Content Based Addressing

The case for multiple Internet service providers

ROBERT MÖRT

KTH Information and Communication Technology

Degree project in
Communication Systems
First level, 15.0 HEC
Stockholm, Sweden

KTH Royal Institute of Technology
Kungliga Tekniska Högskolan

Bachelor's thesis
Version 1.0
Date: 6/12 - 2012

Content Based Addressing: The case for multiple Internet service providers

Bachelor's thesis

Robert Mört

rmort@kth.se

Examiner: Professor Gerald Q. Maguire Jr.

School of Information and Communication Technology
KTH Royal Institute of Technology
Stockholm, Sweden

Abstract

Today's Internet usage is changing from host-to-host communication to user-to-content interaction which proves a challenge for Internet Service Providers (ISPs). Repeated requests lead to transfers of large amounts of traffic containing the same content often over costly inter-ISP connections. Content Distribution Networks (CDNs) contribute to solving this issue, but do not directly address the problem. This thesis project explores how content based addressing could minimize inter-ISP traffic due to repeated requests for content by caching content within the ISP's network. We implemented CCNx 0.6.0 in a network testbed in order to simulate scenarios with multiple ISPs interconnected to each other. This testbed is used to illustrate how caching of popular content minimizes inter-ISP traffic as well as how content independence minimizes the effect of other network problems such as link failures and congestion. These tests shows that the large overhead of the CCNx implementation due to the additional headers brings a 16% performance reduction compared to Hypertext Transfer Protocol (HTTP) transfers. However, these tests also shows that the cost from the inter-ISP traffic of CCNx transfers are constant regardless of the number of repeated requests, due to content caching in the ISP's network. As soon as there is more than one request for the same content there is a gain in using CCNx rather than HTTP for content transfer.

Abstrakt

Dagens användning av internet ändrar form från dator-till-dator kommunikation till användare-till-innehålls interaktion vilket innebär nya utmaningar för internetleverantörer vilka måste överföra stora mängder upprepade förfrågningar av innehåll via kostsamma länkar mellan internetleverantörer. Lösningar som innehållsdistribuerande nätverk (Content Distribution Network) hjälper idag till men addresserar inte kärnan av problemet. Det här examensarbetet undersöker hur innehållsbaserad addressering kan minimera mängden trafik mellan internetleverantörer genom att cachning, att lagra kopior av innehåll, i internetleverantörers nätverket. I det här examensarbetet implementerade vi CCNx 0.6.0 i en testbädd för att simulera scenarion med nätverk mellan internetleverantörer. Denna testbädd används för att illustrera hur cachning av populärt innehåll kan minimera trafik mellan internetleverantörer samt hur innehållets oberoende av plats även hjälper till med andra problem i nätverket såsom länkfel och stockning. Dessa test visar att CCNx implementationen har stor overhead information på grund av ytterligare, extra headers vilket medför en 16% reducering i prestanda jämfört med överföringar som använder Hypertext Transfer Protocol (HTTP). Vidare visar dessa tester även att kostnaden från trafik mellan internetleverantörer är konstant oberoende av antalet upprepade förfrågningar, på grund av cachning av innehåll i internetleverantörens nätverk. Så snart det finns fler än en begäran för samma innehåll finns det en vinst i att använda CCNx istället för HTTP för överföringar av innehåll.

Acknowledgements

I would like to thank my academic supervisor and examiner Professor Gerald Q. Maguire Jr. for introducing me to this field, and his continuous support and feedback during this thesis project. I would also like to thank my family and friends for always being there and reaching out with a helping hand or moral support when needed.

Table of Contents

List of Figures

List of Tables

Abbreviations and Acronyms

BGP	Border Gateway Protocol
CCN	Content-Centric Networking
CDN	Content Distribution Network
CPU	Central Processing Unit
DNS	Domain Name System
DONA	Data-Oriented Network Architecture
ETH	Ethernet
FIB	Forwarding Information Base
FTP	File Transfer Protocol
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
IGP	Interior Gateway Protocol
IP	Internet Protocol
ISP	Internet Service Provider
LFU	Least Frequently Used
LRU	Least Recently Used
RH	Resolution Handler
SCTP	Stream Control Transmission Protocol
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
URN	Uniform Resource Name
VPN	Virtual Private Network

1. Introduction

Today's Internet architecture, based on the TCP/IP protocol stack, was developed as a means for endpoints (also known as hosts) to communicate with each other on a global scale. However, usage of the Internet has evolved from host-to-host communication to user-to-content interaction in recent years. This change is also mirrored in a change in traffic patterns. This change in both traffic and traffic patterns is causing problems, especially for the Internet service providers (ISPs).

Different solutions have been proposed, but unfortunately they do not directly address the issue. For example, many websites with large amounts of content use Content Distribution Network (CDN) solutions (such as Akamai), to reduce the load on their servers and to reduce the bandwidth required for their network connectivity. Akamai does this by placing servers near the consumers and by exploiting the Domain Name System (DNS) to redirect the user to the most suitable (hopefully local) server. In this way the user's requests will be distributed over different servers, thus the traffic will be distributed across these different servers and different portions of the Internet. However, a closer analysis of the traffic shows that many times the large amount of content distribution traffic is actually due to a relatively small amount of data being sent to many different users. Thus, the bandwidth (particularly of backbone links) is used in an ineffective way. Furthermore, it is up to Akamai to locate the copies of content and their servers in such a way as to optimize the load over their servers and the traffic on their links to the different ISPs.

This phenomenon of many users requesting particular items of content combined with the ever-increasing demand for bandwidth by customers and ever higher Internet penetration rates provides a real challenge for ISPs as they attempt to manage their networks and external links to other networks. Akamai's "State of the Internet" [1] report shows that the average connection speed in Sweden, as observed by Akamai, increased by 7.2% over the time period Q3 2010 to Q3 2011. Increased traffic to the ISP from peers (either at peering points or via private links) costs money, therefore ISPs would like to minimize the *repeated* transfer of the same content, hence reducing their peering traffic while still satisfying all (or most) of their customers. Deploying content based addressing technologies can help reduce traffic between ISPs, as well as reduce the ISP's required investment in their own infrastructure.

This report assumes a knowledge of the reader equal to a Bachelor's degree in computer networks. To fully understand the background of this thesis, the reader is encouraged to read the PARC technical paper "Networking Named Content" [2], as this thesis project was inspired by this technical paper.

1.1 Introduction to content based addressing

The main idea behind content based addressing is to address content instead of addressing hosts, thus separating content from its location. In this approach content is addressed using names, typically consisting of a provider name, content name, and version. As content traverses the network, routers and hosts either cache the *location(s)* where they can find a copy of the content or cache an actual copy of the content, enabling future requests for that content to be delivered by any node that currently holds a copy of the content [2]. This enables the use of multiple sources and multiple destinations natively by the network (i.e., creating a multi-path topology to access the content).

This means that network designers needs to consider this change in how content is addressed and how it is delivered when designing and building networks. For example, a network designer must decide whether (and which) routers cache locations or cache named content (and what content) in order to design effective networks. A frequently requested subset of content could be cached close to the users, while awareness of a larger subset of less frequently requested content would be cached at higher capacity backbone or data-centers, which in turn know the location of rarely requested content (the rest of the content on the Internet) via external links. This would lead to faster and cheaper delivery of popular content with the worst case scenario of content retrieved from the source, a transfer similar in speed and cost to traditional methods of addressing content by location, resulting in "win or no loss" with respect to performance and the resources needed to satisfy the users' requests.

The basic difference between addressing content by name and addressing content by location is demonstrated in Figure 1 and Figure 2, where dotted arrows represents requests for content and continuous arrows represents transfers of content. In Figure 2 the content is uniquely identified by its Uniform Resource Locator (URL), meaning that every request for this content can only be satisfied by that host. In Figure 1 content is uniquely identified by its *name* (perhaps as encoded in a Uniform Resource Name (URN)), meaning that any node with a copy of this content can satisfy the request.

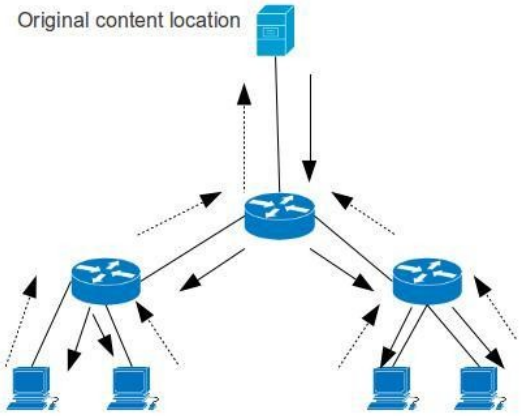

Figure 1: Content addressed by name

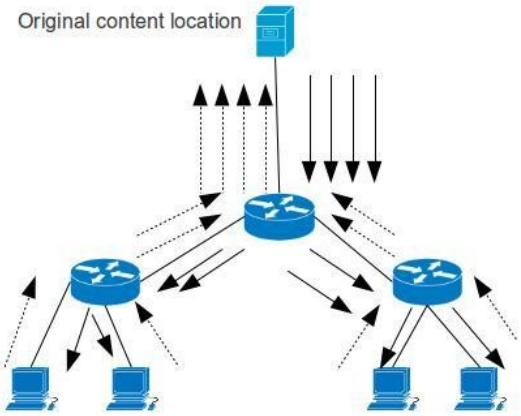

Figure 2: Content addressed by location

1.2 Content Centric Networking

Content Centric Networking (CCN)[3] is a project at Palo Alto Research Center Incorporated (PARC) utilizing content based addressing. It is part of the Named Data Networking[4] project funded as part of the U. S. National Science Foundation's (NSF's) Future Internet Architecture[5] program. CCN is designed to be implemented on top of or alongside TCP/IP, so that it will work with the current networks. Content is made available by repositories announcing the names of content they serve to adjacent routers, who in turn distribute this information into the network via the Interior Gateway Protocol (IGP).

A user who wants some specific content sends out an interest packet[2] on all available interfaces, containing the name of the content, known as a ContentName[2]. A router receiving an interest packet checks if the content is available in its buffer memory, called a ContentStore[2]. If so, then the ContentStore can deliver the content directly to the user. The content matches the request if the ContentName in the interest is a prefix to the ContentName in the ContentStore. If the content is not available, then the router checks if it matches its list of previously sent, unanswered interests stored in a Pending Interest Table (PIT[2]) to ensure that a duplicate interest is **not** forwarded. If the request is in the PIT, then the router adds this new interest to the existing entry in the PIT. If no matches are found in the ContentStore or PIT, then the router checks its CCN Forwarding Information Base (FIB) to see if it knows any other node that might have the content available (longest match lookup is used in cases of multiple matching entries), if so it forwards the interest to them and adds an entry to the PIT with this information. If the router does not have a copy of the content and does not know where to find one, then the interest is simply discarded; the requesting application is itself responsible for re-expressing interest if it is not satisfied (as well as dealing with cases where packets are lost in the network). This process is illustrated in Figure 3.

When the name of the content is matched the node sends the content as an authenticated data packet[2] back on the interface that the interest arrived on. Nodes receiving the data packet should check if it has any entries in its PIT matching the content, if so, then the data packet should also be forwarded to the interface that this interest arrived on. The ContentStore stores a copy of the content if desired. When all of the content has been received, the PIT entry is removed.

Content is transferred backward across the chain of nodes that the interest packet traversed, with each CCN capable node storing either a copy of the content or the location it came from and removing PIT entries until the content is delivered to the application who initiated the request for this content. Any future request for this content can then be served by any of the nodes who have cached a copy in their ContentStore, making the network an efficient content delivery network.

The most suitable cache replacement policies in a content caching network are Least Recently Used (LRU) and Least Frequently Used (LFU). LRU replacement keeps track of the time when the entry was last used and replaces entries that have been in the cache the longest without being requested; thus content that is requested often will stay in the cache. LFU keeps track of the number of times an entry has been requested and replaces entries that have the least amount of requests; thus content that is requested many times will stay in the cache.

This caching of course raises all of the traditional questions of a cache: How long should the entry be cached? How do we know if the cache entry is (still) valid? When should we clear the cache entries? How many cache entries should we maintain? Who can add and delete entries in the cache? Is the cache persistent across reboots? What content should we cache? What content should not be cached? How do we determine what content to cache?

As with peer-to-peer (p2p) networks and applications, there is also the issue of how to deal with freeloaders, nodes that use resources from other nodes but do not share their own resources with other nodes (these resources might be space on a hard drive, network bandwidth, and so on), resulting in a lower total amount of resources being available to nodes in the network (in p2p applications the set of nodes in the p2p network is known as a swarm). Adding countermeasures to discourage freeloaders (such as requirements to offer a minimum cache size, perform content sharing, or similar) could cause more harm than gain by adding extra (unnecessary) complexity to the implementation. Freeloaders could prove to be a non-issue depending on their impact on network performance in well designed networks. Understanding this issue is therefore a prominent subject of further studies.

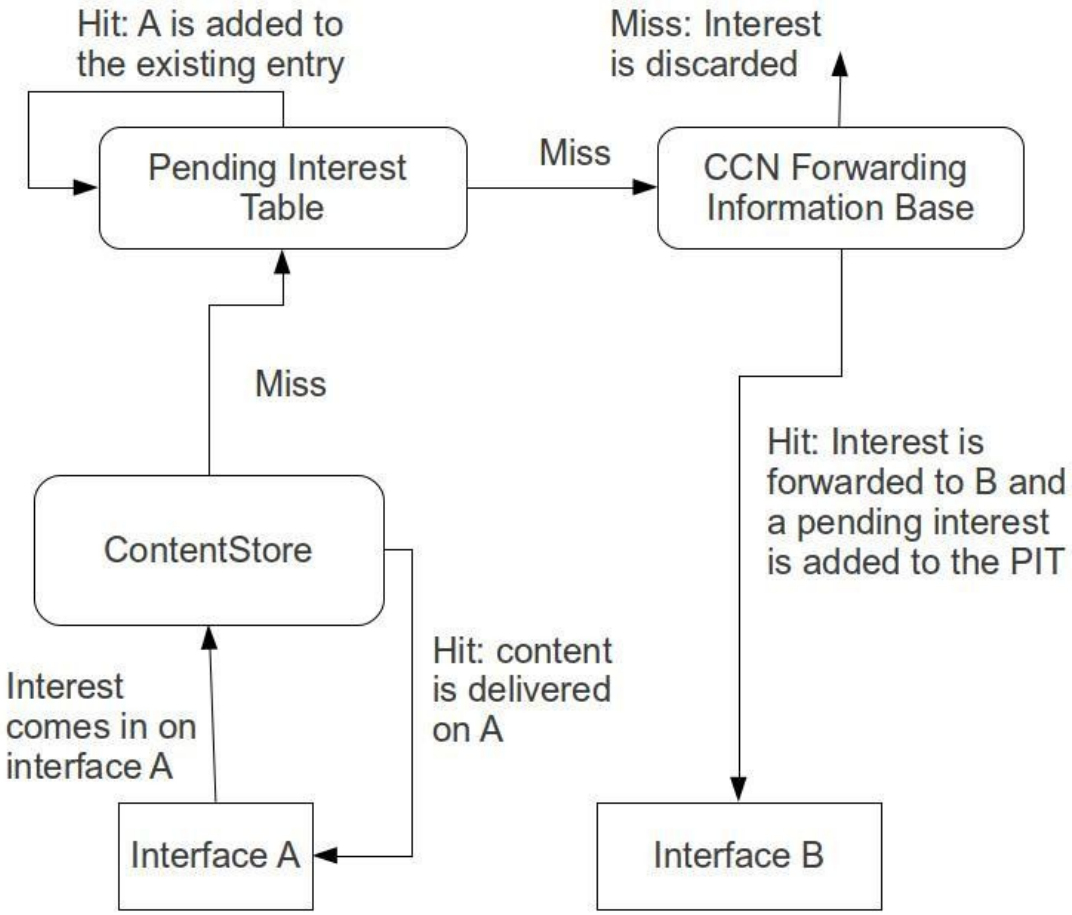

Figure 3: CCN Forwarding Engine Model

1.2.1 CCN Name structure

A name in CCN is always hierarchically constructed in order to allow aggregation of names. A name consists of components. Each component is an arbitrary number of octets that are binary encoded[2]. Names in human readable format use delimiters (for convenience the character / to make the name similar to URIs) between components, but these delimiters are not part of binary encoded names.

A name can be divided into two parts: the first specifying a user or application supplied name and the latter a version and segment of the content. Although the two parts are technically not different, a user will generally only care about the information in the first part (what he/she interprets as the "name"), while the version and segmentation information is useful for applications to identify specific portions of the content. To be able to uniquely identify content with an exact same name, an implicit last component containing a SHA256 digest, a cryptographic hash, of the packet is added, although this is never transferred since it can be computed at any node.

Since the CCN name tree is lexicographically ordered, interest packets can utilize relationships to address content relative to a known entry by using traversal rules, such as *Next* or *RightMostChild* (the *LeftMostChild* traversal rule is used by default). For example , in Figure 4, an interest packet could request ccnx:/Example.com/Articles/Article1/ *RightMostChild* to address the first segment of the second (latest) version of this content (this occurs since the second version is the *rightmostchild,* the first segment according to the default traversal rule *LeftMostChild*). Seg2 could then be retrieved using this information and the *next* traversal rule and so forth. A segment could, for example, have a starting block number or position to playback in a sequence of audio packets. Seg2 could also be computed if the segmentation convention is known by the application. A simple example of this could be: By adding the number of blocks in Seg1 to Seg1's starting block position we know the segment offset for Seg2, assuming the segments are non-overlapping.

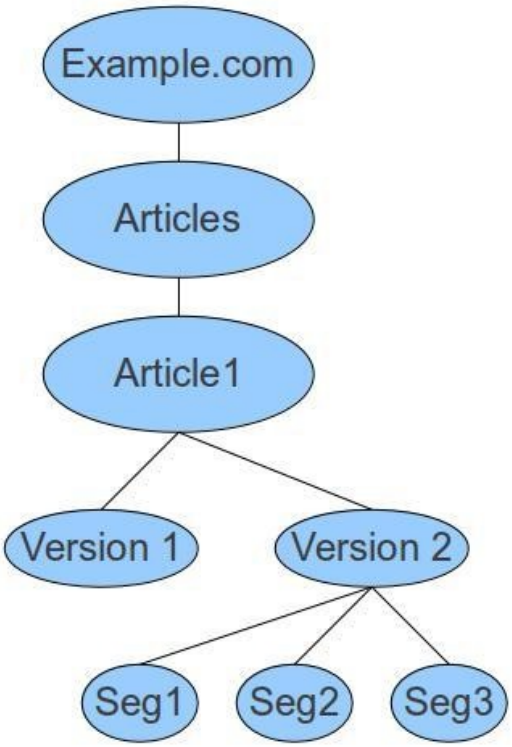

Figure 4: CCN name tree example

1.2.2 CCN and TCP/IP protocol stack

One of the strengths of the TCP/IP protocol stack is the concept that every node runs the Internet Protocol (IP) and this is the only thing that every node *needs* to have in common in order to communicate using these protocols. Protocols above IP are specific to applications and protocols below IP are specific to links. IP handles the global connectivity, by routing and transferring IP packets across links. The operations at the IP layer are invisible to the applications. The idea is that every application uses IP, and IP knows how to utilize every link, the so called "Everything over IP – IP over everything" paradigm.

CCN adopts and simplifies this concept, every node has only the content chunk format in common. As long as all nodes route and forward content chunks, this can be scaled to a global level like IP. As in TCP/IP, protocols above content are specific to applications while protocols below content are specific to links. Links in this concept are not limited to physical links, but rather are simply *connections* to other nodes, as we can utilize any link available to us (including "links" to other local applications). In a sense, the CCN protocol stack can be described similar to TCP/IP as "Everything over content chunks – content chunks over everything". These two stacks are illustrated in Figure 5

One of the major differences between the CCN and TCP/IP protocol stacks is the introduction of the strategy layer in the CCN stack. This layer is described in section 1.2.3.

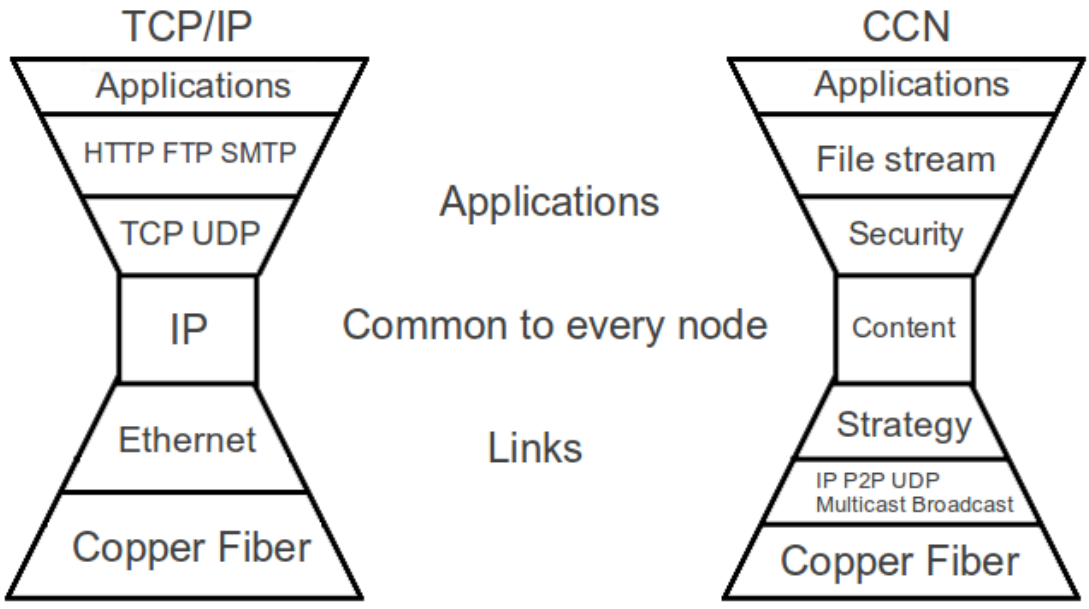

Figure 5: TCP/IP and CCN protocol stack hourglass model comparison

1.2.3 CCN strategy layer

In an IP FIB, entries points to a single destination identified by an IP address and are associated with an interface. In an CCN FIB, entries may point to multiple destinations as there may be multiple sources (or multiple paths to the same source, the important thing is that the content can be found via the associated interface) available. The strategy layer[2] provides strategies on a per-entry basis about how these sources should be queried. This provides fine-grained control over the use of the interfaces and allows CCN to choose the best of these sources based on network conditions or user specified preferences. The strategy layer can react to changes in the available interfaces, determine the best interface, and switch to forwarding packets over this interface if desired.

For example, in a mobile phone we might prefer to use the Wireless LAN (WLAN) interface over the costly cellular network interface. During a voice call, we can use WLAN access-points as they become available when we move around, but use the cell network where no WLAN access is available, without interrupting our voice call.

This network-aware, multi-path transfer ensures implicit resilience and load-balancing between flows in the network. Today resilience and load-balancing can be provided by routing protocols, the transport protocol Stream Control Transmission Protocol (SCTP) and protocols such as the Cisco Hot Standby Router Protocol (HSRP)[6] which enables multiple routers to appear as a single, virtual router to the hosts of the network. Providing resilience and load-balancing at a separate strategy layer has the advantages of making this functionality transparent to higher level protocols, working with any arbitrary higher level protocol without explicit configuration as well as being separately configurable.

1.2.4 CCNx – An implementation of CCN

An early open source implementation of CCN called the CCNx project developed working code in 2009. Since then this code has been further developed. The primary and original contributors of this code were the PARC CCN team. This code implements a CCN overlay network running on top of TCP/IP in order to demonstrate the ideas underlying CCN and to facilitate research in the field. The implementation uses the User Datagram Protocol (UDP) for transfer.

The source code and documentation is available from www.ccnx.org. The code includes the main CCNx daemon, ccnd, which handles node behavior and CCN routing, as well as programs to run a repository, ccnr, on the local machine onto which content can be inserted. The repository is used for persistent storage of Content Objects and can be loaded either as a tree-structure of folders in the local file system or manually file by file. It also includes a control program for ccnd, ccndc, which is used to set up CCN routing. Ccndc can be run in two ways: As a command to add or remove static entries in the CCNx FIB or as a daemon that dynamically creates routes based on DNS Service (SRV) records (DNS SRV records contains the location and port number of servers for a specific service, in this case CCNx nodes). Routes, routing information, and statistics about the cache can be viewed using the ccnd webserver via port 9695 (http://localhost:9695).

Several test programs are also included in the source distribution that utilizes CCNx. The programs that are useful for a simple file transfer are listed in Listing 1.

Listing 1: CCNx test programs useful for file transfer.

- Name: ccnputfile
  Description: Publishes a local file as CCN content in the local repository.
  Syntax: ccnputfile <uri> <file>
  Example: ccnputfile ccnx:/file1 /home/user/file1
- Name: ccngetfile
  Description: Retrieves CCN content and stores it to a local file
  Syntax: ccngetfile <uri> <file>
  Example: ccngetfile ccnx:/file1 /home/user/file1
- Name: ccnls
  Description: Lists all the CCN content available in the local ccnd cache at one level past the given name prefix.
  Syntax: ccnls <prefix>
  Example: Following the example in Figure 4 on page 5, ccnls ccnx:/Example.com/Articles/ would yield Article1.
  Alternatives: The program ccnlsrepo does the same thing and has the same syntax as ccnls, except it lists content available in the local repository instead of the ccnd cache.
- Name: ccndc
  Description: Controls the CCN FIB in the ccnd. Further information is given in section 1.2.4.
  Syntax: ccndc <add/del> <uri> <udp/tcp> <host>
  Example: ccndc add ccnx:/ udp 192.168.1.5

2. Background

This chapter presents proposed architectures with similar design goals as CCN and their fundamental differences in order to illustrate different approaches to content-networking. The chapter also presents the request-routing mechanisms used by CDN networks as these networks currently provide one of the major ways of efficiently distributing content that is desired by large numbers of users. The chapter also presents work done on content-centric networks related to the specific inter-ISP problem that this thesis project focuses on.

2.1 Content Distribution Networks

The Content Distribution Network (CDN) approach to content delivery is based on CDN providers placing clusters of servers in the ISP's network close to the users who will request this content. These servers replicate selected parts of the CDN's customer's website so that static content can be delivered by the CDN operator, for example images, script files, videos, files, and so on. These static parts of the websites are often much larger in size than the dynamic parts, which are usually text.

When web servers are clustered together, they are geographically in the same place, hence a user in Europe visiting a website in North America will experience the one-way delay as the content is transferred between the continents, as well as potentially experiencing lower transfer speed due to bottlenecks in all the networks along the path from the website to the user. Additionally, the delay in the reverse path may also reduce the transfer rate, due to the flow control mechanisms that TCP implements.

When users browse a CDN assisted web page only a small subset of the web page actually comes from the website itself, the rest is delivered by the CDN provider who redirects the user to their servers instead. If these servers can be located close to the user, then the content can be delivered with lower delay and at a higher transfer speed. The methods for redirecting requests for a web page to nearby servers is described in section 2.2. Due to the server's location inside the ISP's network, transferring the content from these CDN servers also reduces the load on inter-ISP links. [7]

One of the major downsides of the CDN approach is that each of the CDN providers must have their own network of servers inside the ISP's network infrastructure. Furthermore, new CDN providers will have to place servers all around the world before they can begin to offer services to websites, requiring a major investment. This creates a closed market where only a small number of major companies can compete.

2.2 Content Request mechanisms

Directing content requests to the most suitable (hopefully local) cache is one of the major mechanisms of CDN networks. RFC 3568 “Known Content Network Request-Routing Mechanisms”[8] specifies several content network request-routing mechanisms, specifically DNS request-routing, transport-layer request-routing, and application-level request-routing. Each of these will be described in more detail in subsections below.

2.2.1 DNS request-routing

DNS request-routing is very common due to the flexibility of the DNS system. In this method a DNS server returns different records based on the location of the user, hence redirecting users to caches close to them. “Close” is frequently computed in terms of Time To Live (TTL) values in the DNS request packets. This method can also be used to balance load over a number of servers and therefore provides both load balancing and server redundancy. One of the downsides of this approach is the recursive nature of the DNS system. In order to achieve the desired spread of requests recursion must be disabled and the time the answer can be cached must be short, otherwise groups of users (of uncontrollable size) may be directed to the same server. This of course increases the load on the DNS server, which must answer all DNS queries itself. While this can be facilitated by using a more powerful DNS server, disabling recursion and limiting caching defeats[9] the original intentions of scalability and performance in the domain name system.

2.2.2 Transport-layer request-routing

Transport-layer request-routing provides more parameters upon which the decision to which server the user is directed can be made. Every user's first request is directed to the same server via the DNS record, which in turn determines a suitable server based on the client's IP address and port. This balancing is one-way only, the requests for content are sent to the original server which directs these requests to nearby servers which deliver the content to the user. This approach could be used in combination with DNS request-routing for faster load balancing as DNS entries persist for some time in the hosts and the spread of requests across content-servers can change quite often and very quickly.

2.2.3 Application-level request-routing

Application-level request-routing examines the application-layer header and is similar to transport-layer request-routing in the sense that the original request always go toward the same server. This server, or special nodes along the network path, replaces embedded references to objects in content being delivered to point to suitable alternative servers.

The redirection decision is made based on either a specific URL or, in the case of websites the decision can be made based on HTTP header fields such as language, user agent, or cookies. Because the decision can be made on a per-object basis different choices can be made for different objects, allowing different servers to handle different portions of objects or different languages. One of the downsides of this is that webpages that have been altered should not be cached, as entries in the cache may point to servers that are no longer available or are no longer the preferred choice. This means that the node that alters the webpage must also add the header line <META HTTP-EQUIV=”pragma” CONTENT=”no-cache”> to the HTML header of the webpage. This tells the user's web-browser to not cache the page, but rather always get the page from the source.

2.3 Breadcrumbs

In content caching networks, a breadcrumb is a small cache entry that records the (most recent) origin and destination of content that passes through the node instead of caching a copy of the content. The advantages of breadcrumbs is described by E. J. Rosensweig and J. Kurose in ther paper “Breadcrumbs: efficient, best-effort content location in cache networks”[10]. These authors also present a way to search for content using their method called Best-Effort Content Search (BECONS). This method is described as a best-effort attempt to locate copies in the caches of the network (where it may or may not be found). However, if a search fails the content can always be found at (or near) the source.

As content traverses the network, it leaves a *trail* of breadcrumbs from the source to the requesting host(s). New requests for this content can try to locate this trail as the request is routed towards the source. The request follows this trail in either direction (upstream towards the source means a guaranteed hit, but downstream might be faster) until it finds a node with a cached copy of the content.

2.4 Data-Oriented Network Architecture

The Data-Oriented Network Architecture (DONA) described by Koponen, et al. [11] aims to improve content delivery by using flat names instead of domain names. These flat names consists of a cryptographic hash of the publisher's public key and a label chosen by the publisher. These names are therefore not human-readable (although the label could be), but can be bound by the user to a name of their own choosing. The idea is that these bindings should be made via webpages, social media sites, or contacts in a contact list. While the content itself is authenticated, the binding between a user's name for an object and the underlying flat name for an object is not. Because of this, an attacker could perform a substitution attack by returning malicious couplings, thereby substituting the attacker's malicious content for the authentic content.

This architecture relies on resolution handlers (RH) that implement an anycast primitive among themselves. These RHs keep track of the location of copies of content. Requests for content are routed among these RHs until a copy is found (for example, in a CDN server). The RH then returns the location of the copy to the user who retrieves the content via traditional IP transfer.

DONA exploits the location-independence of flat names, but does not in itself provide storage of content. Content must be published somewhere (a website or in a CDN) before it can be registered with an RH. Subsequently the RH can return a server's address in response to a user's content request(s). In this sense, the DONA approach is essentially a CDN content locator mechanism rather than a content caching network.

2.5 Related work

In “Active ISP Involvement in Content-Centric Future Internet“[12], Islam and Grégoire discuss ways of evolving current CDN networks in collaboration with ISPs to create a content-centric delivery system using surrogates to accelerate the adoption of content-centric architectures. In their proposed architecture users are directed to surrogates, located close to the user at the edge of the network via a webinterface which is capable of displaying any HTML5 supported content (such as video or audio). The surrogate fetches the requested content directly from content providers or from other surrogates using a content-centric network connected via a Virtual Private Network (VPN). The surrogate places the content in its local cache and makes the content available via a webserver. Thus CDN providers can implement efficient content-delivery to their surrogates while this processing is transparent to the user, who only interacts with a webserver using the standard HTTP based webarchitecture. The advantage of this approach is that it allows the ISP's backbone network to adopt a content-centric network architecture while this change is transparent to the ISP's customers as they do not have to update either their hardware or software.

In “Performance Measurement of Name-Centric Content Distribution Methods” [13], Yuan and Crowley measure the performance of CCNx and compare it with HTTP-caching when downloading files. They conclude that the implementation of CCNx (specifically the version released March 8, 2011) lacks optimization. However, this paper was written over a year ago and during this time CCNx has been under constant development. Thus we will be interested to see if this development has changed the performance of the latest CCNx implementation (released on April 22, 2012).

The original CCN paper “Networking Named Content” [2] measure how CCNx performs versus TCP when distributing a file over a bottlenecked network. This paper concludes that the total download time for a single file increases linearly with the number of simultaneously downloading clients in the case of TCP, while the time required remains constant *independent* of the number of simultaneously downloading clients when using CCNx.

In “Flow-aware traffic control for a content-centric network” [14], Oueslati, Roberts, and Sbihi proposes a traffic control framework based on per-flow fair bandwidth sharing. They conclude that a TCP-like (additive increase, multiplicative decrease) congestion control mechanism implemented as a maximum window of pending interests is sufficiently effective to make up for the lack of CCN specific congestion control. The current traffic control in CCN (one interest packet returns one data packet) might be sufficient in the small-scale testbeds we are using, however this might need additional mechanisms when tests (or operations) are conducted on a larger scale.

In “Routing policies in named data networking”[15], DiBenedetto, Papadopoulos, and Massey explore the changes in inter-domain routing from a economic perspective when comparing border gateway protocol (BGP) policies to policies in future content-centric architectures. While economics is not a focus of this thesis project this paper is interesting because the authors discussion how policies that are not possible (or desired) today may change how an ISP considers transit traffic. In section 4.2 of this paper these authors present a scenario where two ISPs similar in size and volume of traffic collaborate by sharing their cached content between each other over a link with limited connectivity (i.e., they can only reach each other through this link) in order to achieve a lower operating cost than if they were to get the content from other providers. Because of this sharing the two ISPs increase the amount of content each ISP has (virtually) cached, while reducing the amount of money they have to pay their peers. This sharing may or may not reduce the speed with which they can deliver this content, but it can reduce the time before the first packets of the content are delivered.

3. Implementation, Testing, and Analysis

In this chapter we introduce the topologies and setup of the network testbed used to demonstrate CCNx in an environment with multiple ISPs. We describe the four tests that we performed, what we are measuring, how we accomplish this, and what we want to achieve in each test. Finally, we described the analysis of the data resulting from these tests.

3.1 Topology of a single network testbed

The testbed topology consists of three ISPs: ISP1, ISP2, and ISP3. Each ISP has one router through which they are connected to each other by a single (direct) inter-ISP link. ISP1 has 4 hosts connected to the router of ISP1. The content publisher is simulated by inserting our content at the router of ISP2. Network links inside the ISPs are 1 Gb/s links while the bandwidth of the inter-ISP links is 10 Mb/s. The testbed has two different configurations, shown in Figure 6 and Figure 7. The main purpose of Topology 1 is to evaluate *caching* of content and content transfer with *multiple* destinations. The main purpose of Topology 2 is to evaluate content transfer when there are *multiple* alternative sources for the content.

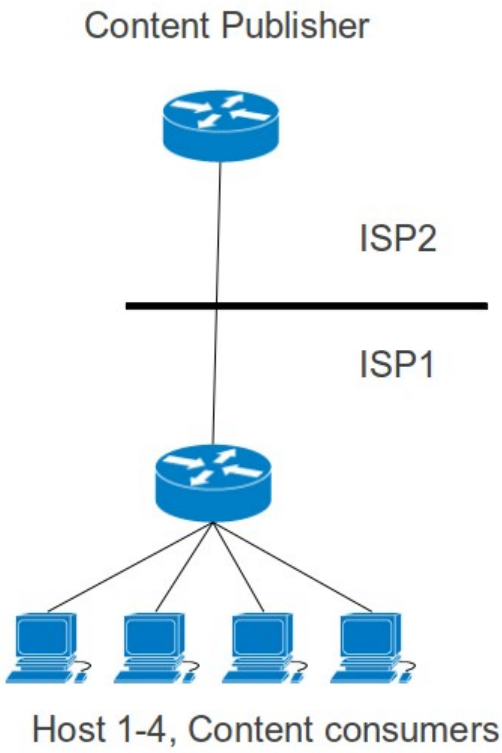

Figure 7: Testbed Topology 1

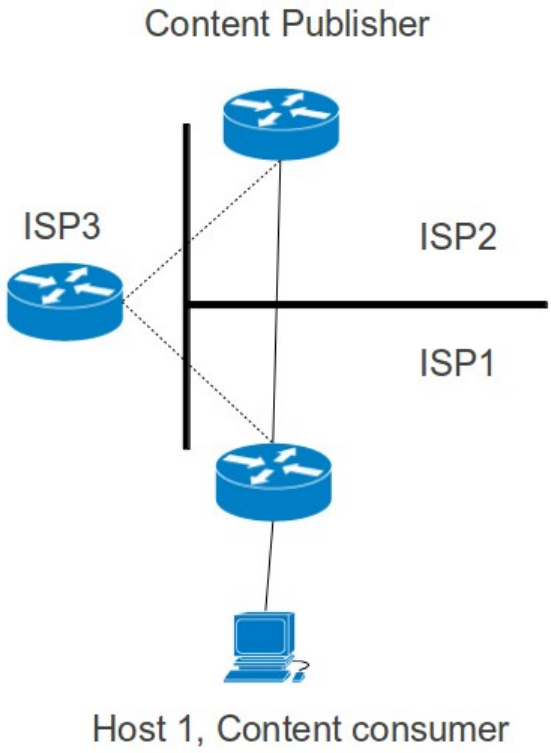

Figure 6: Testbed Topology 2

3.2 Tests and measurements

We will be performing four tests: test A, test B, test C, and test D. For test A we recreate the measurements performed by Jacobson, et al, as described in section 6.2 of the paper "Networking named content" [2]. This test measures the performance when multiple hosts simultaneously retrieve the same file from a server over a bottlenecked link when the transfer is done using both CCN and TCP. The test measures the amount of time until the last host has successfully downloaded the same 6MB file over a 1 Gb/s network with a 10 Mb/s bottleneck link to the content publisher. This test utilizes the program cURL[16] to connect to an HTTP server in order to transfer the file. For CCN this content was pre-staged into the CCN daemon by requesting the file locally so that we only measure communication costs, since the actual CCN content is normally dynamically generated upon request.

For test B we utilize the same topology to transfer a larger 50MB file to one of the hosts. The transfer is done in the same way as in test A. We also perform a single HTTP transfer of the same file to the same host, in order to compare TCP and CCN measurements. This test measures the amount of data and packets transferred on the inter-ISP link in order to evaluate what impact content based addressing *could* have on the traffic between ISPs.

For test C and test D we add another ISP, ISP 3, to evaluate the multi-path capability of CCN. The content is prerequested at ISP3, thus ISP1 has two places to retrieve the content from via identical links. As in test B, we transfer a 50MB file simultaneously to the hosts. During test C we also examine the behavior of CCNx when a HTTP transfer of a 6MB file is started from ISP2 to ISP1. The transfer is started in the middle of the test and completed during the test. The goal of this test is to see if we can utilize the link with the greatest available bandwidth, in order to achieve load balancing between alternative sources of content.

During test D we are simulate link failure over the inter-ISP links by disconnecting the network cable, waiting a few seconds seconds and then reconnecting the network cable. We do this first on the link between ISP2 and ISP1 and later on the link between ISP1 and ISP3. The goal of this test is to achieve an uninterrupted transfer in the event of a link failure when there are alternative sources for the content.

3.3 Testbed implementation and routing

As CCNx is an overlay implementation we are using workstations with multiple network interfaces as routers; thus making them act as CCNx capable routers by running the CCNx daemon.

The hosts are implemented in VMware Workstation 8.0 [17] as virtual machines running the Linux based operating system Ubuntu 12.04 LTS[18].

CCN routing is achieved using static routes in order to route the whole CCN namespace (ccnx:/) to the directly connected nodes; thus the hosts in ISP1 can find all content via adjacent hosts or the ISP1 router and the ISPs routers can find content via each other. The prefix ccnx:/ matches all content (as the shortest match), and is therefore similar in CCNx to the IP default route.

IPv4 routing is achieved using static routes and by enabling IPv4 forwarding of packets on the workstations acting as routers. This forwarding is **not** needed by CCNx to forward packets, but it is needed to forward the HTTP requests and responses.

3.4 Tools

In order to perform these measurements we use the packet analyzer Wireshark[19] to capture packets during our tests. Wireshark can store the captured packets in the well-known ".pcap" format for later use in other packet visualizers, although we will use the built in tools in Wireshark to generate graphs of our results. To generate these graphs we use filters to select the desired traffic based on specific parameters. For CCNx traffic we use the IPv4 source and destination addresses together with UDP port 9695, as this is the port that CCNx uses to transfer interest and data packets.

For the HTTP transfers in these tests we are using the well-known Apache[20] HTTP server implementation and GNU Wget[21]. Wget is a command line HTTP, HTTPS, and FTP client included in many Linux distributions.

In order to create files of specific size we utilize the Unix program "dd", which copies raw data from a file to another file. The special Linux-file /dev/zero, which returns null characters when read, is used as input file to create a file of specific size containing only zeros. In retrospect, it might have been better to use the Linux-file /dev/random to generate random bits instead of the zeroes generated by /dev/zero.

3.5 Test A – Content transfer with multiple receivers

For this test we implement Topology 1 (see Figure 7 on page 13) on a single workstation running all the virtual machines in Topology 1. A 6 MB file is created in the root directory of the webserver using the command "dd if=/dev/zero of=/var/www/file1.dat bs=1M count=6". This file is loaded (four times, incrementing the number in the name of the content) into the ccnd using the command "ccnputfile ccnx:/file1.dat /var/www/file1.dat" to ensure that the same file is used in both the CCNx and HTTP transfers.

The goal of this test is to illustrate the advantages of the content caching and multiple destination transfer in CCNx in an inter-ISP scenario where the inter-ISP link is a bottleneck for the transfers.

3.5.1 Method

The hosts retrieves the file over HTTP using the command "wget http://192.168.2.2/file1.dat" and over CCNx using the command "ccngetfile ccnx:/file1.dat". Wireshark running on the router of ISP1 captures the packets of the transfers and the transfer time is determined based on this capture.

The transfer time for the HTTP transfers is based on the duration of the flow of TCP packets from ISP2 to ISP1, and the transfer time for the CCNx transfer is based on the duration of the flow of UDP packets with destination port 9695 from ISP2 to ISP1. We conclude that once the content is received at ISP1 it is delivered with nominal delay over the gigabit links from the cache (either by the router of ISP1 or the other hosts).

This test is performed in 4 stages, with one, two, three, and four hosts in ISP1 requesting content. The resulting transfer times are displayed in Figure 8. Table 1 on page 17 contains the transfer times (rounded to a precision of 0.5 seconds) for both of the protocols for each of these four stages. The test was performed once.

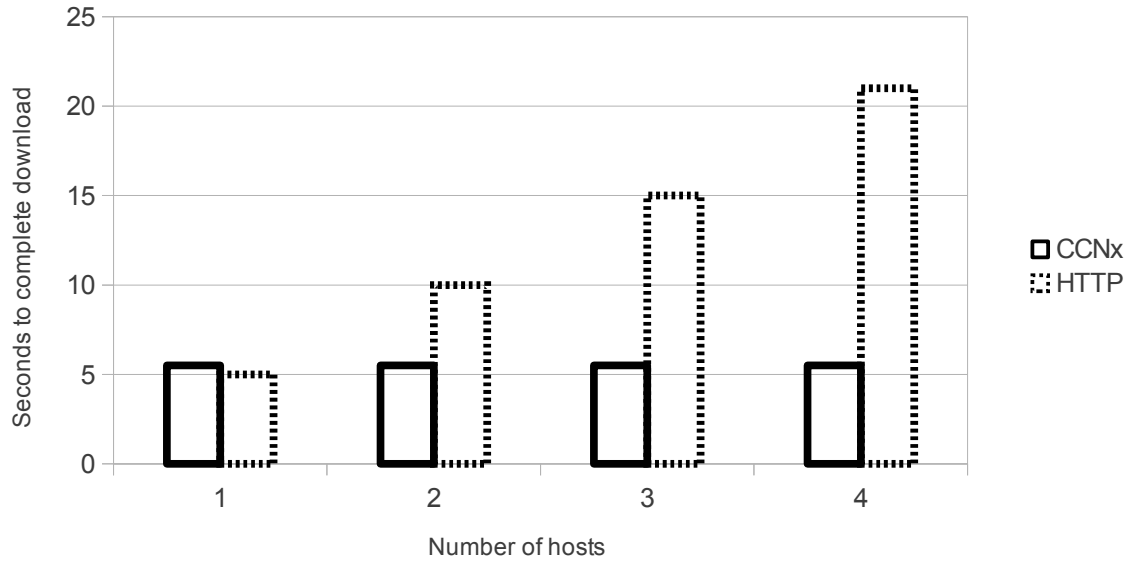

Figure 8: Test A: Time for hosts to download a 6MB file using CCNx and HTTP

Table 1: Time for hosts to download a 6 MB file using CCNx and HTTP

Number of hosts	CCNx	HTTP
1	5.5s	5.0s
2	5.5s	10.0s
3	5.5s	15.0s
4	5.5s	20.5s

3.5.2 Analysis

As seen in Figure 8, the caching capability of the nodes in the network causes the content to be transferred only *once* over the bottleneck link between ISP1 and ISP2 when the content is transferred using CCNx. In transfers using HTTP on the other hand, each host has to get the content from the source at ISP2, i.e. the content crosses the bottleneck link multiple times.

Because of this, the transfer time and the load on the inter-ISP link increases roughly linearly (the transfer time f(x) can be described as f(x) = 5x where x is the number of hosts, i.e. the transfer time increases by a constant 5 seconds for each additional requesting host) with the number of simultaneously downloading hosts when using HTTP; while the transfer time and the load on the inter-ISP link is constant regardless of the number of simultaneously downloading hosts when using CCNx.

As long as a copy of this content remains cached (i.e. due to being requested frequently or many times) in the ISP's network the cached copy can be used to serve future requests for this content, without additional costs of transfers over the inter-ISP connection(s). These transfers are also potentially faster as the source is (local) closer to the user. As soon as there is more than one request for this content (if the content is still in the ISP's cache) there is a significant gain to both the ISP and the ISP's customers by using CCNx over HTTP for content delivery.

In a scenario where there is a CDN provider present in the ISP's network capable of serving requests for this content, HTTP gains some of the content caching advantages. The content would only cross the inter-ISP link once (from the content provider to the CDN server(s)) and transfers could also potentially be faster due to the local source. The disadvantages of this scenario is that it requires explicit configuration and only improves the content delivery of the CDN provider's customer's content (websites), whereas a CCNx solution would not require configuration other than network setup (allowing for easier expansion of the network's cache capacity by simple adding storage to existing nodes or adding new nodes to the network) and could handle any arbitrary content.

3.6 Test B – Comparison of CCNx and HTTP transfer

For this test we implement Topology 2 in a virtual environment on a single workstation running all the virtual machines from Topology 2 (though ISP3 is not used in this particular setup). A 50 MB file is created in the root directory of the webserver and loaded as CCNx content into the ccnd of the router of ISP2.

The goal of this test is to determine the transport efficiency of CCNx compared to HTTP when there is only a single request for some particular content. This setup is the worst case scenario for a CCNx transfer as the content must be retrieved from the original source.

3.6.1 Method

The content was transferred via both CCNx and HTTP from the router of ISP2 to the host in the same way as in Test A. The test was run only once. Packets were captured by the router of ISP1. The results are displayed in Figure 9 and Figure 10. Table 2 on page 19 shows the statistics of these two transfers, where Transferred bytes is calculated as transfer time * bandwidth, IP packet size as transferred bytes / number of IP packets, user data per packet as filesize / number of IP packets, and overhead per packet as packet size – user data size.

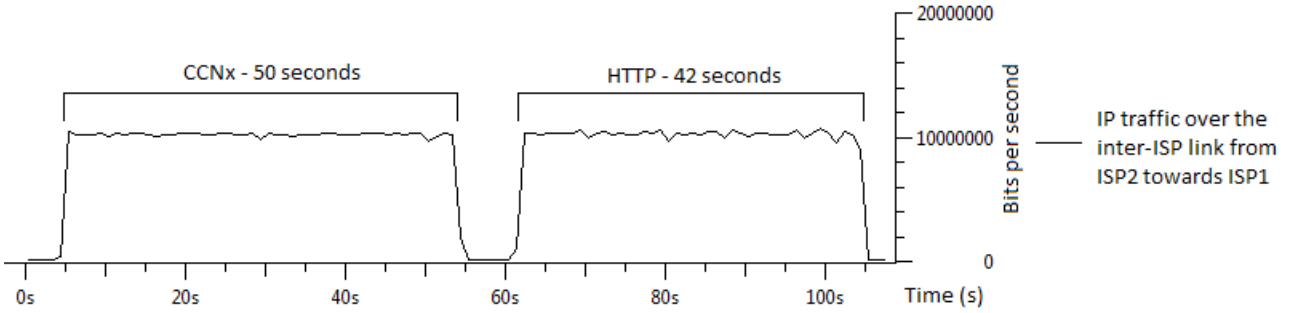

Figure 9: Test B: Bandwidth in bits per second of inter-ISP traffic from ISP2 to ISP1 during the transfer of a single 50 MB file using CCNx and HTTP

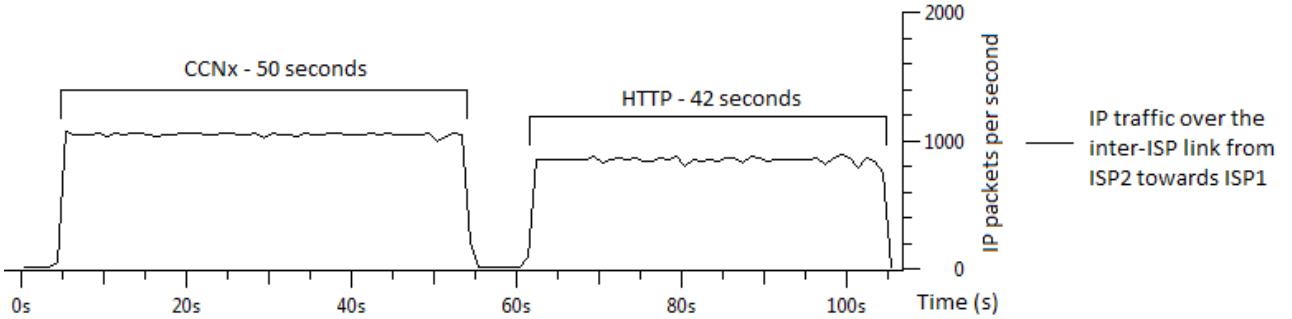

Figure 10: Test B: Number of IP packets per second of inter-ISP traffic from ISP2 to ISP1 during the transfer of a single 50 MB file using CCNx and HTTP

Table 2: Statistics of inter-ISP traffic from ISP2 to ISP1 during the transfer of a single 50 MB file using CCNx and HTTP

	Transfer time (s)	IP packets per second	Number of IP packets	Transferred bytes	IP packet size (byte)	User data (byte) per packet	Overhead (byte) per packet
CCNx	50	988	49391	$62.5*10^6$	1265	1012	253
HTTP	42	822	34543	$52.5*10^6$	1520	1447	73

3.6.2 Analysis

As seen in Figure 9, CCNx utilizes the available bandwidth on the link just as well as TCP which is famous for "filling the pipe". The time to transfer a 50 MB file over a 10 mbit/s link is 50 seconds using CCNx and 42 seconds using HTTP. As both transfers fully utilize the available bandwidth, we conclude that the transfer efficiency of a single-destination transfer using CCNx is 84% (42 / 50) of that of HTTP. As seen in Figure 10, the number of IP packets sent per second for the CCNx transfer is about 988, while this value is about 822 for the HTTP transfer. This is due to the smaller packet size seen in the CCNx transfer compared to the HTTP transfer. However, as seen in Test A, once there is more than one request for some particular content there is a gain in using CCNx over HTTP.

The overlay implementation of CCNx is a disadvantage in this because of the additional IP and UDP headers in addition to the CCNx header leads to a higher overhead than HTTP, as the later only has the overhead of IP and TCP headers. The CCNx overhead is calculated to be 20% (overhead per packet / packet size, 253/1265) while the HTTP overhead is 4.8% (73/1520). The roughly 16% penalty in using CCNx rather than HTTP is due to the larger overhead in the CCNx overlay implementation which in turn also increases the number of packets needed to transfer the same amount of content compared to the HTTP transfer.

3.7 Test C – Multiple transfers

For this test we use the same setup as in Test B, a virtual implementation of Topology 2 running on a single workstation.

Method

A 50 MB file is created (in the same way as in Test A) in the root directory of the webserver on the router of ISP2 and loaded as content into the ccnd. The content is transferred to the router of ISP3 using CCNx *before* the test. As a result ISP1 now has two alternative sources for this content over two identical links.

The file is transferred to the host using CCNx. All times referred to are approximate to a precision of 0.5. After 15 seconds, a HTTP transfer of a 6 MB file is initiated from ISP2 to ISP1. The HTTP transfer finishes in 5.5 seconds, about the same time as in Test A. During this time the CCNx content is transferred over the (other) link to ISP3; thus allowing the CCNx transfer to finish in 50 seconds, the same time as in test B. Details of the various transfers are shown in Figure 11. The statistics of these transfers are shown in Table 3.

This test is further analyzed together with Test D in section 3.9.

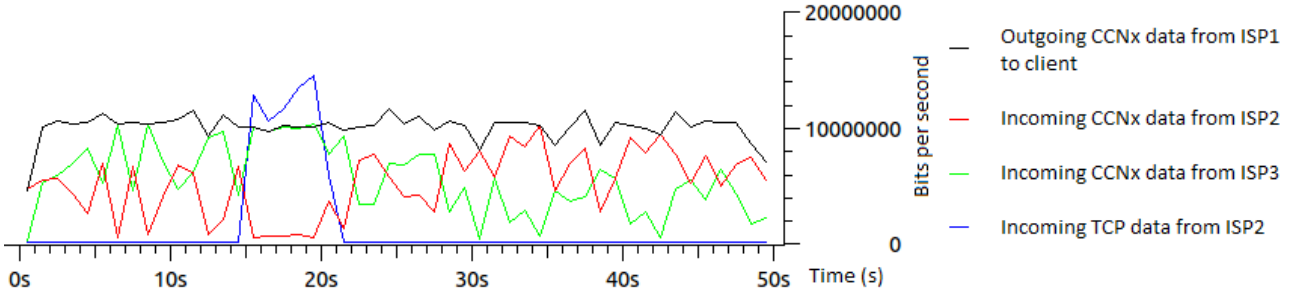

Figure 11: Test C: CCNx transfer from two alternative sources with a HTTP transfer occurring over one of the links during the test.

Table 3: CCNx transfer from two alternative sources with a HTTP transfer occurring over one of the links during the test.

	Number of packets	Percentage of content traffic
Link to ISP2	20016	37.2 %
Link to ISP3	33813	62.8 %

3.8 Test D - Resilience

For this test we implement Topology 2 on an identical, physical topology. The routers are each implemented as virtual machines on a workstation of their own and the physical network interfaces of the workstation are bridged together with corresponding network interfaces on the virtual machine. The host is implemented on a laptop running the operating system Ubuntu 11.04 and the CCNx implementation.

Method

For this test we used the same setup as in Test C, a 50 MB file was created (in the same way as in Test A) in the root directory of the webserver on the router of ISP2, this file was also loaded as content into the ccnd. The content is transferred to the router of ISP3 using CCNx *before* the test. As a result ISP1 now has two alternative sources for this content over two identical links.

The file is transferred to the host using CCNx. All times referred to are approximate. After 10 seconds the link to ISP2 is disconnected and reconnected after 10 seconds. Then 10 seconds after the first link failure, the link to ISP3 is disconnected and reconnected after 8 seconds. The results are displayed in Figure 12. In this figure, the red and green graphs shows the distribution of traffic between the two available sources (links), and the black graph shows the traffic towards the host from the router of ISP1. As there is always at least one available source throughout the test, the transfer was not interrupted by the link failures. Although we can see in Figure 12 that there is a dip in the content traffic just following the link being disconnected. There is a possibility that these dips in content traffic were caused by overloading the Central Processing Unit (CPU) of the virtual machines running as routers as we noticed the CCNx implementation occasionally maxed out the CPU capacity of their guest as well as host operating system.

Note also that there is a tradeoff in the traffic between the incoming link from ISP2 and ISP3 before the link is disconnected to ISP2. Similarly there is a tradeoff in the traffic between the incoming link from ISP3 and ISP2 before the link to ISP3 is disconnected. This tradeoff in the distribution between the link to ISP2 and the link to ISP3 is **not equal** and **will change** if the test is reset and performed again, as this is a result of the strategy layers continuously making decisions during the transfer.

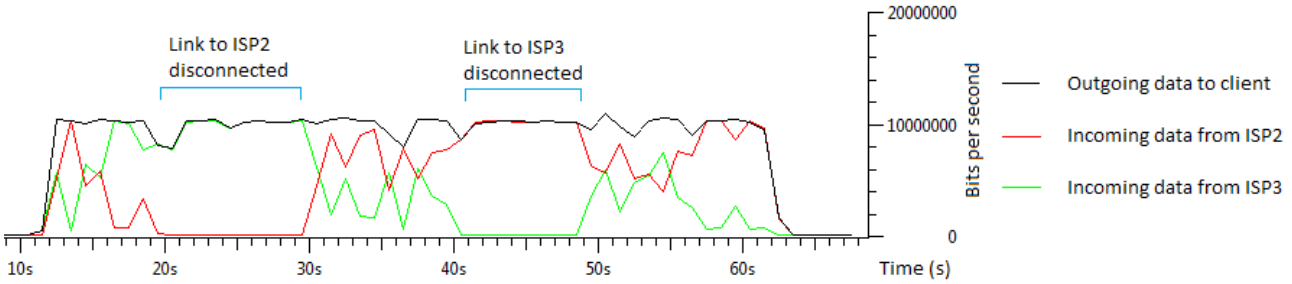

Figure 12: Test D: CCNx transfer from two alternative sources with a (non-overlapping) link failure occurring once for both links

3.9 Analysis of Test C and Test D

These two tests show the advantages of having multiple available sources for content. The inter-ISP links available to ISP1 in the tests are equal (identical in bandwidth), and no other traffic is flowing through the testbed network; therefore the strategy layer at router of ISP1 could potentially choose either of the two links towards the peering ISPs as the best source. Note that it would seem more beneficial to use both links simultaneously (increasing the total transfer rate to the user), however I have not found a way of doing so.

When we look at test C we see that the strategy layer adapts to the HTTP transfer over the link between ISP1 and ISP2 by choosing the other link to ISP3 as the best link. The result is that the performance of neither transfer is affected and both transfers finish in their expected times.

Looking at test D we see that the strategy layer adapts to the link failures by choosing the available source as the best source. The switches between sources are done with minimal delay, as reactions in the strategy layer are triggered by events such as changes to the interfaces' state, this change in source does not affect the performance of the transfer in the test which finish in roughly the expected times. Similarly, the failover time for SCTP as measured by J. Eklund and A. Brunstrom[22] is between 422ms and 874ms depending on the maximum retransmit threshold.

Also interesting is that no interruptions occur which might affect higher level protocols or cause issues that would be visible to the user (i.e. no aborted file transfers, no interrupted or delayed phone calls, and so on). This resilience at the network layer can be provided by the routing protocols in the TCP/IP model, but is associated with some additional delay in adapting as the changes to the routing tables have to propagate throughout the network. If this delay is too large it might cause higher level protocols to attempt to recover (or break the the connection entirely if a recovery is not possible), causing delays or interruptions in the ISP's customer's service.

3.10 Cache size, peer traffic, and reliability

Imagine a scenario with 5 ISPs interconnected in a full mesh, i.e. each ISP directly connected to the other ISPs through a single link. Users in one ISP (the producer) produce over time a lot of content that the users of the other ISPs are interested in. The cache size of the ISPs directly affect their peer traffic as well as the reliability of the peer traffic as described in the following paragraphs.

With a large cache the ISPs will minimize their peer traffic because requested popular content will not be removed from the cache, due to multiple alternative sources for content at the other ISPs as users from different ISPs might request specific content at different times there is an improvement in reliability.

With a small cache the ISPs will frequently remove moderately requested content from their cache, which results in additional peer traffic from new (repeated) requests which potentially also reduces transfer speed and increases the delay as experienced by the customer, as compared to a local source. The content is also less likely to be found at the other ISPs, hence lowering the number of possible alternative sources for content and as a result reducing reliability.

A large cache might naturally seem to be the better solution, but as the amount of content grows the subset of content that the ISP can cache decreases. On another note, this is not the purpose of a cache. The cache of an ISP is most efficient when it keeps the content currently most requested by the ISP's customers (i.e., the currently most popular content) cached. How large an efficient cache is, and how large the subset of popular content is, might have to be analyzed and considered by each ISP as it might differ depending on the ISP's size, number of customers, and so on.

Some evaluation of the benefits of caching or lack thereof can be seen in the recent Master's thesis by Manxing Du [23].

This leads to a trade off between the cost for storage for the cache and the cost for downstream peer traffic. Having a large cache might be a costly investment for the ISP, but could potentially increase the quality of the service they offer to their customers if they are able to serve a larger portion of their content requests from this cache. Servicing a larger number of requests from this cache might also enable the ISP able to offer their customers effectively greater bandwidth while avoiding increasing the ISP's inter-ISP traffic.

ISPs that focuses on hosting or co-location services could benefit more from a large cache by offering to cache all of their customer's content as a service so that a minimal load is put on the customer's servers. By guaranteeing that the customer's content remains cached the customer's cost for hosting might shift from server or bandwidth capacity to cache space within the ISP's network, potentially lowering the customer's costs for hosting while maintaining their capacity to serve their customer's requests.

4. Conclusion

The demand for higher Internet speeds and greater Internet penetration rates leads to increasing costs for inter-ISP traffic for ISPs. By basing addressing of content on names rather than the content's location, content becomes location independent.

This project has shown that such addressing can reduce an ISP's inter-ISP traffic by caching the content, enabling new requests for this content to be served from a local cache without additional costs for inter-ISP traffic, as compared to addressing by location which causes inter-ISP traffic to increase linearly with repeated requests.

We have shown that even though CCNx transfers of new content suffers some performance loss compared to a TCP transfer because of the overhead from additional headers, there is a potential gain to the ISP for caching this content if there are additional requests for this content before it is removed from the cache.

We have shown the benefits of transfers of named content when multiple alternative sources are available, decreasing the load on the original source and increasing resilience against link failures by switching between sources without interruptions or major delay. CCNx also provides load-balancing by using the currently best available source for the content. These advantages provide better quality of service to the ISP's customers as well as better supporting mission-critical services such as phone calls.

4.1 Future work

A suggestion for further work in this area would be to further investigate the CCNx strategy layer in order to exploit multiple sources simultaneously in order to potentially increase transfer rates of inter-ISP transfers. It might also be interesting to look at strategies for cache policies and size, as well as ways of financing the cache for the ISP.

4.2 Required reflections

The work done in this thesis project investigates how changing the underlying infrastructure of our ISPs' networks to accommodate for the changes in how we use the Internet today compared to what the TCP/IP protocol stack was originally designed for. The quality and availability of Internet service is an important part of todays society and the ever increasing demand. The changes investigated in this thesis project could effectively reduce the load on the backbone Internet infrastructure overall, allowing for higher available bandwidth to more users.

One of the strengths of the Internet is that anyone, even someone who knows nothing about computers, can browse the web and access content on the Internet. This new way of addressing content follows the URI format that users are used to, because of this the user will not be confused or require additional education following this change. In fact, the user might not even be aware of these changes. While these new protocols can be automatically installed into the user's operating system with little to no interaction from the user via updates, something that could prove to be a problem is the user's home router. Updating router firmware to support these new protocols is possible, but is often a rather complicated process for an inexperienced user. This could mean an additional expense to the users who might have to buy a new home router.

The additional storage required in an ISP's infrastructure in order to provide a cache of suitable size is currently already present in their networks in the form of CDN operator's servers. Since the CDN operator's business model of efficient content delivery is solved elsewhere it is natural for them to adapt to reformulate their business model around the new problem of storage space. Because of this, the transition to content based addressing does not require major investments from the ISPs and CDN operators who adapt will not be going out of business.

References

[1] "Q3 2011 State of the Internet Webinar, February 14, 2012" Akamai. Available: http://www.akamai.com/dl/akamai/soti_slides_q3_2011.pdf [Accessed: 04-Apr-2012].

[2] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *Proceedings of the 5th international conference on Emerging networking experiments and technologies*, 2009, pp. 1–12.

[3] "Content-Centric Networking - PARC, a Xerox company." [Online]. Available: http://www.parc.com/work/focus-area/content-centric-networking/. [Accessed: 04-Apr-2012].

[4] "Named Data Networking." [Online]. Available: http://www.named-data.net/. [Accessed: 04-Apr-2012].

[5] "NSF Future Internet Architecture Project." [Online]. Available: http://www.nets-fia.net/. [Accessed: 04-Apr-2012].

[6] T. Li, B. Cole, P. Morton, and D. Li, "Cisco Hot Standby Router Protocol (HSRP)," *Internet Request for Comments*, vol. RFC 2281 (Informational), Mar. 1998.

[7] G. Peng, "CDN: Content Distribution Network," *CoRR*, vol. cs.NI/0411069, 2004.

[8] A. Barbir, B. Cain, R. Nair, and O. Spatscheck, "Known Content Network (CN) Request-Routing Mechanisms," *Internet Request for Comments*, vol. RFC 3568 (Informational), Jul. 2003.

[9] P. Vixie, "What DNS Is Not," *Queue*, vol. 7, no. 10, pp. 10:10–10:15, Nov. 2009. DOI :10.1145/1647300.1647302. ISSN:1542-7730

[10] E. J. Rosensweig and J. Kurose, "Breadcrumbs: Efficient, Best-Effort Content Locatio n in Cache Networks," Rio de Janeiro, Brazil, 2009, pp. 2631–2635. DOI: 10.1109/INF COM.2009.5062201. ISBN: 978-1-4244-3512-8

[11] T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, and I. St oica, "A data-oriented (and beyond) network architecture," *SIGCOMM Comput. Com mun. Rev.*, vol. 37, no. 4, pp. 181–192, Aug. 2007. DOI: 10.1145/1282427.1282402. IS SN: 0146-4833.

[12] S. Islam and J.-C. Grégoire, "Active ISP Involvement in Content-Centric Future Intern et," in *New Technologies, Mobility and Security (NTMS), 2011 4th IFIP International Conference on*, 2011, pp. 1 –5. DOI: 10.1109/NTMS.2011.5720586.

[13] H. Yuan and P. Crowley, "Performance Measurement of Name-Centric Content Distrib ution Methods," in *Proceedings of the 2011 ACM/IEEE Seventh Symposium on Archit ectures for Networking and Communications Systems*, Washington, DC, USA, 2011, pp . 223–224. DOI: 10.1109/ANCS.2011.43. ISBN: 978-0-7695-4521-9.

[14] S. Oueslati, J. Roberts, and N. Sbihi, "Flow-aware Traffic Control for a Content-Centric Network," in *Proc of IEEE INFOCOM conference*, 2012.

[15] S. DiBenedetto, C. Papadopoulos, and D. Massey, "Routing policies in named data net working," in *Proceedings of the ACM SIGCOMM workshop on Information-centric ne tworking*, New York, NY, USA, 2011, pp. 38–43. DOI: 10.1145/2018584.2018595. ISB N: 978-1-4503-0801-4.

[16] "cURL and libcurl." [Online]. Available: http://curl.haxx.se/. [Accessed: 31-May-2012].

[17] "VMware Workstation." [Online]. Available: http://www.vmware.com/products/workstation/. [Accessed: 05-Jul-2012].

[18] "Desktop | Ubuntu." [Online]. Available: http://www.ubuntu.com/download/desktop. [Accessed: 05-Jul-2012].

[19] “Wireshark · Go deep.” [Online]. Available: http://www.wireshark.org/. [Accessed: 26-Jun-2012].
[20] “Welcome! - The Apache HTTP Server Project.” [Online]. Available: http://httpd.apache.org/. [Accessed: 26-Jun-2012].
[21] “GNU Wget.” [Online]. Available: http://www.gnu.org/software/wget/. [Accessed: 26-Jun-2012].
[22] J. Eklund and A. Brunstrom, “Performance of Network Redundancy Mechanisms in SCTP,” Karlstad University, Division for Information Technology, 2005:48, 2005.
[22] M. Du, “Analyzing Caching Gain in Small Geographical Areas in IP Access Networks,” Master's thesisKTH, Communication Systems, CoS, 2012. Number 2012:246.

TRITA-ICT-EX-2012:300

www.kth.se