공학석사 학위논문

인터넷과 컨텐트 중심 네트워크의 상호 연동 구조

Design and Implementation of Internet/Contents Oriented Networks Interworking Architecture

2013년 2월

서울대학교 대학원
전기·컴퓨터공학부
정 효 기

인터넷과 컨텐트 중심 네트워크의 상호 연동 구조

Design and Implementation of Internet/Contents Oriented Networks Interworking Architecture

지도 교수 권태경

이 논문을 공학석사 학위논문으로 제출함

2012년 12월

서울대학교 대학원

전기 · 컴퓨터공학부

정효기

정효기의 공학석사 학위논문을 인준함

2012년 12월

위 원 장	김 종 권	(인)
부위원장	권 태 경	(인)
위 원	전 화 숙	(인)

Abstract

Design and Implementation of Internet/Contents Oriented Networks Interworking Architecture

Hyogi Jung

School of Computer Science and Engineering

The Graduate School

Seoul National University

A content oriented network (CON) brings a paradigm shift in the present Internet communication model by addressing named-data instead of host address. Because of totally different primitives for data transmission from the Internet, a user in a CON could not communicate with Internet directly. In this paper, we consider design of architecture for internetworking between legacy Internet and CON. We designed a proxy name server (PNS) which translates the Internet protocol into CON protocol and vice versa, because it is compatible with both protocols. To this purpose, we implement the proposed framework in the open source CCNx prototype and perform experiments in a real testbed. We expect that this

interworking framework can be used as a proposal for increasing deployment of CON in the future.

Keywords : Contents Oriendted Networks, Interworking, Future Internet, CCNx, Proxy Name Server
Student Number : 2011-20933

Contents

List of Figures

List of Tables

Chaper 1

INTRODUCTION

1.1 Background

The Internet has been one of the most important communication systems for various applications such as email, web surfing, and video/audio streaming. One of reasons for the success of the Internet is the flexibility, which it supports various protocols and wide spectrum of applications.In the Internet, unique IP addresses should be assigned to hosts and hosts communicate each other based on the addresses. However, communication environments are changing rapidly due to advanced network technologies and end devices. The recent traffic measurements reveal that traffic of contentoriented applications such as P2P and content delivery networks (CDNs) is increasingly more dominant [1]. These applications fucus on how to deliver contents efficiently rather than where to get contents from. In other words, users do not matter what the addresses of servers who originally published the contents.

Recently, there have been proposed emerging architectures for

content oriented network (CON) to address this issue [2]–[4]. In these architectures, a host does not need to know the address of content server. It only needs to know the content name that it desires to get. Correspondingly, routers forward content requests and content objects according to the contents names instead of hosts addresses. This makes it easy that networks cache the contents on their own storages, which reduce redundant content deliveries and speed up the content access.

Although existing CON architectures provide efficient content delivery service, it still needs massive effort to figure out how to deploy the protocols to some routers and to the hosts who want to use these services. There are two approaches for deploying the new network paradigm: clean slate approach and the overlay approach.

In the clean slate approach, legacy Internet protocols are replaced with new CON protocols. However, as of today, Internet connects billions of nodes and has millions of application that have been developed over the last 40 years on top of the existing architecture. So it is difficult for internet service providers (ISPs) to replace legacy Internet protocols with new ones immediately. Therefore, it needs large efforts to deploy the CONs in large scale testbeds [5]. With the overlay approach, CONs are implemented on top of the Internet

as applications. It makes newly proposed architectures coexist with the Internet, However they are constrained to deliver contents with higher throughput [2].

Therefore it is expected that the interworking architecture between CON and the legacy Internet takes a role as an intermediate stage from the current Internet to the future CON architectures. It can extend CON services to users of the Internet and it also provides plentiful contents in the Internet to CON services. In this manner, CON can be deployed gradually with less risk of introducing new architecture at a time.

1.2 Proposition

In this work, we propose an interworking framework between the Internet an the CON driven by the following design principles:

- *Easy Deployment:* The interworking architecture should be easily deployed in legacy Internet with investigation cost.
- *Transparency:* The interworking architecture should provide transparency of networking architecture to the end users. In other words, end users should be able to access any contents without any concern of the location of contents (i.e., either

CON or legacy Internet).

- *High Performance:* The interworking architecture should provide low latency communication between legacy Internet and CON since user experience (UX) is sensitive to the contents access latency.

Based on the design principles, we propose a novel interworking architecture between CON and Internet. Specifically, we introduce the PNS that provides the mapping service between content names (used in CONs) and a URL (used in the Internet). We verify the proposed framework between the Internet and the CCNx prototype [9] in two interworking scenarios: requesting contents from the Internet to a CON and requesting contents from a CON to the Internet. Experimental results validate the interworking operation and demonstrate that caching and retrieving improve the interworking performance.

The rest of this paper is organized as follows. Firstly, a general CON architecture and the CCNx details are introduced in Section II. In Section III, the proposed framework including the PNS functionalities are described. In Section IV, we evaluate performance with prototype implementation. And we propose some discussions for the proposed framework in Section V. Finally we conclude this

paper in Section VI.

Chaper 2

OVERVIEW OF CONTENTS ORIENTED NETWORKS

In this section, we introduce a brief overview on the architecture of CON.

Recently, the content-oriented networking approach has been explored by a number of research projects such as CCN [2], DONA [3] and NDN [4]. Although these approaches have differences with respect to their specific architectural design, they share common motivations. That is, they aim to develop network architectures that are better suited for content distribution.

These communication paradigms are different from what it is with the Internet. The Internet stands on an address-based conversation model, thus the delivery of data in the network follows an address-based. On the contrary, in CON, a user requests a content without address of a host which can provide it, and the communication follows a name-based, and the data follows the reverse path [6].

Contents naming in CONs is distinct from that in the legacy Internet. The names of contents in the legacy Internet are based on

the endpoints where the contents are located. Subsequently, a content hosted in a server is characterized by a universal resource locator (URL), which is the concatenation of the retrieval protocol, the host name, and the path name (e.g.,*http://www.snu.ac.kr/mmlab/videos/movie.avi*). In contrast, contents names in CONs hardly include hosts where the contents are located because the same content can be located in multiple servers and routers.

One of the well-known CON networking solutions is content centric networking (CCN), designed by the alo Alto Research Center. In CCN, users request named data packets by issuing Interest packets. Interest packets are forwarded by CCN routers in a hop-by-hop manner. Upon receiving an Interest packet, the router first looks up its content store (CS). If a copy of the requested data packet is found from CS, it instantly sends the copy back. Otherwise the router performs the longest prefix match on its forwarding information base (FIB) and forwards the Interest packet to the next hop towards the content. Routers keep track of each forwarded Interest in a data structure called pending interest table (PIT).

When the Interest reaches a node who has the content, the requested data packet is forwarded back along the reverse path. At

each hop, routers check their PIT for the Interest whose name is matched with the data name. If a match is found, the data packet is forwarded and a copy of the packet is kept in a local cache for future use. Incoming data packets that do not have a match at the PIT are considered as unwanted packets are simply discarded. After a data packet is forwarded, if the Interest packet is satisfied, the router deletes its entry from the PIT. In this way, CCN ensures that a user receives at most one data packet per issued Interest packet. Since CCN is the most representative work in CONs, the description of the proposed interworking architecture is based on CCN throughout this paper.

Chaper 3

THE INTERWORKING ARCHITECTURE

Since paradigms of CONs require totally different primitives for data transmission compared with the Internet, a user in a CON cannot communicate with Internet directly. Thus the interworking framework is needed for communications between Internet and CON. To this end, we define the PNS which performs a major role of interworking between a CON and the Internet. Fig. 1 shows the overall interworking architecture.

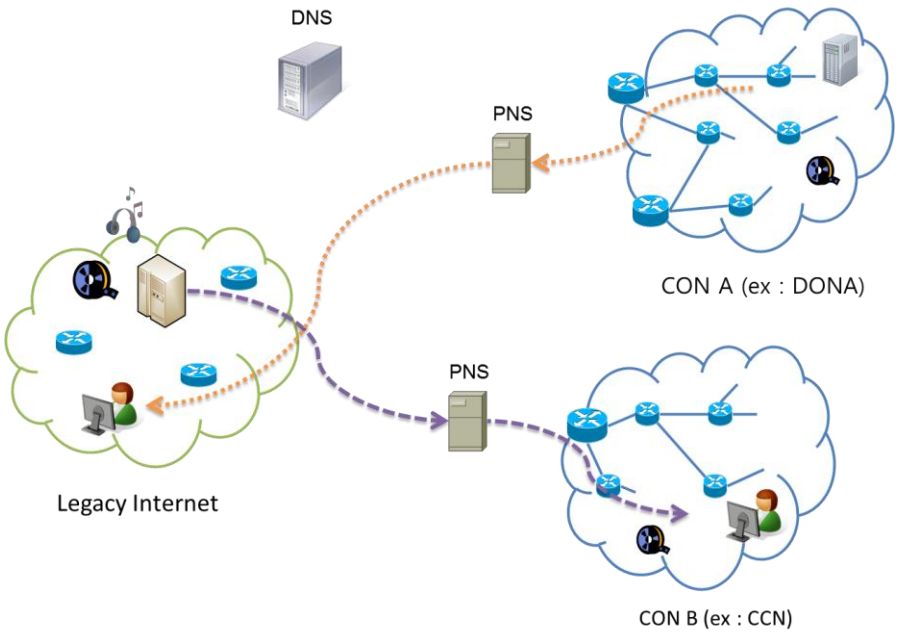

Fig. 1 : Internet/CON interworking architecture.

In this framework, PNS provides an adaption facility between the

Internet and a CON. It also provides mapping service between a content name (used in a CON) and a URL (used in the Internet). We describe the PNS functionalities and interworking procedures details in Section III-B1 and Section III-B2. In Section III-C, we explain how to enable the partial request in a PNS to obtain just part of a large content.

3.1 Proxy Name Server

As we mentioned above, there are many differences between Internet and CON such as network architecture and naming scheme. So, PNS should implement both Internet and CON protocols. And it has databases to map a content name (used in CON) to a URL (used in Internet) and vice versa. It also has to translate packets between Internet and CON. Fig. 2 shows the PNS architecture which includes such elements as two interfaces, databases and packet translator.

The database are divided into two parts, in this case, a database of the Internet and a database of a CON (DB Internet and DB CON for short). It maintains binding information to resolve name between a content name and a URL. If a content request packet

comes from the Internet, it searches DB Internet for binding names between the Internet and a CON, and vise versa. We assume that this binding information between a content name and a URL is registered by content providers. It is expected that content providers are in favor of disseminating binding information to PNSes since they desire to provide their contents to the users as much as possible. By registering, a content name in a CON and a corresponding URL are stored in DB CON for the content located in a CON. And the PNS registers its IP address for a domain of the URL to the DNS. Thus, a user in the Internet can obtain the PNS IP address by a DNS query. Similarly, a content name and a corresponding URL are stored in DB Internet. Thus, when a Interest for a content reaches a PNS, it can verify it corresponding URL.

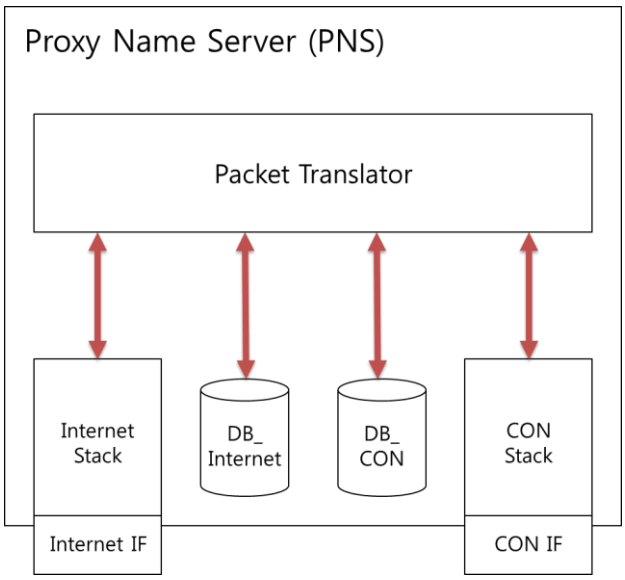

Fig. 2 : Proxy Name Server (PNS) architecture.

3.2 Interworking Procedures

We illustrate two interworking scenarios between a CON and the Internet via a PNS. The first scenario is the case which a user in the Internet requests a content in a CON and a CON node provides the requested content to the Internet user. The other scenario is the case which a user in CON requests a content in the Internet and an Internet server responds to the user in a CON. We describe these two scenarios in Section III-B1 and III-B2. And we also describe a scenario in which only some parts of a content is requested in Section III-C.

1) Content request from Internet to CON: We assume that user A in the Internet requests the content by the HTTP [7] request. Fig. 3 describes this procedure.

- Step 1: User A queries the DNS with a URL of the content.
- Step 2: The DNS returns a PNS IP address to the user.
- Step 3: A request message is transmitted to the PNS. For example, the following is the HTTP request message for a content object named *'/mmlab/documents/index.html'* from a CON.

GET /mmlab/documents/index.html HTTP/1.1

Host: ccn.snu.ac.kr

- Step 4: After the PNS receives the request message from the Internet, it extracts a URL from the request message. And it searches through DB CON with the URL and identifies a content name. After that, it generates a new Interest message that included the content name and disseminates the Interest message to the CON. The format of a generated Interest packet is as following.

con://ccn.snu.ac.kr/mmlab/documents/index.html

- Step 5: The Interest packet is broadcasted and the node in which the requested content located delivers the content object to the PNS by delivery mechanism of the CON.
- Step 6: After the PNS receives the content object from the CON node, it generates a HTTP response. The format of a generated HTTP response is shown as following.

HTTP/1.1 200 OK

Date:Mon, 10-Nov-12 13:15:14 GMT

Server: webserver/1.0

Content-length: 2048

Content-type: text/html

(Body of the document)

The status code "200" indicates that this is the successful response to its request. The message body includes the incoming content object from the CON. By this procedure, the user A receives the HTTP response from a PNS. That is, the PNS acts as an Internet server (i.e., Web server), which provides the transparency to user A.

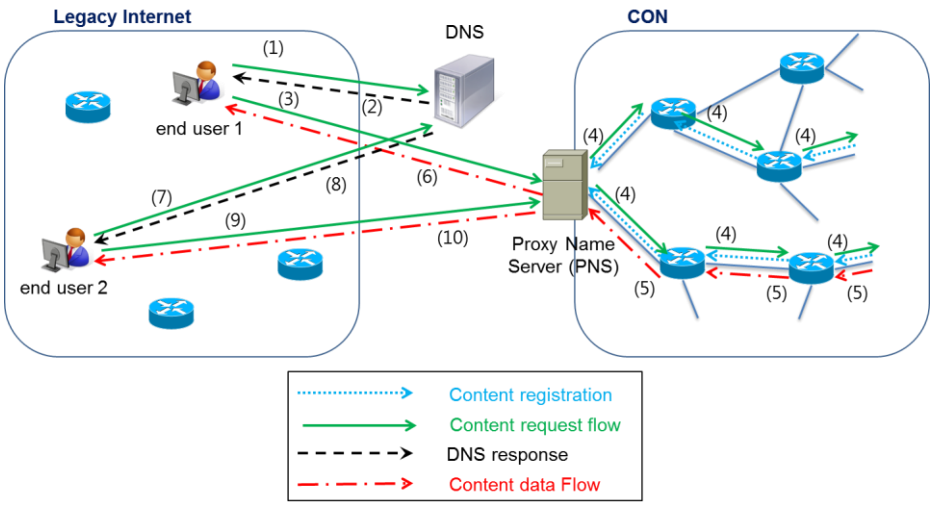

Fig. 3. Content request from Internet to CON.

Suppose that a PNS node caches the same content object in its storage. Then, another user, user B can retrieve the content object as followings.

- Step 7: User B sends a request message to the PNS as user A did.
- Step 8: The DNS returns a PNS IP address to the user.
- Step 9: After the PNS receives the request message from the Internet, it returns the requested content object if it cached it. If not, it disseminate the Interest message to the CON. Any node in the CON which cached the requested content object deliver it from its own storage to the PNS.
- Step 10: After the PNS receives the content object from the CON node, it sends a HTTP response with it to user B.

2) Content request from CON to Internet: In this scenario, a user in a CON requests a content and the requested content is delivered from the Internet to the user in the CON. Fig. 4 describes this procedure.

- Step 1: User A disseminates an Interest message to get a content. And the Interest message is forwarded to the PNS eventually if no intermediate CON node has the content.

The format of the Interest packet includes the content name, i.e.,

con://www.snu.ac.kr/mmlab/documents/about.html

- Step 2: After the PNS receives the Interest message, it extracts a content name from that message. And the PNS searches through DB Internet with the requested content name and identifies a URL. The PNS generates the HTTP request message that includes a URL and sends it to an Internet server that corresponds to the URL. The format of the HTTP request message is as following.

GET /mmlab/documents/about.html HTTP/1.0

Host: www.snu.ac.kr

- Step 3: After the Internet server receives the HTTP request message, it sends a response message with the content object to the PNS who sends a request message.
- Step 4: After the PSN receives the response message from the Internet server. it converts the HTTP response message

into a CON data message. The format of the CON data message is shown as following.

con://www.snu.ac.kr/mmlab/documents/about.html/v00/c00

Signature

Signed Info

Data (2048 Kbyte Chunk)

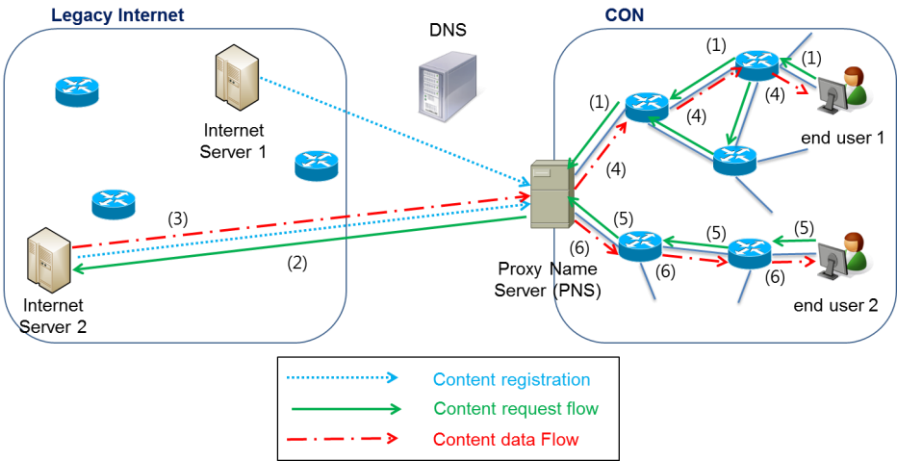

Fig. 4. Content request from CON to Internet.

And this content delivered from the Internet can be cached in the storage of the PNS or nodes in a CON. Suppose that the PNS or CON nodes cached the content object in its storage.

- Step 5: Another user, user B disseminates a Interest message for the same content object.

- Step 6: If the Interest message meets the PNS or CON node which cached the content object, it delivers the content from its own storage to user B.

3.3 Partial request

A user may desires to request a content with byte range option (or some of chunks independently) rather than an entire content. In this section, we describe how the proposed framework support this scenario. There is an example. Adobe acrobat provides a byte-serving service from servers, to serve individual pages of a large PDF file, using the byte-ranges feature [8]. If an Internet user requests just the second 4096-bytes of an entire document, the request would include the following option:

Range: bytes=4096-8171

For sake of simplicity, we assume that the requested bytes are mapped to chunks. In this case, a PNS calculates chunk numbers based on the CON chunk size and converts it to Interest messages.

Actually, a content is divided into small-size chunks (e.g.,4 KB[①] in the CCN) And each chunk is identified by a unique name based on a hierarchical structure. So, the resulting requests includes content name as the following.

con://ccn.snu.ac.kr/mmlab/documents/info.pdf/v00/c01

A single request message from the Internet may requires more than one chunk. For example, if it asks for the first 8192 bytes of a file, the PNS issues multiple Interest messages and it will send back each part in a several responses. In the case of a request from a CON, similar procedure is conducted. The PNS issues a request message with byte-range option for the requested parts.

① This is the default data packet size in the current CCNx implementation.

Chaper 4

EXPERIMENTAL STUDY

We leveraged CCNx open source project [9] to develope a proof-of-concept prototype of our Internet/CON interworking architecture. In CCNx project, a software prototype of CCN architecture is implemented and some simple experiments are presented to validate CCN principles [2] [10]. In this paper we implemented based on CCNx version 0.4.0, released in May 2011, which is composed of the following main modules:

- *ccnd* : the core networking daemon providing caching, forwarding, and packet authentication functionalities.
- *repository* : provides persistent storage of CCNx content backed by a filesystem, and respond to Interests to the available content.
- *utilities* : a variety of simple utilities developed in C or Java.

Some notable utilities are provided as following:

1) *ccndc* : configures the inter-machine forwarding of interest/content.
2) *ccnputfile* : command-line tool to publish a file as CCNx content.

3) *ccngetfile* : command-line tool to retrieve a file published as CCNx content and save it to a local storage

4) *VLC* : a plugin to the standard video player VLC that can read ccnx data.

We extend CCNx prototype to support the content delivery between Internet and CCN as described in Section III-B1 and Section III-B2. We add some functionalities as separate utilities instead of modifying the existing CCNx modules. The newly introduced utilities are followed:

1) *httpdownloader* : generates a HTTP request message and listens a HTTP response to obtain a content from Internet.

2) *httpserver* : listens for a HTTP request message from Internet and responds with a content in CCNx as a HTTP response message.

3) *httprecorder* : connects to a streaming server in Internet, downloads continually the streaming contents and publishes it as CCNx.

Also, we extend the original utilities, *ccnputfile* and *ccngetfile* to handle the contents in Internet.

A shown in Fig. 5, the testbed consists of three desktop PCs running CCNx and one laptop PC as a Internet client. One of the

desktop PCs is PNS and the others are CCN nodes. We consider a linear topology to show caching effect and configure nodes' forwarding table with the ccndc control program. In this testbed, CCN nodes are equipped with 3.10 GHz Intel Core i3-2100 and 4 GB of RAM. Each machine runs Linux Ubuntu 2.6.38.8 and it is connected to a switch through a 10 Gbps line card. And Internet client is equipped with 2.53 GHz Intel Core i5-460M and 4 GB of RAM in which Windows 7 Home premium is running.

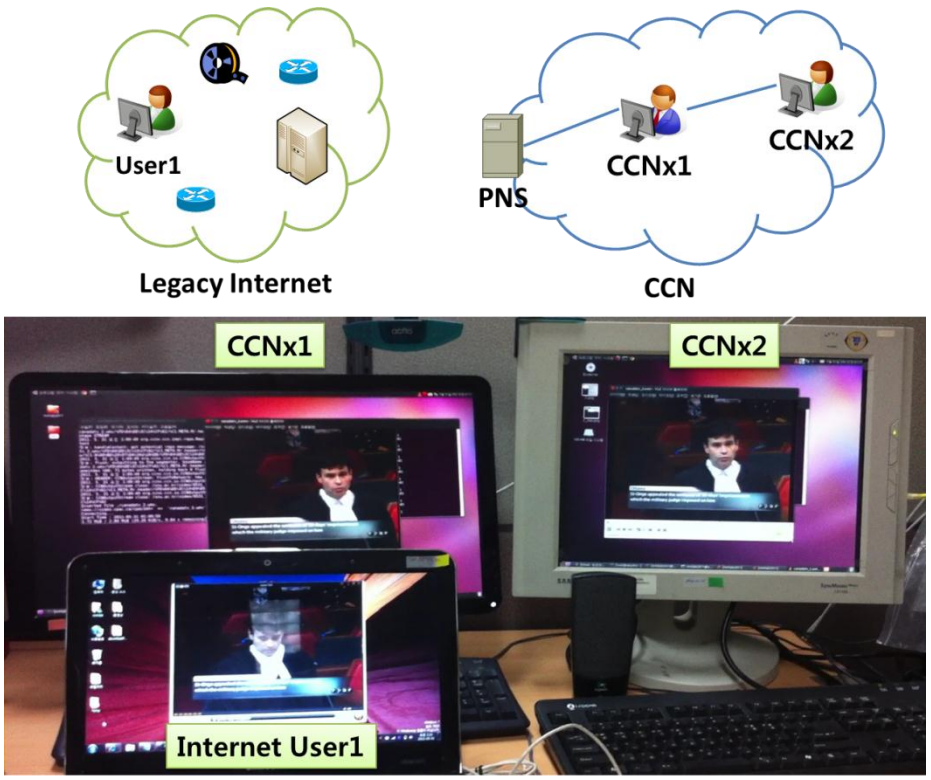

Fig. 5. Testbed configuration and demonstration of video streaming.

We conduct two kinds of test scenarios with this testbed. The first scenario is designed to demonstrate a content downloading between Internet and CCN. Another scenario is to demonstrate a live streaming from Internet to CCN.

4.1 Two test application

1) Contents downloading scenario: A CCN user retrieves a file published in CCN using the *ccngetfile* utility. It also could request a content (e.g., ccnx://www.snu.ac.kr/mmlab/images/pic.jpg) in Internet using same utility. An Interest message is generated by *ccngetfile*, and then forwarded to its neighbor nodes. If the requested content is not located on CCN, PNS will receive the Interest message eventually. The name of requested content is extracted by ccnd which is the core networking daemon in CCNx. Using this, PNS searches the URL (e.g., http://www.snu.ac.kr/mmlab/images/pic.jpg) in DB Internet and generates a HTTP request message with *httpdownloader*. And PNS requests the content to the content server in Internet and receives the content using HTTP protocol. PNS puts all the received contents in a repository using the *ccnputfile* utility. Finally, the user in CCN receives the requested

content.

Similarly, an user in Internet also retrieves a content that is published in CCN networks. We assume that a domain name of CCN (e.g., http://ccn.snu.ac.kr) is registered previously and the IP address of PNS is announced as a front end server by DNS. As we mentioned above, the *httpserver* utility is exploited to receive a HTTP request and to response with the content in PNS. It is implemented based on a simple web server, Sanos' HTTP server② [11].

If an Internet user1 requests a content (e.g., http://ccn.snu.ac.kr/mmlab/documents/doc.txt) in a CCN node (CCNx2) through a web browser. A HTTP request message is transmitted to PNS after DNS query. After PNS receives the HTTP request message, it extracts the name of requested content using *httpserver*. And PNS searches the content (e.g., ccnx://ccn.snu.ac.kr/mmlab/documents/doc.txt) in DB CON and generates an Interest packet using ccngetfile. After PNS receives the corresponding content from CCNx2, PNS generates a HTTP response message that contains incoming CCN content. Finally, the user1 receives the requested content. In this way, an user can download

② Sanos is a minimalistic 32-bit x86 OS kernel for network server appliances running on standard PC hardware.

various types of contents such as web pages, images, documents and videos without modification of Internet client.

2) Video streaming scenario: An user in CCN can receive a live steaming from Internet with PNS. If an user in CCN (CCNx1) wants to subscribe the live steaming content (e.g., ccnx://video.cpac.ca/cpac1e) from Internet, it broadcasts a request message to CCN nodes. After PNS receives the request message eventually, it sends a HTTP request message to a corresponding server in Internet after finding URL (e.g., http://video.cpac.ca/CPAC1E) in the DB Internet. We assume that this binding information (content name, URL) is registered previously by a content provider. Then PNS can receives the streaming content using *httprecorder* which is an utility in PNS which is implemented based on *TubeMaster++*[③] [12]. And PNS forwards incoming packets to the user. Finally, the user receives the live streaming using VLC Plugin [13] which can read ccnx data. If another user (CCNx2) requests the same live steaming, the user can receive the cached data not duplicated transmission. This can reduce the network congestion and improve the video stream's delivery speed. Fig. 5 shows the

[③] TubeMaster++ is a open source stream recorder allowing to capture HTTP multimedia streams.

demonstration of live steaming between Internet and CCN.

4.2 In-network caching effects

In CCN, the in-network caching feature provides several attractive advantages such as low dissemination latency and network transport load reduction [2]. The content is stored in each node (for a period of time), the number of repeated trips across the entire network to fetch the same content can be significantly reduced and improve the delay/throughput performance.

We also can obtain these advantages in our interworking architecture. To show how much time we can reduce a content delivery using caches, we conduct the following experiment. When an Internet user1 request the content that exists on CCNx2, we measure the content delivery time. If the user1 request the same content again, the user1 can retrieve the content from PNS that is 2 hops away from PNS because this content is cached in PNS after first request. We also measure the content delivery time with different size of contents.

Table I shows the result of this experiment. We observe that content delivery time is reduced when content is cached in PNS.

Furthermore, as the content size increases, the decrement ratio increases with PNS cache. Because a larger number of packets send to an Internet user using the cached data in PNS, it can reduce the content delivery time more than a few packets.

Type of Content	Content delivery time (sec)		Decrement Ratio
	No-cache	Cache	
Image (0.43 MB)	1.5	1.0	29.6%
Document (1.21 MB)	2.4	1.3	45.1%
Video (933 MB)	912.2	130.1	85.7%

Table. 1. Content delivery time.

Chaper 5

DISCUSSION

There are a few issues to efficiently support the architecture for interworking Internet and CON. Frist, load-balancing should be provided because PNS located between CON and Internet could be a bottleneck. In addition to, since all content providers register their content names to PNS, we have to reduce registration overhead.

5.1 Bottleneck of PNS

In our interworking architecture, PNS handle every interworking request. So, with heavy traffic - more users, more transactions, and more data - the more likely PNS becomes a bottleneck. When PNS got overloaded with requests, long response times retrieving data or transactions processing bottlenecks. To reduce the response time and provide users with the best available quality of service, it needs to distribute architecture that can spread incoming requests among several PNS nodes. Load-balancing among the PNS nodes can be achieved by several approaches with different degrees of

effectiveness: client-based, DNS-based, dispatcher-based and server-based [15]. But load-balancing approaches must be transparent to users, making a distributed system appear as single server to the outside world.

We can use a dispatcher-based architecture, where one of the processors (the dispatcher) receives all incoming requests and distributes them among the PNS nodes. In this architecture typically use simple algorithms for the selection of the PNS nodes (e.g., round-robin, server load) because the dispatcher has to manage all incoming flow and the amount of processing for each request has to be kept to minimum. Load-balancing among the PNS nodes can solve the performance and scalability problems of PNS, but we also consider synchronizing or distributing databases that is used naming convention. So we will further extend our interworking architecture to solve bottleneck and related problems.

5.2 Naming convention

CON naming scheme has some characteristics distinct from Internet. So PNS has databases to resolve content name or URL. The databases store binding information (content name, URL) that is

registered by content provider on Internet and CON. The content providers have to offer users their contents name since if users don't know contents name, they cannot request content. If they request contents, PNS can simply map between content name and URL using its databases. This mapping system is simple and fast. It can also reduce redundant traffic that is occurred different names for the same content because contents registration is managed by PNS. But there are several disadvantages. It is a burden content provider with registering all their binding information to PNS. It has also a scalability problem because the name of binding information needs to be unique.

If an user know an URL or content name through a searching engine (e.g., Google, Yahoo), PNS can resolve a more promising way without contents registration overhead. For example, the following is the URL of a searching engine that is a content in CON.

http://ccn.snu.ac.kr/mmlab/documents/index.html

PNS change just the URL prefix "http://" with the string "con://", which indicates the routing information needed by CON routers to form the new name of the corresponding content name used in

CON. The resulting content name is like the following.

con://ccn.snu.ac.kr/mmlab/documents/index.html

It will be routed to the certain content server, provided that all the CON routers have announced the prefix "con://" to the routing system.

Chaper 6

CONCLUSION

CON is a new paradigm that addresses the gap between the content-centric needs of a user and the current widespread address-based Internet architecture. In this paper, we proposed design of architecture for interworking between Internet and CON. We define PNS transforming HTTP request and HTTP response into CON Interest and Data respectively and illustrate two interworking procedures between CON and Internet via PNS. And we verify the proposed architecture by conducting experiments over a testbed. Thanks to the interworking architecture, CON can smoothly co-exist whit Internet and therefore such CON could be deployed gradually with less risk.

Chaper 7

REFERENCES

[1] Forecast, "Cisco Visual Networking Index: Forecast and Methodology, 2011-2016," Cisco Public Information, May 2012.

[2] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. Briggs, and R. Braynard, "Networking named content," in Proc. of ACM CoNEXT 2009.

[3] T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, and I. Stoica, "A data-oriented (and beyond) network architecture," in Proc. of ACM SIGCOMM 2007.

[4] Named Data Networking(NDN) Project, http://www.named-data.net/

[5] J. Pan, S. Paul, and R. Jain, "A Survey of the Research on Future Internet Architectures," IEEE Communications Magazine, vol. 49, no. 7, pp. 26-36, 2011.

[6] A. Ghodsi, T. Koponen, B. Raghavan, S. Shenker, A. Singla, and J. Wilcox, "Information-Centric Networking: Seeing the Forest for the Trees," in Proc. of ACM Workshop on HotNets-X 2011.

[7] Hypertext Transfer Protoco, http://www.w3.org/Protocols/rfc2616/rfc2616.html

[8] Adobe Acrobat, http://www.adobe.com/products/acrobat.html

[9] CCNx project, http://www.ccnx.org

[10] V. Jacobson, D. K. Smetters, N. H. Briggs, M. F. Plass, P.Stewart, J. D. Thornton, and R. L. Braynard, "Voccn: voice-over content-centric networks," in Proc. of ACM Workshop ReArch 2009.

[11] Sanos Operating System Kernel, http://www.jbox.dk/sanos/webserver.htm

[12] TubeMaster++, http://www.tubemaster.net/index.html

[13] VideoLan, http://www.videolan.org/vlc/

[14] CPAC, http://www.cpac.ca/eng

[15] V. Cardellini, M. Colajanni, and P.S. Yu, "Dynamic Load Balancing on Web Server Systems," IEEE Internet Computing, vol. 3, pp. 28-39, 1999.

요 약

컨텐트 중심 네트워크가 현재 인터넷의 호스트 주소 기반의 커뮤니티 모델에서 이름 기반 주소 방식으로 패러다임의 변화를 가져왔다. 이처럼 현재 인터넷과 전혀 다른 데이터 전송 기법 때문에, CON의 사용자들은 인터넷과 바로 통신 할 수 없다. 본 논문에서는 현재의 인터넷과 CON이 상호 연동 할 수 있는 구조를 제안한다. 우리는 두 개의 프로토콜에 호환하는 PNS를 제안하여 인터넷 프로토콜과 CON 프로토콜를 서로 변환을 가능하게 하였다. 그리고 제안한 프레임워크를 오픈소스인 CCNx를 이용하여 구현하였고, 실제 테스트 베드를 통하여 성능 실험을 하였다. 이러한 연동 구조를 통하여 CON이 현재의 인터넷과 공존 가능케 함으로써 새로운 인터넷 서비스로 자연스럽게 진화 할 수 있기를 기대한다.

주요어 : Contents Oriendted Networks, Interworking, Future Internet, CCNx, Proxy Name Server
학번 : 2011-20933