$ B cache $==>$ C
D $-->$ R2 store $==>$ D
E $-->$ D cache $==>$ E

The above is only one potential order. C might have received content either from R2 or from B cache (as discussed in Section 9). Either is correct, and the outcome varies depending on link bandwidth and network load. While retrieving mutiple chunks of content, part of the content could come from B cache and part from R2. Similarly, E might receive content from the D cache or R2 cache. In any case, there are only two full content fetches - one from each repository. The other retrievals come from various caches.

We note that in CCN, the caches automatically get filled based on requests from content consumers. For content distribution systems deployed over DHTs, content publishers need to explicitly inform the DHTs about their content location before the content can be retrieved. Also, unlike CCN, there are no guarantees that this retrieval will be from the closest available copy.

There are many potential policy decisions about content distribution, but these are well above the level of the CCN protocol, which is only intended to provide a solid mechanism for realizing those higher level policies. A example of a policy decision is implementing access control for the content.

14.2 Mobility

In this example, we discuss two different styles of mobility and describe how CCN can be used in each case. These examples are intended for illustration only. We stress that other approaches are also feasible.

Copyright © 2013-2014 Palo Alto Research Center, Inc.,
All Rights Reserved.

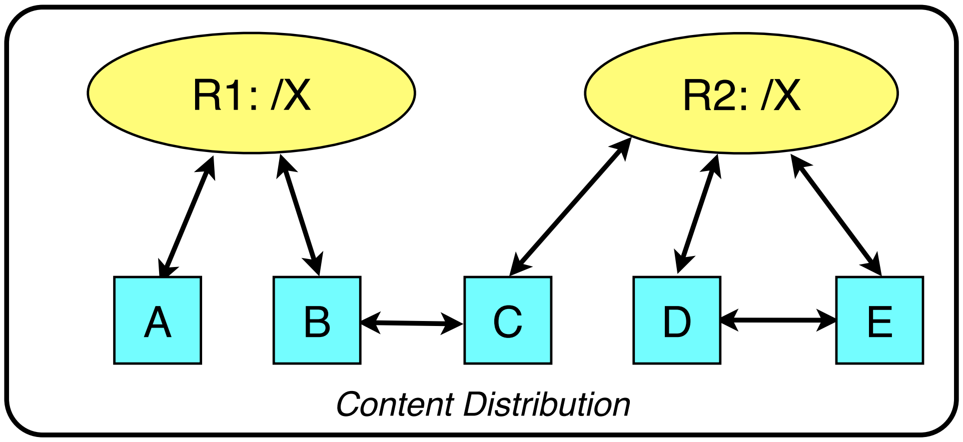

Figure 3. Content Distribution Example

Mobile Phone Scenario

This example involves a mobile user communicating using CCN. Consider a user U in control of a mobile unit M. We also have nodes A, B, and X. We use /PM, /PX, /PA, and /PB to denote prefixes that are unique for each of the nodes. The goal is to show how X communicates to U through M.

Since M is mobile, routing becomes an important issue. We assume that the provider network where M is registered provides an agent connected to node A that is used to as a static routing point for forwarding traffic to M.

When M connects to a provider network it is connected through node B and assigned a temporary prefix /T that can be routed by the network through B. The provider network for B can reach A either internally (if they share the same network) or through peer network routing. M publishes content under /T prefix, so Interests in /T can route to M. M expresses an interest in /PA/update/PM/T, which lets A know the routable new prefix T should be used for M.

Moving packets from some client X to M can either use A as a proxy, or can query A to determine /T and transfer to M using the /T prefix directly. If X is also mobile then the pattern is more indirect, since there will be an agent for X, but the rest of the protocol is similar.

If the connection between B and M is broken then the address T becomes invalid. When M reconnects to a network, a new address T' is assigned, and communications can resume once agent A becomes aware of the new address. This renewal may use B if the disconnection is transient, or it may use a new node C, which may be in a different network (since M may be in a new location). When T becomes invalid, then any client X that was using T must contact A to determine T'.

Mobile Peer Scenario

This scenario assumes a network of nodes, many of which are presumed to be potentially mobile, and communication is handled by sending messages over one or more faces to temporarily reachable nearby nodes. An example of this pattern would be a disaster relief operation where the more static infrastructure is unavailable.

In a CCN node, interest forwarding chooses which faces to use to reach the desired content. The mechanism is to store prefixes and paths into the FIB. In the mobile peer case there is unlikely to be a single solution, because communication costs are extremely variable with configuration and scale, and FIB entries can get stale quickly.

Perhaps the simplest example is where flooding is used to contact adjacent nodes to forward interests within a region. This method works well for ad hoc networks that have highly mobile nodes or unreliable communications [2]. While broadcast is not suitable for large scale networks it can be quite efficient in regions that have a small number of mobile nodes that need to mutually communicate. Further, if used infrequently (e.g. when the best path is unknown) then flooding can be practical even in cases where the number of nodes is moderate.

In the above figure we show a simple hybrid scenario where one node, X, is a relatively static connection to an external network. The local region (A, B, C, D) is small enough to permit flooding as a viable means to forward Interests. The policy may use direct forwarding if the Interest reaches X.

For the larger network to obtain content published within the local region, there needs to be advertisement of prefixes by some agent in the larger network, such as X in the above example. It is also possible for a local region to be connected to multiple static networks.

15. Quality of Service

The widespread successful deployment of VoIP has demonstrated that real-time conversational traffic can be easily handled by datagram packet architectures. PARC has experimented with Voice over CCN (VoCCN) [3], which is similar to VoIP, but has some additional advantages such as better handling of mobility, security, and advanced services such as conference calls.

Copyright © 2013-2014 Palo Alto Research Center, Inc.,
All Rights Reserved.

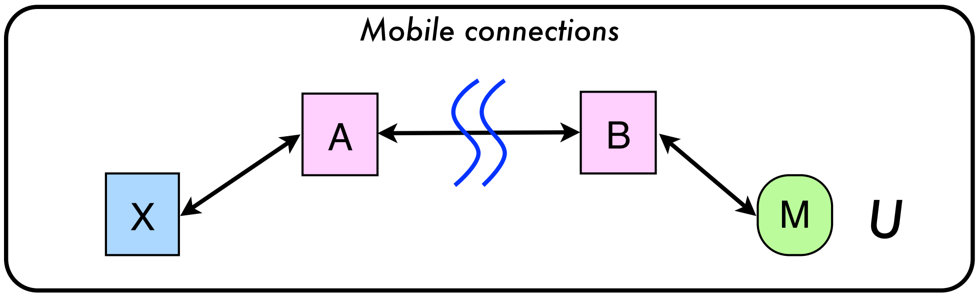

Figure 4. Mobile Connections

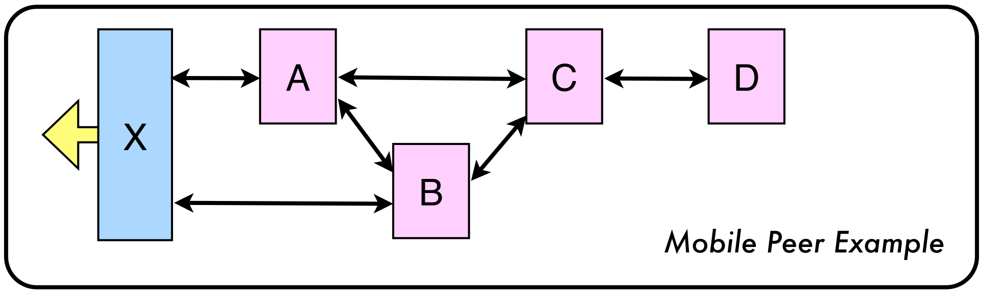

Figure 5. Mobile Peer Example

ISPs have identified two areas where better QoS capabilities are needed (for example, see [4]):

> At the peering edge: content-driven control of the traffic mix on congested peering links and in the pop-to-core aggregation network. E.g., if there are 10,000 customers watching eight TV shows from Hulu.com and two from ESPN.com, the sharing granularity should be the ten shows (content items), not the 10,000 conversations or two source addresses.
>
> At the customer edge: customer-driven control of the traffic mix on the congested "last mile" link from the ISP to that customer. E.g., a customer should be able to tell the upstream ISP router that updates for his multi-player game are most important, streamed movie is second priority and everything else third.

The first of these problems is difficult to solve with IP/TCP since packets contain only endpoint identifiers. Solutions include using Deep Packet Inspection, Content Sensitive Load Balancers and/or Transparent Proxies but they are expensive, brittle and difficult to scale. Since CCN data packets always start with the content name, the CCN forwarding lookup can be followed by a simple multi-level queuing or scheduling structure to implement inexpensive, robust and scalable bandwidth sharing in the content domain. The French Agence Nationale Recherche has recently funded Orange (France Telecom), Alcatel Lucent, Inria, Telecom Paris Tech and University of Paris Marie Curie (Sorbonne) to implement and analyze this architecture as part of the CONNECT project [5].

The second problem, customer control of the last mile link, is difficult to solve in IP/TCP because IP only identifies 'endpoints' and offers no way to identify or communicate with intermediaries such as the next hop router. In CCN, local intermediaries are an explicit part of the architecture. In particular a downstream can tell its upstream, via a 'priority' field in an Interest, the relative importance of different content items. This is a totally new capability, impossible in IP, that gives a customer simple, fine-grained, real-time control over queuing in the ISP-to-customer direction.

References

[1] Van Jacobson, D. K. Smetters, James D. Thornton, Michael Plass, Nick Briggs, and Rebecca Braynard. Networking Named Content. In *CoNext*, 2009.

[2] Michael Meisel, Vasileios Pappas, and Lixia Zhang. Ad hoc networking via named data. In *MobiArch'10*. ACM, 2010.

[3] Van Jacobson, D. K. Smetters, Nick Briggs, Michael Plass, Paul Stewart, James D. Thornton, and Rebecca Braynard. VoCCN: Voice-over Content-Centric Networks. In *ReArch*, 2009.

[4] James Roberts. What QoS for the future internet? In *The proceedings of FISS09*, July 22, 2009.

Copyright © 2013-2014 Palo Alto Research Center, Inc.,
All Rights Reserved.

[5] INRIA. CONNECT project. https://www-roc.inria.fr/twiki/bin/view/RAP/ContentCentricNetworks.

Copyright © 2013-2014 Palo Alto Research Center, Inc.,
All Rights Reserved.