SATELLITE-ASSISTED CONTENT DELIVERY NETWORK USING MOBILITYFIRST FUTURE INTERNET ARCHITECTURE

BY SHASHIKANTH RANGAN PENUGONDE

A thesis submitted to the

Graduate School—New Brunswick

Rutgers, The State University of New Jersey

in partial fulfillment of the requirements

for the degree of

Master of Science

Graduate Program in Computer Science

Written under the direction of

Prof. Dipankar Raychaudhuri

and approved by

New Brunswick, New Jersey

May, 2017

ABSTRACT OF THE THESIS

Satellite-Assisted Content Delivery Network using MobilityFirst Future Internet Architecture

by Shashikanth Rangan Penugonde

Thesis Director: Prof. Dipankar Raychaudhuri

This thesis presents the design of a satellite-assisted CDN that utilizes the MobilityFirst Future Internet Architecture (MF-FIA) for content delivery. Current CDN solutions require an overlay control framework on top of IP to deal with content caching and delivery. However these features can be integrated into the network layer in an ICN based design. The proposed CDN solution uses MobilityFirst, a representative example of ICN to show the benefit of an integrated network layer solution. The framework uses in-network caches to reduce the latency of content delivery relative to overlay CDN designs. Content requests are efficiently routed to the nearest content source using the MobilityFirst routing protocol. As content providers are aware of content popularity and its distribution, the framework supports a pushing mechanism where content providers can pro-actively push popular contents to designated cache locations. This can be efficiently done using services such as multicast or broadcast which are natively supported in MobilityFirst. The routing mechanism can choose to deliver contents either through a terrestrial network or through a broadcast medium such as a satellite. Finally the proposed architecture uses self-certifying content names that enable efficient content validation.

A proof of concept prototype was created to demonstrate the feasibility of this solution and to conduct performance studies on the effectiveness of content caching in the network. The prototype includes Click based MobilityFirst routers, a GNRS, a content provider and a Click based satellite emulation. The benefit of in-network caches was studied by requesting contents with popularity drawn from a Zipf distribution. Results obtained with the prototype system demonstrated that clients experienced reduced delay due to in-network caches and that a fairly high cache hit ratio was achieved in some content distributions even when the cache only stored a small portion of the total contents. The evaluation was also used to study the impact of key parameters such as the Zipf distribution parameter and cache size on metrics such as delay and cache hit ratio.

Acknowledgements

I would like to express my sincere gratitude to my adviser Dr. Dipankar Raychaudhuri for his invaluable guidance and support through out. Working under him has been a great learning experience. I would also like to thank the rest of my thesis committee: Dr. Wade Trappe and Dr. Badri Nath.

I am thankful to Francesco Bronzino and Jiachen Chen for their insights and guidance that gave direction my work. I would also like thank my friends at WINLAB for their wonderful company. Last but not the least, I am thankful to my family for their constant support and encouragement.

Table of Contents

List of Figures

Chapter 1

Introduction

Media consumption is a crucial part of our daily activities. There has been a proliferation in hand held devices which are one of the most important sources of media consumption. Global mobile data traffic grew 63 percent in 2016 and by 2021 it is estimated that mobile will represent 20% of the entire IP traffic [16]. Content Delivery Networks serve as an important tool to facilitate content dissemination. Current IP based CDN solutions implement an overlay framework to optimize content delivery by using techniques such as web caching, server-load balancing and request routing [18]. These features can be integrated into the network layer in an ICN based design. Several Information Centric Network architectures [3], [4] have been proposed and MobilityFirst [3] is one such architecture. It is a mobility-centric architecture that supports large-scale, efficient and robust network services with mobility as the norm. In MobilityFirst, every network attached object (hosts, mobiles, content, context...) is assigned a global unique identifier (GUID), which is decoupled from its current network address or location. A dynamic global name resolution service (GNRS) [6], [7], [9], [10] is used to track and resolve the mapping between a GUID and its network addresses. MobilityFirst aims to support smooth mobile content delivery by enabling in-network content storage along with real-time GNRS updates and queries required to locate content objects.

In this work we utilize the MobilityFirst architecture to build a satellite-assisted content delivery network (S-CDN). We use Self-certifying content names in our design in order to provide low cost content validation and an object resolution service that helps clients get content names. We built in-network caches to reduce the latency of content retrieval and performed evaluations on ORBIT to demonstrate this. A unified push model was designed and built in order to support pro-active caching of contents.

Our solution also supports the use of a broadcast medium (satellite) in cohesion with the terrestrial network. This was motivated by an analytical study by Professor Mung Chiang's group at Princeton that confirms cost and performance gains of a satellite based CDN [15].

A proof of concept prototype was built to demonstrate the feasibility of the proposed architecture. MobilityFirst routers and the satellite were built using the open source Click platform. The MF Proxy and the content provider were built on top of the MobilityFirst protocol stack.

Chapter 2 provides a brief overview of the MobilityFirst architecture as it serves as the base framework for the S-CDN architecture. Chapter 3 builds on this and provides a detailed description of the S-CDN architecture. Chapter 4 presents a proof of concept prototype to demonstrate its functionality. We proceed to present our experimental evaluations in chapter 5. Finally, the conclusion and future work are presented in Chapter 6.

Chapter 2

MobilityFirst Future Internet Architecture

2.1 Motivation

Content Providers such as Netflix, YouTube and Facebook extensively use CDN Services provided by Akamai,AT&T etc. Thus the efficiency of content-retrieval is crucial. Existing IP based CDN solutions need to utilize an overlay framework for efficient content delivery.

We believe that the root cause of the problem lies in the fact that the IP Network only understands locations (IP Addresses) but not contents or URLs. Thus, the network can only forward a request to a predefined location (decided by DNS) rather than locate and fetch a piece of content based on the dynamic network status.

To Address this, Information Centric Networks (ICNs) were proposed that shift the network to focus on content identity,services and nodes. MobilityFirst [3], NDN [4] are alternative proposed ICN architectures, which use content identities as the routing label. After receiving a content request, the network can locate and fetch the replica from the best source or in-network storage location based on the network and provider status. We proceed to describe the S-CDN design on top of MobilityFirst.

2.2 Architecture Overview

MobilityFirst is a clean slate architecture designed to directly address the challenges of wireless access and mobility at scale, while also providing new services needed for emerging mobile Internet application scenarios. MobilityFirst follows the design principle of adding functionality to the core network rather than leaving it to the end host which is vital for mobility. It also inherently supports various services such as such as multicast, multi-homing, anycast, multi-path and IoT. Figure 2.1 depicts the overall architecture of MobilityFirst along with its routing protocols.

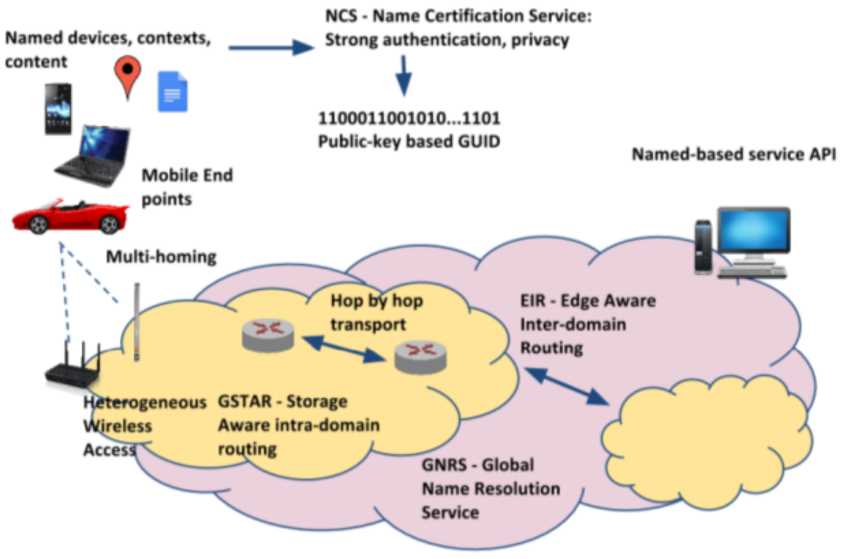

Named devices, contexts, content
NCS - Name Certification Service: Strong authentication, privacy
1100011001010...1101 Public-key based GUID
Named-based service API
Mobile End points
Multi-homing
Hop by hop transport
EIR - Edge Aware Inter-domain Routing
Heterogeneous Wireless Access
GSTAR - Storage Aware intra-domain routing
GNRS - Global Name Resolution Service

Figure 2.1: MobilityFirst architecture overview [2]

The existing IP based solution for addressing mobility is inadequate as there is a conflation of location with identity. MobilityFirst aims to solve this by separating name/identity from location. Each network attached object is assigned a Globally Unique Identifier (GUID). Entities can represent a hand held device, service, vehicle, content, group of these objects or even a context.

The GUID is a 20 byte identifier generated by the Name Certification Service(NCS). It is essentially a public key and is hence self-certifying. This alleviates the need of an external authority for node authentication.

The mapping of GUIDs to locations is maintained by the logically centralized Global Name Resolution Service(GNRS) [7]. The GUID serves as a narrow waist for the MobilityFirst protocol stack similar to how IP serves as the narrow waist for the current Internet Architecture. Figure 2.2 demonstrates this.

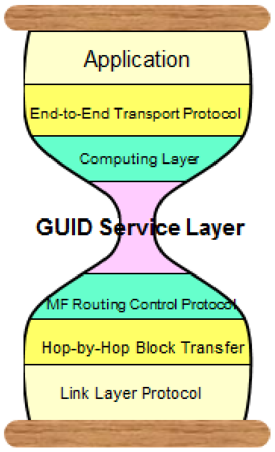

Application
End-to-End Transport Protocol
Computing Layer
GUID Service Layer
MF Routing Control Protocol
Hop-by-Hop Block Transfer
Link Layer Protocol

Figure 2.2: MobilityFirst stack with GUID as the narrow waist [19]

The Name Based Service API in MobilityFirst uses the unique names of end point network objects instead of their network addresses or interfaces. This allows for abstract services based on multi-homing, multicast, named content, anycast etc.

2.3 Intra-Domain Routing: GSTAR

GSTAR [5] is essentially a storage-aware link-state routing protocol that enables routers to temporarily store and/or replicate data in response to detected network problems. GSTAR operates on the principle that mobility related challenges are best handled directly at the networking layer itself.

Each node in the network uses two types of topology and path quality information (Figure 2.3) while deciding to pro-actively store or push out messages. The first, called intra-partition graph, is maintained via flooded-link State Advertisements (F-LSA). F-LSAs carry fine-grained, time-sensitive information about the links in the network.

The second, called DTN graph, is maintained via epidemically Disseminated Link State Advertisements (D-LSA). D-LSAs carry connection probabilities between all nodes in the network.

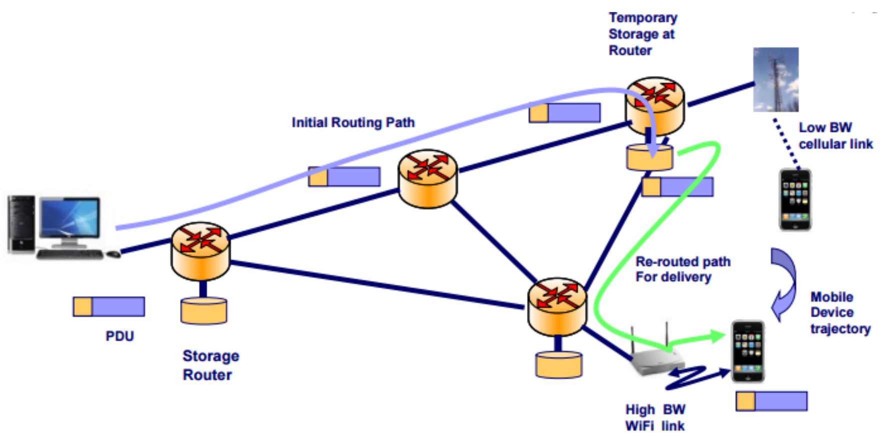


Figure 2.3: GSTAR [17]

2.4 Inter-domain Routing: EIR

Edge Aware Inter-domain routing or EIR [20] is a clean slate inter-domain routing protocol designed to meet the needs of the future mobile Internet. EIR provides enhanced information about network topology and edge network properties through a telescopic network state dissemination protocol and by enabling autonomous systems to optionally expose internal network topology and aggregate properties such as bandwidth, availability and variability. EIR uses aNodes and vLinks as abstractions for optionally exposing aggregated internal network topology.

Network State Packets (NSPs) encapsulate information about aNodes and vLinks and are flooded through out the network in a telescopic manner. Telescopic flooding provides fast updates to nearby networks while eventually providing a global view of the network topology to all networks. EIR is also designed to work in conjunction with late binding of names to addresses and in-network storage in order to provide robust services in environments with dynamic mobility and disconnection.

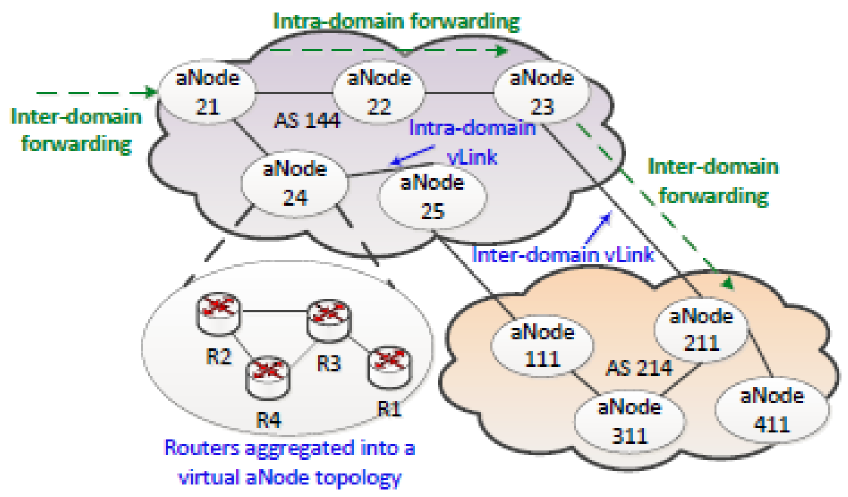
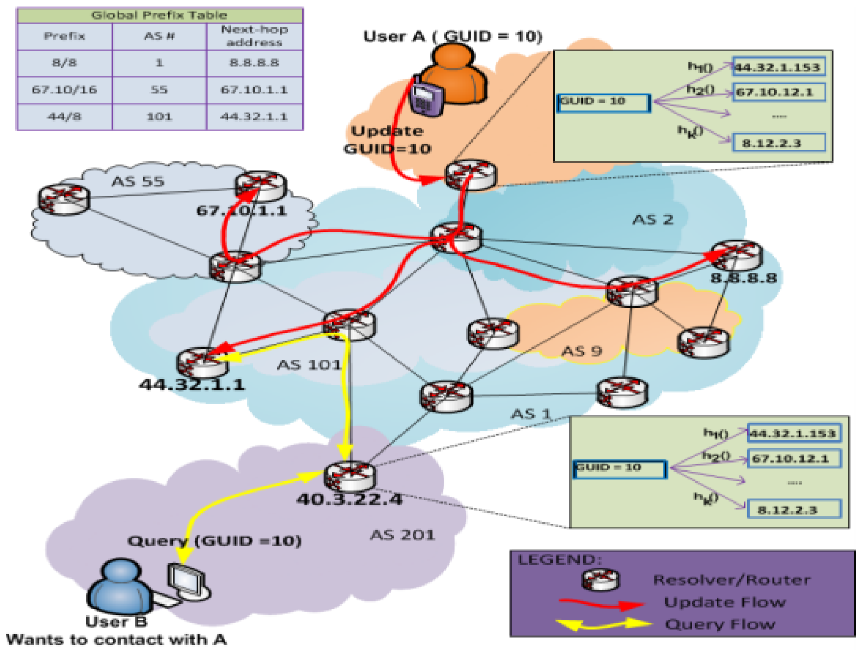


Figure 2.4: aNode-vLink topology abstraction for an AS [20]

2.5 GNRS

The Global Name Resolution Service [7] is one of the core components of the Mobility-First framework that helps in separating location from identity. It does this by storing mappings of GUIDs to Network Addresses. Each GUID is unique and flat and thus in cohesion with the GNRS provides better support for mobility when compared to the existing IP solutions where disconnections are frequent as a new location implies a new IP address.

The GNRS is designed and built to be a logically centralized but physically distributed service that significantly enhances and simplifies a number of network-layer functions such as seamless host, network, or service mobility, multi-homed traffic engineering, content retrieval, multicast, context-aware services, etc.



Global Prefix Table		
Prefix	AS #	Next-hop address
8/8	1	8.8.8.8
67.10/16	55	67.10.1.1
44/8	101	44.32.1.1

Figure 2.5: GNRS [6]

The current implementation of the GNRS uses the Direct Mapping(DMap) scheme to achieve a scalable distributed system for shared hosting of the GUID to NA mappings. In DMap, each GUID to NA mapping is stored in a set of ASs. Each GUID is directly hashed to existing network addresses and its mapping is thus stored within the ASes corresponding to these network addresses.

2.6 Name Certification Service

The name certification service(NCS) assigns globally unique identifiers (GUID) to network objects. The NCS generates and returns a public key/private key pair for each GUID request. The public key generated is the GUID for the network object and the private key is used as its signature. The NCS stores mappings of human-readable names(HRN) to GUIDs. The chain of trust can be adopted in this process to help identify the network object.

2.7 Object Resolution Service

Object resolution service (ORS) is a service that helps users translate requirements, keywords or other features to network IDs [1]. The meta-data associated with the network object is registered with the ORS. The ORS indexes the information associated with the network object with a search engine. The ORS can be queried with keywords and it displays information of the network objects retrieved such as GUID and other relevant meta-data.

Chapter 3

CDN Requirements and Architecture Overview

3.1 Requirements

To ensure efficient and low-cost content delivery, a CDN should satisfy the following requirements [1]:

- **Efficient content retrieval:** Efficient content retrieval is the key requirement to all CDNs. When a CDN provider has multiple caches in the network, it is desirable that the request is sent to a nearby cache rather than going all the way to the content provider. Due to the variations in bandwidth,latency and some other factors in the network, it is difficult for the CDN caches to locate a nearby copy. Therefore, it is important for the architecture to fully utilize the functions provided by the network to improve efficiency.

- **Efficient content validation:** Cache pollution is a big issue for CDNs (especially in ICN) since illegitimate users may respond to content requests with invalid content. To avoid storing corrupted content, the CDNs should be able to verify the integrity of a content file and the mapping between the content and its identity, at a low cost.

- **Content control and awareness:** Content provider must be able to control 1) whether content can/cannot be stored in the CDN 2) where they should be stored. Since this is application-specific, the architecture should not impose network-layer modifications to satisfy the CDN applications.

- **Consumer mobility support:** With the wide use of mobile devices like smartphones and wearable devices, content delivery is expected to support consumer mobility. It could be possible that devices switch to a different domain during a content retrieval session, or even get off-line for hours.

3.2 Architecture Overview

The architecture of the proposed solution is shown in Figure 3.1. Query for a content is forwarded to an in-network cache. On a cache miss the request is forwarded to the ”nearest” content replica. The nearest content replica,which could be a cache in another network or a content provider is efficiently computed by the MF routing protocol [5].

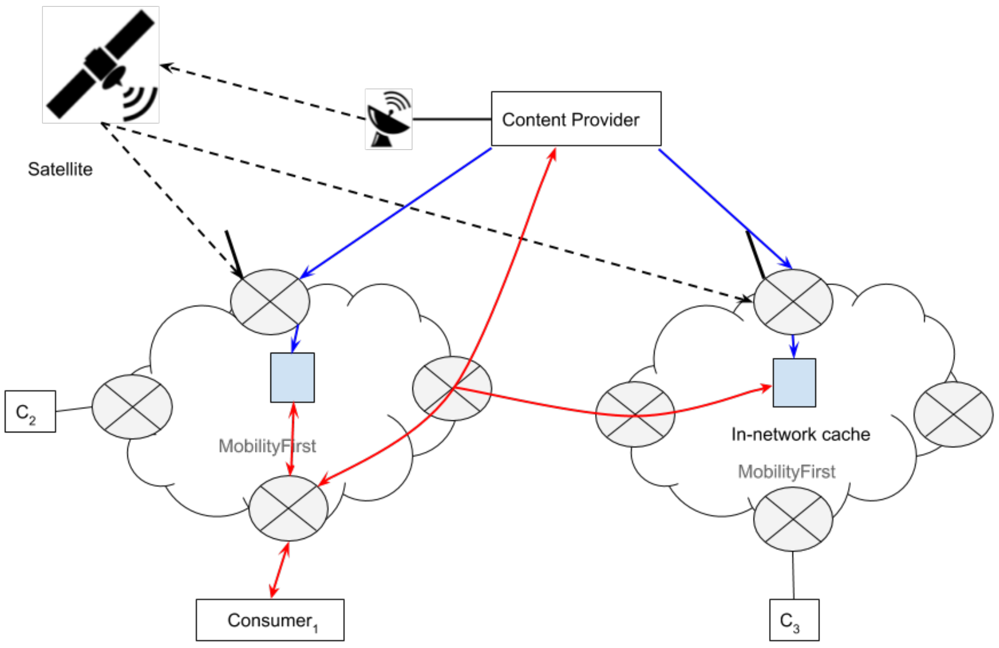


Figure 3.1: Content delivery in the proposed architecture

Figure 3.1 shows the content delivery scenario in the proposed architecture. Red lines indicate content query, blue lines indicate pro-active caching initiated by a content provider and the black dotted lines indicate content broadcast through satellite.

Since each CDN can listen for different GUIDs and/or join different multicast groups, we enable the server to push content files to predefined cache locations for proactive caching. To facilitate efficient content validation, we use self-certifying names so that the CDN can verify the integrity of the content without relying on other services in the network. MobilityFirst provides the architecture with efficient consumer mobility support thanks to the late-binding and store-and-forward mechanisms.

3.3 Detailed Design

This section starts with the discussion of how contents are named and validated. We proceed to describe how we use these names for content delivery in our CDN. The final section describes the optimizations provided in our architecture such as pro-active caching and multicast support.

3.3.1 Content Naming and Validation

Content Validation is one of the key requirements in networking and communication. To validate the integrity of a content file, the content provider and the consumer must possess the content's signature and public key.Self-certifying content names are highly desirable in CDNs as the cost of content validation is much lower than validating their non self-certifying counterparts. However in the IP world the consumer can only secure a location-to-location channel rather than the content itself.

MobilityFirst provides a suitable way to achieve self-certification by securing the content rather than the channel itself. It requires each content to be signed by the content generator. The content generator's signature along with its public key is carried with the content so that consumers/content providers can validate the integrity of the content source.

However this solution is still incomplete as there is still an issue with self certification as there is no relationship between the content name/ID and the content itself. As a result attackers could respond to a content request with an invalid or malicious content, the attacker's public key and the signature generated by the attacker. Unfortunately the consumers are unable to identify these kind of attacks by checking the content as the signature and the public key match each other. One way to address this problem is to rely on a trusted third party which stores the relationship between the content ID and public key. However this violates the self certifying property and prompts for a better solution.

An easy way to capture the relationship between the content ID and content is to use either the signature or public key as the content ID. We chose to proceed with using public key as content ID due to high likelihood of signature collision. In order to obtain the content's public key, the content generator must send the meta data of the content to the Name Certification Service (NCS). The chain of trust can be adopted in this process to help consumers identify the content generator [1].
On receiving a request for a public key, the NCS would generate a public-private key pair for that content. The content generator would then sign this content with the private key it obtained and use the public key as the content GUID. In our solution we use the Elliptic Curve Cryptography (ECC) to generate the content GUIDs.

After receiving the content, the consumer/content provider can easily verify if the content's signature matches the public key/content GUID. Hence it becomes extremely difficult for an attacker to replace the content and its signature since the signature is computed by the private key which only the content generator produces. The attacker cannot use its own public key as it means using a different content GUID. Therefore the validation process in this architecture does not reply on a trusted third party and the cost of content validation is low. Hence this solution satisfies the self-certifying requirement.

3.3.2 Network layer CDN Support

In the proposed architecture we assign each content a GUID. A content source such as a content provider or an in-network cache registers itself as a new content location for all the contents it possesses to the GNRS.

The mapping of content GUID to network addresses of the devices that contain it is stored in the local GNRS. It is important to note that there might be multiple devices (NAs) that serve the same content GUID.

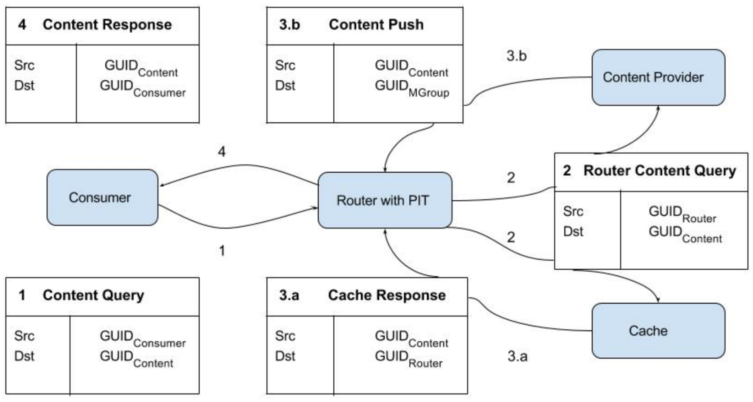


Figure 3.2: Protocol exchange of content retrieval

A consumer that requests a content sends a request packet to the first hop router. The first hop router then queries the GNRS for the location of this content and performs on the following actions:

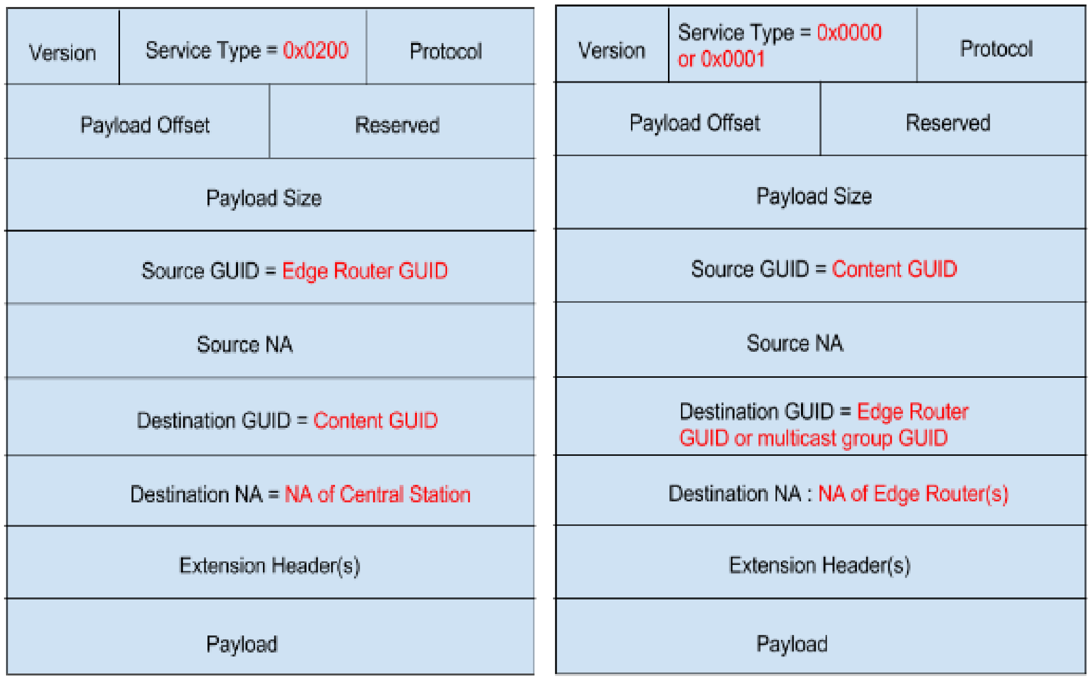
- If the requested content is present in the in-network cache, it responds with the content from the cache. Note that either the first hop router is connected to the in-network cache or it forwards it to a router connected to the in-network cache.
- On a cache miss, it needs to forward the request to the nearest content source. The content can be present in nearby in-network caches as well. The MF router uses anycast to determine the nearest content source and forwards the request to it.

However there is a possibility of multiple consumers requesting for a content in a small window of time. Therefore in order to avoid multiple requests for the same content to the content provider, we introduce a Pending Interest Table (PIT) into each first hop router.

Each entry in the PIT comprises of a mapping between pending content GUID to the GUIDs of the consumers that requested it. A new entry in the PIT implies a new content request. If a content GUID already exists in the PIT,it implies that a request for this content GUID was already made. Hence it appends this request (consumer GUID) to the existing content GUID.

On a content response which can either come from a content provider or a neighboring cache, if the content is cached, then the GNRS needs to be updated with this new content location. This is an important step as the GNRS must be up to date with the latest content locations in order to aid the nearest replica routing. This is also true in case a content is evicted from the cache.While updating the GNRS for each piece of content would incur some control overhead in the network, [13] suggests that this design choice is feasible and it enables more efficient Nearest-Replica Routing (NRR) which can save much more network traffic compared to the overhead. Figure 3.3 depicts request and response packets used for content retrieval. Figure 3.3a shows a request packet made by the edge router with pending interest table (PIT). Figure 3.3b shows a response packet from the central station (router capable of sending packets through satellite and terrestrial MF network).

In our design we have assigned GUIDs to contents and consumers. Thus this architecture can take full advantage of the mobility support offered to end hosts/consumers by MobilityFirst Routers can query the GNRS for the latest NA of the consumer and redirect the content to the latest consumer location. The "store and forward" mechanism allows the network to cache the content when the consumer is off-line forward the content when the consumer is back online. Network assisted mobility prediction can be used to improve the user experience by routing content to a location even before the consumer appears there [12].


Version
Service Type = 0x0200
Protocol
Payload Offset
Reserved
Payload Size
Source GUID = Edge Router GUID
Source NA
Destination GUID = Content GUID
Destination NA = NA of Central Station
Extension Header(s)
Payload
Version
Service Type = 0x0000 or 0x0001
Protocol
Payload Offset
Reserved
Payload Size
Source GUID = Content GUID
Source NA
Destination GUID = Edge Router GUID or multicast group GUID
Destination NA : NA of Edge Router(s)
Extension Header(s)
Payload

(a) Request Packet

(b) Response Packet

Figure 3.3: Request and Response Packets

3.3.3 Pushing and Multicast Support

In existing CDN solutions, the cache fetching policy is reactive i.e the cache will retrieve and store a content object only when the requested content is missing in the cache. Therefore a consumer/consumers that request for a content not available in the cache experience long latencies as the content has to be fetched from a distant location. This issue becomes more prominent if a content that gets many requests is not available in the cache yet.

To address this inefficiency, our architecture allows content providers to pro-actively push content replicas to cache locations(Step 3.b in figure 3.2. A provider can choose to push content in two scenarios:

- A content provider can push content to the desired cache locations even without receiving a query by sending a content push message to the desired destinations.
- On receiving a content request, the content provider can also choose to push this

content to multiple locations while also including the consumer as a target.

Thus the network allows the content provider to make intelligent decisions regarding where to push the content to, while the network makes informed decisions on how to forward these contents to the required destinations.

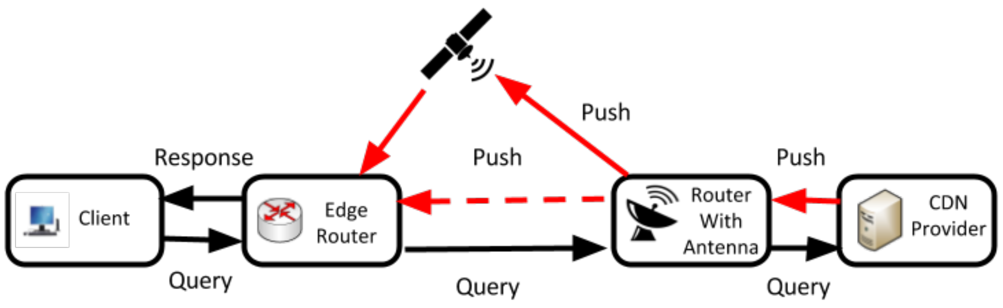


Figure 3.4: Unified Push Model

This mechanism is extremely useful for applications like video delivery, online gaming where the content providers have a good understanding of content popularity. The content providers can also get a good view of a content's location in different locations. Thus contents can be placed intelligently on caches achieving good cache hit rates.

Our architecture utilizes the multicast functionality natively supported by MobilityFirst. Instead of doing multiple unicast pushes, a content provider can create different multicast groups based on choices such as location,popularity distribution etc.

In order to push a content to a multicast group, the content provider must just specify the multicast group GUID ($GUID_{MGroup}$) as the destination in the content push message. The routers at the bifurcation point in the multicast tree would replicate and forward the content to the next set of bifurcation routers. This results in reduction of network traffic when compared to multiple unicast pushes [8].

Chapter 4
Proof of Concept Prototype

A proof of concept framework was built to demonstrate the benefits proposed in the previous sections. The components for this architecture were built on top of the MobilityFirst framework. Evaluation of the prototype was done on the ORBIT research testbed.The following sections detail the components of the MobilityFirst framework and the components of the sCDN architecture.

4.1 S-CDN Prototype

Figure 4.1 shows the components of the S-CDN prototype along with their sub modules. It is important to note that a few of these sub modules are incomplete such as the "unreliable transport module" in the MobilityFirst routers and "relay with certain loss probability" module in the satellite component.

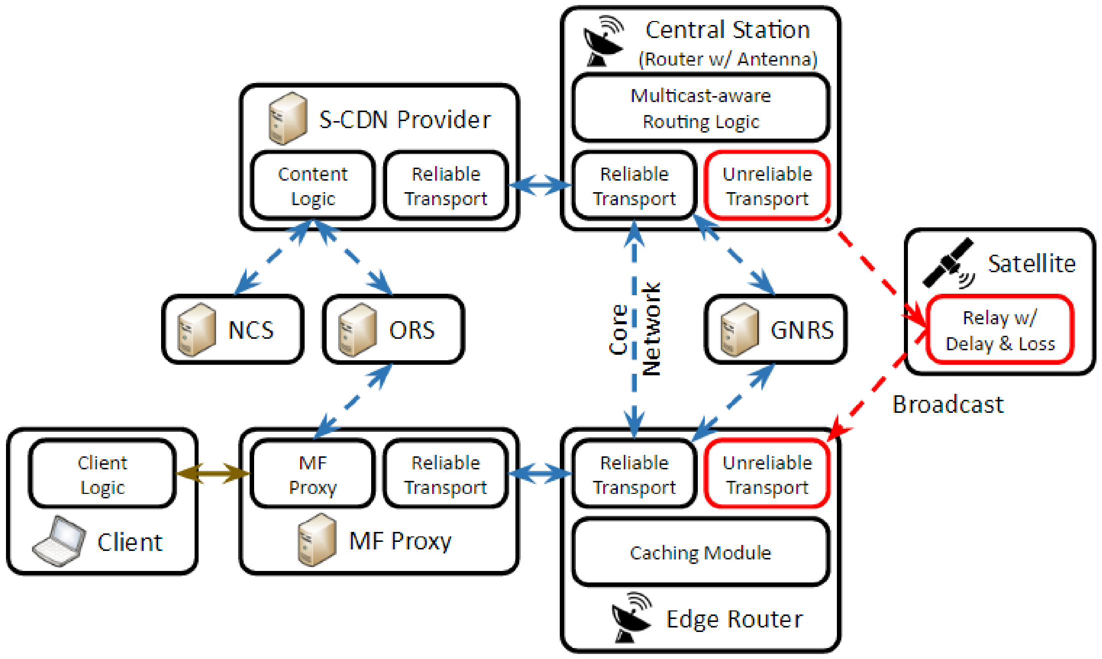


Figure 4.1: S-CDN Prototype

4.1.1 Client

Clients request for contents using a browser. An HTTP request is made to the MF Proxy requesting content specified by a GUID.Figure 4.2 demonstrates this. The setup works fine With Internet Explorer and Mozilla Firefox using flashplugin 12.04.

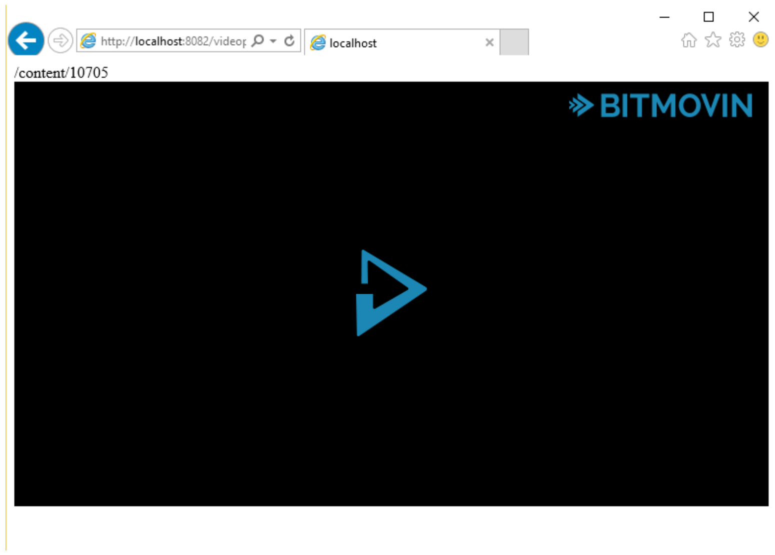


Figure 4.2: Client

4.1.2 MF Proxy

The MF Proxy is used to translate HTTP requests from the client into MobilityFirst requests. It marks the entry point into the MobilityFirst network. It creates an MF Packet and sets the content GUID requested in the HTTP request as the destination. It forwards this packet to a first hop MobilityFirst router.

When a content response arrives at the proxy, it translates this request back to an HTTP response. The MF Proxy is built as an application on top of the MobilityFirst host-stack in C++.

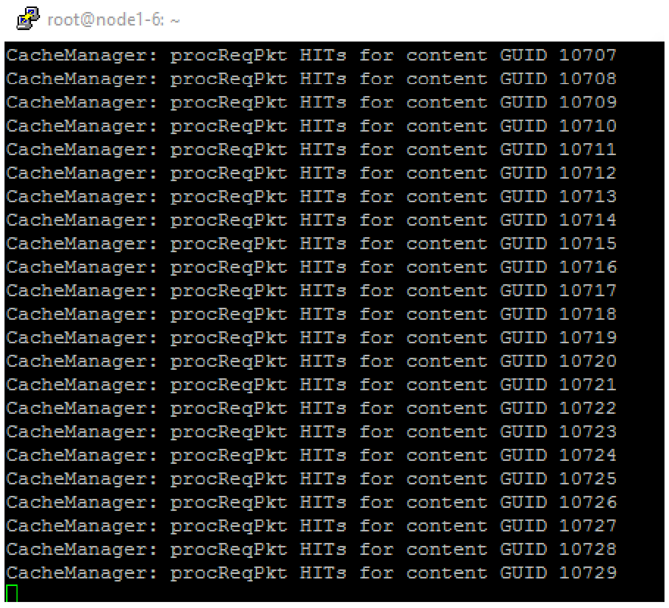
4.1.3 GNRS

The global name resolution service (GNRS) is a DMAP [6] based implementation built in JAVA. It resolves globally unique identifiers(GUIDs) to Network Addresses(NA) which is essential for determining the location of contents.

4.1.4 Edge Router With Cache

The Edge router with cache comprises of modules that provide the core MobilityFirst functionalities along with a cache module. The cache module is currently implemented using the "Least Recently Used" replacement policy. This is implemented in C++ and it communicates with the edge router with a cache interface built into the router.

On receiving a content request, the router performs a GNRS lookup to determine the content location. If the content is present in the in-network cache,it responds with content from the cache. However if it determines the content in different locations, it forwards the request to the "nearest" content replica. The "nearest" replica is calculated using a metric called the Estimated Transfer Time (ETT) that estimates the time taken to transfer content taking real time network status into consideration. Figure 4.3 shows cache hits in the in-network cache.



```
CacheManager: procReqPkt HITs for content GUID 10707
CacheManager: procReqPkt HITs for content GUID 10708
CacheManager: procReqPkt HITs for content GUID 10709
CacheManager: procReqPkt HITs for content GUID 10710
CacheManager: procReqPkt HITs for content GUID 10711
CacheManager: procReqPkt HITs for content GUID 10712
CacheManager: procReqPkt HITs for content GUID 10713
CacheManager: procReqPkt HITs for content GUID 10714
CacheManager: procReqPkt HITs for content GUID 10715
CacheManager: procReqPkt HITs for content GUID 10716
CacheManager: procReqPkt HITs for content GUID 10717
CacheManager: procReqPkt HITs for content GUID 10718
CacheManager: procReqPkt HITs for content GUID 10719
CacheManager: procReqPkt HITs for content GUID 10720
CacheManager: procReqPkt HITs for content GUID 10721
CacheManager: procReqPkt HITs for content GUID 10722
CacheManager: procReqPkt HITs for content GUID 10723
CacheManager: procReqPkt HITs for content GUID 10724
CacheManager: procReqPkt HITs for content GUID 10725
CacheManager: procReqPkt HITs for content GUID 10726
CacheManager: procReqPkt HITs for content GUID 10727
CacheManager: procReqPkt HITs for content GUID 10728
CacheManager: procReqPkt HITs for content GUID 10729
```

Figure 4.3: In-Network Cache Hit

All the routers in MobilityFirst are built using an open source framework called Click [21] that supports building efficient modular routers.

4.1.5 Satellite

The Satellite module is simulated in click. It serves as a broadcast medium for content dissemination. It is important to note that it only broadcasts content responses and does not take in content requests. VLAN tags were used to achieve broadcasting and will be described in more detail in the upcoming section.

4.1.6 Central Station

Central Station is implemented as a router that has the capability to send content response through the satellite and/or the terrestrial network. It has an up-link to the satellite. Based on the service id field in the packet header, it decides the service type(content request,content response, satellite response etc) and acts accordingly.

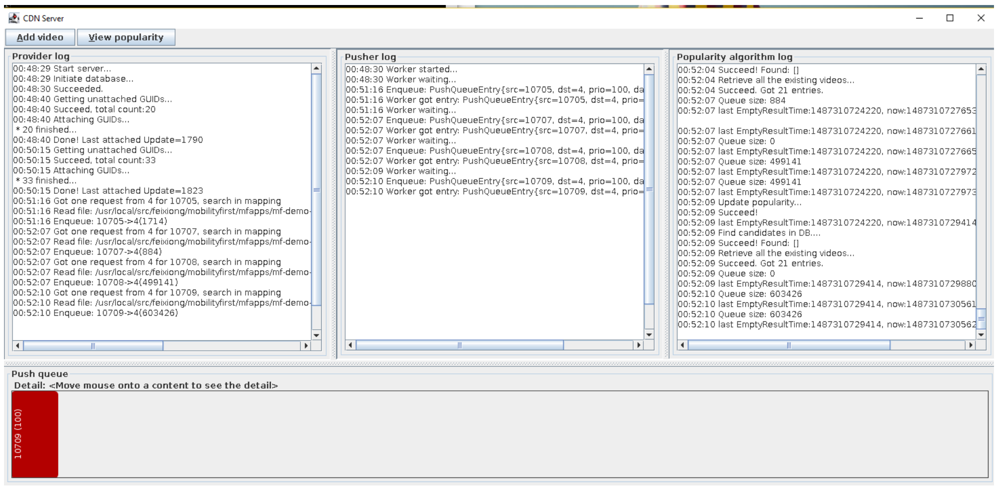
Since MobilityFirst natively supports multicast we plan to add functionality to the central station that allows it to compute the total hop distance of the destinations in order to decide whether a terrestrial transmission is more efficient when compared to satellite transmission or vice versa.

4.1.7 S-CDN Provider

The S-CDN Provider (also known as Content Provider) is an application that is implemented on top of the MobilityFirst host-stack. It is implemented in Java and uses MobilityFirst content APIs to listen to content requests and respond to them over the MobilityFirst network.

In addition to responding to contents reactively, we have implemented support for pro-active caching.The S-CDN provider can pro-actively decide where and when to cache content as it has a better view of content popularity. The S-CDN provider has the option of forcing which path to chose for content dissemination (satellite or terrestrial) or it can allow the routers to decide which path is optimal.

For scheduling of contents the S-CDN provider uses a priority queue. The priority is determined by factors such as content popularity and urgency. Content requests have the highest urgency value and hence those contents are placed in the head of the queue. Content push is ordered by popularity. Figure 4.4 demonstrate the S-CDN provider responding to content requests.



```
00:48:29 Start server...
00:48:29 Initiate database...
00:48:30 Succeeded.
00:48:40 Getting unattached GUIDs...
00:48:40 Succeed, total count:20
00:48:40 Attaching GUIDs...
 * 20 finished...
00:48:40 Done! Last attached Update=1790
00:50:15 Getting unattached GUIDs...
00:50:15 Succeed, total count:33
00:50:15 Attaching GUIDs...
 * 33 finished...
00:50:15 Done! Last attached Update=1823
00:51:16 Got one request from 4 for 10705, search in mapping
00:51:16 Read file: /usr/local/src/feixiong/mobilityfirst/mfapps/mf-demo-
00:51:16 Enqueue: 10705->4(1714)
00:52:07 Got one request from 4 for 10707, search in mapping
00:52:07 Read file: /usr/local/src/feixiong/mobilityfirst/mfapps/mf-demo-
00:52:07 Enqueue: 10707->4(884)
00:52:07 Got one request from 4 for 10708, search in mapping
00:52:07 Read file: /usr/local/src/feixiong/mobilityfirst/mfapps/mf-demo-
00:52:07 Enqueue: 10708->4(499141)
00:52:10 Got one request from 4 for 10709, search in mapping
00:52:10 Read file: /usr/local/src/feixiong/mobilityfirst/mfapps/mf-demo-
00:52:10 Enqueue: 10709->4(603426)
```

```
00:48:30 Worker started...
00:48:30 Worker waiting...
00:51:16 Enqueue: PushQueueEntry{src=10705, dst=4, prio=100, da
00:51:16 Worker got entry: PushQueueEntry{src=10705, dst=4, prio=
00:51:16 Worker waiting...
00:52:07 Enqueue: PushQueueEntry{src=10707, dst=4, prio=100, da
00:52:07 Worker got entry: PushQueueEntry{src=10707, dst=4, prio=
00:52:07 Worker waiting...
00:52:07 Enqueue: PushQueueEntry{src=10708, dst=4, prio=100, da
00:52:07 Worker got entry: PushQueueEntry{src=10708, dst=4, prio=
00:52:09 Worker waiting...
00:52:10 Enqueue: PushQueueEntry{src=10709, dst=4, prio=100, da
00:52:10 Worker got entry: PushQueueEntry{src=10709, dst=4, prio=
```

```
00:52:04 Succeed! Found: []
00:52:04 Retrieve all the existing videos...
00:52:04 Succeed. Got 21 entries.
00:52:07 Queue size: 884
00:52:07 last EmptyResultTime:1487310724220, now:1487310727653

00:52:07 last EmptyResultTime:1487310724220, now:1487310727661
00:52:07 Queue size: 0
00:52:07 last EmptyResultTime:1487310724220, now:1487310727665
00:52:07 Queue size: 499141
00:52:07 last EmptyResultTime:1487310724220, now:1487310727972
00:52:07 Queue size: 499141
00:52:07 last EmptyResultTime:1487310724220, now:1487310727973
00:52:09 Update popularity...
00:52:09 Succeed!
00:52:09 last EmptyResultTime:1487310724220, now:1487310729414
00:52:09 Find candidates in DB....
00:52:09 Succeed! Found: []
00:52:09 Retrieve all the existing videos...
00:52:09 Succeed. Got 21 entries.
00:52:09 Queue size: 0
00:52:09 last EmptyResultTime:1487310729414, now:1487310729880
00:52:10 Queue size: 603426
00:52:10 last EmptyResultTime:1487310729414, now:1487310730561
00:52:10 Queue size: 603426
00:52:10 last EmptyResultTime:1487310729414, now:1487310730562
```

Detail: <Move mouse onto a content to see the detail>

Figure 4.4: Content Provider

The S-CDN provider and the MF Proxy use content APIs provided by MobilityFirst. These APIs are built both in C++ and JAVA using the MobilityFirst host-stack and provide support such as socket management, content transmission etc.

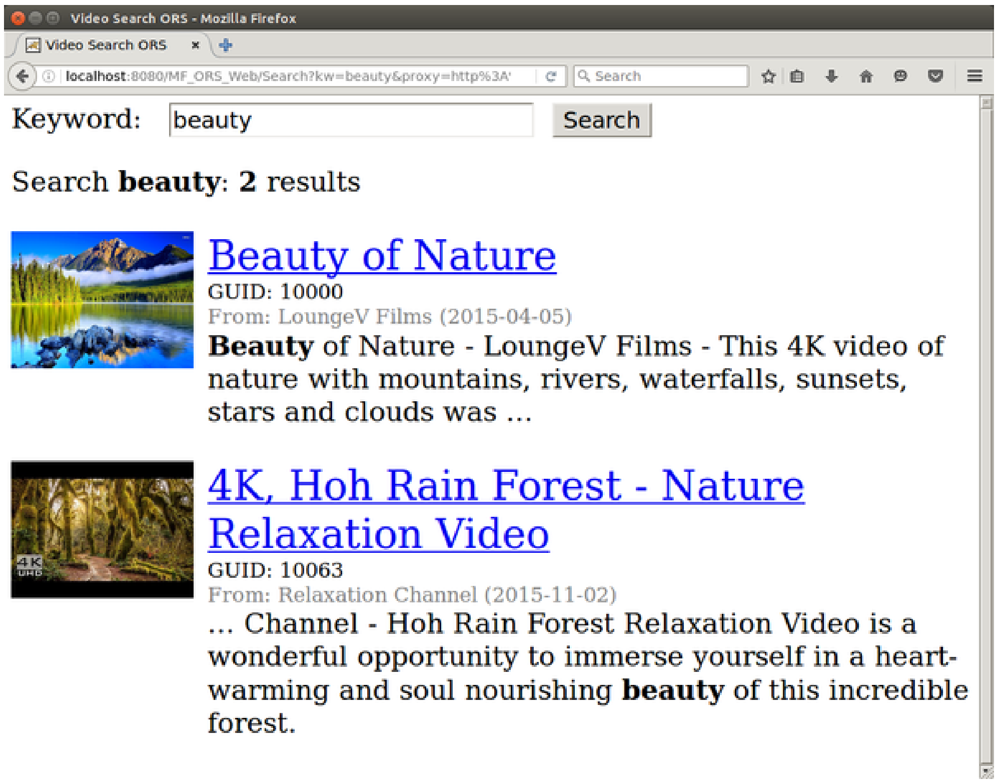
4.1.8 Name Certification Service

The Name Certification Service (NCS) is an application implemented in Java that generates and grants 20 byte GUIDs for an entity (S-CDN Provider in our case) that requests GUIDs. The provider sends information such as provider name, provider information and content metadata to the NCS in order to obtain the content GUID.

4.1.9 Object Resolution Service

The Object Resolution Service(ORS) is a service hosted on Apache Tomcat that indexes content information.It obtains the necessary content information either from the provider or through web crawling.

It takes in keywords from clients (mostly Human Readable Names) and responds with tuples containing content GUID,screen-shots(if available) and a brief description of the content. Figure 4.5 shows the working of the ORS.


Video Search ORS - Mozilla Firefox
Video Search ORS
localhost:8080/MF_ORS_Web/Search?kw=beauty&proxy=http%3A'
Search

Keyword: beauty Search

Search **beauty**: **2** results

Beauty of Nature
GUID: 10000
From: LoungeV Films (2015-04-05)
Beauty of Nature - LoungeV Films - This 4K video of nature with mountains, rivers, waterfalls, sunsets, stars and clouds was ...

4K, Hoh Rain Forest - Nature Relaxation Video
GUID: 10063
From: Relaxation Channel (2015-11-02)
... Channel - Hoh Rain Forest Relaxation Video is a wonderful opportunity to immerse yourself in a heart-warming and soul nourishing **beauty** of this incredible forest.

Figure 4.5: Object Resolution Service

4.2 Prototype Setup on Orbit Testbed with VLAN tags

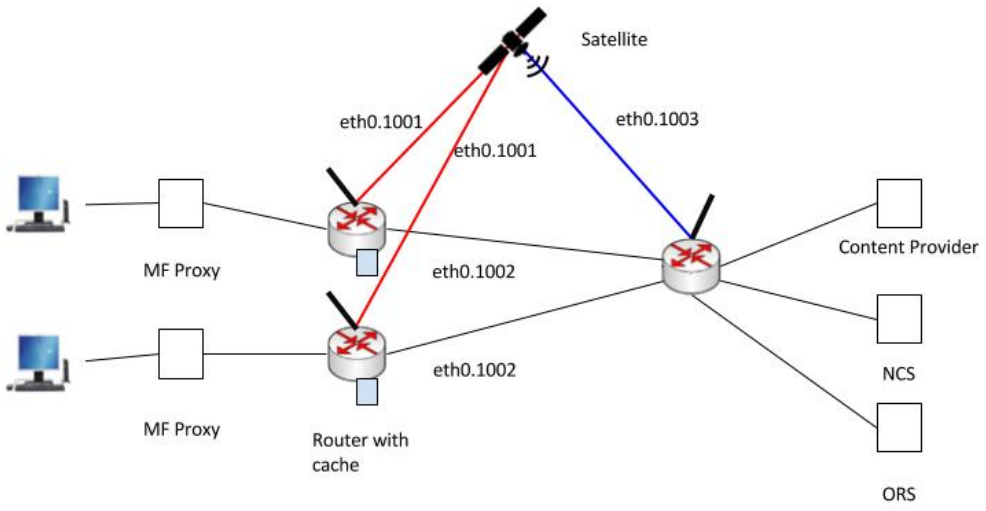


Figure 4.6: Example Topology with VLAN tags

Figure 5.1 demonstrates an example topology with VLAN tags. VLAN tags are created in order to separate broadcast domains. Separate tags are required for satellite up-link, satellite down-link and for the terrestrial network. The above setup is setup on the ORBIT testbed where the nodes of certain sandboxes support creation of VLAN tags [11].

Chapter 5 Evaluations

The topology setup for the set of experiments described below is as shown in Figure 1. The set of MobilityFirst routers and satellite are programmable components developed using an open source framework called Click.

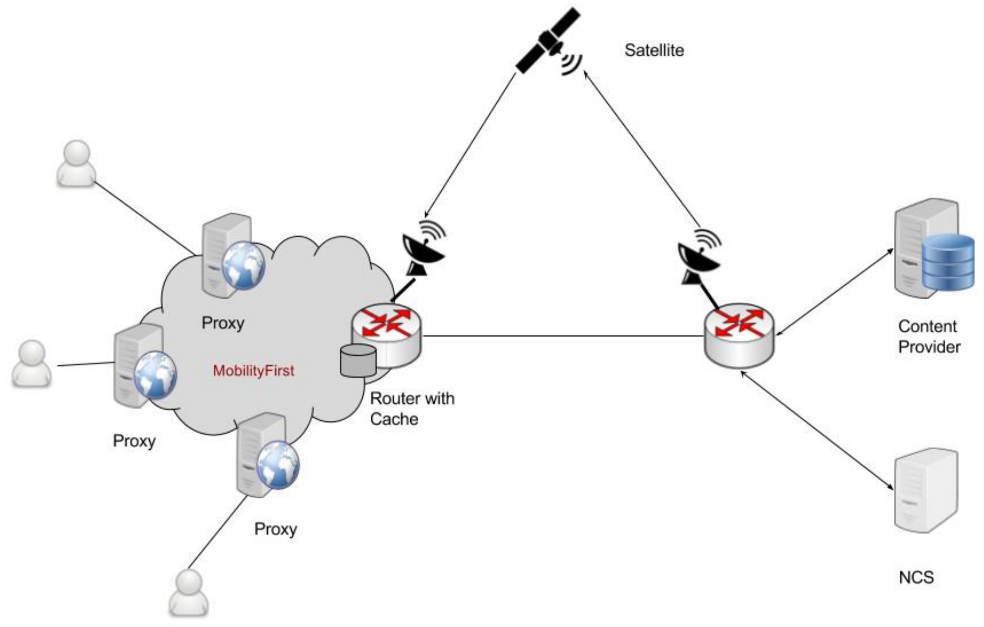


Figure 5.1: Topology setup for evaluation

The Proxy, Content Provider and the Name Certification Service (NCS) are applications that run on top of the MobilityFirst host-stack. All these components are brought up on individual nodes on the ORBIT testbed as individual nodes. The clients here request content via a browser which are translated into MobilityFirst requests by the Proxy. The location of the content is known from the GNRS and the content is obtained either from the Content Provider or the cache.

Note :

1. Routers in MobilityFirst comprise of both routers with in-network caches and routers without caches.

2. Server bandwidth is constrained to 50 Mbps.

5.1 Performance of In-Network Cache for Single Content Request under varying load

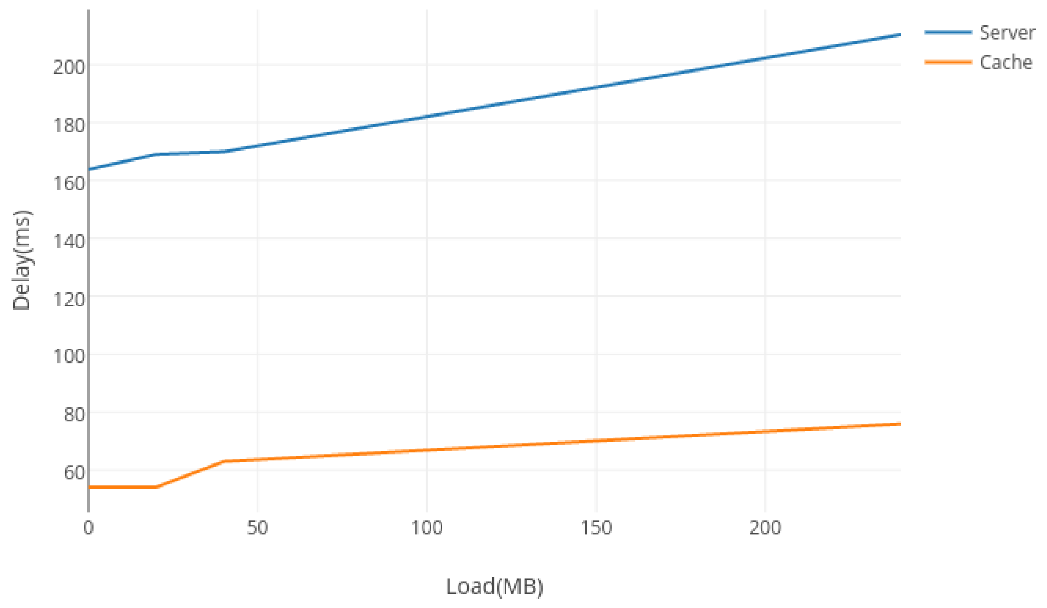


Figure 5.2: Delay vs Load (Single Content Request)

The first experiment was a simple experiment to demonstrate content retrieval latencies when retrieving content from the server (content provider) and an in-network cache. A content of size 16 MB was retrieved from the cache and server under varying loads as shown. To inject load into the network, additional clients were using to request other contents. As seen in Figure 5.2, the latencies seen for content retrieval from cache is much lower than that from the server. Also as the load increases the rate of increase in delay is higher for the server when compared to that of the cache.

5.2 Analyzing Delay and Cache hit ratios for N contents following a Zipf distribution

In today's Internet, content distribution follows a Zipf distribution. Hence the remaining evaluations were performed to simulate this scenario where contents were requested in a Zipf distribution. Each content is assigned a popularity value and popular contents were requested more than the less popular contents. Contents in batches of 5, 10 and 15 were used to study the overall delay experienced by clients over a period of time and the cache hit ratios of in-network caches. The Zipf exponent (alpha) was also varied to study its impact on delay and cache hit ratio. A total of 50 content requests were made in each of the experiments.

5.2.1 Case study 1 : Five contents distribution

A set of 5 contents were created of total size 81 MB. Cache sizes were fixed as a fraction of the total content size. Cache sizes were set to 20%(17MB), 30%(25MB) and 40%(33MB) of the total content size and were evaluated with zipf exponents of 0.75, 1.0 and 1.3. As stated above a total of 50 content requests were made in order to study metrics such as delay and cache hit ratios.

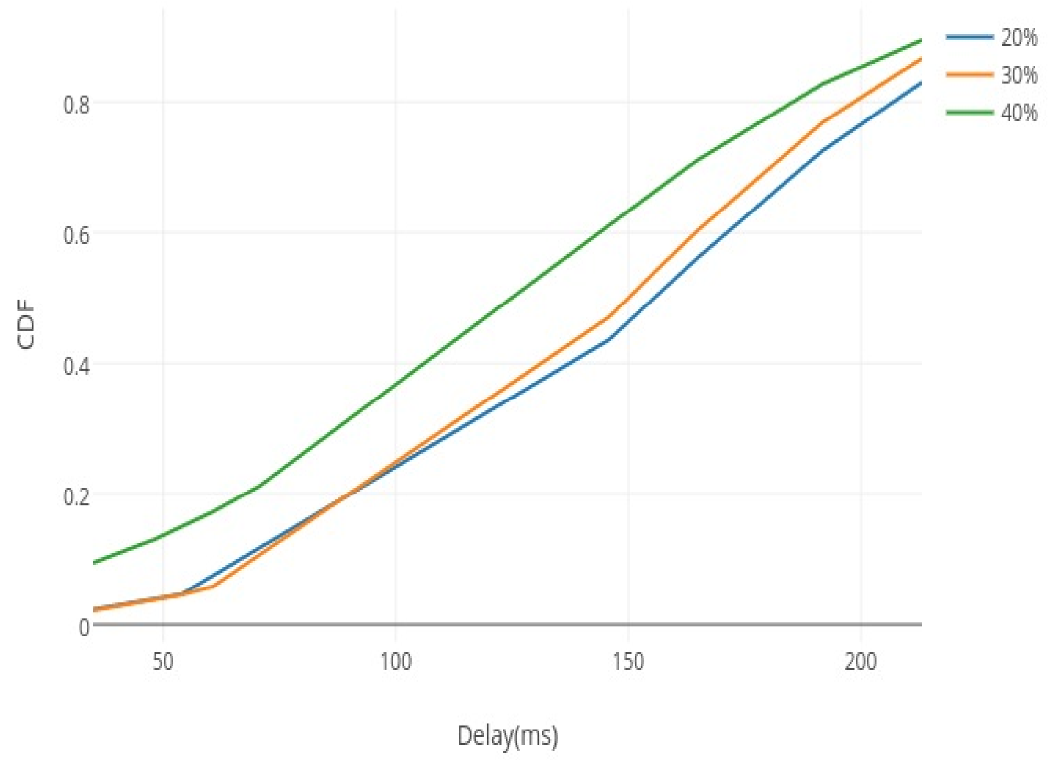


Figure 5.3: Delay CDF for alpha = 0.75

Figure 5.3 demonstrates the delay CDF for cache sizes 20%,30% and 40% of the total content with a Zipf exponent of 0.75. As seen in the figure, the overall delay for caches 20% and 30% are almost identical whereas the overall delay is lower for cache size 40%.

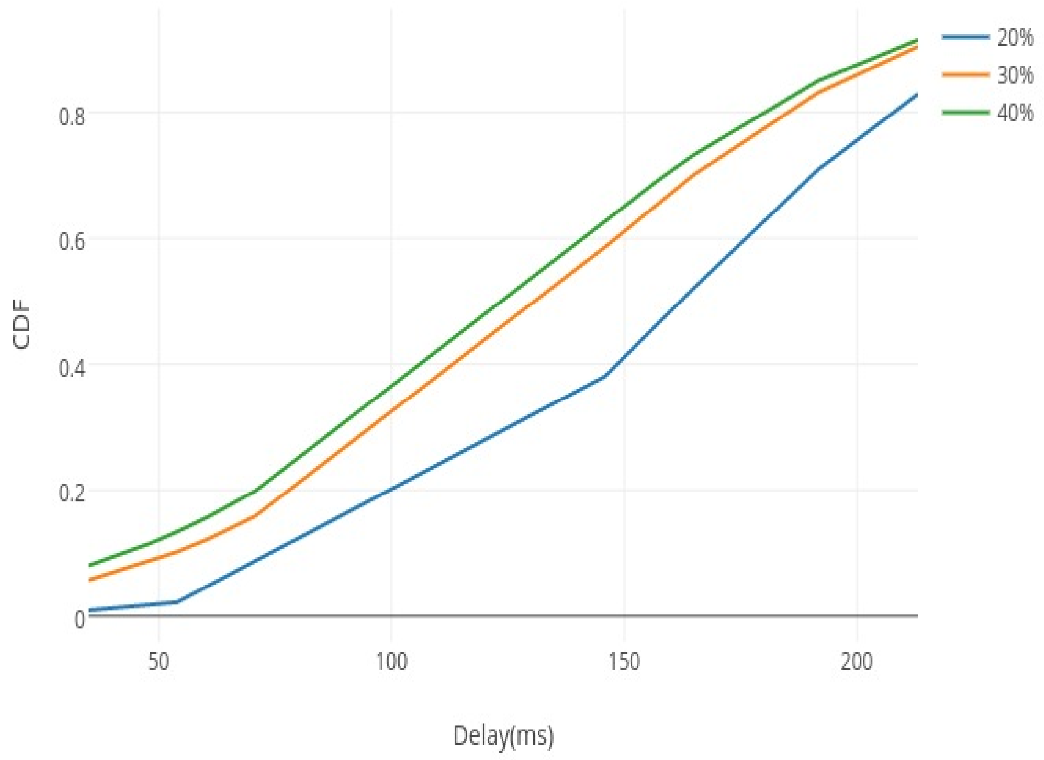


Figure 5.4: Delay CDF for alpha = 1

Figure 5.4 demonstrates the delay CDF for cache sizes 20%,30% and 40% of the total content with a Zipf exponent of 1.0. Here we see that the overall delay for cache size 30% and 40% is similar and is lesser than that with cache size 20%. This is because the number of requests for popular contents increases with increase in alpha and as a result caches with higher capacities capture a higher percentage of popular contents.

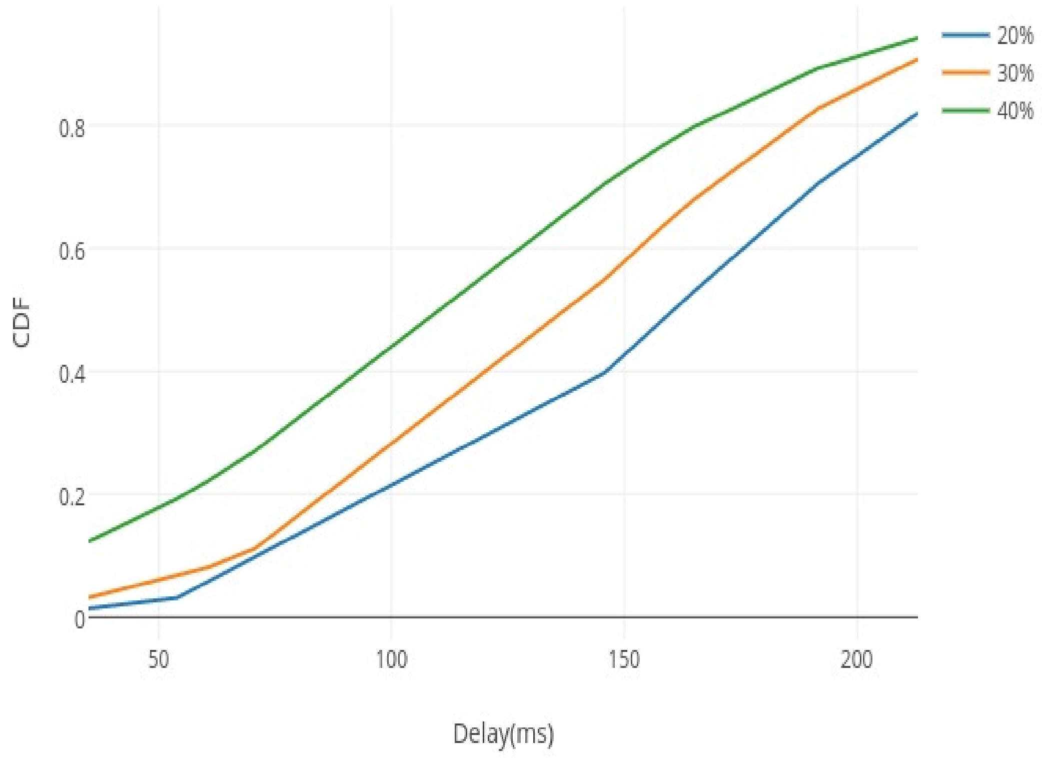


Figure 5.5: Delay CDF for alpha = 1.3

Figure 5.5 demonstrates the delay CDF for cache sizes 20%, 30% and 40% of the total content with a Zipf exponent of 1.3. Here we see that the delay is lesser in cache size 30% when compared to that of cache size 20% and similarly for cache size 40% when compared to cache size 30%. In all of the plots above we see that as the cache size increases there is a reduction in the overall delay. This is due to the fact that some contents are retrieved from the in-network caches. Hence for those contents, the delay experience is lower than that if it were retrieved from the server. This has also been demonstrated in Figure 5.2.

Cache Performance for different Zipf exponents

Figure 5.6 depicts the cache performance for 5 contents with varying cache sizes and Zipf exponents. As seen in the diagram the cache performance is better as the Zipf exponent increases. This is because the cache size is big enough to store the most popular content(or contents in some cases). Hence as the Zipf exponent increases, the most popular contents are requested much more than the least popular ones. Hence the cache hit ratio improves with increase in the Zipf value. Also it must be noted that the cache hit ratio improves as the cache size increases.

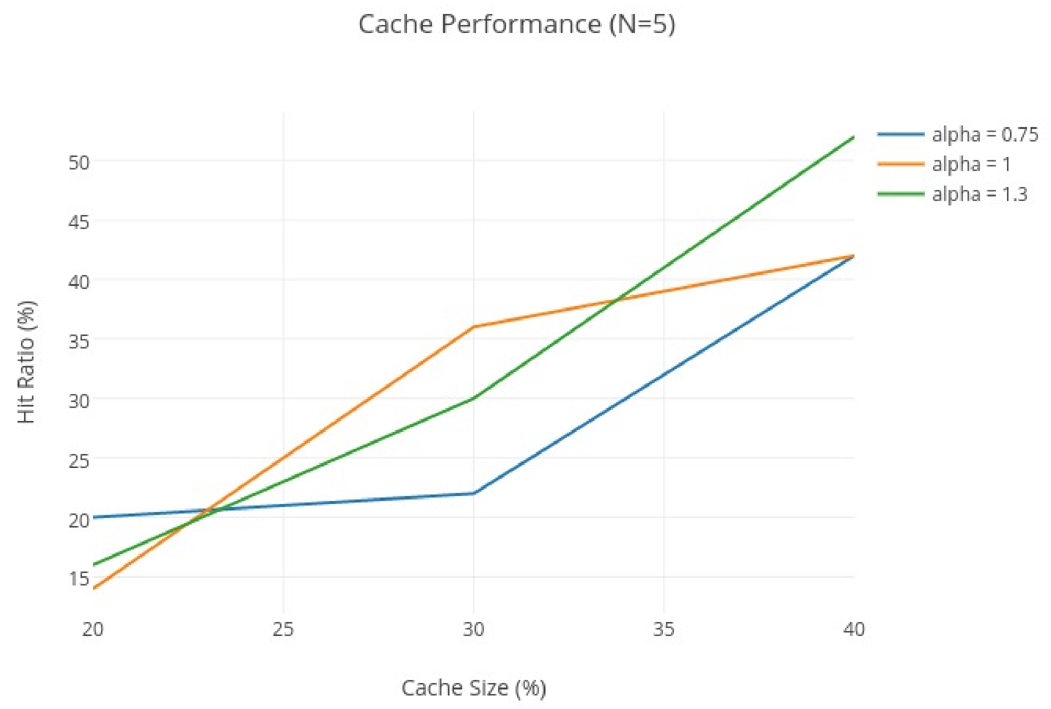


Figure 5.6: Cache Performance

5.2.2 Case study 2 : Ten contents distribution

A set of 10 contents were created of total size 265 MB. Cache sizes were fixed as a fraction of the total content size. Cache sizes were set to 10%(28MB), 20%(53MB) and 30%(80MB) of the total content size and were evaluated with Zipf exponents of 0.75,1.0 and 1.3. Figure 5.7 demonstrates the delay CDF for cache sizes 10%, 20% and

30% of the total content with a Zipf exponent of 0.75. A total of 50 content requests were made in order to study metrics such as delay and cache hit ratios.

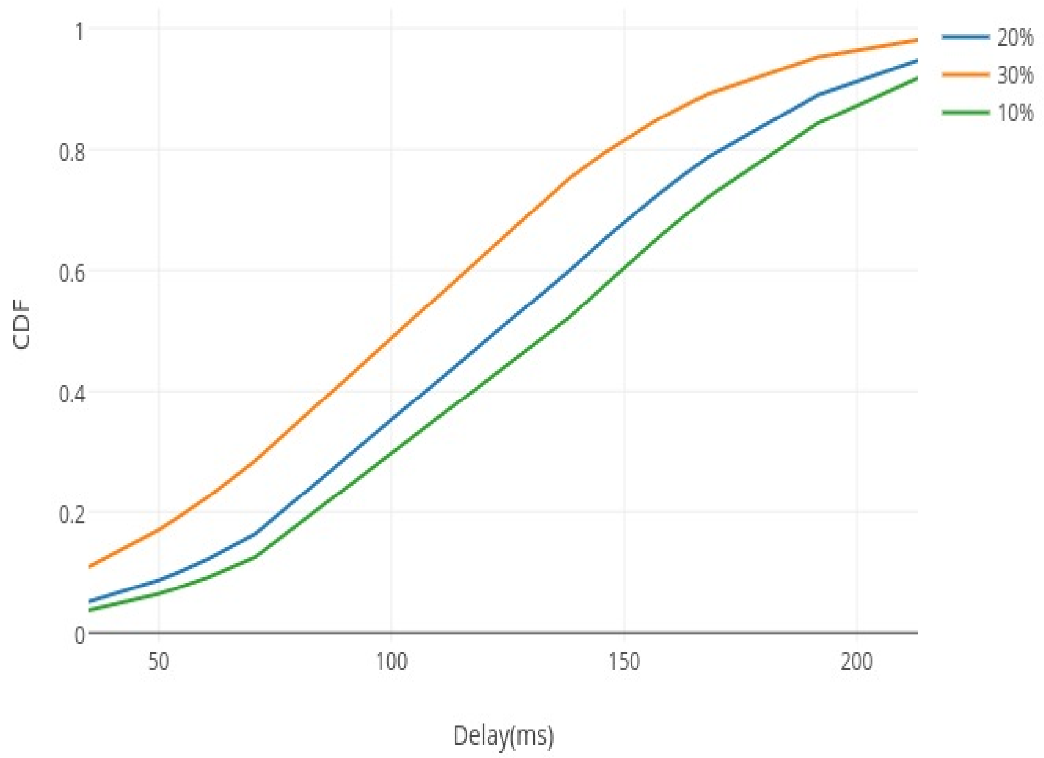


Figure 5.7: Delay CDF for alpha = 0.75

As seen in Figure 5.7, having a cache size of 30% results in lower overall delay when compared to that of cache size 20% and cache size 10%.

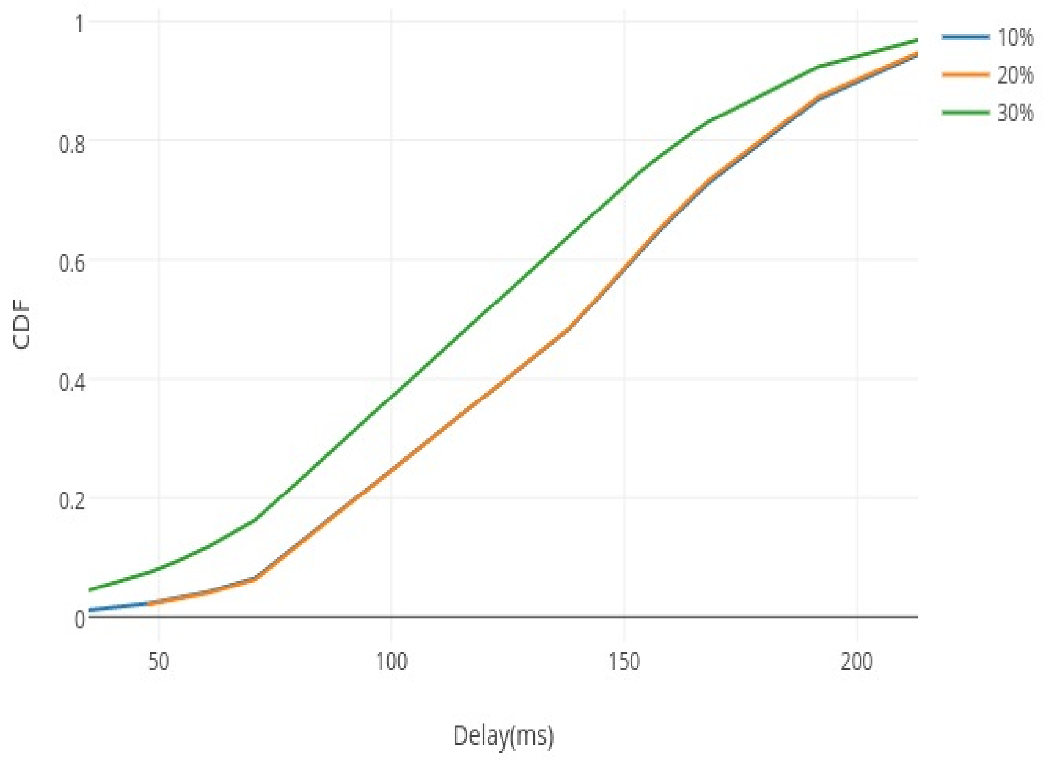


Figure 5.8: Delay CDF for alpha = 1

Figure 5.8 demonstrates the delay CDF for cache sizes 10%, 20% and 30% of the total content with a Zipf exponent of 1.0. As seen in Figure 5.8 the CDF curve is overlapping for cache sizes 10% and 20% which implies similar overall delay. However we see that for cache size 30% we get a reduction in the overall delay. Cache size 30% captures a higher fraction of popular contents when compared to that of 20% and 10%. However it must be noted that none of these cache sizes capture the most popular content which is nearly 37% of the total size of all contents.

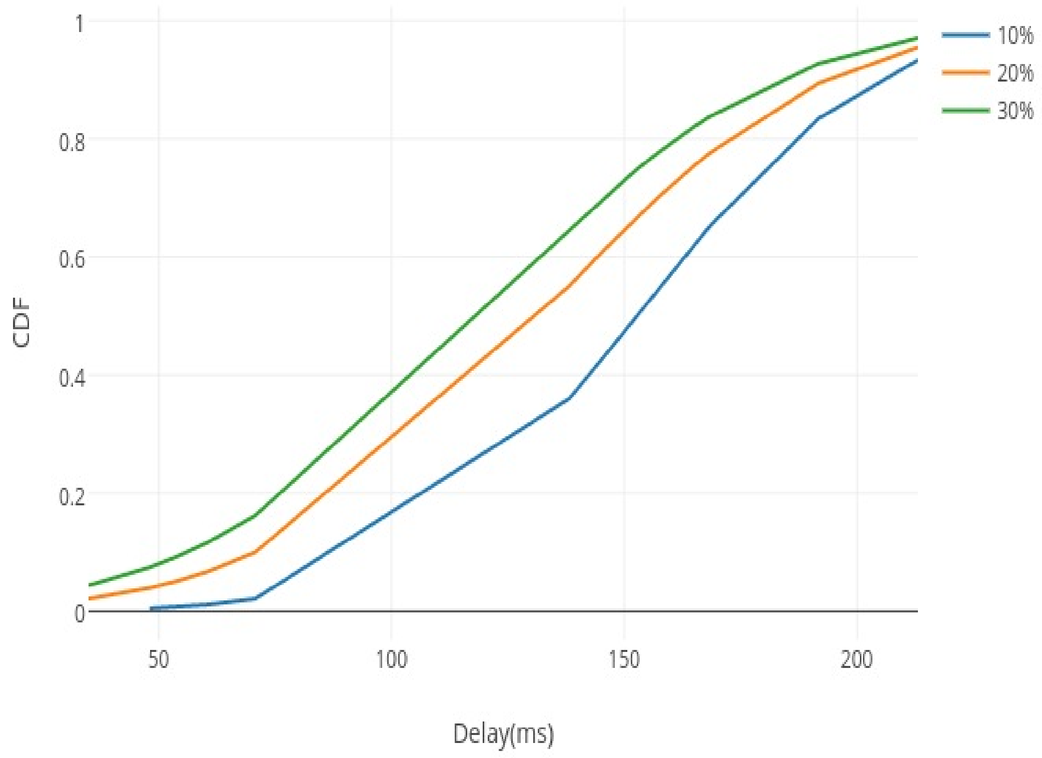


Figure 5.9: Delay CDF for alpha = 1.3

Figure 5.9 demonstrates the delay CDF for cache sizes 10%,20% and 30% of the total content with a Zipf exponent of 1.3. As seen in Figure 5.9, the overall delay is the least for cache size 30% followed by cache size 20% and cache size 10% respectively.

In all of the plots above, similar to N=5 contents, we see that as the cache size increases there is a reduction in the overall delay. This is due to the fact that some contents are retrieved from the in-network caches. Hence for those contents, the delay experience is much lesser than that if it were retrieved from the server.

Cache Performance for different Zipf exponents

Figure 4.4 depicts the cache performance for 10 contents with varying cache sizes and Zipf exponents. The cache performance degrades in this case as the Zipf exponent increases. This is because the most popular content is so big that it is either too big to fit in the cache or occupies more than 75% of the available cache size. Hence as the Zipf exponent increases, the most popular contents are requested much more than the least popular ones and result in more cache misses. However it must be noted that the cache hit ratio improves as the cache size increases. For distributions with alpha = 1.3, caches with smaller sizes experience higher cache misses when compared to that with alpha = 1.0 as alpha plays a dominant factor.However the increase in cache size compensates for alpha and hence distributions with alpha = 1.0 and alpha = 1.3 tend towards similar cache hit ratios with increase in cache size.

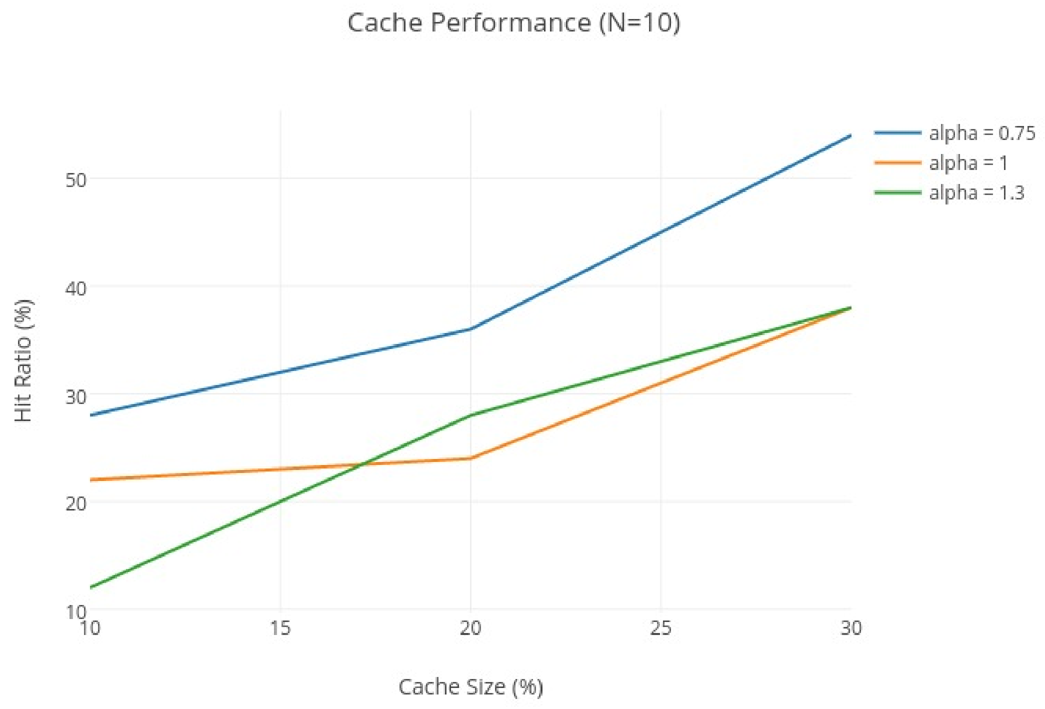


Figure 5.10: Cache Performance

5.2.3 Case study 3 : Fifteen contents distribution

For the final batch of experiments,a set of 15 contents were created of total size 335 MB. Cache sizes were fixed as a fraction of the total content size. Cache sizes were set to 10%(34MB) and 20%(67MB) of the total content size and were evaluated with Zipf exponents of 0.75, 1.0 and 1.3. A total of 50 content requests were made in order to study metrics such as delay and cache hit ratios.

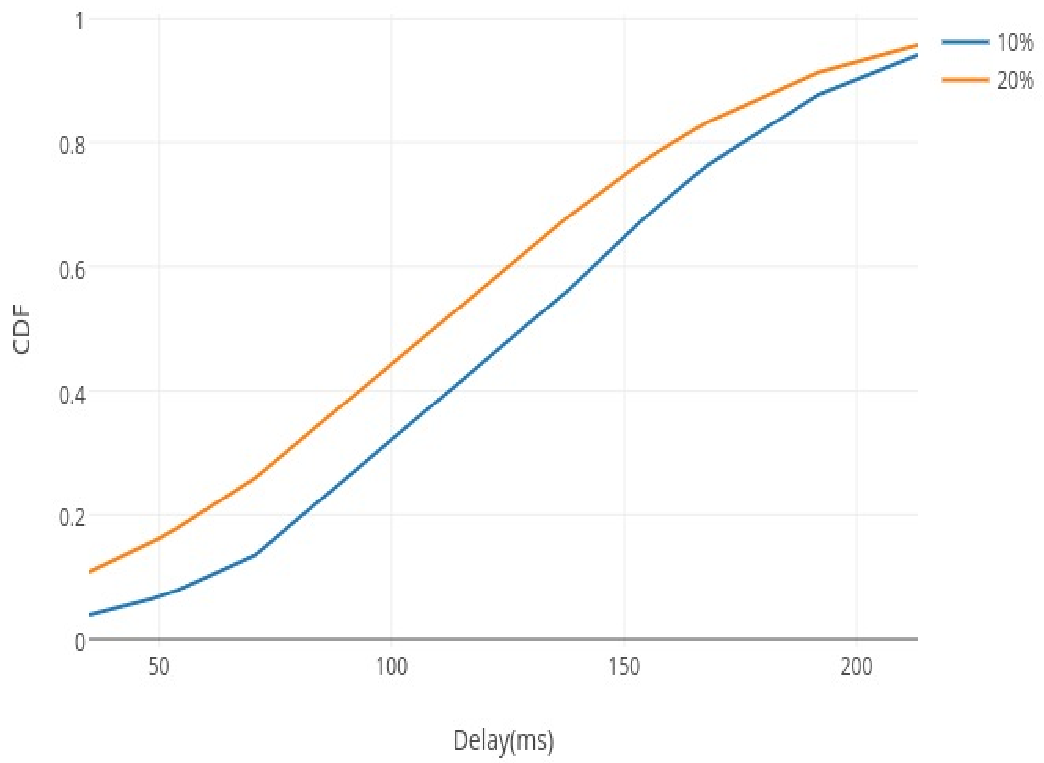


Figure 5.11: Delay CDF for alpha = 0.75

Figure 5.11 demonstrates the delay CDF for cache sizes 10% and 20% of the total content with a Zipf exponent of 0.75. In the above figure overall delay is lesser for cache size 20% when compared to that of cache size 10%.

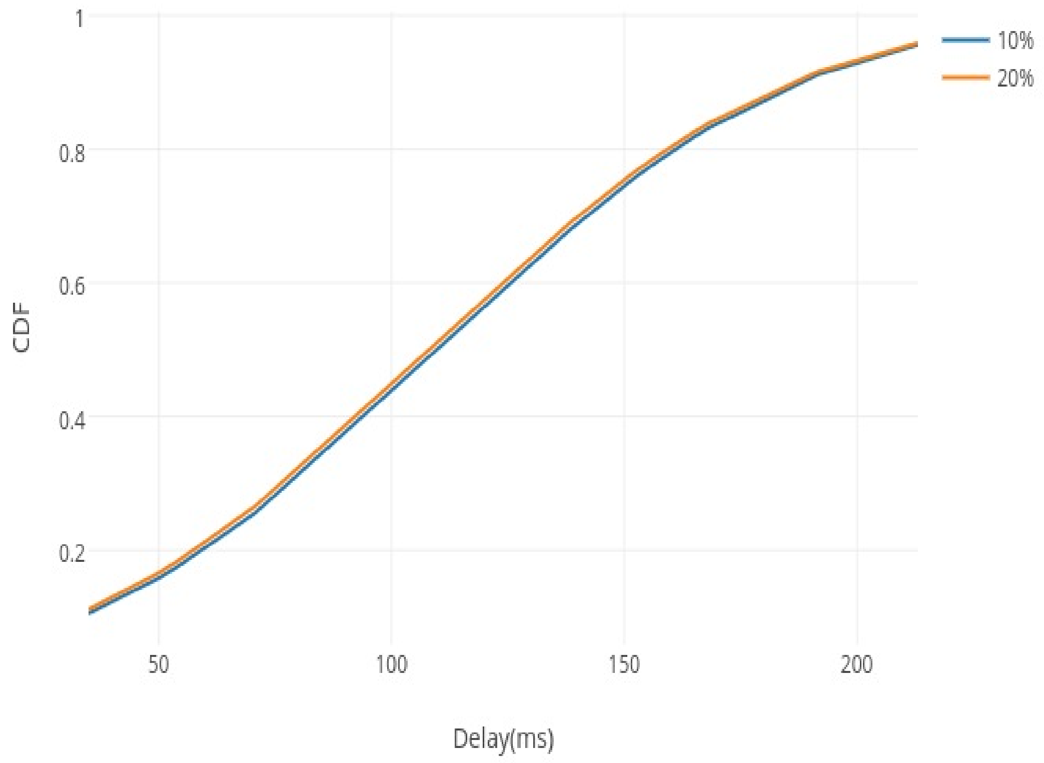


Figure 5.12: Delay CDF for alpha = 1

Figure 5.12 demonstrates the delay CDF for cache sizes 10% and 20% of the total content with a Zipf exponent of 1.0. In the above figure overall delay is identical for cache size 20% when compared to that of cache size 10%.

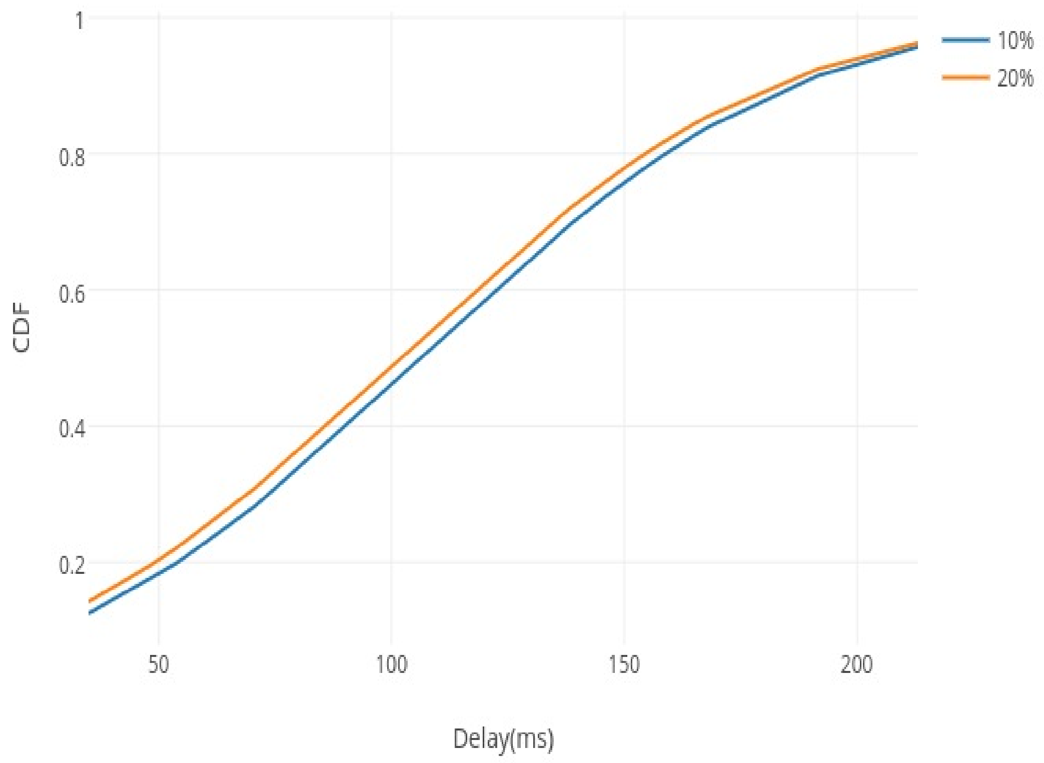


Figure 5.13: Delay CDF for alpha = 1.3

Figure 5.13 demonstrates the delay CDF for cache sizes 10% and 20% of the total content with a Zipf exponent of 1.3. In the above figure overall delay is slightly lesser for cache size 20% when compared to that of cache size 10%. In all of the plots above,similar to N=5 and N=10 we see that as the cache size increases there is a reduction in the overall delay. This is due to the fact that some contents are retrieved from the in-network caches. Hence for those contents, the delay experience is much lesser than that if it were retrieved from the server. Thus from all of the above cases (N=5, N=10 and N=15) , the cache hit ratio improves with increase in cache size.

Cache Performance for different Zipf exponents

Figure 5.4 depicts the cache performance for 15 contents with varying cache sizes and

Zipf exponents. The cache performance improves as the zipf exponent increases.Also it can be noted that even having a small cache size (10% of the total content) yields a good cache hit ratio. In fact it is much higher than that for N=10 with a similar cache size (10% of total content size). This is because the most popular content occupies less than 50% of the total cache size in all cache sizes. Hence as the Zipf exponent increases, the most popular contents are requested much more than the least popular ones and results in increased cache hits. Also, it is noted that the cache hit ratio improves as the cache size increases.

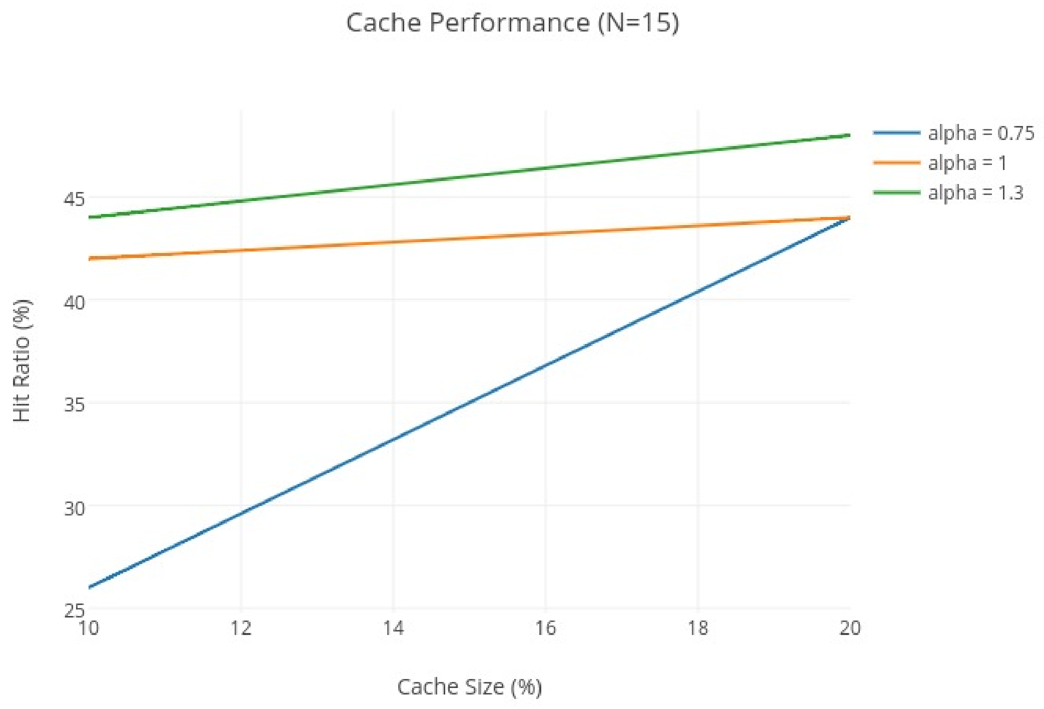


Figure 5.14: Cache Performance

Chapter 6
Conclusion and Future Work

In this work, a satellite-assisted content delivery network was proposed over the MobilityFirst Future Internet Architecture. Our architecture uses self-certifying content names to enable low cost content validations for consumers,content provider and the network. Content objects are fetched efficiently from the ”nearest” available replica taking the network status into consideration. We designed a push mechanism that provides support for services such as multicast and broadcast and also enables pro-active caching of contents. Content providers maintain priority queues and have the ability to pro-actively push out popular contents to multiple cache locations. Finally we use in-network caches in order to improve the efficiency of content delivery. We built a prototype system to demonstrate the feasibility of this architecture and evaluations on the ORBIT testbed showed that in-network caches achieve a reasonably high hit ratio even by storing only a portion of the total content and reduce the overall content delivery latency.

6.1 Future Work

Pro-active caching of popular contents improves the efficient of content delivery. However efficient methods of determining content popularity needs to be further evaluated. Content providers have the best view of content distribution and hence they can utilize efficient algorithms to determine content popularity.

Transmitting data through the satellite implies a certain probability of losing data. Hence modules have to be placed in the MobilityFirst routers to efficiently detect and retrieve lost content from the nearest available content replica. Efficient queue scheduling mechanisms need to be adopted at the MobilityFirst routers and the content provider as the satellite could be a potential bottleneck. Finally different caching techniques should be studied and evaluated in order to maximize the benefits of in-network caches.

References

[1] Jiachen Chen,Haoyuan Xu,Shashikanth Penugonde,Yanyong Zhang and Dipankar Raychaudhuri.Exploiting ICN for Efficient Content Dissemination in CDNs. *2016 Fourth IEEE Workshop on Hot Topics in Web Systems and Technologies (HotWeb)*, 2016

[2] MobilityFirst project, http://mobilityfirst.winlab.rutgers.edu/

[3] Dipankar Raychaudhuri, Kiran Nagaraja, and Arun Venkataramani. Mobilityfirst: a robust and trustworthy mobility-centric architecture for the future internet. *ACM SIGMOBILE Mobile Computing and Communications Review*, 2012

[4] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard. Networking Named Content. In *CoNEXT*, 2009.

[5] S. C. Nelson, G. Bhanage, and D. Raychaudhuri. GSTAR: Generalized storage-aware routing for MobilityFirst in the future mobile Internet. In Proceedings of *MobiArch*, pages 19-24, 2011.

[6] Tam Vu, Akash Baid, Yanyong Zhang, Thu D. Nguyen, Junichiro Fukuyama, Richard P. Martin, Dipankar Raychaudhuri. DMap: A Shared Hosting Scheme for Dynamic Identifier to Locator Mappings in the Global Internet. In *IEEE ICDCS, 2012*

[7] Yi Hu, Roy D. Yates, Dipankar Raychaudhuri A hierarchically aggregated in-network global name resolution service for the mobile internet.

[8] Shreyasee Mukherjee, Francesco Bronzino, Suja Srinivasan, Jiachen Chen and Dipankar Raychaudhuri. Achieving Scalable Push Multicast Services Using Global Name Resolution. *IEEE GLOBECOMM*, 2016.

[9] X. Tie, A. Sharma, and A. Venkataramani. A global name service for a highly mobile internetwork. Technical report, UMASS, Tech. Rep., 2013.

[10] Arun Venkataramani, Abhigyan Sharma, Xiaozheng Tie, Hardeep Uppal, David Westbrook, Jim Kurose, Dipankar Raychaudhuri. Design requirements of a global name service for a mobility-centric, trustworthy internetwork. In *IEEE COMSNETS*, 2013.

[11] Dipankar Raychaudhuri, Ivan Seskar, Max Ott, Sachin Ganu, Kishore Ramachandran, Haris Kremo, Robert Siracusa, Hang Liu, and Manpreet Singh. Overview of the orbit radio grid testbed for evaluation of next- generation wireless network protocols. In *In Wireless Communications and Networking Conference, 2005 IEEE*, volume 3, pages 16641669. IEEE, 2005.

[12] Feixiong Zhang, Chenren Xu, Yanyong Zhang, K. K. Ramakrishnan, Shreyasee Mukherjee, Roy Yates, Thu Nguyen. EdgeBuffer: Caching and prefetching content at the edge in the MobilityFirst future Internet architecture. In *World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, IEEE 2015.

[13] Feixiong Zhang,Yanyong Zhang,Dipankar Raychaudhuri. Edge caching and nearest replica routing in information-centric networking in *Sarnoff Symposium*,IEEE 2016

[14] Gerhard Hasslinger, Konstantinos Ntougias. Evaluation of Caching Strategies Based on Access Statistics of Past Requests. in *MMB & DFT 2014 Proceedings of the 17th International GI/ITG Conference on Measurement, Modelling, and Evaluation of Computing Systems and Dependability and Fault Tolerance - Volume 8376*,2014

[15] Christopher G. Brinton,Ehsan Aryafar,Steve Corda,Stan Russo,Ramiro Reinoso,Mung Chiang. An Intelligent Satellite Multicast and Caching Overlay for CDNs to Improve Performance in Video Applications. in *31st AIAA International Communications Satellite Systems Conference Florence, Italy*,2013

[16] Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 20162021 White Paper

[17] Nehal Somani,Abhishek Chanda,Samuel C. Nelson,Dipankar Raychaudhuri. Storage aware routing protocol for robust and efficient services in the future mobile Internet. in *2012 IEEE International Conference on Communications (ICC)*,2012

[18] Hofmann, Markus; Leland R. Beaumont (2005). Content Networking: Architecture, Protocols, and Practice. Morgan Kaufmann Publisher

[19] http://mobilityfirst.orbit-lab.org

[20] Shreyasee Mukherjee, Shravan Sriram, Tam Vu, Dipankar Raychaudhuri. EIR: Edge-Aware Inter-Domain Routing Protocol for the Future Mobile Internet

[21] http://read.cs.ucla.edu/click/click