國立交通大學

資訊科學與工程研究所

碩 士 論 文

以內容為中心的網路動態調整快取系統

Dynamic Age-Adjusted Cooperative Caching in Content Centric Networks

研 究 生：楊文翔

指導教授：陳 健 教授

中 華 民 國 一 百 零 二 年 十 二 月

以內容為中心的網路動態調整快取系統

Dynamic Age-Adjusted Cooperative Caching in Content Centric Networks

研 究 生：楊文翔　　　　Student：Wen-Xiang Yang

指導教授：陳　健　　　　Advisor：Chien Chen

國 立 交 通 大 學

資 訊 科 學 與 工 程 研 究 所

碩 士 論 文

A Thesis

Submitted to Institute of Computer Science and Engineering

College of Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer Science

December 2013

Hsinchu, Taiwan, Republic of China

中華民國一百零二年十二月

以內容為中心的網路動態調整快取系統

研究生：楊文翔　　　　　　　　　　　　　　　　　指導教授：陳　健

國　立　交　通　大　學　資　訊　科　學　與　工　程　研　究　所

中文摘要

新一代的網路架構以內容為中心的網路架構已成為近年來最熱門的網路研究主題，以資料名稱取代傳統 IP 的方式尋找資料，並且在每個路由器上裝備快取記憶體以儲存轉發的資料，使得資料可以再次讓使用者就近存取，但是快取記憶體空間是有限的，所以有效利用有限的快取記憶體空間是很重要的一個議題。Age-based cooperative caching 的方法中提到利用資料的熱門程度以及快取設備的地點來決定資料的存活時間(time-to-live)，利用此方法可有效減少頻寬消耗。而我們延伸這個想法，利用動態的調整資料的存活時間來進一步的改善網路效能。當在短時間內多數的使用者針對同一筆資料進行存取時，我們利用延長資料的存活時間來改善快取命中率的問題，使得客戶端的使用者可以持續在快取記憶體中取得熱門的資料。另外，為了減少不同快取記憶體中儲存同一筆資料的問題，當下層路由器的快取記憶體儲存同一筆資料時，即通知上層路由器將快取記憶體中的資料存活時間減低，使得上層的資料可以提早過期而被其它資料替換。除此之外，當快取記憶體中的資料異動時，我們利用有限廣播的方式來達到互相傳遞快取資訊的位置，以達最短距離繞徑的目的。而模擬的結果顯示，我們的方法可以有效的增加資料命中率以及減少頻寬消耗、減低伺服器負載。另外，我們利用軟體自定網路(Software Defined Networks)的方法，將以內容為中心的網路架構以及我們的演算法實作出來，並以軟體自定網路 Mininet 擬真器進行效能測試，其效能數據和模擬的結果非常接近，皆顯示出我們的方法能有效的改進以內容為中心的網路效能。

關鍵字：以內容為中心網路、網路快取系統、最近最少使用替換演算法、軟體自定網路、最短距離繞徑

Dynamic Age-Adjusted Cooperative Caching in Content Centric Networks

Student: Wen-Xiang Yang **Advisor: Dr. Chien Chen**

Institute of Computer Science and Engineering
National Chiao Tung University

Abstract

Content-Centric Networking (CCN) is a novel networking paradigm. It changes the host-centric model to a content-centric model and allows each CCN router to cache content. Since the cache size of a router is limited, design of a cooperative caching scheme is an important issue in CCN. Among them is the Age-Based Cooperative caching scheme (ABC) which gives an age (i.e., time-to-live) to each piece of content in the cache based on the content popularity and the location of the cached node to reduce the bandwidth consumption and server loading. In this thesis, we focus on extending the idea of ABC by adding a dynamic age adjustment scheme. To handle short-term burst requests to the same content, our age adjustment scheme increases the content's age value when the cache is hit. To reduce redundant data, our age scheme decreases the age value of an upstream node when the downstream node has cached the same content. Besides, we also propose a simple cache information update scheme to further enhance the routing performance. Simulation results show that our scheme can further enhance cache hit ratio and reduce server loading and bandwidth consumption. Furthermore, we realized the CCN architecture with our caching and routing schemes using Software Defined Networks (SDN). We use SDN emulator Mininet to verify the performance. The emulation results show that they are very close to the simulation results.

Keywords: Content Centric Networks, Cooperative Caching, Least Recently Used, SDN, Shortest Path Routing

誌謝

本篇論文的完成，我要感謝這兩年給予我支持與鼓勵的人。首先要感謝我的指導教授 陳健博士，雖然在研究上常常遇到挫折，但陳老師常常鼓勵我要不屈不撓，勇往直前，除此之外也給了我許多的指導和啟發，都讓我在遇到困境時能找到解決的方法，也因為老師在實驗上以及論文寫作的細心指導，我才能順利完成此篇論文。除了在科學上的知識，老師也常鼓勵我們要好好的規劃未來的人生，並且要把握當下不要虛度光陰，在此表達最誠摯的感謝。同時也感謝我的口試委員，簡榮宏教授、王國禎教授、朱煜煌博士，在口試時提出了許多寶貴意見，讓我受益良多。

感謝我的同學劉郁鈞、黃健忠、曾國郡、林俊宇陪我渡過這豐富又充實的碩士生涯，在修課時大家總是會互相幫助，互相切磋。研究時若遇到瓶頸，他們就像是我的智囊團一樣，常常能幫忙想出一些好主意。特別感謝張哲偉以及陳盈羽兩位博班學長，對我的論文給予許多寶貴和實用的建議，還有同學黃俊憲，在我實作 SDN 擬真實驗的過程中，幫了我許多忙，讓我可以排除萬難，順利的完成實驗。

最後，我要感謝家人對我的支持，有了他們我才能無後顧之憂的專心於研究所的課程，在此要向他們致上最高的感謝。

本文目錄

圖目錄

表目錄

Chapter 1 Introduction

Many content delivery services have been developed in the Internet such as Youtube, Hulu, Netflix, etc. Saving the network bandwidth consumption and the server load are the important issues of this kind of services. Content Centric Networking (CCN) or Information Centric Network (ICN) [1-10] is a new type of Internet architecture developed to solve this issue. The basic idea of CCN is to replace the narrow waist at the network layer from the physical network location (i.e., Internet Protocol (IP)) to the content name itself. Each CCN router is equipped with the storage to temporally cache contents. If one of the CCN routers on the routing path has stored the corresponding content that a user requested, then it immediately sends it back to the user. Since the user can obtain its requested content from an intermediate router not necessarily from server, then CCN can save server load and bandwidth consumption in the Internet.

In CCN [1], a content consumer sends out an interest packet, which carries a name that identifies the requested content. Once the interest reaches a router that has the requested content, a data packet is sent back. There are three data structures in router to support such routing process as follows: (1) Content Store (CS), (2) Forwarding Information base (FIB), (3) Pending Interest Table (PIT). CS is used to store content when data packets pass through. However, the caching policy is managed by caching algorithm, such as FIFO, LRU, etc. FIB is like a routing table in IP. Instead of IP address, the content name in the interest is compared with the entries in FIB to decide which face (port) to forward the Interest packets. When an Interest arrived, PIT table will record the arrival face of that interest if there is no corresponding content in CS. Then, the content can be send back reversely to the content consumer according to the PIT information.

Efficient caching algorithms profoundly influence the performance of CCN. Least Recently Used (LRU) is a common caching strategy, where a node always replaces data with a least recently used when the cache overflows. Since LRU lacks of the coordination between nodes and cache contents, thus a lot of cooperative caching schemes have been proposed[2-12].[8] considers a hierarchical cache. Each cache computes an average maximum document access interval without a cache miss to decide whether the content should be cached or not. In [9], each node collects the miss rate, miss penalty, cost loss of evicting objects, and access frequency to decide which content to be cached and the optimal caching locations. In [2], the upstream node adds a flag in the content to suggest that the downstream node to cache this content based on the popularity of this content.

Age-based Cooperative Caching(ABC) [10] is a simple and effective cooperative caching scheme in CCN. ABC computes an age of content (i.e., the time-to-live of the content in the cache) based on the popularity of that content and the location of the cached node. To pass the age information to the downstream node, ABC adds an age field to the content. When the cache overflows, ABC replaces one of the out-of-date contents in its cache. The benefits of ABC are as follows: (1) ABC gives the popular content to a larger age to stay in the network. (2) ABC gives a cache near the network edge a larger age to achieve a better response time, and (3) the content near the server has a smaller age such that the less popular content has a better chance of being cached.

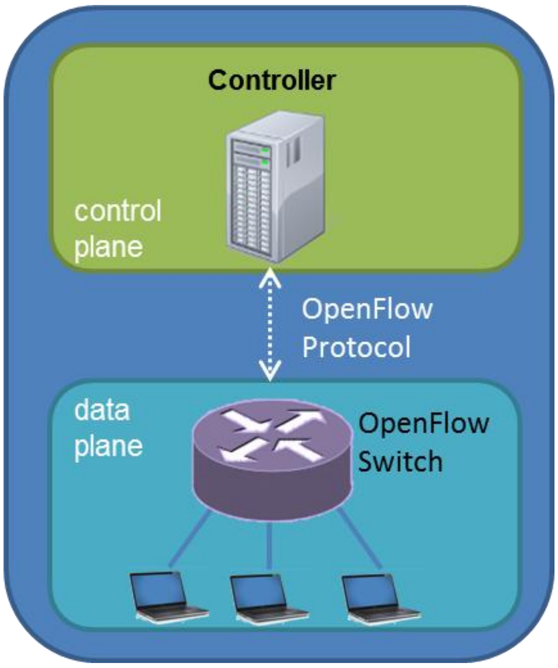
However, some issues still need to be solved in ABC. In most content-centric services have the burst requests to the same content within a short period of the time due to a new lease of video or a block buster movie. Although ABC has considered content popularity (i.e., the long- term statistical information) to give the age of the content in the cache, it does not adjust the age taking this temporal-locality factor into consideration. Besides, there are still some redundant contents among CCN routers because ABC cached content in each router on the routing path even though each router gives a different age to content.

On the other hand, the process of getting the content-location information (i.e., the cache information of other node) is another important factor to achieve a better performance in CCN. When a node knows the content-location information, it can help that node to redirect its request to the content caching node which is closer than content server. Complete broadcast is a naive way to disseminate the cache information among routers, but this way may cause a lot of extra overhead. Therefore, this study further designs a restricted broadcast scheme to reduce the overhead of the content location information dissemination.

This study extends the ABC protocol by adding a dynamic age adjustment scheme. To overcome the short-term burst request to the same content issue in CCN, an age-refresh scheme is added in ABC. When a cache hits, it will refresh that content's age to its original assignment. Since the cache hits will continue to refresh the age of the content and prolong the survival time of the content in CCN, the age-refresh scheme can handle the temporal-locality. To solve the redundant contents in ABC, an age-decrease scheme is proposed. When a downstream node decides to cache content in its local cache, it will send an acknowledgement (ACK) to the upstream node. When the upstream node receives the ACK, it will immediately decrease the age of that content exponentially. When more downstream nodes store redundant contents, upstream node will decrease the age of that content more such that that content will edge out from the upstream nodes earlier. In this way, the cache space can be used more efficiently. This study further designs a cache information update scheme to inform the near-by routers. The cache information update scheme is a restricted broadcast scheme. When a node caches content, it will broadcast this information within *k* hops, where *k* is the average length of shortest paths between nodes and content server. When a node receives this information, it will update its routing table if the content cached node is closer than original content server. We evaluate our scheme under GEANT topology and 3-ary tree topology. The simulation results show that our age adjustment scheme has an improvement of 31% and 20% in bandwidth consumption compared to LRU scheme and ABC scheme, respectively.

Considering implementing the CCN architecture on the other hand, it is difficult to substitute all the traditional switches in the whole world with CCN switches. One of the best ideas is using Software Defined Networks (SDN) [13] , as shown in Figure 1-1. The main idea of SDN is to separate the network into the control plane and the data plane. The ISP can develop its own application to control the network operation. Hence, the CCN mechanism can be implemented over IP architecture using SDN without too many costs. OpenFlow is a communication interface between the control and forwarding layers of SDN architecture. In the control plane, the controller can manage the network flow through OpenFlow protocol.

In our emulation, we use Mininet [14] with Open-vSwitch and implement the three



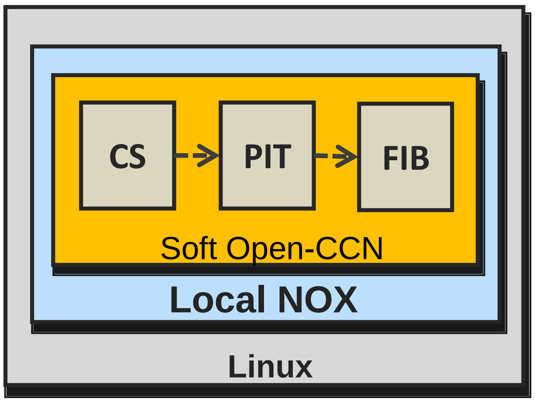


Figure 1-2 Nox Controller

Figure 1-1 SDN Architecture

tables (CS, PIT, FIB) of CCN in nox controller [15] as shown in Figure 1-2. We evaluate our scheme under Abilene topology. The emulation results show that our age adjustment scheme has an improvement of 26% and 13% in bandwidth consumption compared to LRU scheme and ABC scheme, respectively. Besides, the simulation results in Abilene topology are almost match the emulation results.

The rest of this thesis is organized as follows. Chapter 2 introduces the related works. Chapter 3 presents details of our dynamic age adjustment scheme. Chapter 4 shows how to realize CCN using SDN. Chapter 5 discusses simulation and emulation results, and Chapter 6 concludes this thesis.

Chapter 2 Related Works

CCN is different from Content Distribution Network (CDN) where cache space is distributed to each router. The idea does not only reduce the cost in hardware equipment (compared to the equipment at the server level) but also saves the bandwidth consumption and server loading of the network. Designing efficient cooperative caching schemes is an important issue in CCN. Leave Copy Down (LCD) [17] stores another data in *l*-1 level node only when a cache hit in *l* level node. Move Copy Down (MCD) [17] is similar to LCD. The difference is that MCD only caches the data in the *l*-1 level node and deletes the data in the *l* level node. In this way, the cache space in MCD strategy is more efficiently utilized, but it may cause some requests to traverse more hops to reach their corresponding contents, and result in more bandwidth consumption in comparison to LCD. WAVE [2] deals with a chunk-level caching strategy also similar to LCD. The difference is that WAVE uses piggyback suggestion bit to inform the downstream node, and a downstream node can ignore or accept the suggestion from the upstream node. Another difference is that the suggestion from upstream is based on the popularity of content. The research of [8] and [9] are designed for a hierarchical cache. However, [8] maintains a characteristic time which is the average time between two replacements in the cache. According to the access frequency and the characteristic time, a low-pass filter algorithm is designed to decide which one of the nodes upon the routing path from the request node to the content server should cache the data. The authors of [9] on the other hand collect miss rate, miss penalty, cost loss of evicting objects and access frequency from the routing path. Based on this information, the optimization problem is formulated to decide the best locations of contents. The idea of [11] is calculating the distance between client and server and cache size to decide the cache probability. The data closer to client, the higher probability the data will be cached. [18] use the Shield function to keep track the request rate of Interest. The primary purpose of the Shield function is to discourage unpopular data being cached. Hence, the data will be cached when the Interest is requested over a threshold.

ABC [10] adds an ‘age’ field to each content packet representing the content age of the upstream node. ABC dynamically configures content’s age based on (a) distance (i.e., the number of hops) of cached router toward the server, and (b) popularity of content. When content reaches a router, the router first checks cache space for the content. If there is no available space, the router removes the most expired content in its local cache, and gives the new-arrival content an age as follows. If the router is the first router next to the content server, then the age for the content is given as:

$$age = BASE\_AGE * weight \tag{1}$$

where *BASE_AGE* is the initial value of the age and *weight* is the ratio of access frequency of the cached content. Otherwise, the router extracts the age of incoming content (C_u) and calculates the new age as:

$$age = \min\left(C_u * (1 + weight), MAX\_AGE\right) \tag{2}$$

where *MAX_AGE* value is used to avoid the age growth infinitely. Based on Equations (1) and (2), ABC has the following two features: 1) The more hops away from server, the more age it has. 2) The more popular the content is, the longer age it has.

On the other hand, the routing protocol is also an important aspect of this scheme. Breadcrumbs [19] leaves routing message (breadcrumb) when a router forwards content. The routing message includes where the data is from, where the data is forwarded to, and when the arrival time of the content is. If a request matches the index of breadcrumb, the request is forwarded to the caching router according to breadcrumb information. Potential Based Routing (PBR) [20] uses the capacity and the traffic load of caching node to define the quality of the content, and broadcast the quality of content to its neighbor node. The node calculates the potential value according to the value of content quality and hop count. The lower the potential value it has, the more chances interest will be routed to that node. INFORM [21] uses 2 phases to make the routing decision. First phase is “Exploration phase” which is calculating the Q value to keep track efficiency of routing path. Second phase is “Exploitation phase” which is to decide the routing path using different interface according to the Q value. The lower Q value, the higher priority will be used to forward packet. The idea of Stateful Forwarding Plane[22] is that use color to track the state of interface. If the data come back overtime, then router will try alternative interfaces to send the packet and mark the interface is yellow state. Otherwise, the interface will be mark green state when data come back within time limit. Hash routing [23] is using hash function to designate the router to cache the data. Edge router is responsible for forward data to designated core router. There are different ways to forward data when data sent back from server. The Symmetric is using the same routing path and Asymmetric is using shortest path when send packet out. The Multicast is sending one copy of data to original routing path to cache in designated core router and another copy of data is sending to client by shortest path.

Chapter 3 Age Adjustment Scheme

Even though ABC has taken the popularity into consideration, and achieves better performance than LRU, it cannot handle the short-term burst requests to the same content. Besides, ABC still suffers from redundant data problem because ABC cached content in each router on the routing path. This section introduces the principle of our age adjustment scheme which is aimed at further improving the performance of ABC, and solves the problems of routing information dissemination. Subsection *3.1* introduces the detail of our age adjustment scheme. Subsection *3.2* is a cache information update scheme.

3.1 Dynamic Age-Adjusted Cooperative Caching

ABC is an efficient cooperative caching scheme, which outperforms the LRU and FIFO. This study extends the ABC scheme by adding a dynamic age-adjusted scheme, named Dynamic Age-Adjusted Cooperative Caching (DAAC). DAAC has two parts: (1) age-decrease scheme and (2) age-refresh scheme. DAAC uses an acknowledgement-based scheme to decrease the age of the content on the upstream routers in order to reduce the redundant data. When cache hits, age-refresh scheme refreshes the content's age to its original assignment to overcome the temporal-locality.

1) Age-decrease scheme: When content is forwarded to a downstream router, if the downstream router has free space for data caching, then the downstream router sends an acknowledgement back to the upstream router. When the upstream router receives the ACK, it decreases the age of content using equation (3).

$$age = age * (1 - \frac{1}{f})^{Number\ of\ ACK} \tag{3}$$

where $f = K - 1$ and K is the node degree of the upstream router. According to Eq(3), the redundant data can be dropped sooner due to exponential age decrease.

2) Age-refresh scheme: Our scheme refreshes the age of its original assignment when content hit to overcome the short-term burst requests. It’s very common in content services that the same content getting lots of hits in a short period of time such as a new release of TV series, sport broadcasts, news events, etc. Since a burst of request will be sent within a small period of time to the content servers, it may cause a lot of traffic and bandwidth consumption to the networks. Therefore, the age-refresh scheme can keep the content in cache to serve this kind of burst requests without content servers involving.

3.2 Caching Information Update Scheme

We design a caching information update scheme to redirect the request to the near-by router which caches the content. When a router has stored content in its cache, that router sends content caching information to other routers using limited broadcast. The broadcast distance (i.e., the number of hops of rebroadcast) is the average number of the hops from the routers to the content server. It can be defined as in equation (4):

$$Broadcast\ Distance = \frac{\sum_{n \in U \setminus \{S\}} \#\, hops\ from\ n\ to\ S}{|U|} - 1 \tag{4}$$

where U is the set of all routers in the network and S is the content server. Since broadcast distance is the average number of the hops from the routers to the content server, most of the routers in the network can redirect the request to the closest router hold the content within its neighborhood. When a router receives cache information, it will check if the distance to the location of the cache replica is less than the distance to the content server. If so, it will modify the routing table to the cached router. The caching information update scheme can redirect the interest to the near-by caching router, even though the caching router is not on the original routing path.

Chapter 4 Implement CCN using SDN

The CCN network is an efficient way to solve the current IP network traffic problem. Several methods to implement CCN have been published. These are detailed below:

1. Clean-slate approach：This approach follows the CCN mechanism instead of TCP/IP and uses CCN routers in place of traditional IP routers. But this method is too difficult to realize.

2. Overlay approach：This approach uses software to develop the CCN mechanism such as CCNx. The CCNx can be used for current IP network. However, the CCNx cannot be ported to routers and it does not support QoS or load balancing functions.

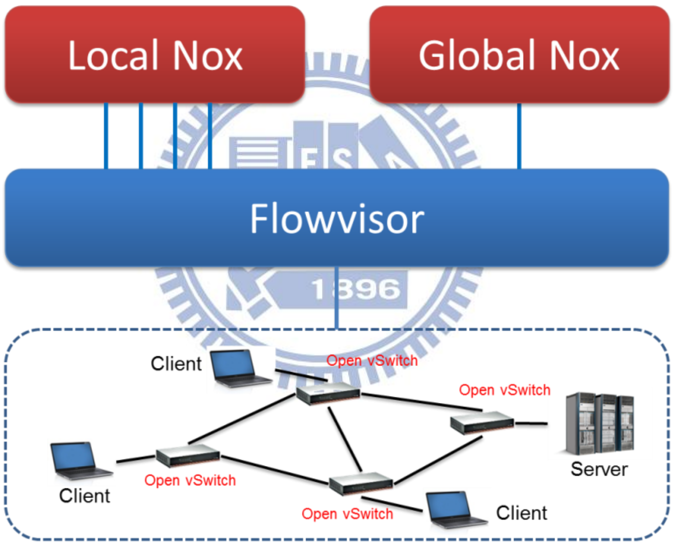


Figure 4-1 Open-CCN Architecture

In order to implement CCN in current IP architecture, the OpenFlow is used to design the CCN architecture. The Open-CCN architecture is shown in Figure 4-1. Mininet is used to emulate the OpenFlow network and Flowvisor [16] is used to forward the packet to the corresponding nox controller. In Mininet, each Open-vSwitch connects to one host and two nox controllers: one controller is the corresponding Local Nox, and the other is the Global Nox. All hosts are running a ccn-client program, except the node that is chosen as the server which is running a ccn-server program.

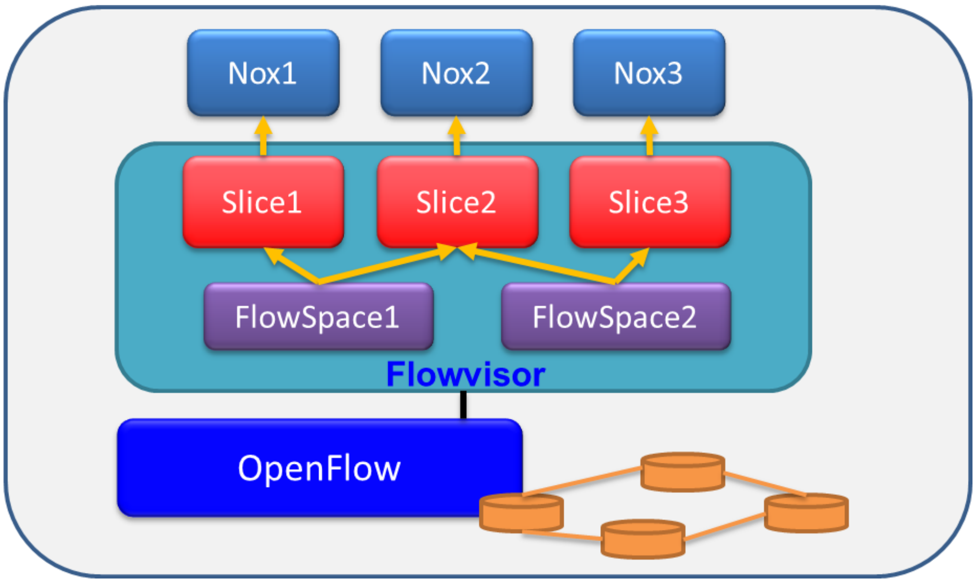


Figure 4-2 Architecture of Flowvisor

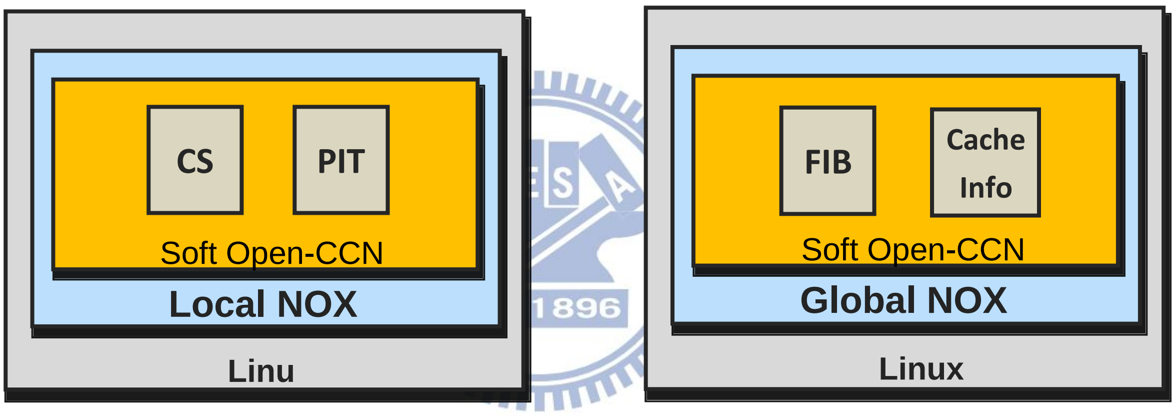


Figure 4-3 Local Nox and Global Nox

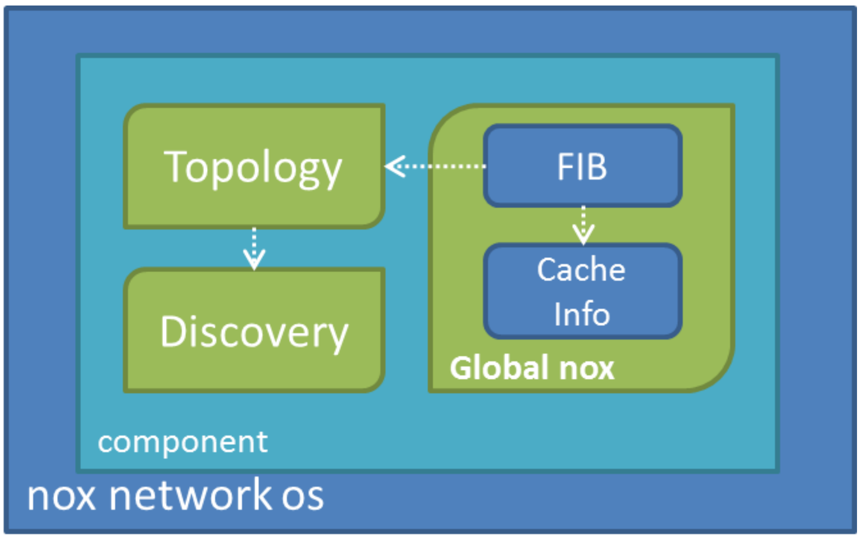


Figure 4-4 Nox Network OS and Components

However, there is a problem that OpenFlow switch cannot connect to multiple controllers.

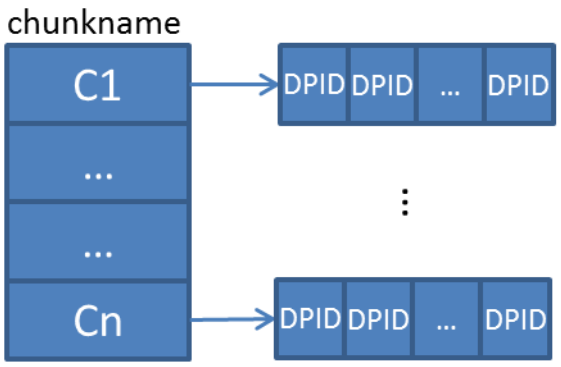


Figure 4-5 Data structure of Cache Info Table

Therefore, we use the Flowvisor software to forward packets to the corresponding nox controller. Flowvisor is configured between Mininet and Nox as shown in Figure 4-1. There are two main components in Flowvisor: one is Flowspace and the other is Slice. The Flowspace is a rule which is consisted of different setting of IP header. A different Flowspace can be set to send the packet to different Slice. Each slice corresponds to a Nox controller. The architecture of Flowvisor is shown in Figure 4-2. In our emulation, each Open-vSwitch corresponds to a Local Nox controller and every Open-vSwitch connects to the Global Nox controller.

For implementing our DAAC_CIU scheme, the three tables are separated into two parts. The CS table and PIT table are implemented in “Local Nox”, and the FIB table is implemented in “Global Nox” as shown in Figure 4-3. In the Nox network operating system, there are 2 components which can be used to establish the topology information as shown in Figure 4-4. First, the “Discovery” component sends LLDP messages to every OpenvSwitch to test the link between each OpenvSwitch and the message is stored in a data structure. The next step is that “Topology” component will read the data structure to establish the topology information. The Global Nox uses Dijkstra’s algorithm to calculate the shortest path between each node when topology is changed. The out port of shortest path and the hop count between each node will be stored in the FIB table. Furthermore, a Cache Info Table is also maintained in Global Nox. The data structure of the Cache Info Table is shown in Figure 4-5. The Cache Info Table will maintain all the cache information of the router. Therefore, the same chunk may be stored in two or more routers. The Global Nox will compare the FIB table and Cache

Info Table to find the closest Local Nox which stored the content in the cache when a request of Interest is received.

Type	Message(rule)	Action of Switch
general rule	Interest	send to flowvisor
general rule	Content	send to flowvisor
general rule	Content(last packet)	send to flowvisor
general rule	ACK	send to flowvisor
modify rule	CIU_routing	modify packet header of Interest and then sent to flowvisor
modify rule	CIU_caching	modify packet header of Interest and then send to flowvisor

Table 1 Type of packet & Action of Switch

The CRC-64 is used to encode the file name into a unique interest name and store the interest name in IP header. For the sake of implementing easily, UDP protocol is used to send the packet, and raw socket is used to receive the content back. The message list in Table 1 will be installed in Open-vSwitch when Nox controller starts up and all the rules as shown in Table 2 will install in Flowvisor.

Message	Action of Flowvisor
Interest	send to corresponding Local Nox
Content	send to corresponding Local Nox
Content(last packet)	send to corresponding Local Nox
ACK	send to corresponding Local Nox
CIU caching	send to Global Nox
CIU routing	send to Global Nox

Table 2 Action of Flowvisor

The step of sending interest is that ccn-client sends a packet with interest name as shown in Figure 4-6. The Open-vSwitch will send the packet to Flowvisor when match the rule of Interest that is installed early and then Flowvisor sends the packet to corresponding nox according to DPID (mac address) of Open-vSwitch. The corresponding Local Nox follows CCN rules to check the CS table. The content will be sent back to the client when cache hit in

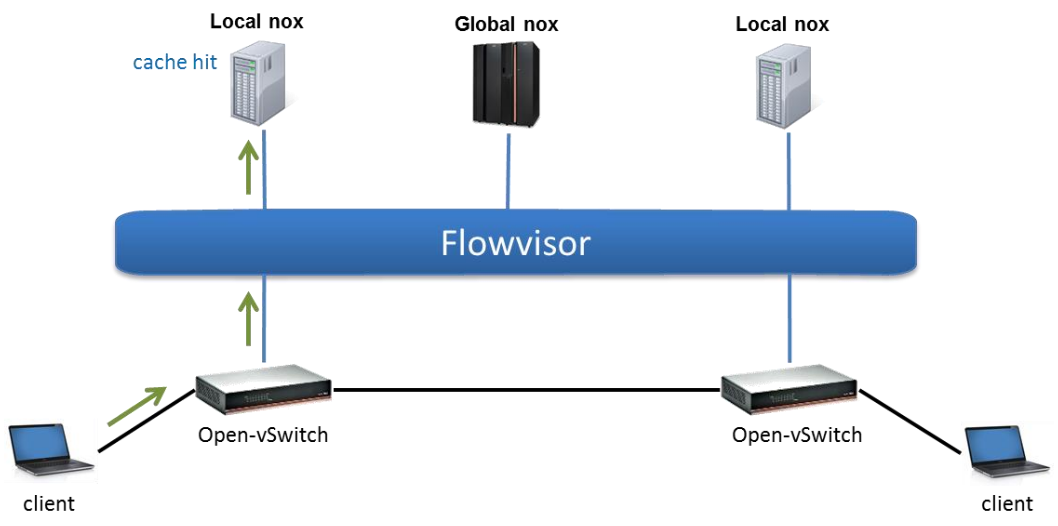


Figure 4-6 CCN-Client send interest

PIT table and then send the interest to next router according to the routing information in FIB table when cache miss in CS table. In our CIU scheme, the Local Nox will send the packet to Global Nox instead of searching FIB table in Local Nox as shown in Figure 4-7. The Open-vSwitch modifies the Interest message to CIU_routing message and then sends to Global Nox through Flowvisor. If there is caching information within *K* hops in FIB table of Global Nox, then Global Nox ask Open-vSwitch modifies the CIU_routing message to Interest message and sends to corresponding output port. The *K* hops is calculated by equation (4). Otherwise, the interest packet will be sent to the output port towards the content server.

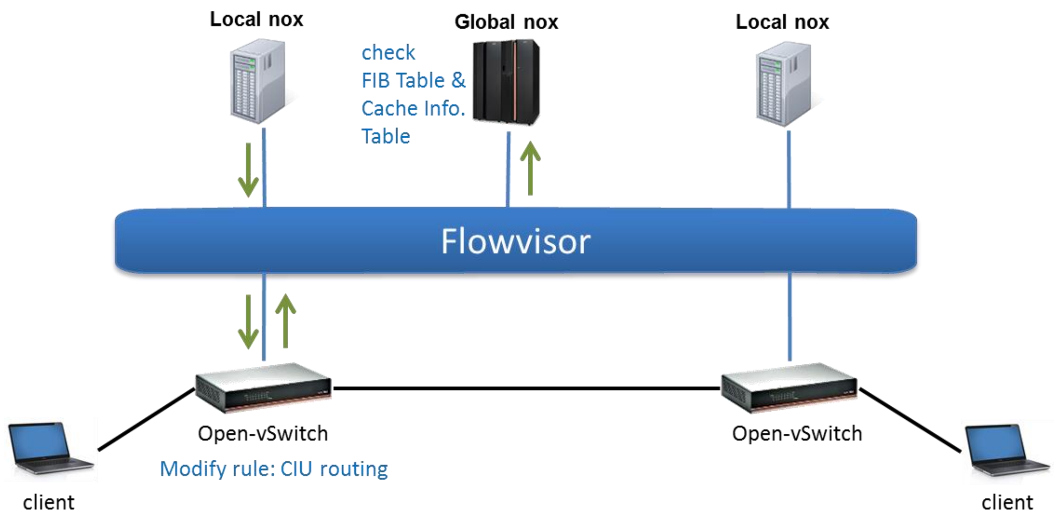


Figure 4-7 Retrieve routing information from Global Nox

The server send the content back when receives request as shown in Figure 4-8. The

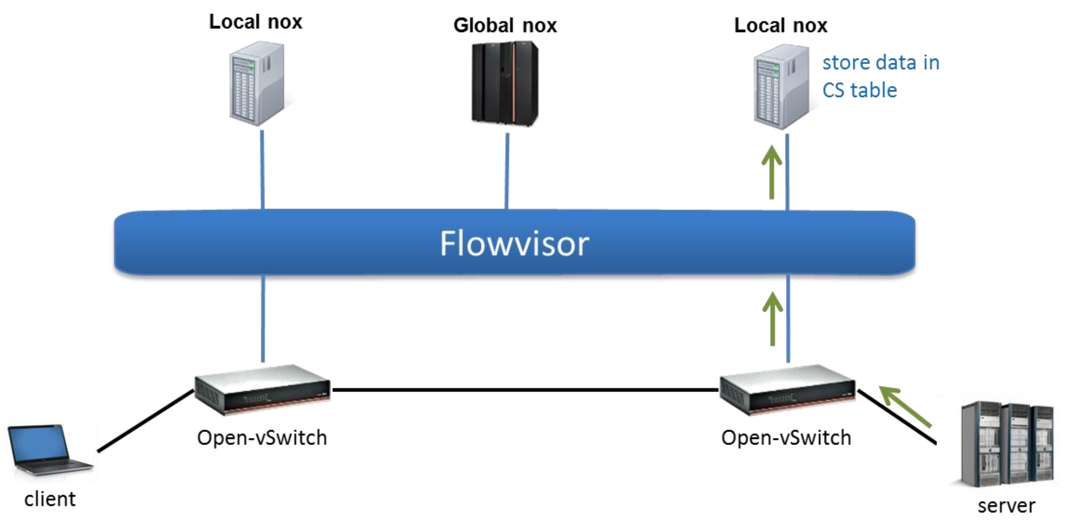


Figure 4-8 Server send content back

Open-vSwitch sends the content to Flowvisor, and then Flowvisor forward the content to corresponding Local Nox. In our CIU scheme, if data is stored in CS Table in Local Nox, the Local Nox will send a cache information packet to Global Nox as shown in Figure 4-9. If the CS table is full and the replacement occurs, the name of the removed data will be store in the payload of the packet of cache information. When Global Nox receives the cache information packet, it will store the cached name and DPID of Open-vSwitch in FIB table. If there is the name of the removed data in payload, the cached name will be removed in FIB table. Besides, The Local Nox will store the data in CS table and then send data back according to the output port of PIT table as shown in Figure 4-10. Furthermore, our DAAC scheme will send an ACK

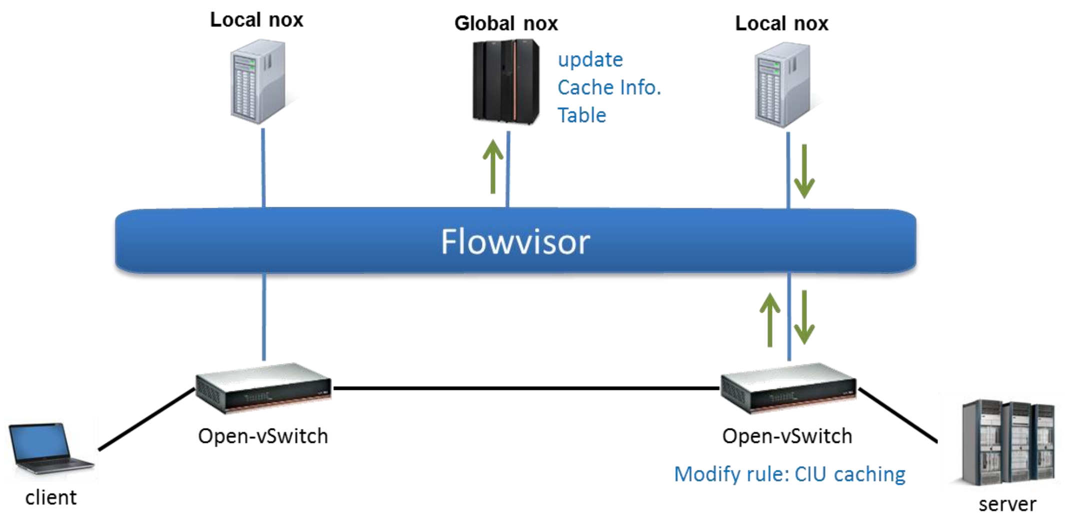


Figure 4-9 Send cache information to Global Nox in CIU scheme

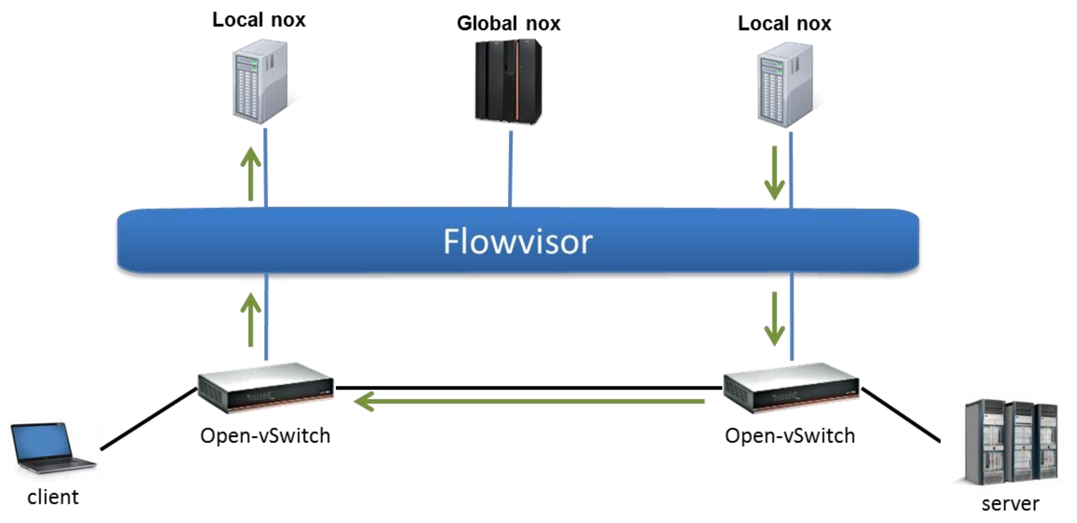


Figure 4-10 Send content back to downstream router

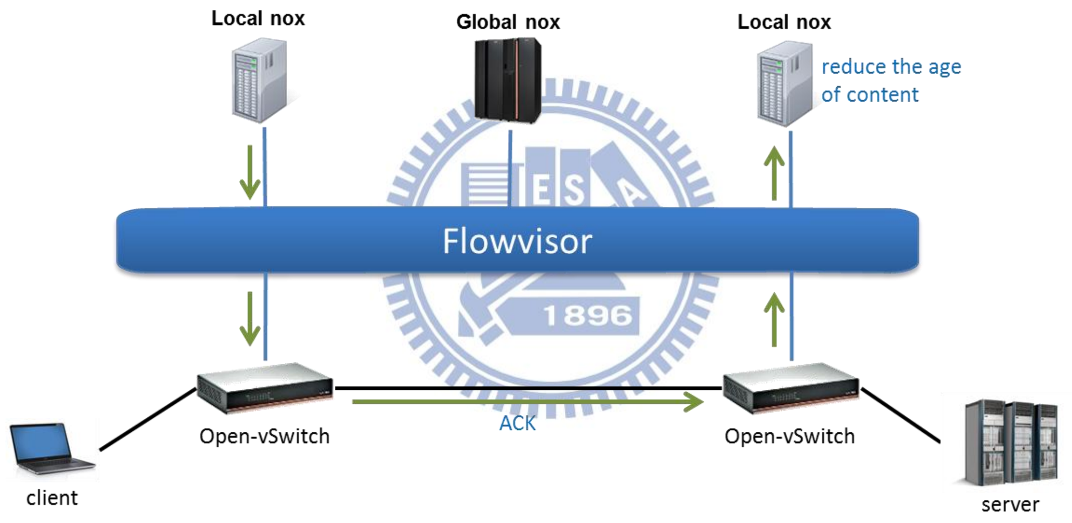


Figure 4-11 Downstream router send ACK back

back to upstream router to reduce the age of content when downstream router caches the same content as shown in Figure 4-11.

Chapter 5 Performance Evaluation

We conduct a series of simulations under different network scenarios to evaluate DAAC with other caching schemes in CCN. Subsection *5.1* introduces the simulation settings and results. Subsection *5.2* show the Emulation scenario and results.

5.1 Simulation

We evaluate the performance of our scheme using ccnSim [24]. CcnSim is a Content Centric Networks simulator in the OMNet++ framework. The implementation of ccnSim is based on the architecture of CCN in [1], and the routing path based on a default routing path which calculate by Dijkstra’s algorithm.

We evaluate our dynamic age-adjusted scheme with two other caching schemes: ABC and LRU. We also implement two kinds of dynamic age-adjusted schemes, DAAC and DAAC_CIU, where DAAC_CIU is the DAAC scheme with cache information update scheme (CIU). The parameters of ABC and DAAC are as follows: *BASE_AGE* is set to be 50 second and *MAX_AGE* set to 60 second. To enlarge the effect of ABC method, we multiple 5 times age when pass a hop. There are a total of 1000 files in CCN and the number of the chunks for a file follows the geometric distribution with an average of 10. The chunk size is 6.4Kbytes and the cache size of each node is from 50 to 250 chunks (5 files to 25 files). Each router generates request interests according to Poisson process with intensity $\lambda = 50$ requests per second. Each request selects a file according to Zipf-Mandelbrot distribution [25]. Zipf-Mandelbrot distribution can be used to imitate the data access pattern in the content distributed services such as video on demand, IPTV, web caching, etc. The distribution equation is shown as follows.

$$Zipf(\alpha) = \frac{K}{(i+q)^{\alpha}} \tag{5}$$

where α is the distribution parameter, *q* is the shift parameter, *i* is the rank of data and *K* is the normalizing constant. Zipf-Mandelbrot distribution can reflect the different skew of the data access pattern in the network. When α is going higher, the distribution of the content access requests becomes steeper, and vice versa. Our simulations set $\alpha = 0.9$ 、$\alpha = 1$ and $\alpha = 1.1$. All parameters can be summarized as Table 3.

Parameters	Value
Number of File Request	1000 per node
Poisson Intensity	50 request/s
Distribution	Zipf's law
Skew(α)	0.9,1,1.1
Number of File	1000
File Size	10 chunks
Chunk Size	6.4kB
Cache Size	50~250 chunks per node
Base Age	50s
Max Age	60s

Table 3 Simulation Parameters

There are three performance metrics: hit ratio, server loading and average hop count.

- Hit Ratio: the ratio of the number of interest served by caches to the total number of interest arrived at caches.
- Server loading: the number of the interests served by content server.
- Average hop count: the average number of hops that content takes to reach the requester

We use three different network topologies in our simulation: GEANT as in Figure 5-1 and 3-ary tree structure as in Figure 5-2, Abilene as in Figure 5-3. GEANT is a realistic pan-European topology and tree structure is a synthetic 3-ary tree. In GEANT topology, there are 22 nodes and node 12 is chosen as the content server. All of the other nodes can send requests except the content server. In 3-ary tree topology, there are 40 nodes and only leaf nodes send request (i.e., nodes 13-39). The root of the tree (i.e., node 0) is chosen as the content server. In Abilene topology, there are 11 nodes and node 2 is chosen as the content server. The simulation result of Abilene topology is shown in subsection *5.2* with emulation result.

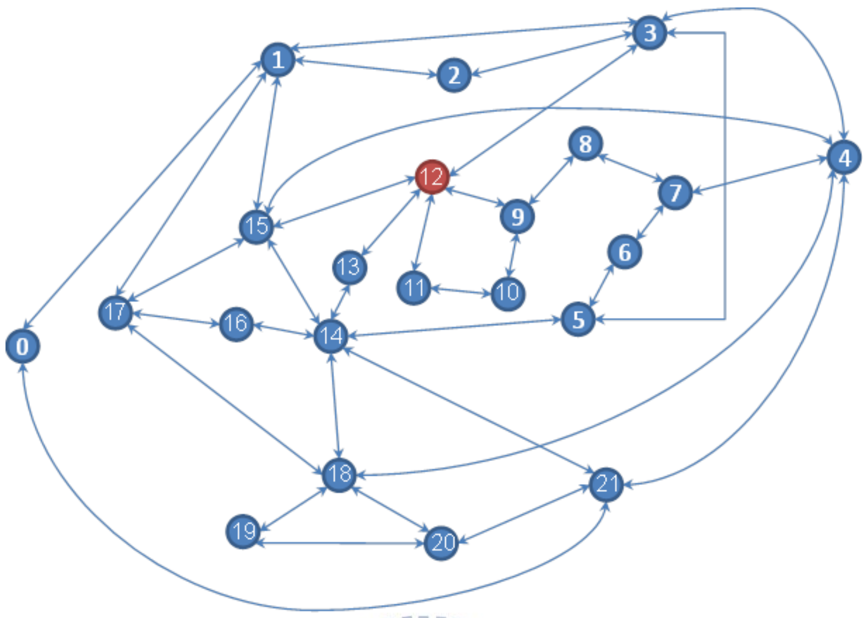


Figure 5-1 GEANT Topology

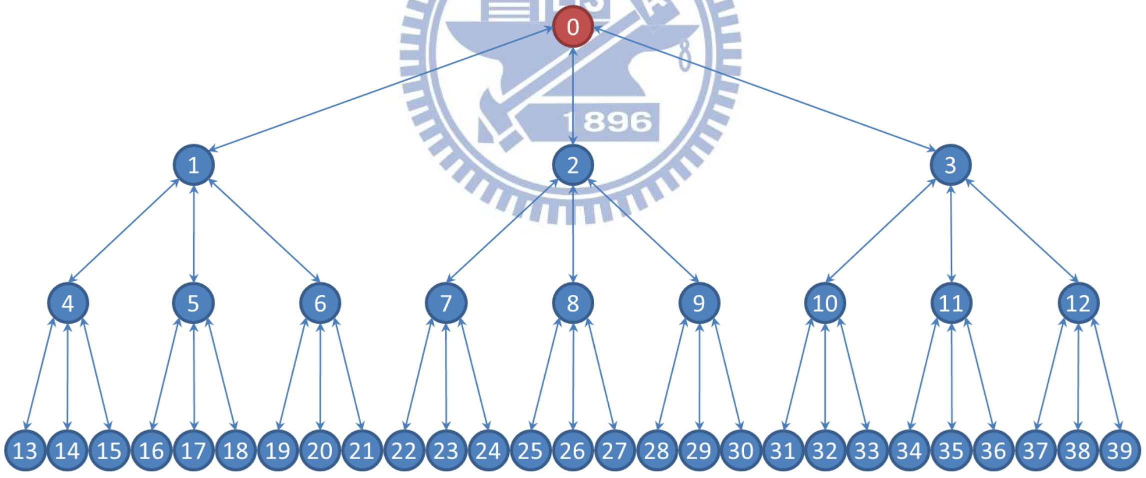


Figure 5-2 3-ary Tree Topology

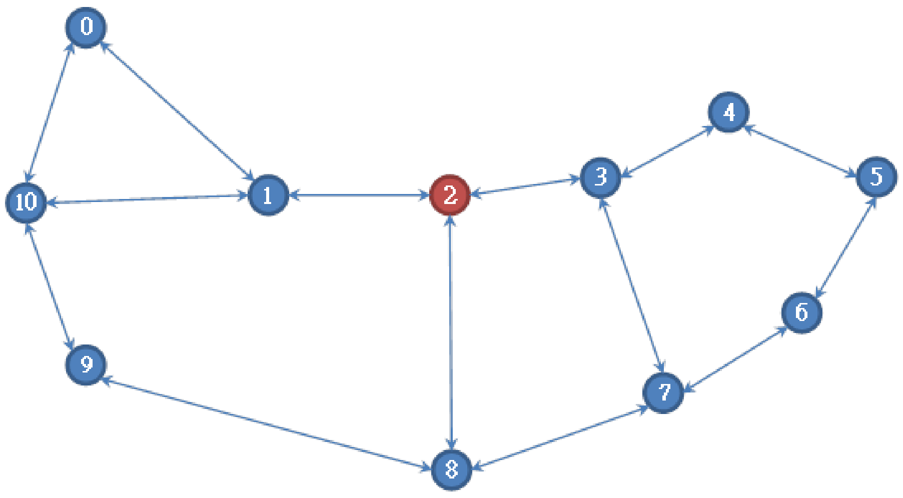


Figure 5-3 Abilene Topology

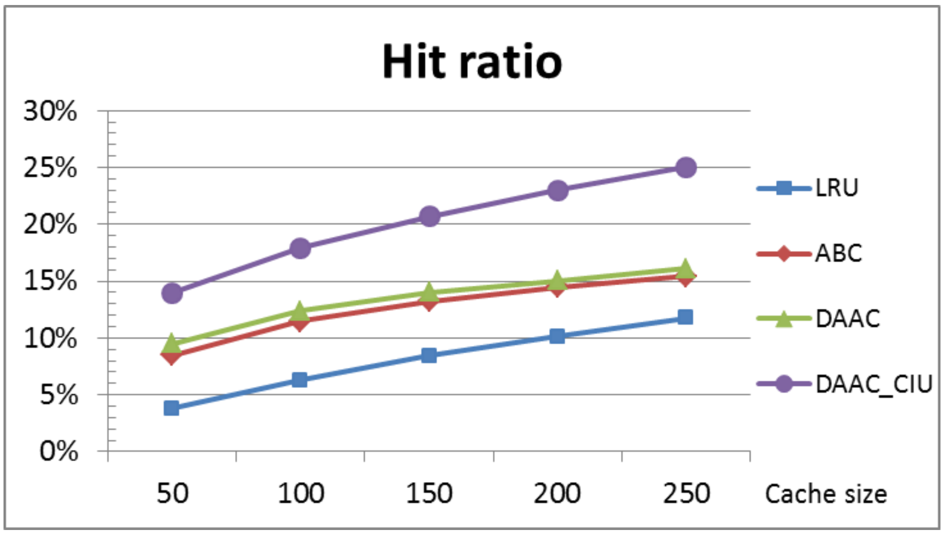


Figure 5-4 Hit ratio of GEANT Topology ($\alpha = 0.9$)

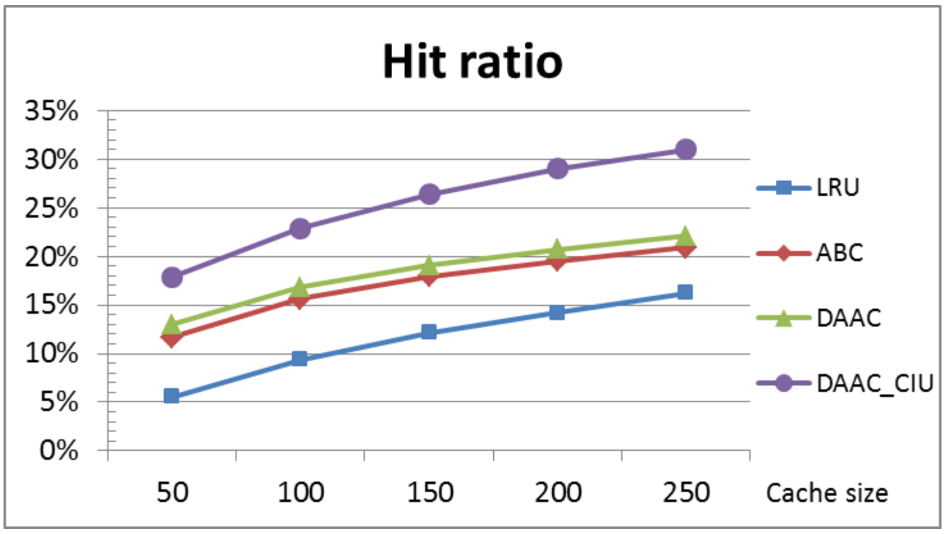


Figure 5-5 Hit ratio of GEANT Topology ($\alpha = 1$)

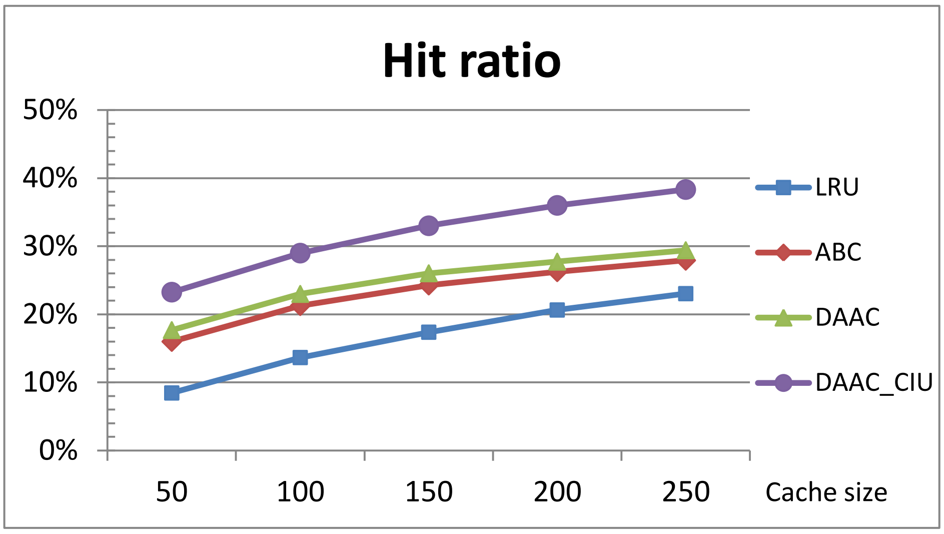


Figure 5-6 Hit ratio of GEANT Topology ($\alpha=1.1$)

Figure 5-4、Figure 5-5 and Figure 5-6 shows the simulation results under the GEANT topology with 3 different distribution parameters $\alpha=0.9$ 、 $\alpha=1$ and $\alpha=1.1$ respectively and which also show the cache hit ratio of all the caching schemes under different cache sizes. DAAC achieves a hit-ratio difference of 0.7%-1.0% ($\alpha=0.9$)、1.1%-1.3% ($\alpha=1$) and 1.5%-1.7% ($\alpha=1.1$) compared to ABC. DAAC_CIU achieves a hit-ratio difference of 5.5%-9.6% ($\alpha=0.9$)、6%-10% ($\alpha=1$) and 7.3%-10.4% ($\alpha=1.1$) compared to ABC. Since both DAAC and DAAC_CIU use the age-refresh scheme to increase the age of a cache being recently hit, they have a higher probability to have a cache. Since DAAC_CIU provides the content's lately caching location information, it performs best in cache hit ratio. Moreover, hit ratio in $\alpha=1.1$ is one order of magnitude better than when $\alpha=1$ and $\alpha=0.9$ for our schemes. Since our algorithm relay on the temporal locality, the content requests distribute in steeper, the hit-ratio improvement is much higher.

Figure 5-7、Figure 5-8 and Figure 5-9 show the server loadings of all the caching schemes under different cache sizes. DAAC reduces server loading by about 1.0-2.8% ($\alpha=0.9$)、2%-3.4% ($\alpha=1$) and 2.6%-4.0% ($\alpha=1.1$) compared to ABC, and DAAC_CIU also reduces server loading by about 18.1%-30.5% ($\alpha=0.9$)、20%-33% ($\alpha=1$) and 25%-35.1% ($\alpha=1.1$) compared to ABC. Since most of the requests are redirected to the near-by routers rather than to the content server, DAAC_CIU can reduce server loading much better than other schemes.

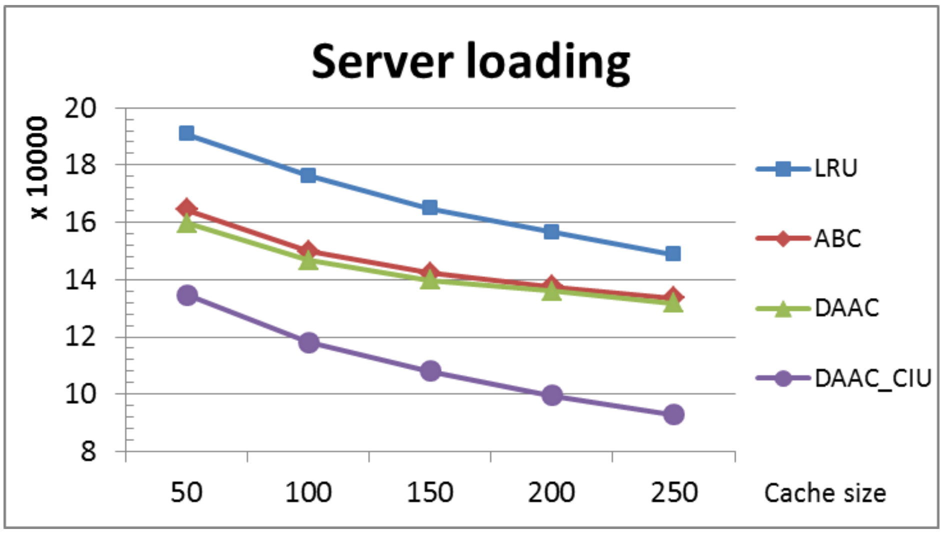


Figure 5-7 Server loading of GEANT Topology ($\alpha = 0.9$)

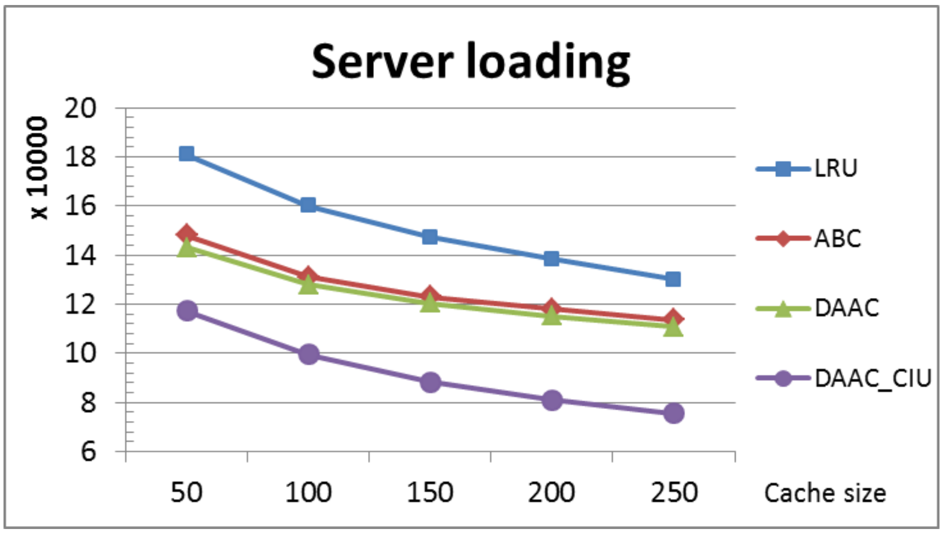


Figure 5-8 Server loading of GEANT Topology ($\alpha = 1$)

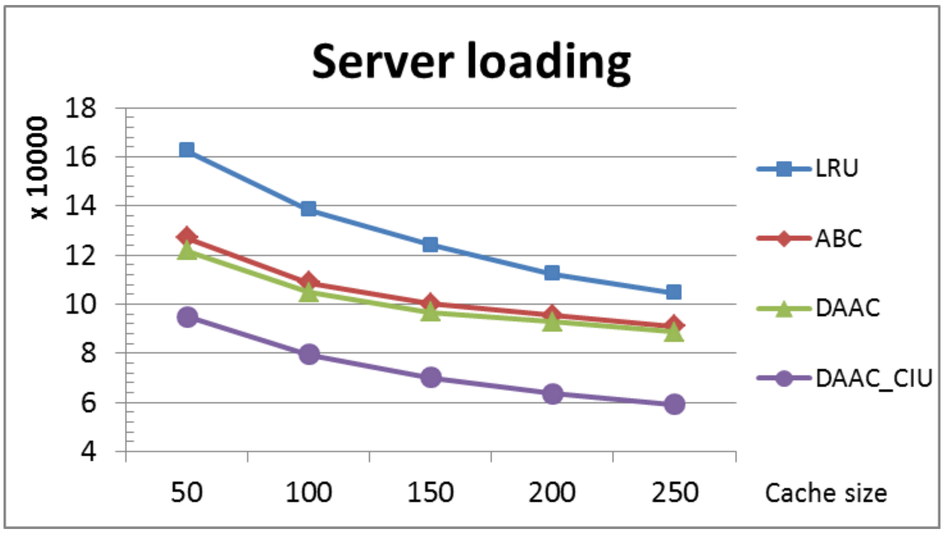


Figure 5-9 Server loading of GEANT Topology ($\alpha = 1.1$)

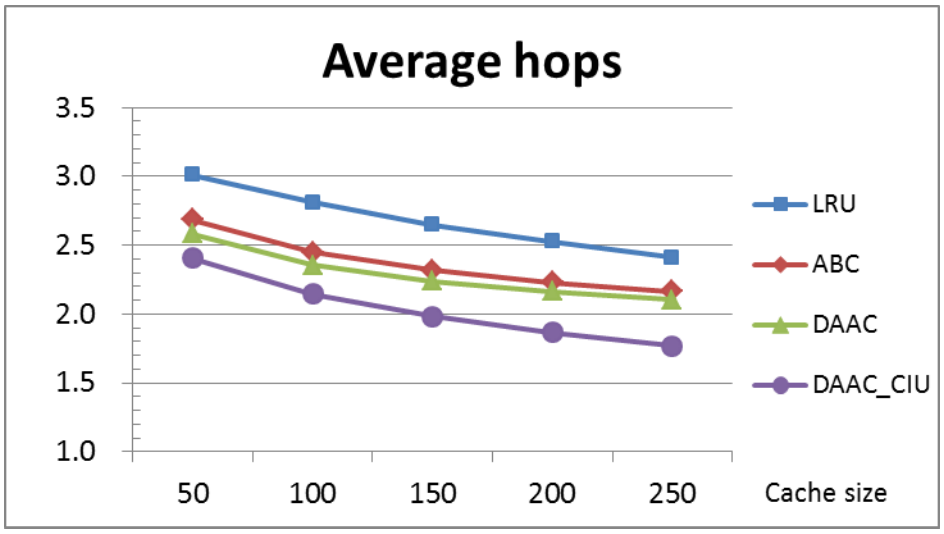


Figure 5-10 Average hop count of GEANT Topology ($\alpha = 0.9$)

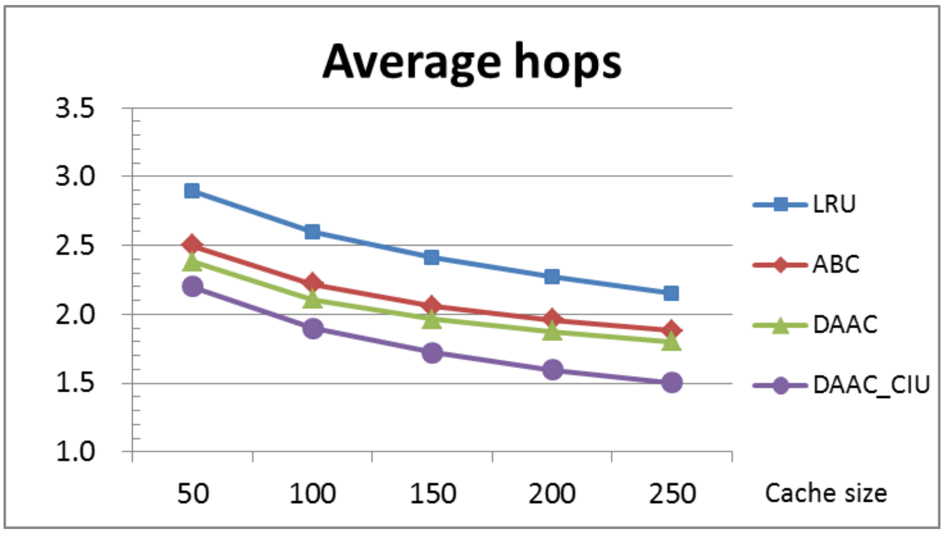


Figure 5-11 Average hop count of GEANT Topology ($\alpha = 1$)

Figure 5-10、Figure 5-11 and Figure 5-12 show the average hop count of all the caching schemes under different cache sizes. DAAC reduce about 3%-3.9% ($\alpha = 0.9$)、4.2%-4.9% ($\alpha = 1$) and 5.2%-6.2% ($\alpha = 1.1$) in average hop count compared to ABC. DAAC_CIU reduce about 10.5%-18.4% ($\alpha = 0.9$)、11.9%-19.9% ($\alpha = 1$) and 14%-21.5% ($\alpha = 1.1$) in average hop count with comparison to ABC. Since DAAC and DAAC_CIU increase the average hit ratio, it has a more chance to obtain contents from the intermediate routers rather from the content server. Thus, DAAC and DAAC_CIU have less average hop count. Since average hop count can also reflect bandwidth consumption per request, DAAC_CIU can enjoy the least bandwidth consumption in the network. The results in the figures show that

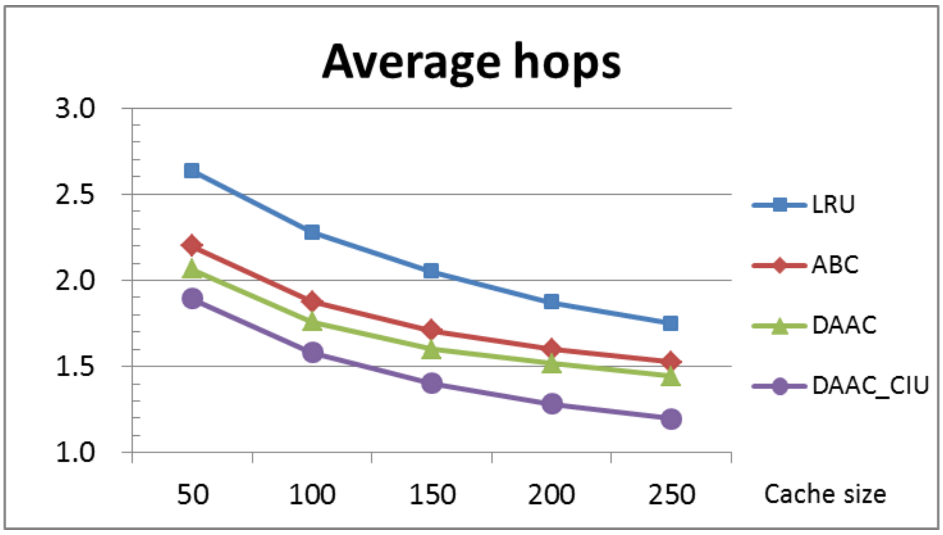


Figure 5-12 Average hop count of GENAT Topology ($\alpha = 1.1$)

there are significant gains through using DAAC and CIU. The content consumers will benefit from lower download latency. On the other hand, content providers will be able to greatly reduce server loading as well as network bandwidth consumption.

We also evaluate the broadcast overhead and average hop distance of CIU mechanism with the other two information update mechanisms: *ALL* and *1-hop*, where *ALL* is to broadcast the cache update information to all nodes and *1-hop* is to broadcast the cache update information within 1-hop. As shown in Figure 5-13, the average hop count in *ALL* is almost same as the CIU scheme. However, in Figure 5-14, the broadcast overhead (i.e., the total number of the messages of the cache update information) in *ALL* is 1.4-17 times higher than

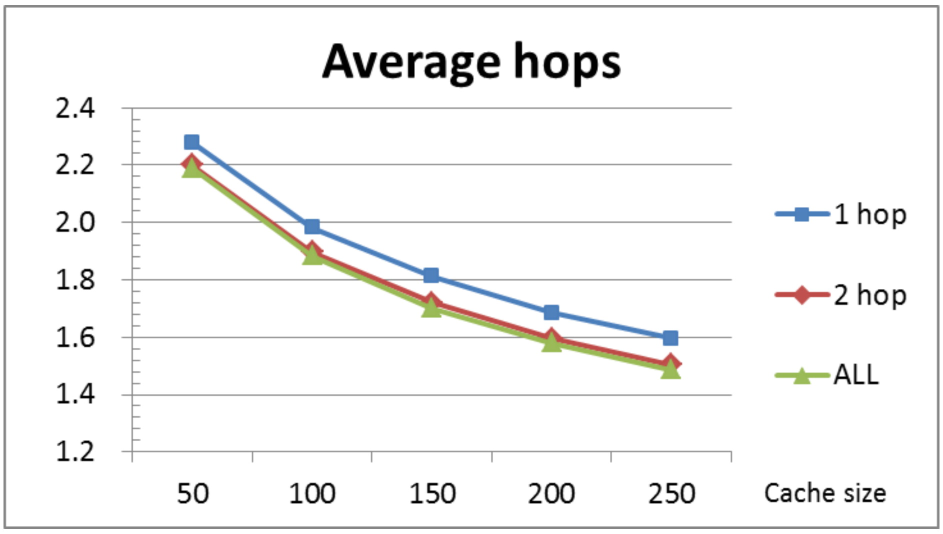


Figure 5-13 Average hop count of Broadcast Distance

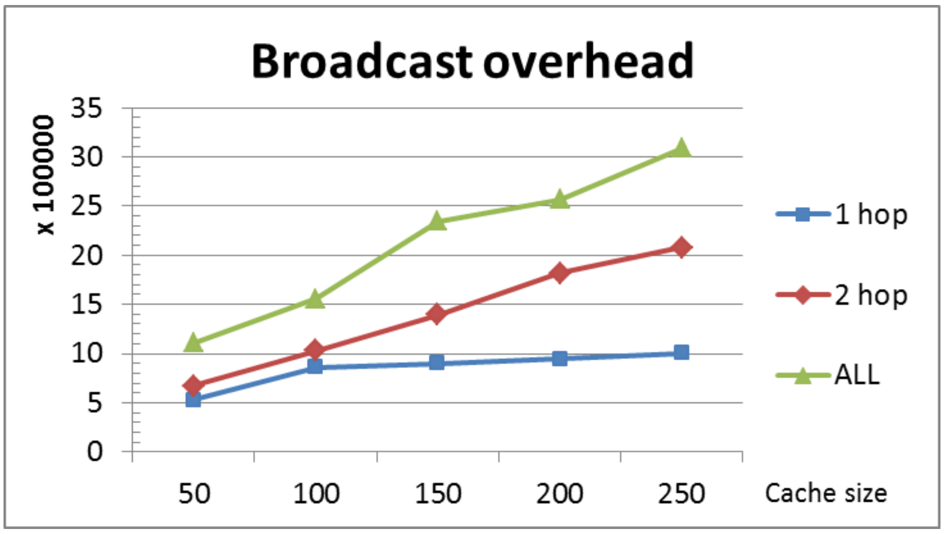


Figure 5-14 Broadcast overhead

the CIU scheme. The CIU can achieve a lower average hop count than *1-hop* with price on moderate increase of broadcast overhead. According to Figure 5-13 and Figure 5-14, the CIU scheme can achieve a balance between the average hop count and the broadcast overhead.

Figure 5-15、Figure 5-16 and Figure 5-17 shows the simulation results under the 3-ary tree topology. We can observe that the performance of DAAC and DAAC_CIU are performing better in all performance metrics. The hit ratio of DAAC_CIU is the highest. The difference of the hit ratio between DAAC and ABC is about 0.8%-1.1%、1.1%-1.4% and 1.7%-2% for $\alpha = 0.9$、$\alpha = 1$ and $\alpha = 1.1$ respectively. On the other hand the difference of the hit ratio between DAAC_CIU and ABC is about 2.6%-4.6%、3.1%-4.8% and 3.4%-5.3% for $\alpha = 0.9$、$\alpha = 1$ and $\alpha = 1.1$ respectively.

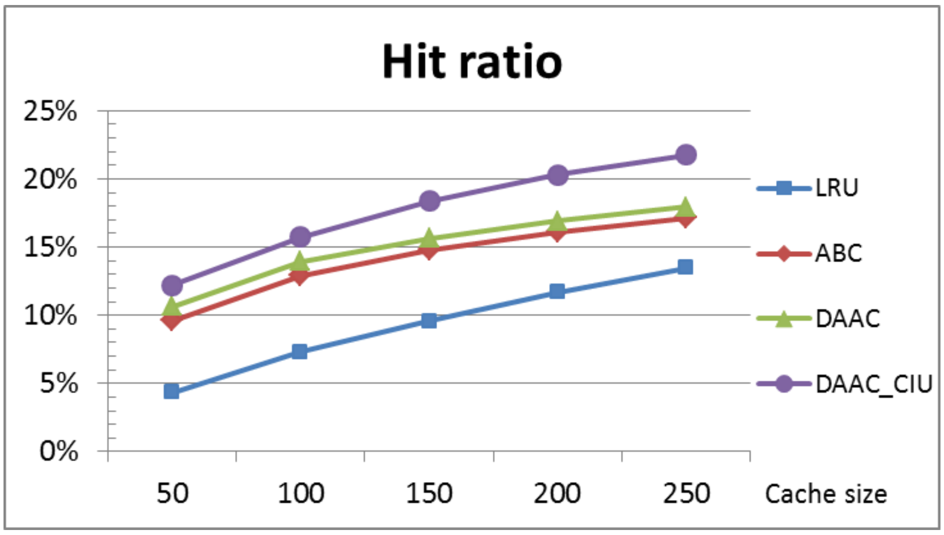


Figure 5-15 Hit ratio of 3-ary Tree Topology ($\alpha = 0.9$)

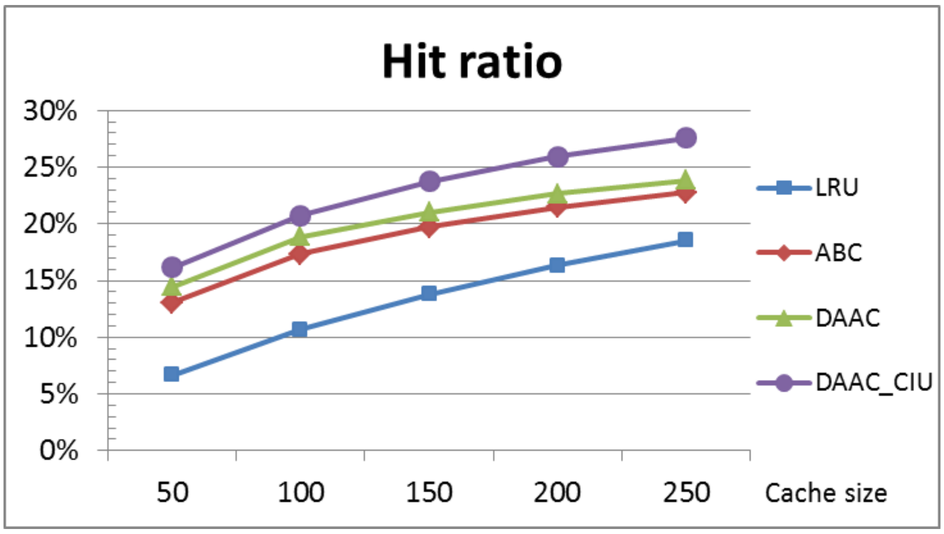


Figure 5-16 Hit ratio of 3-ary Tree Topology ($\alpha = 1$)

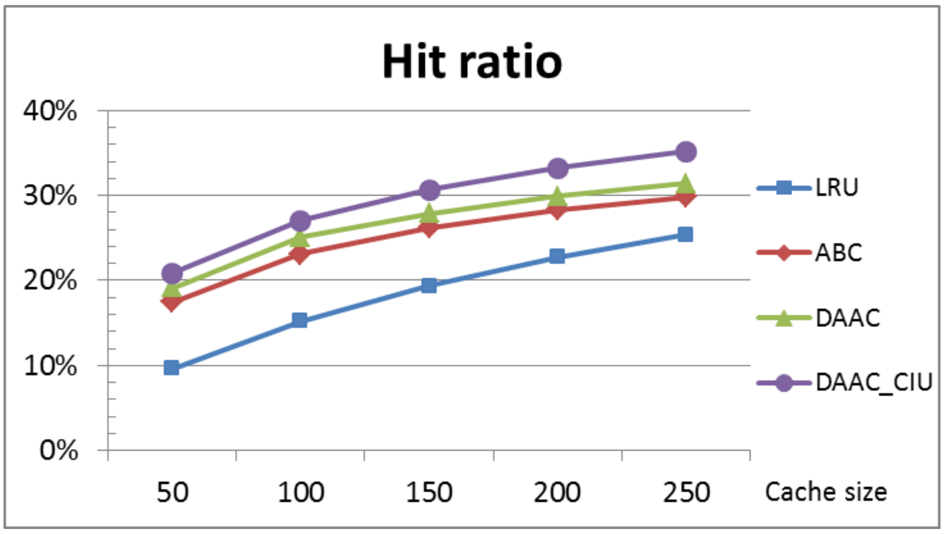


Figure 5-17 Hit ratio of 3-ary Tree Topology ($\alpha = 1.1$)

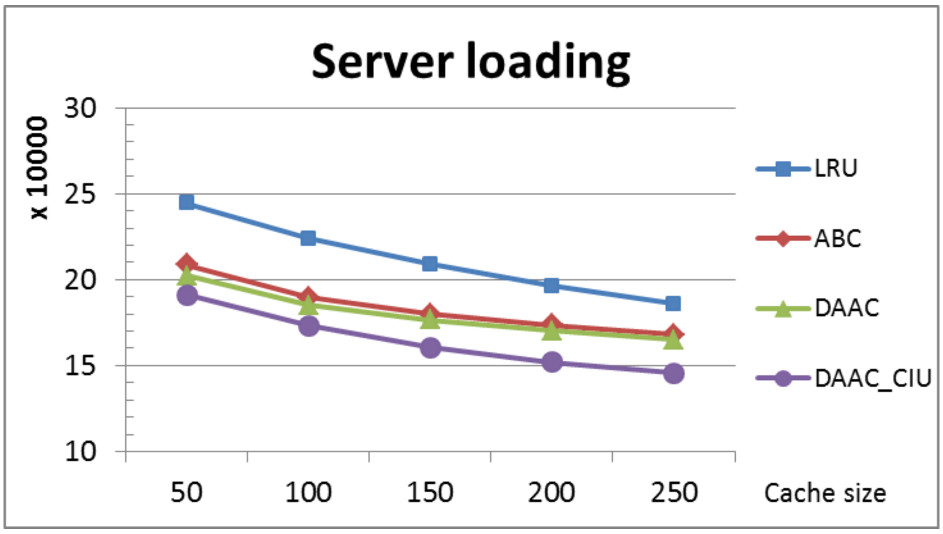


Figure 5-18 Server loading of 3-ary Tree Topology ($\alpha = 0.9$)

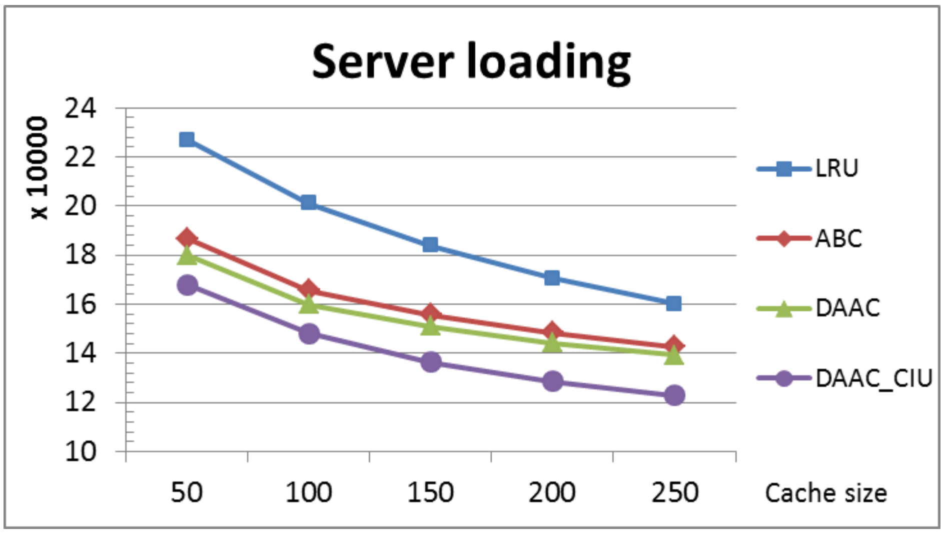


Figure 5-19 Server loading of 3-ary Tree Topology ($\alpha = 1$)

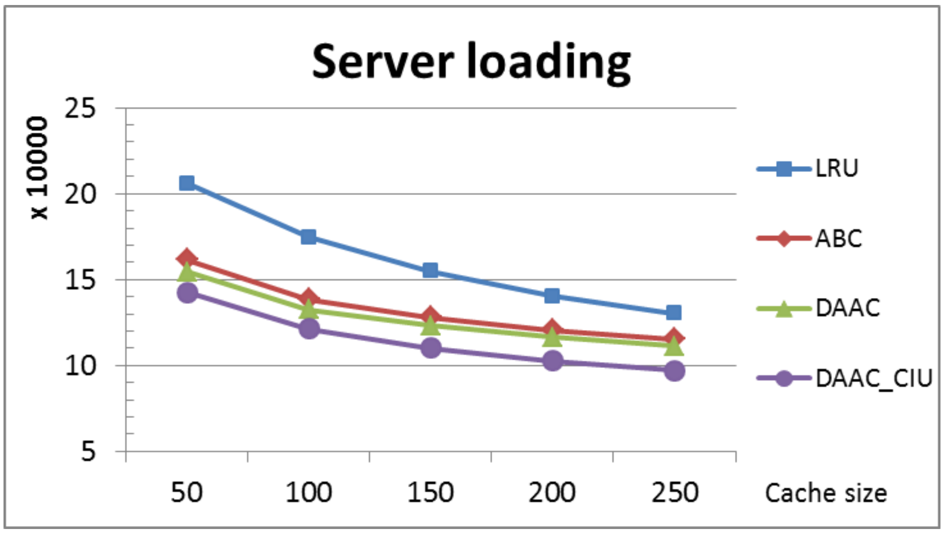


Figure 5-20 Server loading of 3-ary Tree Topology ($\alpha = 1.1$)

In Figure 5-18、Figure 5-19 and Figure 5-20, DAAC_CIU can reduce the server loading by about 8%-13.4% ($\alpha = 0.9$)、10.1%-14% ($\alpha = 1$) and 11.8%-15.7% ($\alpha = 1.1$) in comparison to ABC.

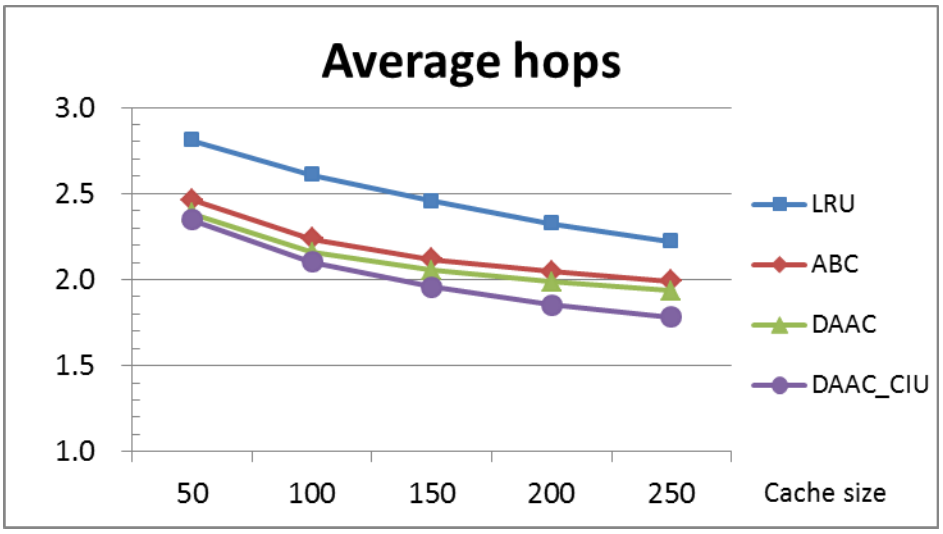


Figure 5-21 Average hop count of 3-ary Tree Topology ($\alpha = 0.9$)

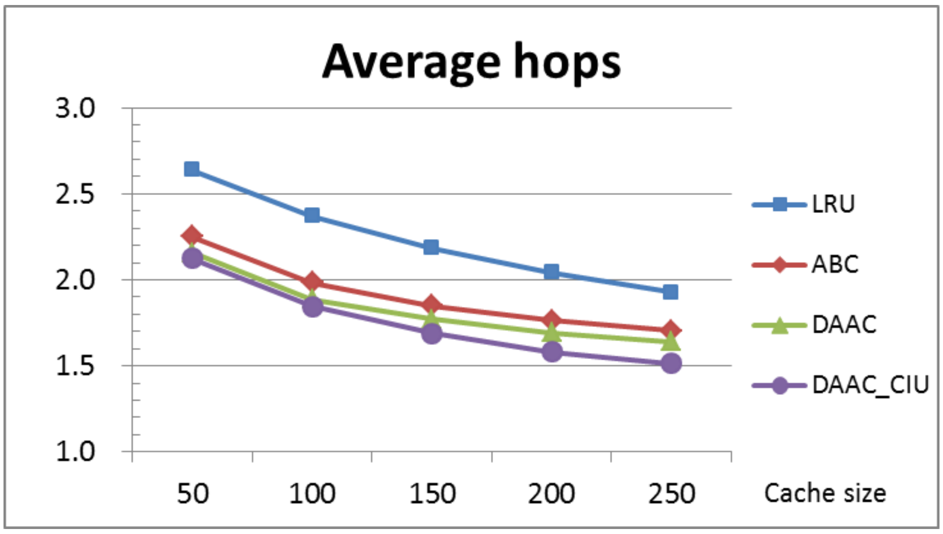


Figure 5-22 Average hop count of 3-ary Tree Topology ($\alpha = 1$)

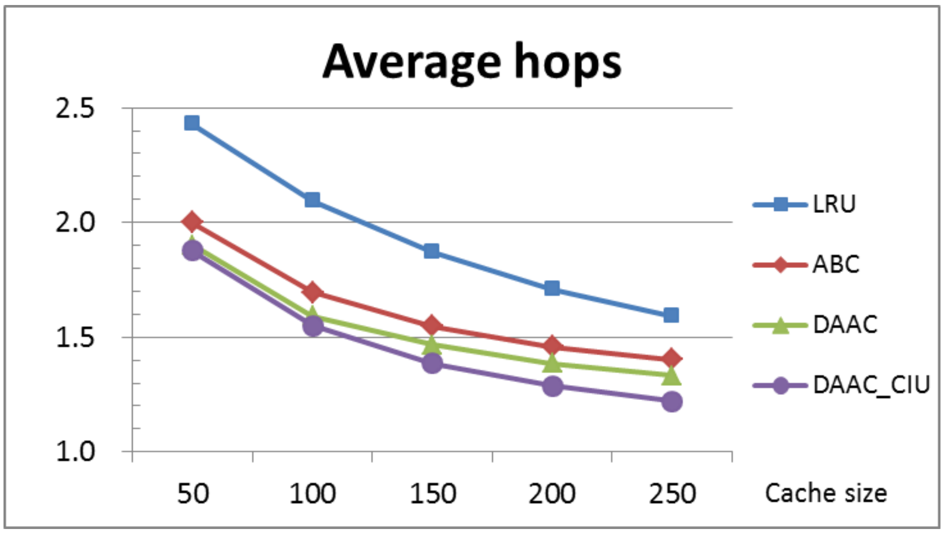


Figure 5-23 Average hop count of 3-ary Tree Topology ($\alpha = 1.1$)

DAAC_CIU scheme can reduce the hop count approximately 4.6%-12.8% compared to ABC. Finally DAAC also gained 2.9%-6% reduction of the average hop count as shown in Figure 5-21、Figure 5-22 and Figure 5-23.

5.2 Emulation

In our scenario of emulation is similar with simulation, the parameters of ABC and DAAC are as follows: *BASE_AGE* is set to be 50 second and *MAX_AGE* set to 60 second. The age is multiple 5 times age when pass a hop same as simulation. There are a total of 1000 files in CCN and the number of the chunks for a file follows the geometric distribution with an average of 10. The chunk size is 6.4Kbytes and each chunk contains 10 packets. The cache size of each node is from 50 to 250 chunks (5 files to 25 files). Each router generates request interests according to Poisson process with intensity $\lambda =$ 50 requests per second. Each request selects a file according to Zipf-Mandelbrot distribution and the $\alpha = 0.9$ 、 $\alpha = 1$ and $\alpha = 1.1$. All parameters can be summarized as Table 4. The Abilene topology in our Emulation is shown in Figure 5-24.

Parameters	Value
Number of File Request	1000 per node
Poisson Intensity	50 request/s
Distribution	Zipf's law
Skew(α)	0.9,1,1.1
Number of File	1000
File Size	10 chunks
Chunk Size	6.4kB
Cache Size	50~250 chunks per node
Base Age	50s
Max Age	60s

Table 4 Emulation Parameters

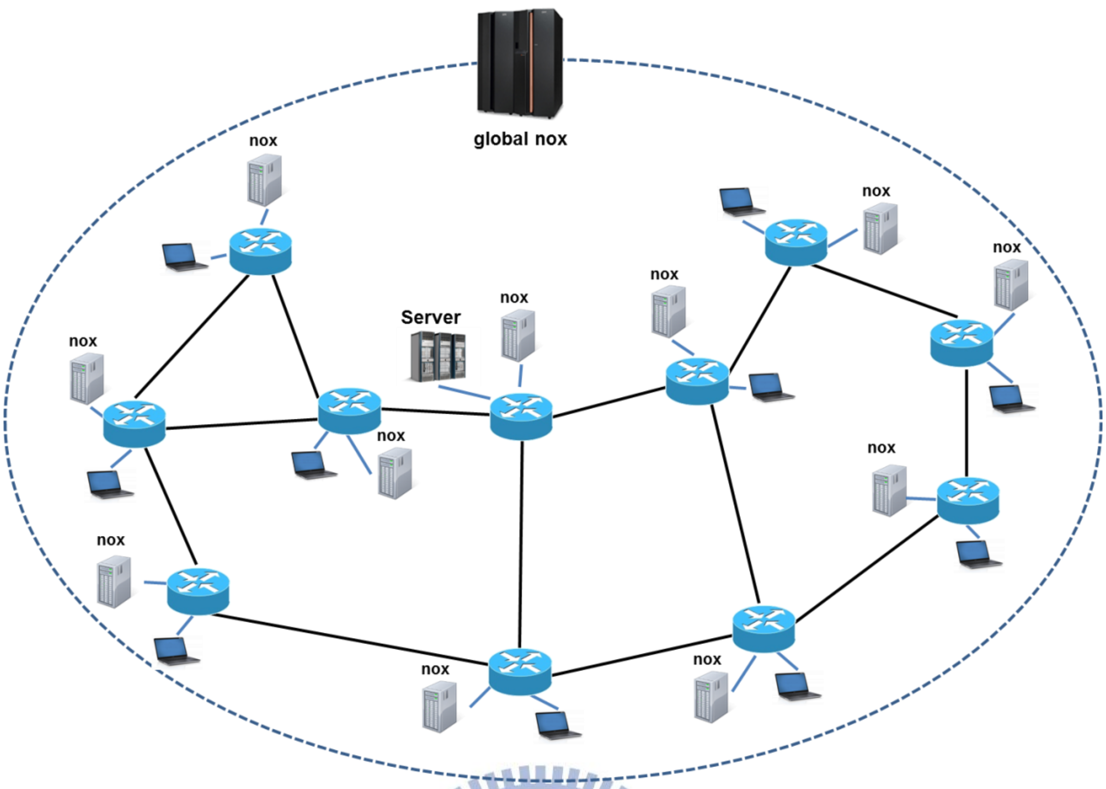


Figure 5-24 Abilene Topology in Emulation

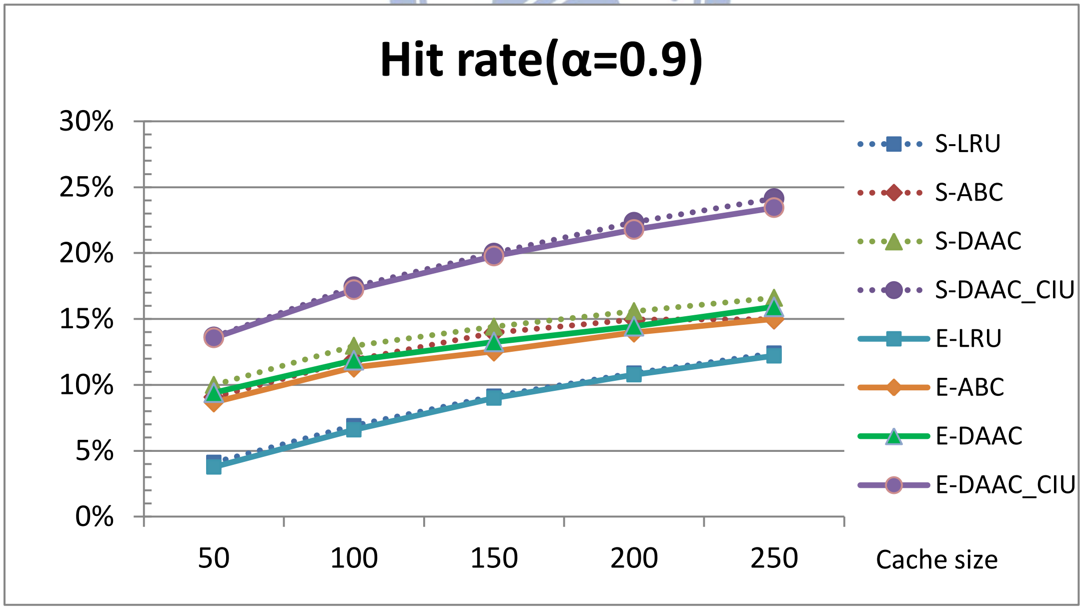


Figure 5-25 Hit ratio of Abilene Topology ($\alpha = 0.9$)

In Figure 5-25 shows the simulation and emulation results under the Abilene topology and also show the cache hit ratio of all the caching schemes under different cache sizes. The simulation result is shown in dot line and emulation result is shown in solid line.

DAAC achieves a hit-ratio difference of 4.2-6.1% and 0.5-1.6% compared to LRU and ABC respectively. DAAC_CIU achieves a hit-ratio difference of 9.5%-11.7% and 4.6%-9.2% compared to LRU and ABC.

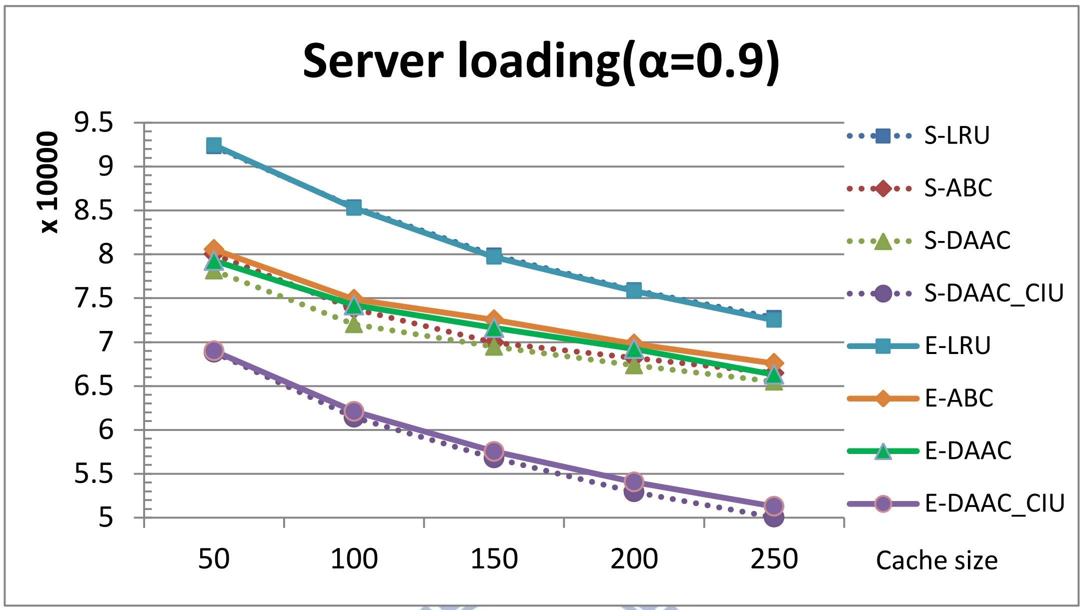


Figure 5-26 Server loading of Abilene Topology ($\alpha = 0.9$)

Figure 5-26 show the server loadings of all the caching schemes under different cache sizes. DAAC reduces server loading by about 10%-15.6% and 0.7%-2.4% compared to LRU and ABC, and DAAC_CIU also reduces server loading by about 25.4%-31.2% and 14%-24.7% compared to LRU and ABC. The server loading in emulation is a little higher

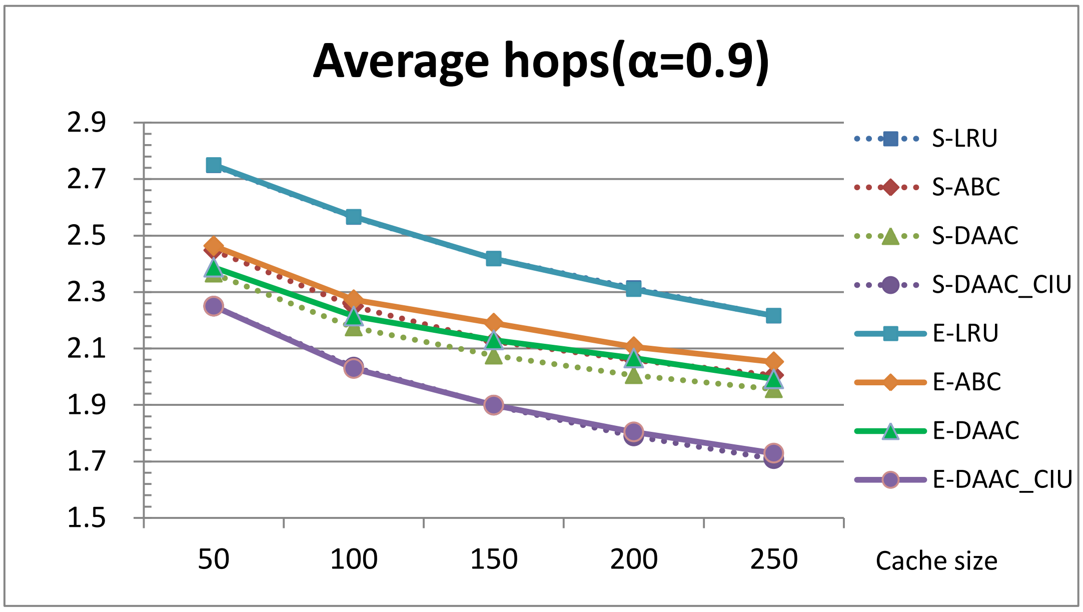


Figure 5-27 Average hop count of Abilene Topology ($\alpha = 0.9$)

than simulation is because the emulation is sent the real packet data.

Figure 5-27 show the average hop count of all the caching schemes under different cache sizes. DAAC reduce about 11.7%-15.2% and 2.4%-3.4% in average hop count compared to LRU and ABC. DAAC_CIU reduce about 18.1%-22.8% and 8.1%-14.8% in average hop count with comparison to LRU and ABC.

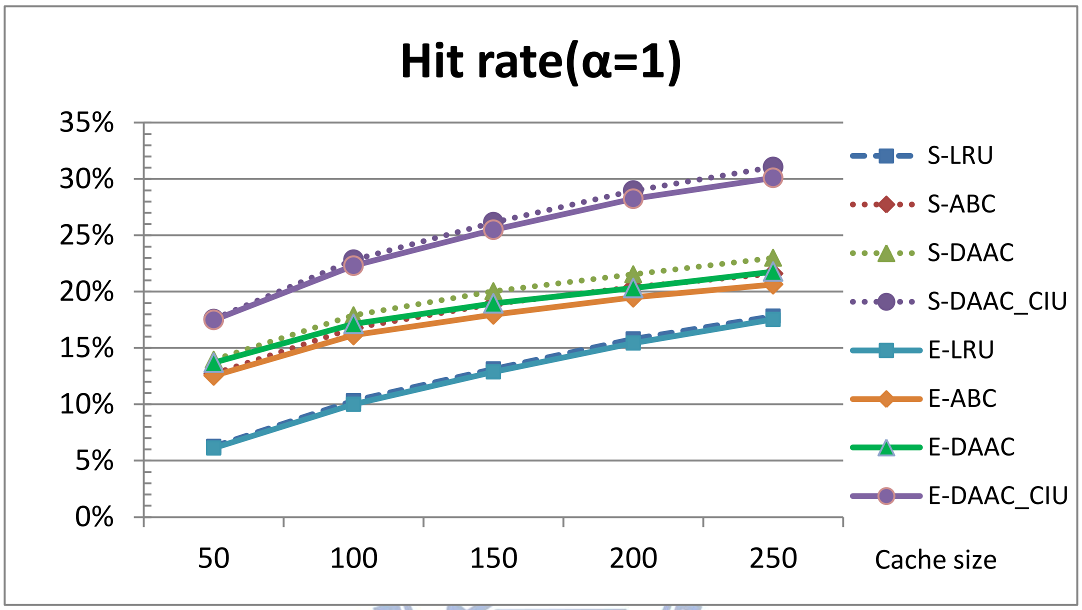


Figure 5-28 Hit ratio of Abilene Topology ($\alpha = 1$)

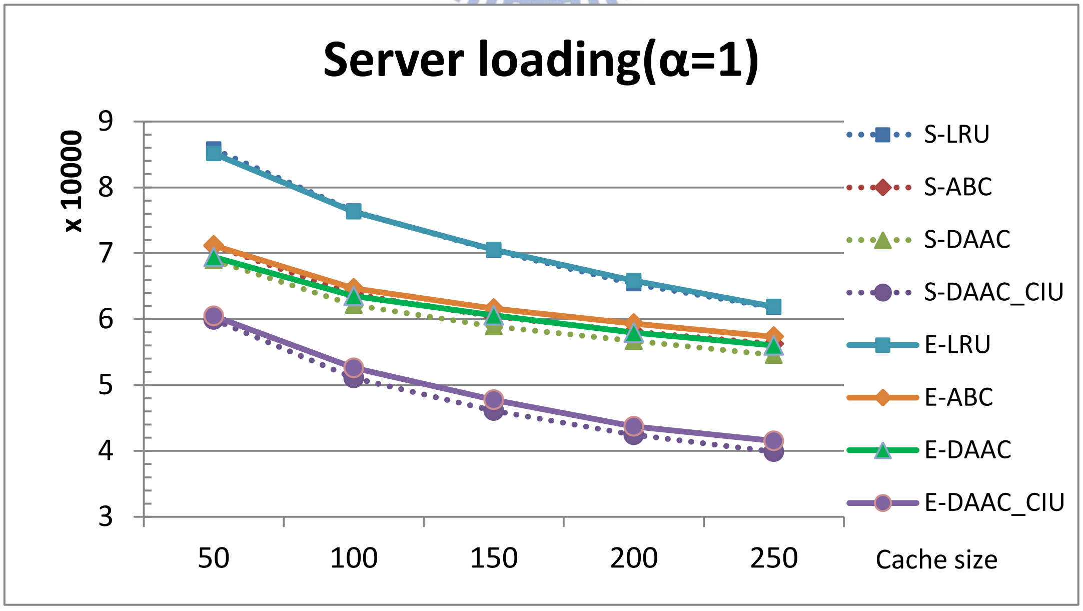


Figure 5-29 Server loading of Abilene Topology ($\alpha = 1$)

In Figure 5-28 shows show the cache hit ratio of all the caching schemes under different cache sizes. DAAC achieves a hit-ratio difference of 5.2-7.7% and 1.1-1.4% compared to LRU and ABC respectively. DAAC_CIU achieves a hit-ratio difference of 11.3%-13.2% and 4.9%-9.5% compared to LRU and ABC.

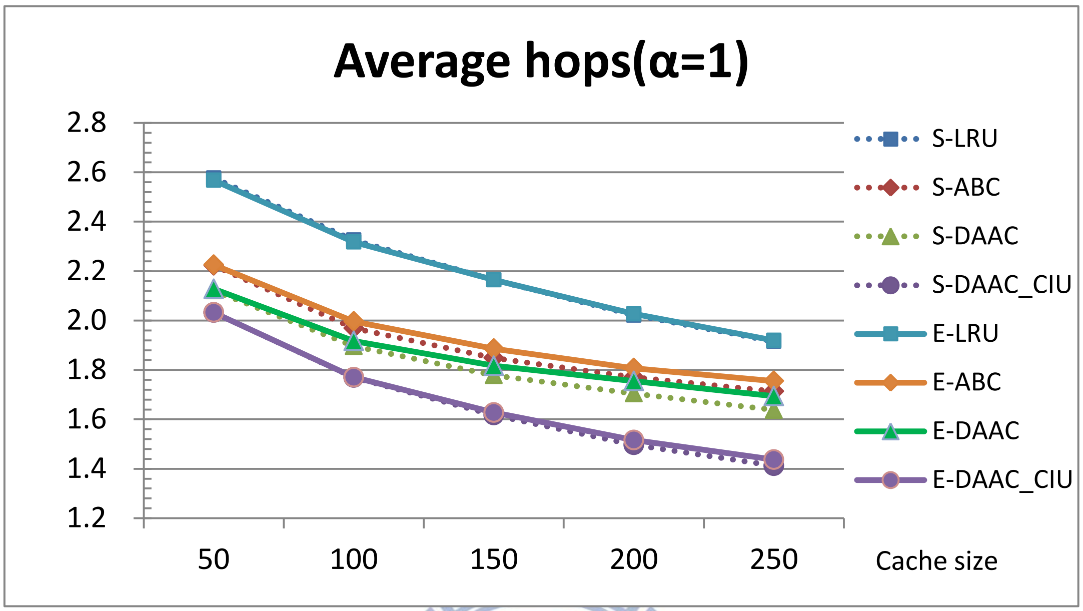


Figure 5-30 Average hop count of Abilene Topology ($\alpha=1$)

Figure 5-29 show the server loadings of all the caching schemes under different cache sizes. DAAC reduces server loading by about 11.7%-19.7% and 2.4%-3.1% compared to LRU and ABC, and DAAC_CIU also reduces server loading by about 30%-35.5% and 15.7%-29.1% compared to LRU and ABC.

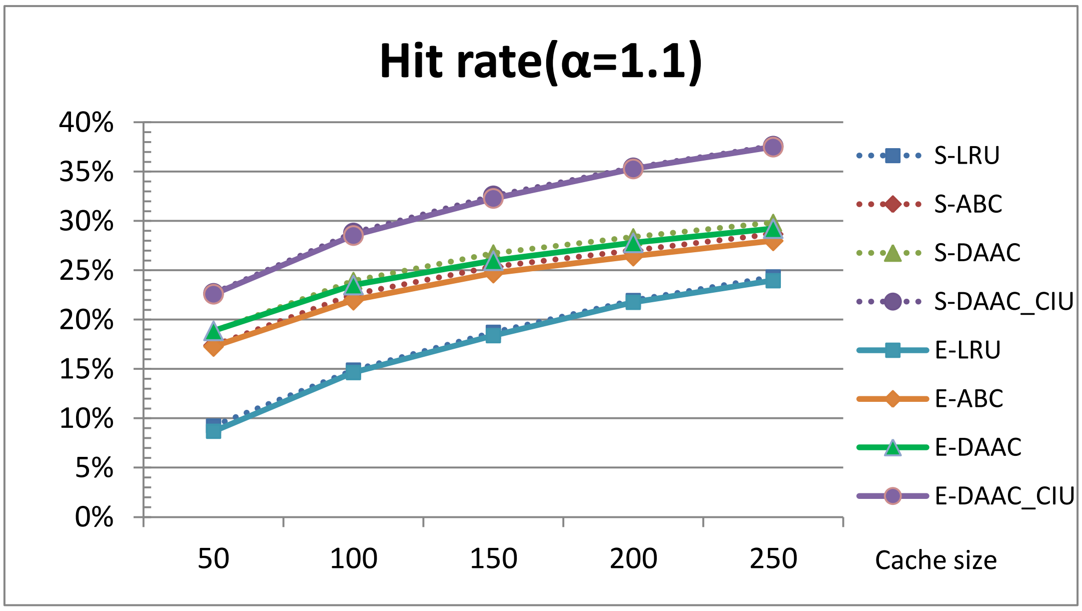


Figure 5-31 Hit ratio of Abilene Topology ($\alpha=1.1$)

Figure 5-30 show the average hop count of all the caching schemes under different cache sizes. DAAC reduce about 14.5%-18.5% and 3.7%-4.4% in average hop count compared to LRU and ABC. DAAC_CIU reduce about 21.1%-26.3% and 8.5%-17.6% in average hop count with comparison to LRU and ABC.

In Figure 5-31 shows show the cache hit ratio of all the caching schemes under different cache sizes. DAAC achieves a hit-ratio difference of 5.5-9.6% and 1.2-1.5% compared to LRU and ABC respectively. DAAC_CIU achieves a hit-ratio difference of 13.3%-13.9% and 5.3%-8.9% compared to LRU and ABC.

Figure 5-32 show the server loadings of all the caching schemes under different cache sizes. DAAC reduces server loading by about 12%-24.2% and 2.2%-3.9% compared to LRU and ABC, and DAAC_CIU also reduces server loading by about 35%-36.7% and 17.5%-28.3% compared to LRU and ABC.

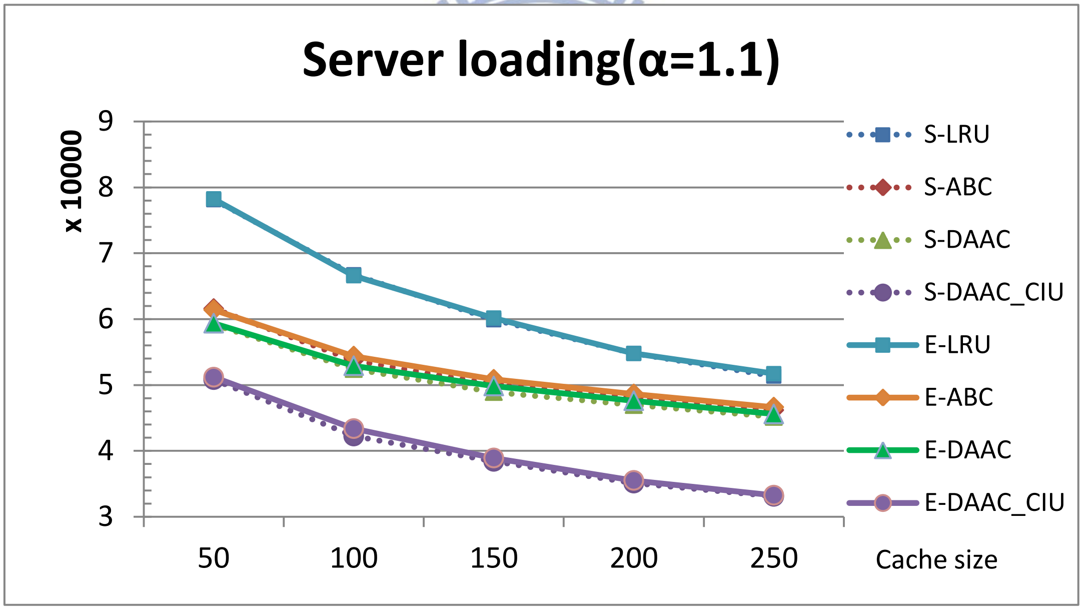


Figure 5-32 Server loading of Abilene Topology ($\alpha = 1.1$)

Figure 5-33 show the average hop count of all the caching schemes under different cache sizes. DAAC reduce about 14.9%-22.2% and 3.8%-4.9% in average hop count compared to LRU and ABC. DAAC_CIU reduce about 25.3%-27.9% and 9.3%-17.9% in average hop count with comparison to LRU and ABC.

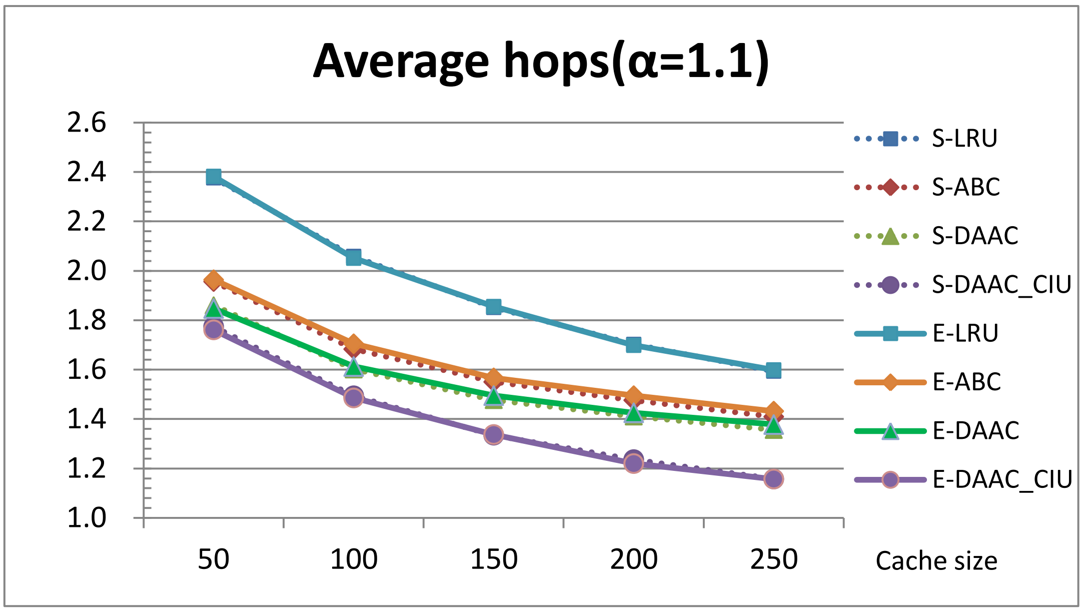


Figure 5-33 Average hop count of Abilene Topology ($\alpha = 1.1$)

Figure 5-34、Figure 5-35 and Figure 5-36 shows the average response time in simulation and emulation. The response time in simulation only take the link delay into consideration. On the other hand, in emulation, the response time higher than simulation because interests and contents are sent to Local Nox or Global Nox to check the CS table, PIT table and FIB table.

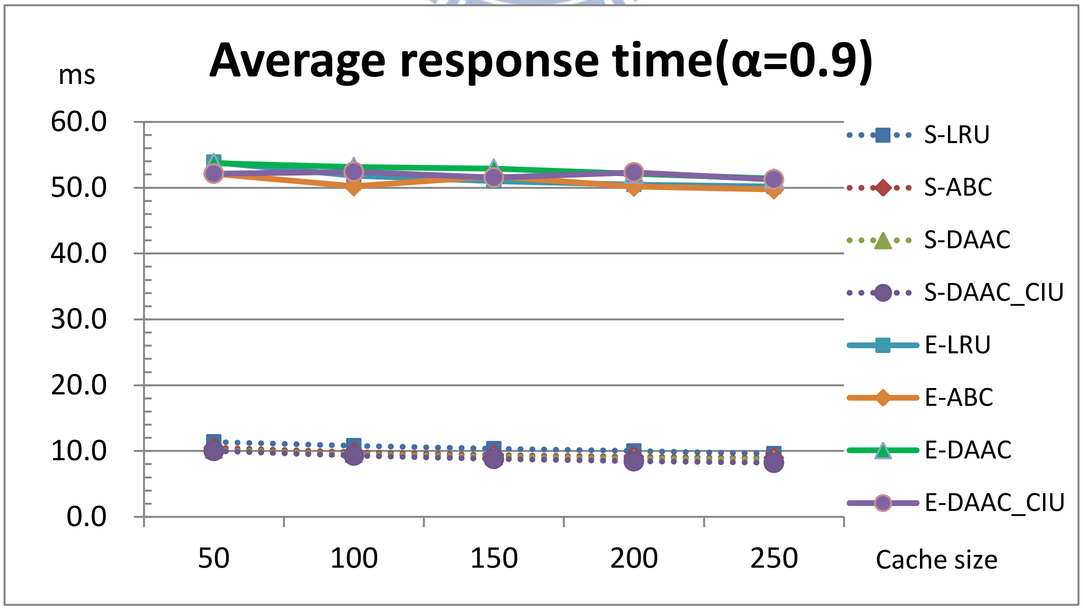


Figure 5-34 Average response time of Abilene Topology ($\alpha = 0.9$)

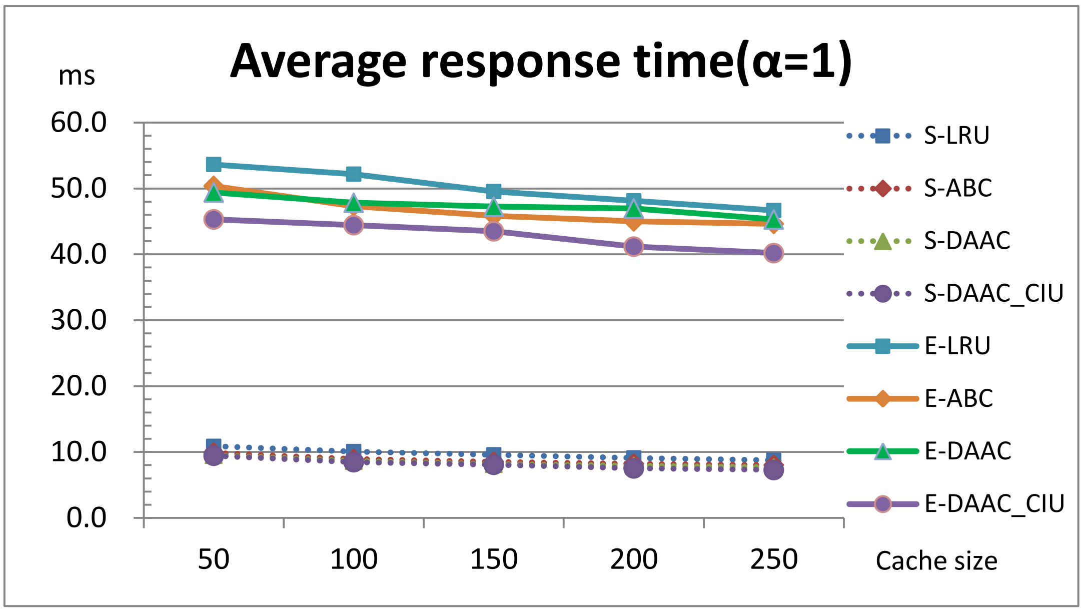


Figure 5-35 Average response time of Abilene Topology ($\alpha = 1$)

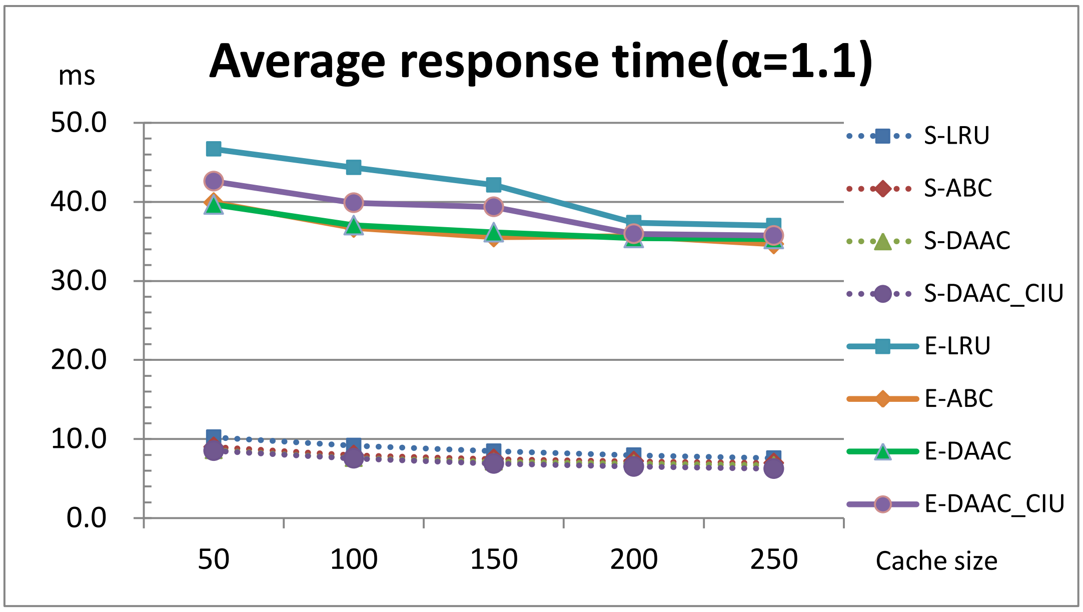


Figure 5-36 Average response time of Abilene Topology ($\alpha = 1.1$)

Chapter 6 Conclusion

We propose a Dynamic Age-Adjusted Cooperative caching scheme (DAAC), which is an extension of Aging-Based Cooperative caching scheme. DAAC has an age-refresh scheme to solve the temporal-locality problem in the content delivery services. It also proposes an age-decrease scheme to reduce the redundant data problem. The age-refresh scheme operates as follows. When a cache hits on a router, that router will reassign the age of that cache to its original assignment. Since each cache hit can prolong the life time of the cache being replaced, the age-refresh can help to resolve the temporal-locality problem. The age-decrease scheme works as follows. A downstream node sends an ACK back to the upstream node. If the upstream receives the ACK, it will decay the age of that content exponentially. In addition, the Caching Information Update scheme is proposed to provide the caching information for the router to obtain the routing efficiently. When a router stores a new content in its local cache, it disseminates this information using limited broadcast. The broadcast distance is the average hop distance of all shortest paths between the routers and the content server. When a router receives this information, it will check if the distance to the new caching router is smaller to the current one. If so, that router will update its routing table to modify the routing path to the new caching router. Simulation results show that DAAC_CIU can reduce the bandwidth consumption by at least 11% and server loading by 16% compared to ABC. Furthermore, we use the Software Defined Networks (SDN) to realize the CCN network. The simulation and emulation result show that there are almost the same, the difference only 3~4% at most. Hence, we can further prove our cache strategy not just a theory but also can be used to improve the network efficiency.

References

[1] V. Jacobson, *et al.*, "Networking named content," in *Proc. ACM CoNEXT*, 2009.

[2] K. Cho, *et al.*, "WAVE: Popularity-based and collaborative in-network caching for content-oriented networks," in *INFOCOM Workshop*, 2012.

[3] M. Gritter and D. R. Cheriton. (2000, Jul.). *TRIAD: A new next-generation Internet architecture*. Available: http://gregorio.stanford.edu/triad/index.html

[4] T. Koponen, *et al.*, "A data-oriented (and beyond) network architecture," *ACM SIGCOMM Computer Communication Review*, vol. 37, pp. 181-192, 2007.

[5] L. Zhang, *et al.* (2010, Oct.). *Named data networking (ndn) project*. Available: http://www.named-data.net/techreport/TR001ndn-proj.pdf

[6] V. Jacobson, *et al.*, "VoCCN: voice-over content-centric networks," in *Proc. of the 2009 workshop on Re-architecting the internet*, 2009.

[7] M. Caesar, *et al.*, "ROFL: routing on flat labels," *ACM SIGCOMM Computer Communication Review*, vol. 36, pp. 363-374, 2006.

[8] H. Che, *et al.*, "Analysis and design of hierarchical web caching systems," *Proc. IEEE INFOCOM'01*, vol. 3, pp. 1416-1424, 2001.

[9] X. Tang and S. T. Chanson, "Coordinated en-route web caching," *IEEE Transactions on Computers*, vol. 51, pp. 595-607, 2002.

[10] Z. Ming, *et al.*, "Age-based cooperative caching in information-centric networks," in *Workshop on Emerging Design Choices in Name-Oriented Networking*, 2012.

[11] I. Psaras, *et al.*, "Probabilistic in-network caching for information-centric networks," in *Proceedings of the second edition of the ICN workshop on Information-centric networking*, 2012, pp. 55-60.

[12] I. Psaras, *et al.*, "Modelling and evaluation of CCN-caching trees," in *NETWORKING 2011*, ed: Springer, 2011, pp. 78-91.

[13] N. McKeown, *et al.*, "OpenFlow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, pp. 69-74, 2008.

[14] B. Lantz, *et al.*, "A network in a laptop: rapid prototyping for software-defined networks," in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, 2010, p. 19.

[15] N. Gude, *et al.*, "NOX: towards an operating system for networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, pp. 105-110, 2008.

[16] R. Sherwood, *et al.*, "Flowvisor: A network virtualization layer," *OpenFlow Switch Consortium, Tech. Rep*, 2009.

[17] N. Laoutaris, *et al.*, "The LCD interconnection of LRU caches and its analysis," *Performance Evaluation*, vol. 63, pp. 609-634, 2006.

[18] M. Xie, *et al.*, "Enhancing cache robustness for content-centric networking," in

INFOCOM, 2012 Proceedings IEEE, 2012, pp. 2426-2434.

[19] E. J. Rosensweig and J. Kurose, "Breadcrumbs: efficient, best-effort content location in cache networks," in *IEEE INFOCOM*, 2009.

[20] S. Eum, *et al.*, "Potential based routing for ICN," in *Proc. of the 7th Asian Internet Engineering Conference*, 2011.

[21] R. Chiocchetti, *et al.*, "INFORM: a dynamic INterest FORwarding Mechanism for Information Centric Networking."

[22] C. Yi, *et al.*, "A case for stateful forwarding plane," *Computer Communications*, 2013.

[23] L. Saino, *et al.*, "Hash-routing Schemes for Information Centric Networking," 2013.

[24] D. Rossi and G. Rossini, "Caching performance of content centric networks under multi-path routing (and more)," *Technical report, Telecom ParisTech*, 2011.

[25] L. Breslau, *et al.*, "Web caching and Zipf-like distributions: Evidence and implications," *in Proc. IEEE INFOCOM,* vol. 1, pp. 126-134, 1999.