Master’s Thesis
submitted in partial fulfillment of the
requirements for the course “Applied Computer Science”

ORICE: An Architecture for Object Resolution
Services in Information-Centric Environment

Sripriya Srikant Adhatarao

Institute of Computer Science

Bachelor’s and Master’s Theses
of the Center for Computational Sciences
at the Georg-August-Universität Göttingen

18. September 2015
I hereby declare that I have written this thesis independently without any help from others and without the use of documents or aids other than those stated. I have mentioned all used sources and cited them correctly according to established academic citation rules.

Göttingen, 18. September 2015
Abstract

Information Centric Networks (ICN) enable accessing data oblivious of its location, by allowing end-systems to retrieve content based on names. But, architectures such as Named Data Networking (NDN) and Content Oriented Publish/Subscribe System (COPSS) do not yet provide a mechanism for end-system applications to obtain these names. There is a need for an object resolution system that addresses a most important and as yet unimplemented component of obtaining names in ICN. This work proposes ORICE, an architectural design for Object Resolution services in Information-Centric Environment that satisfies this need. The architecture enables intelligent resolution service by placing the service in the application layer and allows for the service diversity by separating the name space management from resolution service. Through evaluation, this work shows that with the help of ORICE, the states stored in the network can be dramatically reduced while ensuring complete data delivery. A prototype is also built to demonstrate the feasibility of the design and the importance of such a system in ICN. Part of this work titled ‘ORICE: An Architecture for Object Resolution Services in Information-Centric Environment’ was published in IEEE LANMAN 2015 workshop and also in ICN 2015 a demonstration of the proposed architecture with the title ‘Prototype of an Architecture for Object Resolution Services in Information-Centric Environment’ has been accepted.
Contents

1 Introduction 1

2 Background and Related Work 5
  2.1 TCP/IP .................................................. 5
  2.2 Information Centric Networking .......................... 6
    2.2.1 Named Data Networking .......................... 6
    2.2.2 COPSS: An Efficient Content Oriented Pub/Sub System .... 8
    2.2.3 Differences between TCP/IP and CCN ................. 10
  2.3 Object Resolution .................................. 11
  2.4 Sampling ............................................. 12

3 Requirement Analysis 15
  3.1 Intelligence ......................................... 15
  3.2 Diversity ........................................... 15
  3.3 Scalability .......................................... 16
  3.4 Name Space Maintenance ............................. 16
  3.5 Efficiency .......................................... 16
  3.6 Cross Architecture Support ......................... 16

4 Design Rationale 19
  4.1 Separate Application and Network Layer Functionality .... 19
  4.2 Separate Name Space Management & Object Resolution .... 20
  4.3 Exploit Benefits Offered by ICN ..................... 21

5 Architecture 23
  5.1 Components ......................................... 23
  5.2 Architecture Description ............................. 24
    5.2.1 Query/Response ................................ 26
    5.2.2 Subscribe ....................................... 26
    5.2.3 Publish .......................................... 26
CONTENTS

5.3 Support for Multiple Layouts ................................................. 27
  5.3.1 Multiple Resolution Services ......................................... 27
  5.3.2 Private Resolution Service ........................................... 27
  5.3.3 Distributed Servers .................................................... 28

6 Implementation ................................................................. 31
  6.1 Scenario [Query/Subscribe] .............................................. 31
  6.2 Scenario [Post] ............................................................. 34
  6.3 Scenario [Reconnect] ...................................................... 35

7 Evaluation ........................................................................... 37
  7.1 Data Set ........................................................................... 37
  7.2 Small Scale Evaluation ...................................................... 42
  7.3 Large Scale Analysis ......................................................... 45

8 Conclusion and Future Work .................................................. 49
  8.1 Future Work ................................................................. 49

Bibliography ............................................................................ 54
# List of Figures

2.1 CCN forwarding engine model [1] ............................................. 7  
2.2 Pictorial representation of data processing in CCN nodes [1] ............... 8  
2.3 NDN Forwarding Engine with COPSS Subscription Table [2] ............... 10  
4.1 Layer division comparison among different approaches .................... 20  
5.1 Communication in ORICE .................................................. 24  
5.2 ORICE application-layer example ......................................... 25  
6.1 Scenario Query/Subscribe .................................................. 32  
6.2 Query/Subscribe View ...................................................... 32  
6.3 Scenario Post ............................................................. 33  
6.4 Post and Messages View ................................................... 33  
7.1 Wikipedia data set statistics: CDs per category ................................ 38  
7.2 Wikipedia data set statistics: Pages per category ............................ 38  
7.3 # of Ancestors/Category ................................................... 39  
7.4 # of Descendants/Category ................................................. 40  
7.5 # of Request/Category ..................................................... 41  
7.6 # of states in the network with $1.7 \times 10^6$ interests ...................... 43  
7.7 Message loss rate on different % of known CDs ............................. 44  
7.8 # of states needed for complete data delivery with $1 \times 10^9$ interests .... 45
Chapter 1

Introduction

Users primarily seek information from the network without necessarily focusing on its location or the underlying mechanisms used to retrieve that information. Conceptually, this functionality can be divided into three parts resolution of the name of the information object that the user is seeking (in this work it is called object resolution), name resolution (determining the location of the named object) and location based routing (delivering the user request to the repository of information). The object resolution system translates users inputs (keywords, etc. specifying the desired content) into names (URLs in the current IP networks). These systems are provided at the application layer since the translation might involve sophisticated logic and require access and processing of a large amount of data. Separating these functions from the network layer can keep the network simple and efficient.

There are various object resolution systems for example search engines like Google [3], Yahoo [4], etc., recommender systems like Last.fm, Netflix (that combine recommendations with object resolution) as well as publish/subscribe systems such as RSS feeds, etc. The name resolution system in the form of Domain Name System (DNS) [5] then translates these URLs to IP addresses. The IP network then uses these addresses for data distribution and retrieval.

Information-Centric Networking (ICN [6]) optimizes the data distribution and retrieval by integrating name resolution and routing in the network. It treats content as the first-class entity, rather than location of the end-points containing the information of interest. In ICN, nodes exchange information based on the Names of the content instead of the IP addresses of the end points requesting or providing the content. This shift from a “location-based” network to a “content-centric” network allows more efficient data dissemination, especially when the content may be available at multiple points, or the provider or consumer is mobile. Additional performance benefits accrue thanks to the widespread use of in-network caches.

However, the functionality that maps users inputs into identity of the content is often assumed to pre-exist in many ICN solutions. ICN solutions typically assume that the data consumers know the
identity (name) of content before they send requests or subscriptions. Recent ICN proposals such as Named Data Networking (NDN [1]) and Content Oriented Publish/Subscribe System (COPSS [2], which enhances NDN with an efficient publish/subscribe capability), adopt human-readable, hierarchically structured Names and Content Descriptors (CDs). This work clarifies the distinction between content names and content descriptors. A content name is the name (unique identity) of the container of the information which the user seeks. On the other hand, a content descriptor is used to identify specific items and attributes the data contains such as keywords, tags, date of publication, identity of the publisher etc.

CDs are specified by the publisher, and enables the publisher to control what properties should be associated with the data they publish. However, the proposed solutions so far do not explicitly specify how or from where a user finds the “ICN-names/CDs” for the information they seek, and assume that there are separate mechanisms to provide the names. There is a need for object resolution systems to fill this gap and thus enhance the usability of ICN.

Inherently different users have different preferences regarding the content they seek. There is a need for an architecture that can serve as the foundation for supporting any object resolution system, allowing the object resolution to focus on providing the service logic while leaving the communication primitives to ICN. With multiple resolution systems sharing the same underlying network architecture, it provides an opportunity for them to communicate with each other and provide better services. In this work it is believed that by exploiting the capabilities of ICN, object resolution systems can achieve the benefits such as:

1) Diversity – with multiple copies of a same data “aggregated” by the same name, the results from the resolution system can offer higher performance compared to the IP-based solutions.

2) Service scalability – service distribution and load balancing can be more easily achieved in ICN. In-network caches can further reduce the content server load by leveraging temporal locality of the search queries.

3) Retrieval/Dissemination efficiency – both data retrieval and publication can benefit from name-based routing and in-network caches, and

4) Data integrity – the integrity of the content and provenance can be assured by the signature of the original content provider being associated with the content.

This work proposes ORICE, an Object Resolution architecture for Information-Centric Environment. In particular, ORICE provides a foundation for building object resolution services that can help users to translate keywords to the names/CDs they might be interested in, and also find related CDs for publishers at the time of publication of their content. The proposed architecture can take advantage of the capabilities provided by object resolution services such as Google and Yahoo.

In the current Internet model, search engines such as Google and Yahoo which perform the object resolution maps the keywords into URLs by crawling the web to create an index of the web and
use these indexes to satisfy user’s search queries, thus deciding what content is made available. But this may not be what is desired from a publisher’s perspective. Publishers may desire control over what content should be used by object resolution systems to initially expose to users.

In ORICE architecture, CDs are used as the key feature to provide a clear separation of responsibility between publishers and object resolution systems. The publisher decides what CDs (including the name of the container) are associated with a piece of content and the object resolution systems uses these CDs for indexing. The object resolution systems is a repository of CDs. The object resolution system (functioning just as search engines do in the current Internet) may be used to suggest CDs appropriate for content to users. The publisher can potentially further enhance the control over the content by encrypting the content and only leaving the associated CDs to be used by the object resolution systems for providing suggestions to users. To allow multiple resolution systems to work on the same CD space, ORICE separates the logic of CD certification (that manages the CD space) and the logic of CD resolution (that finds proper CDs in the CD space).

The key contributions of this work are listed below:

- An architecture that enables object resolution systems to provide intelligent functions at the application layer and leverages efficient data dissemination in the network,
- A separation between the name certification and object resolution that can satisfy more diverse requirements and content reliability,
- An implementation by building a prototype of the architecture to showcase the efficiency and necessity for an object resolution system in ICN,
- An evaluation of ORICE via Wikipedia data. Evaluation shows that with the help of ORICE, the states in the network can be dramatically reduced while still ensuring complete data delivery.
Chapter 2

Background and Related Work

This chapter describes the current Internet architecture along with the recent Information Centric Network proposals and mechanism of object resolution in these architectures. A brief description on sampling technique is also included here to provide the necessary background knowledge for the evaluation carried out on the Wikipedia data set described in detail in the Evaluation chapter.

2.1 TCP/IP

The classic model of today’s Internet architecture revolves around hosts and connections between them. There is a host which is the source and it has some data or resource(s) and there is another host which is a receiver and it needs this data or wants to use these resources. The receiver initiates a request for data and establishes a connection with the source and once a successful connection is established the data is transferred from the source to the receiver on this established connection. This design was created to satisfy the needs of the Internet in the 1960’s and has been in use since then.

This architecture was designed to solve the problem of resource sharing as the resources were expensive and scarce during the old times and it has successfully addressed this problem. But now the problem has changed. The resources such as hardware are now less expensive and are not scarce anymore. Today’s problem is; there is an ever increasing need for content and its distribution to multiple receivers in a timely, reliable, effective and efficient manner irrespective of the location of the data.
2.2 Information Centric Networking

The success of TCP/IP architecture has resulted in usage of Internet to such a great magnitude that it has resulted in the emergence of research to find a solution that better fits today’s problem. One such class of research is Content Centric Networking (CCN). Here the emphasis is on retrieving the content independent of the location and identity of its hosts. In CCN the term ‘named data’ is used to identify the content and to retrieve this named data from anywhere in the network irrespective of its location or the identity of the host containing the data.

2.2.1 Named Data Networking

Recently there have been several proposals for CCN [1, 7–14]. In NDN there are two types of packets Interest and Data. The consumer initiates a request for content by sending Interest into the network and this Interest is routed to any location in the network where the content or a copy of it is present and Data will be sent back to the consumer. There are times when an Interest may be sent by a consumer for Data that does not exist (yet), in which case the publisher will dynamically generate the Data which will consume this Interest. In CCN the Interest will be broadcasted by the consumer on all available connections.

Fig. 2.1 shows the forwarding engine model of NDN and Fig. 2.2 shows data processing in NDN architecture. The Forwarding Information Base (FIB) gives a list of potential sources that may satisfy the Interest. The Content Store (CS) keeps a copy of the Data as long as possible (depending on the caching policies) to increase the probability of availability of the Data for future Interests as this will reduce the response time to the consumer and also the demand for upstream bandwidth. Pending Interest Table (PIT) is used to keep track of the Interest that have been forwarded upstream to the potential data sources in order to send the data back to the consumer when the Data arrives.

Whenever an Interest is received by any node in the network a longest match look up is made on its name. The node checks if it has a stored copy of the requested content in the content store and if a copy is present then it sends Data back to the consumer. If there is no copy then PIT is checked to see if an Interest with the same name has already been forwarded to fetch the Data. If an entry is found then the interface from which the Interest was received is added to a list maintained in PIT for that entry to send the Data when it arrives. Otherwise FIB is checked and if an exact match is found then the Interest is forwarded and a PIT entry is made. If there is no match found for the Interest then the Interest is discarded as the node does not have the requested data and it has not been able to find any source that could provide it. The Data follows the trail left by the Interest in the network. When Data arrives at a node in the network a lookup of the longest matching name of the content is made. If there is an exact copy present in the Content Store then this is considered to be a duplicate and it is discarded. On the other hand, if it matches a FIB entry then it means that there is no matching PIT entry and hence this is an unsolicited piece of data and is discarded. If a
matching entry is found in PIT then this indicates that the Data needs to be sent downstream and is sent out on all the interfaces that had requested for this Data via Interest. When Data is travelling in the network intermediate nodes can store a copy of it in Content Store to satisfy subsequent request for the same data.

From the above description of the NDN architecture we can see that CCN is trying to address some of today’s requirements with its proposal. We can say that CCN architectures try to address today’s problem in contrast to current Internet architecture where the consumers want content as fast as possible and this can be achieved due to the stored copies of data in the intermediate node if present. In NDN the Data is secured and not the link it traverses to reach the consumer so this provides a secure delivery of the data. NDN also suggests re-expressing of the Interest if Data has not arrived after waiting for it for a period of time (this may happen if there was delay in arrival of Data due to congestion or PIT might have discarded the Interest entry after timeout while waiting for Data) and Data is still required by the consumer and this ensures reliability. So when content is required the consumer can just send out an Interest containing the name of the data and let the network handle this request. This architecture tries to adopt the features of TCP/IP such as simplicity, scalability, robustness by having a network layer which is similar to IP and makes less demand on the underlying layer. On the whole CCN is an Internet architecture that is targeted to address the current requirements.

Figure 2.1: CCN forwarding engine model [1].
2.2.2 COPSS: An Efficient Content Oriented Pub/Sub System

In COPSS [2] the authors say that recently, the publish/subscribe (pub/sub) systems have gained popularity and are used increasingly to remove the need for temporal dependency on users to explicitly query the Internet to retrieve each piece of data. But these systems are still location based and the users still have to know the location of the publisher to query for updates. This results in inefficient use of the design specifically with data forwarding and management at the user’s end. There have been various proposals including NDN, IP multicast [15–17] and overlay multicast [18–20] to improve the current pub/sub situation, however they do not provide a complete pub/sub solution for content based network. Hence, the authors propose COPSS to extend the initial NDN architecture by providing publish/subscribe capability in CCN.

The existing proposals for pub/sub systems can be broadly classified into two types: one is a pull based method and the other is a push based method. In pull based method the users continuously or periodically poll the publishers (proxy servers) for any update to their subscribed piece of information. In the pull based method the servers incur unnecessary overhead in network bandwidth and the servers incur unnecessary computational overhead as the information may not be updated as frequently as the server is polled by users. Additionally the pull based mechanism necessitate the need for users to know the location of the publisher for polling.

Whereas, in the traditional push based pub/sub mechanisms various ways including maintaining a long-lived TCP connection between the server and subscribers [21] or using Instant Messaging
or Rendezvous points to inform subscribers about the available updates. The traditional pub/sub solutions have to maintain a lot of connections between the subscribers and publishers and also the number of states that need to be stored is high and hence are not scalable. Also, these solutions bring forth the concern of publishers and the subscribers revealing each others identities but, due to network address translations the subscribers cannot practically have a global visibility. This further complicates the push based solutions operations in a wide scale.

On the other hand there have been a few overlay proposals like the SpiderCast and the Astrolabe but these solutions are unaware of the underlying network topology and lead to increased overhead. Another class of proposals include solutions from ONYX, TERA, Sub-2-Sub and SpiderCast where the subscribers express their interest in any of the topic/content rather than the location from where the information has to be subscribed. In RSS feeds and XMPP based solutions subscribers subscribe to topics/publishers and the system publishes the frequently updated messages to the subscribers. These solution seem to be push based however they are actually doing a pull based publication because the RSS and XMPP sources are polled frequently for updates.

In COPSS, the authors investigate and find that there is need for enhancements to the initial query/response design of NDN to support publish/subscribe applications in NDN. A recent proposal in NDN ChronoSync synchronizes the repositories of data providers to provide a pub/sub like communication but it needs the users to know the prefix of all the data providers who has the content of their interest and it is still polling based. The IP multicast solutions also provide mechanism for pub/sub but they are still not efficient for a content based architecture. COPSS enhances NDN by providing a push based multicast capability in CCN.

COPSS uses the concept of Content Descriptors (CDs) similar to XTreeNet and SEMANDEX for pub/sub in CCN. A CD can be any information that the publisher wishes to associate with any piece of Data that they wish to publish. COPSS uses context based and hierarchical CDs to help in aggregation.

To support the pub/sub communication COPSS introduces two new types of packets namely the Subscribe packet and the Publish packet. There are multiple senders and multiple receivers and various Rendezvous points (RP). The subscribers subscribe to CDs associated with the content of their interest. Each CD is in turn associated with a RP. It is the responsibility of the publishers to associate every piece of information with the related CDs. Whenever the publisher wishes to publish any Data, they associate the Data with CDs and send it to the network. When a message is published, it is forwarded towards the respective RP along the COPSS multicast tree. The COPSS aware routers maintain a Subscription Table (ST) that records the subscriptions downstream as shown in Fig. 2.3. ST maintains the CDs in a distributed and aggregated fashion. Whenever a COPSS aware router receives a Data, it is forwarded along any face if there are subscribers downstream for the CDs in received Data packet.
2.2.3 Differences between TCP/IP and CCN

Some of the noticeable differences between CCN and TCP/IP would be that in TCP/IP the data transfer involves establishing a connection between two hosts and the link is secured for data transfer whereas in CCN the data packet carrying the data will be secured so no matter which link the data packet uses to reach the consumer the data will be secured. In IP with plain TCP/IP the Forward Interest Table gives a single best outgoing face that can be used for the communication between the hosts whereas in CCN the Forward Interest Table gives a list of outgoing faces which can be used in parallel to forward the Interest and retrieve Data. In TCP/IP the data packet is forwarded and forgotten at intermediate nodes whereas in CCN the storage capability in the intermediate nodes is utilized and the Data is stored while in transit at intermediate nodes so that subsequent request if any for the same Data can be satisfied at a much faster pace than having to relay the request to the actual source and waiting for the data to be sent all way from the source to requester. But the Interest and/or Data might get lost or damaged while in transit and may not reach the consumer and it turns out that if this Data is required then it is the responsibility of the Consumer and not the Sender to re-express the Interest and due to this the senders in CCN are stateless whereas in TCP/IP the senders and receivers are stateful.
2.3 Object Resolution

In the current IP architecture the object resolution is usually performed with the help of the search engines like Google, Yahoo, Bing, etc. in the web. According to Wikipedia, the web search engine is designed for searching information in the world wide web. The search engines first crawl the web with a Web crawler - an automated system that crawls every link in the Internet to its associated site. After crawling, the search engine analyzes the content from each of the web pages and indexes them for object resolution. When a user issues a query to a search engine usually by providing some search keywords, the search engine looks up the index and returns a list of URL's to the user. This is object resolution. The user can then use any of the URLs from the suggestion and start browsing. Internally the browser will send the URL to the Domain Name System (DNS) server for obtaining the IP address of the destination. According to Wikipedia DNS is a distributed naming system for any resources in the Internet. Domain names in DNS are organized and maintained in a hierarchical fashion. The main function of the DNS is to translate the URLs to IP address and this is referred to as name resolution.

Existing object resolution architectures are built for operating in IP networks which is location centric. However, ICN with a “content-centric” architecture provides numerous benefits. The existing architectures are well suited for IP network but they are not suitable candidates for ICN because they do not exploit the benefits that ICN offers. Such systems can only translate the user inputs into the IP address of the location whereas in ICN the requirement is to just translate the user input into name/CDs then the underlying network takes care of finding the location of the content in an efficient manner. Hence, this work observes that the existing solutions are not well suited for ICN environment and there is a need for a different architecture that is well suited for operating in ICN.

In ICN (e.g., MobilityFirst, globally unique identifiers (GUID) are used to represent objects attached to network such as users, devices, etc. A global name resolution service maps GUIDs to network address (similar to a name resolution service). However, there is no system that assists users in obtaining the actual name of the content. In DONA, flat names are used for replacing the names stored in DNS. It uses anycast primitives to support its functions. DONA assumes that users can obtain these flat names from some external mechanisms such as search engines, recommender services, etc. PSIRP proposes to use rendezvous points for pub/sub however, there is still a need for a system to recommend name for the object of interest to subscribers and publishers. All of the above mentioned proposals either assume that user knows the name of interested content or can obtain it. However, none yet provide a concrete solution for addressing this issue. Sub2Sub, SpiderCast and ONYX provide object resolution in the (overlay) network layer with the help of brokers. But, such a design unnecessarily complicates the network functions by incorporating the intelligence required for translation at the network.
layer. These solutions also assume that users know the identity of the data object where as lack to provide a mechanism for its retrieval.

2.4 Sampling

Sampling techniques for graphs can be broadly classified into two categories namely node sampling and edge sampling. The BFS and MHRW are node sampling techniques whereas FS is an edge sampling technique. BFS is a widely accepted and used graph sampling algorithm whereas MHRW and FS are relatively new techniques. During sampling one has to take in to consideration the fact that the graph exhibits various properties which need to be retained and verified in the sample graph before the sample graph can be used as a representative of the original graph. Two of the most important properties that need verification are Node Degree Distribution (NDD) and Clustering Coefficient (CC).

Node Degree Distribution (NDD): Every graph has a set of nodes and edges. In a directed graph the nodes can have edges flowing towards the node and also edges flowing outwards from the node to another node in the graph. The number of edges pointing towards the node is called as the in-degree of the node whereas the number of edges pointing outwards from a node is called as the out-degree of the node. In a graph sampling algorithm it is very important that a sample of the original graph retains the same in-degree and out-degree distribution of the edges as that of the original graph.

Clustering Coefficient (CC): The Wikipedia definition of clustering in sampling states “Cluster sampling is a sampling technique used when “natural” but relatively homogeneous groupings are evident in a statistical population. It is often used in marketing research. In this technique, the total population is divided into these groups (or clusters) and a simple random sample of the groups is selected” [37]. So this is a measure of how the nodes in a graph tend to cluster together. In order for a sampled version of the graph to be a good representative of the original graph, the sampled version of the graph should exhibit same clustering coefficient as the original graph.
Chapter 3

Requirement Analysis

In order to testify itself as a complete object resolution system, there are quite a few requirements which should be addressed by such a system. Some of these requirements that “ideal” object resolution systems should incorporate are described in this chapter.

3.1 Intelligence

It has been observed that the scale (number and size) of data objects in the Internet is increasing at an enormous rate [38]. To efficiently process (index) such a large amount of data to make it available to users, object resolution systems (e.g., search engines, recommender systems) adopt complex machine learning/artificial intelligence techniques, using very large scale data centers. Such an intelligent logic needs to be supported by advanced hardware in order to achieve higher quality. But this complexity should not affect the transmission efficiency over the network. Since the object resolution systems (e.g., search engines, recommendation systems) usually involve sophisticated Artificial Intelligence (AI) logic and might need to access a large database while processing user requests, the architecture should allow such intelligence implemented in a flexible manner while not affecting data transmission efficiency.

3.2 Diversity

Data consumers usually have different preferences on the resolution services, e.g., researchers would prefer papers to films while looking for technical material; artists might favor images rather than text while looking for inspiration. We can learn from the different resolution systems deployed in the Internet, which help to satisfy these preferences. Similarly, the target architecture should also provide support for diverse resolution systems.
3.3 Scalability

Modern object resolution systems have to deal with large amount of user requests. Many of them take advantage of distributed systems (e.g., MapReduce, NoSQL database) to provide scalable services. The desired object resolution architecture should allow the distributed system to be easily designed and deployed in the network.

3.4 Name Space Maintenance

A topic-based name space in the form of an ontology for information is more intuitive to users compared to other kinds of name spaces. Such name spaces can also maximize the data dissemination efficiency and minimize forwarding overhead. Consider a name space having /sports as a top-layer interest and /sports/football, /sports/basketball, etc. as its children. When a data consumer tries to express the interest of “everything related to sports”, he can choose to use the top-layer interest instead of specifying all the lower-layer names. This aggregation reduces the entries (states) stored in the network and also simplifies the procedure with which users express interests. However, an unorganized name space can do the contrary – a user might have to use many interests just to satisfy one requirement. Therefore, a name space maintenance (certification) system should enable the resolution architecture to optimize the organization of the topic ontology.

3.5 Efficiency

Object resolution systems need to be supported by efficient data delivery. The underlying network should be responsible for propagating “interests” (or queries) and “data” in a scalable and efficient manner which in turn renders the object resolution system efficient.

3.6 Cross Architecture Support

Object resolution systems should be able to operate in a cross platform environment. With multiple proposals for a future Internet architecture, the data objects a user wants may end up residing in multiple networks. Therefore, the systems should have a global view of these data objects in various networking environments to achieve universal data availability. It should be possible to resolve the object irrespective of the networks in which it exists and provide a unified list of candidates. The architecture should support the object resolution systems to provide the various services offered by the networks such as query/response and pub/sub in this unified view. The systems should then intelligently pick a candidate network for name resolution and routing. Such
3.6. CROSS ARCHITECTURE SUPPORT

A cross platform architecture can provide the benefit of access to content in various networking environments.

An efficient data delivery network is also important for a data finding solution. The network should provide efficient routing that disseminates interests and data in a scalable and efficient manner. The system should be able to address user needs at the application layer by providing a platform where users can express their ideas in the form of keywords/search strings and request for related content. At the same time, the system should be capable of transforming these inputs into names (CDs) that CCN can operate on. Such segregation allows for a clear separation of concerns in the application layer and underlying network layer functionality. There should be a Certifier that can function as a Certification Authority (CA) responsible for managing the names in the network. Certifier must approve the creation of any new CDs in the network since, creation of similar CDs or multiple CDs to the same content can turn out to be a problem. For instance, similar CDs such as "/sports/football" and "/football/sports" should not be allowed for creation as this may lead to CD explosion. On the other hand it is possible to have multiple CDs for some content. However, Certifier must ensure that such a situation does not occur by controlling the creation of multiple names for the same content. This can ensure a smooth operation of the network and will contribute towards standardizing the CDs in the network. There can be a component that operates as an intermediary between users and Certifier. Such a component should also be able to provide suggestions for user requested keyword/search strings. It should forward the request for creating new CDs to Certifier and request for approval on behalf of users and provide appropriate response to users. It is partly similar to the search engines in IP network w.r.t to the probable suggestions to CDs that it offers as response to users search. Additionally there should be a a Graphical User Interface (GUI) for supporting client side functions.
Chapter 4

Design Rationale

This chapter describes the design rationale of ORICE to meet the requirements mentioned earlier in chapter three. The application and network layer functions are separated to provide intelligence and efficiency along with service diversity. ORICE also separates the name space management from object resolution to achieve efficiency along with security. In this work ICN is chosen as the primary example underlying network in ORICE but it still provides the possibility for ORICE to operate with other networks.

4.1 Separate Application and Network Layer Functionality

There are two main trends in the object resolution architecture designs – as service networks and as service applications. The service network designs (e.g., ONYX [25]) process user requests on every node (broker) in the network and disseminate packets in a hop-by-hop manner (by calculating XML predicates). Such a design has the benefit of lowering network traffic since the intermediate nodes can decide which interface to send the packet to. However, they face the scalability issue as the state stored and the computation needed in the network can be overwhelming for current deployable network nodes.

On the other hand, in service application designs, the object resolution services (search engines) are deployed as separate functions at the application layer. Data consumers express the features of data of interest via explicit calls to these services while the underlying network handles the routing of requests and responses. Although this design consumes more network traffic, it keeps the design of network functions simple and scalable. Additionally, this approach is more deployable since any changes in the object resolution logic will not affect the underlying network. At a high level, this is the same approach used currently for information access in the Internet, albeit without a content-oriented network layer.
ORICE also chooses to use an architecture similar to the service application design. The object resolution functions are separated from the underlying network which is responsible for routing. Such a design choice allows ORICE to put the complex intelligence logic in the application layer while allowing the network functions to be efficient and scalable.

### 4.2 Separate Name Space Management & Object Resolution

Architectures that combine name space management with object resolution complicate the object resolution functions and lose the flexibility of providing diverse object resolution designs. The synchronization of the name space among various systems leads to unnecessary overhead in the network.

It is beneficial to believe that name space management should be separated from the object resolution function. Name space management should be concerned with the authorization, creation, modification, removal, etc., on the name space while the object resolution system should use this name space for its distinct functions. ORICE provides a logically centralized name certification service that controls authorization and modifications on the name space whereas, the object resolution service translates user inputs into data identities. The resolution services maintain a copy of the name space that is synchronized by the certification service through a management channel upon any update to the name space. Such a design enables efficient management of the name
4.3 Exploit Benefits Offered by ICN

An object resolution architecture can exploit the benefits provided by the underlying network to enhance its efficiency. Hence, ORICE chooses ICN as very suitable network to support efficient object resolution systems.

ICN provides the following benefits for ORICE:

**Diversity:** An object resolution system should provide quality results which includes a diverse set of candidate data objects. However, the current Internet-based solutions do not fulfill this requirement, as a particular piece of data (with potentially multiple copies) is usually represented by multiple URLs. This can lead to multiple URLs in the result list pointing to the same data object. Hence ORICE chooses ICN where a data object has only one name, irrespective of the number of copies stored in the network. Such a design choice allows object resolution services to provide more diversified results to the user.

**Service scalability:** Current object resolution systems utilize distribution techniques (e.g., MapReduce, NoSQL, etc.) to scale their services. Ideally, the underlying network should enable easy deployment of such distribution. With the help of ICN, ORICE can achieve scalability easily by distributing load based on the hierarchical name structure. Additionally in-network caching can reduce the load on servers by storing previous responses. Such a design enables object resolution systems to be timely and efficient.

**Retrieval/Dissemination efficiency:** The ultimate need of users is to retrieve data which needs to be supported by efficient routing in the underlying network. With name based routing and in-network caching, ICN can provide improved efficiency for ORICE compared to an IP-based network.

**Data integrity:** Verifying the integrity of data is a basic need. ICN requires the original content provider to sign the content. This allows consumers to verify the content. ORICE can use ICN to satisfy this need.
Chapter 5

Architecture

This chapter describes the details of the ORICE architecture. ICN is used as the exemplary underlying network for routing.

5.1 Components

As shown in Fig. 5.1, ORICE consists of five main components. The components communicate with each other to achieve the complete functionality of translating user inputs to names/CDs followed by retrieval of the requested content.

The functionality is described below:

Consumers/Subscribers: Consumers (those who generate queries) and subscribers are the clients that seek information from the network. Clients need suggestions for names that represent the data of their interest. Similarly, subscribers need suggestions for names (we sometimes refer to these as recommendations) to subscribe to the data of their interest.

Providers/Publishers: Data providers and publishers have the responsibility to provide the appropriate names/CDs when publishing their data in the network. They may depend on the object resolution service to provide these names.

Certification service: The certification service manages the name space in ORICE. It authorizes the creation, removal and modification of names and CDs and keeps the resolution service synchronized with any modification to the name space.

Object resolution service: All user requests for names are processed by the object resolution services. The object resolution service uses the name space to provide these names to users based on input features.
5.2 Architecture Description

In ORICE, object resolution services are provided in the application layer. The Fig. 5.1 and Fig. 5.2 will be used to describe the details of the architecture. Fig. 5.1 provides the architectural overview of ORICE whereas the figure Fig. 5.2 provides an application layer example of ORICE. Fig. 5.2 consists of 2 certification servers ($CS_1, CS_2$), 4 resolution servers ($RS_1$-$RS_4$) providing 3 different object resolution services marked in different colors, and 4 users of the services ($U_1$-$U_4$). The certification services and object resolution services register their prefixes to enable ICN to route the request for name/CD authorization and object resolution to them respectively. Note that since ICN is in the underlay, entities send requests/updates with names/CDs and it is the responsibility of the network to forward these packets to the proper destinations. A simple object resolution service with ORICE having only 1 Certification Server ($CS_2$) and 1 Resolution Server ($RS_3$) will be used to describe the architecture.
5.2. ARCHITECTURE DESCRIPTION

Figure 5.2: ORICE application-layer example.
5.2.1 Query/Response

As shown in Fig. 5.1, users send explicit requests to the resolution service to receive a list of recommended names. The clients provide their input in the form of features such as keywords to the object resolution service. In contrast to IP solutions, ORICE users do not need to get the location of the data. Instead, they provide the name of the data to ICN. The object resolution service first responds with a list of recommended names for the query. This list is forwarded back to client by ICN. Upon receiving the list, users can then select any of the recommended names in the list and send a request using the name. Subsequent retrieval of the data uses standard ICN procedures. For example, in Fig. 5.2 when $U_4$ wants to get recommended names for a topic of interest, she provides keywords that describes the topic to the client application. The client application then sends a request with name $/ServicePrefix/\text{Search/KEYWORDS}$ to the network. The network will forward the request towards $RS_3$ and $RS_3$ responds with a set of candidate names. $U_4$ selects one (or more) candidates and requests ICN for the data using those names.

5.2.2 Subscribe

Subscription begins with subscribers searching for CDs of their interest. As shown in Fig. 5.1 subscribers provide features of the interested data as input to the object resolution systems, similar to query/response scenario described above. ICN then forwards it to resolution service and forwards the list of recommended names back to subscribers.

For example, in Fig. 5.2 similar to query/response $U_4$ receives a set of candidate names/CDs and can choose to subscribe to any of the candidate CDs.

5.2.3 Publish

For publication a publisher needs a list of all names/CDs associated with the data from the name space hierarchy. In Fig. 5.1, publishers provide features of the data to the object resolution system similar to subscribers. For example, in Fig. 5.2 similar to subscribers, $U_4$ receives a set of candidate CDs and can choose to publish content associated with any of the candidate name/CDs. If $U_4$ wants to generate a new CD in the name space (when the result is not satisfying), she can send a request with name $/CertPrefix/\text{NEW_NAME}$ and content indicating the “add” action and the user identity if required. The network forwards the request towards $CS_2$. $CS_2$ can either approve or decline the request and send a response back. On approving the request, $CS_2$ needs to multicast an update with CD $/ManageChannel/\text{NAME}$ and the action as content. All the resolution servers that subscribe to this CD can get the update and modify their local name space accordingly.

In a hierarchical name space, it is common for a topic to have multiple parents. E.g. a topic “football” can have multiple CDs like $/sports/football$, $/ballgame/football$, etc. ORICE
5.3. SUPPORT FOR MULTIPLE LAYOUTS

puts the responsibility of carrying all the CDs related to a data object on the publisher. Therefore
the subscribers can subscribe to any of the many CDs associated with their topic of interest while
still assuring complete data delivery. Such a design can reduce the states stored in the network.

5.3 Support for Multiple Layouts

Other than the basic service layout described above, ORICE is designed to also support the
following scenarios:

5.3.1 Multiple Resolution Services

Users might want different object resolution services to support different preferences, while
operating on the same name space. In ORICE, every object resolution service provider can register
a different prefix (\texttt{/ServicePrefix}) and listen to the requests under that prefix. E.g., in Fig. 5.2
the provider of $RS_1$ registers \texttt{/BLUE} while the providers of $RS_2$ and $RS_3$ register \texttt{/RED}. All the
resolution servers subscribe to a management channel \texttt{/ManageChannel} so that the certification
service can keep them synchronized in the name space.

ORICE also facilitates smaller service providers that only focus on a minority of users. Consider
\texttt{/BLUE} ($RS_1$) as a service that serves resolutions related to ICN. It only needs to maintain the subset
of the name space it might operate on, by subscribing to \texttt{/ManageChannel/ICN}. The reduced
name space size can lower the storage and computation requirements of the resolution server for
that service provider. $RS_1$ can also refer to other resolution services (e.g., \texttt{/RED}) when receiving
some requests it cannot handle.

5.3.2 Private Resolution Service

An object resolution server can also reside in a private network or even on the same machine as the
client. But since the resolution server still subscribes to the updates from the certification server(s),
it can maintain an up-to-date name space. This service layout can provide an extended private and
more personalized resolution service.

Consider a film provider for e.g. ($U_2$) in Fig. 5.2 who wants to find the proper CDs for a movie to
be released soon. Even if there is a publicly available resolution service that can find related CDs
for video clips, the film provider still faces the risk of content leakage since he has to upload the
video to that service. In such situations, the film provider can deploy a resolution server in his
private network ($RS_4$) instead of using a public one. To accept requests, $RS_4$ propagates its own
prefix (\texttt{/GREEN}) just like public services but only within the private network. The communication
between $U_4$ and $RS_4$ remains the same as public services.
5.3.3 Distributed Servers

In ORICE, both the certification and the resolution service can be distributed based on the name structure. E.g., $CS_1$ and $CS_2$ can listen to /CertPrefix/Sports and /CertPrefix/ to handle different name space modification requests. A request of name modification in sports will go to $CS_1$ based on the longest prefix match. Similarly, the requests under other prefixes can go to $CS_2$. When $CS_1$ receives too many requests under /Sports/Football, a new certification server can listen to /CertPrefix/Sports/Foot-ball to reduce the load on $CS_1$. Similar mechanism can be adopted on resolution servers. The automatic load-balancing provided by NDN can also be leveraged even when two resolution servers are listening to a same prefix.
Chapter 6

Implementation

This work implemented a prototype of ORICE using ICN as the underlying network to demonstrate the feasibility of ORICE in fulfilling the necessity for object resolution services in ICN. This prototype demonstrates how the users can translate keywords to the names/CDs they might be interested in before retrieving the data from the underlying network.

The prototype is built on CCNx 0.8.0 and COPSS. The working of the prototype is explained with the help of the figures which includes two object resolution services (ORS₁ and ORS₂, having 1 server each), 1 certification server (CS) and 2 users (U₁ and U₂) attached to a network consisting of 6 routers (R₁–R₆). The prototype also includes the implementation of the broker design from COPSS (Broker in Fig. 1) to provide support for asynchronous data dissemination. The broker subscribe to responsible CDs and receive publications that can be requested by offline users. The broker, certification and resolution servers register their respective prefixes (i.e., /BrokerPrefix, /CertPrefix and /ServicePrefix) to handle the incoming interests so that CCNx can route each Interest to the respective destination based on the prefixes. The resolution servers subscribe to the management channel (/ManageChannel) on which they receive updates in the name space from the certification server.

The prototype is described with the following three uses cases.

6.1 Scenario [Query/Subscribe]

If a user is interested in a topic, in the network (s)he can either issue a query (for past data) or perform a subscription (for future data). To simplify the procedure, the two processes are unified in the implementation. The user can simply type the key words (s)he is interested in as shown in Fig. 6.2.

When the “Search” button is clicked, the application sends a request using the default search
CHAPTER 6. IMPLEMENTATION

Req/Resp: /ORS1/Search/sigcomm
Req/Resp: /acm/sigcomm/...
Req/Resp: /Broker/ACM/...

Figure 6.1: Scenario Query/Subscribe.

Figure 6.2: Query/Subscribe View.
6.1. SCENARIO [QUERY/SUBSCRIBE]

Figure 6.3: Scenario Post.

Figure 6.4: Post and Messages View.
engine prefix (/ORS1/ Search/sigcomm in the example, the green arrow in Fig. 6.1). ORS1 responds with a list of candidate names and CDs and the application lists the results in the middle column in Fig. 6.2. The Fig. 6.2 shows search result for “sigcomm” and the recent publications of CD /ACM/SIGCOMM/2015 (selected). If a user is interested in a piece of data, (s)he can click on the item and the application would send a request using the ContentName in the entry. In the example, application requests for the data /acm/.../callforpapers.pdf and the packet is forwarded in the network. This kind of requests can fully exploit the benefits provided by ICN. E.g., if a router has a cache or Broker already has the data, the request will be redirected before it goes all the way to the publisher.

When a CD entry (starting with “[CD]”) is clicked, the application would try to request the broker for the most recent messages. It would send a request with name /Broker/CD and put “latest” as the selector and the broker would reply with a list of data published recently as is shown in the GUI. These messages help the user to decide if the CD is the one that (s)he is interested in. When the user decides to subscribe to the CD, (s)he can click the “Subscribe” button in the bottom right corner and the application will subscribe to the CD.

When the clients come back online, the application can request the broker system using the name /Broker/CD and the selector to filter out all the received messages. Broker responds with messages accordingly.

### 6.2 Scenario [Post]

Data providers have the responsibility for associating appropriate names/CDs to the content they post in the network. ORICE allows them to seek names/CDs from the object resolution services. As shown in Fig. 6.3, when a user $U_1$ wants to publish a piece of data, he would type the title and content of the message in the graphical user interface (GUI) as shown in Fig. 6.4. Instead of typing the name and CDs himself, the user can get suggestions from an object resolution service. The prototype has 2 basic object resolution services and the user can set the preference by setting the parameter “default search engine prefix”. The application would send a request using the format /ORSx/Search/Message and the request will be routed to the corresponding object resolution service and receive the suggestions (the green arrow in Fig. 6.3 shows a request goes to ORS2).

$U_1$ can choose any name/CD for the message before publication. If he is not satisfied with the existing identities, he can request to add a new identity by sending an Interest with name=/CertPrefix/ID and content=“add”. This packet will be forwarded to the certification service $CS$ and $U_1$ will get the response indicating if the request is approved (flow not shown in Fig. 6.3). Upon approval, $CS$ will notify the changes to the name space via the management channel (with CD=/ManageChannel/CD) to the respective resolution servers (the red arrows in Fig. 6.3).
When $U_1$ clicks the “Publish” button, the application will start to serve the message and also multicast the message to the users who have subscribed to the related CDs. In the example, $U_2$ is subscribing to the message and her GUI will place a notification on receiving the multicast. In the prototype, a broker is used (described in [2]) that listens to all the CDs as a backup server for the application. The blue arrows in Fig. 6.3 shows the flow of the multicast and Fig. 6.4 shows that the received messages are grouped into different topics based on the CDs. The user has 3 new messages in topic “conferences” and 7 new messages in CD /CNN/news. She can browse through the topics, CDs and also read the details of the messages in the GUI. User can browse through the topics, CDs and also read the details of the messages in the GUI. Fig. 6.4 shows a publication in progress along with the recent messages under the selected CDs.

### 6.3 Scenario [Reconnect]

To avoid clients from missing messages when they are offline, the prototype implements brokers that receive all the publications. When a client comes back online, the application requests for any missed publications using name /BrokerPrefix/CD//NA-OF_LAST_MSG. The broker will respond with the list of messages published after the last message.
Chapter 7

Evaluation

In this chapter discusses the evaluation carried out with ORICE and the benefit of ORICE with a data set extracted from Wikipedia. Wikipedia has relatively well-organized categories similar to a topic ontology. Therefore, the category structure can be used as the topic-hierarchy in the name/CD space. Two sets of evaluations at different scales are carried out to show the need of object resolution services especially in a name-based network like ICN.

7.1 Data Set

This work took the April 2014 dump from English Wikipedia as the basic data set. To have a meaningful category logic, all of the 989,468 descendants from “Category:Main topic classifications” as the topic ontology were used. Data consumers in the evaluation will subscribe to categories and they should receive all the data that are related to these categories and their descendants. E.g., a consumer subscribing (everything related to) “Category:The Big Bang Theory” should also get publications from “Category:The Big Bang Theory characters” (a sub-category of the previous one). To get the topic-based hierarchical names of each category, the traversal process covered all of the ancestors of that category till the root is reached. Each path is seen as a name, e.g., /Architecture/ Landscape_Architecture/Fountains. Since each category can belong to multiple parent-categories, a category can have multiple names. Around 6.5 billion names from these categories were retrieved in the end. The number of names per category is shown in Fig. 7.1. The analysis also incorporated the extraction of around 5 billion articles that are related to the 1 million categories. Each article is seen as a data object during the evaluation. The page-category relationship is shown in Fig. 7.2.
CHAPTER 7. EVALUATION

Figure 7.1: Wikipedia data set statistics: CDs per category.

Figure 7.2: Wikipedia data set statistics: Pages per category.
7.1. DATA SET

(a) # of Ancestors/Category in Wikipedia full Data Set

(b) # of Ancestors/Category in five MHRW Samples

(c) # of Ancestors/Category in five Random Samples

Figure 7.3: # of Ancestors/Category.
CHAPTER 7. EVALUATION

(a) # of Descendants/Category in Wikipedia full Data Set

(b) # of Descendants/Category in five MHRW Samples

(c) # of Descendants/Category in five Random Samples

Figure 7.4: # of Descendants/Category.
7.1. DATA SET

(a) # of Request/Category in Wikipedia full Data Set

(b) # of Request/Category in five MHRW Samples

(c) # of Request/Category in five Random Samples

Figure 7.5: # of Request/Category.
7.2 Small Scale Evaluation

The Wikipedia data set consists of relatively a large number of categories. So a study on sampling techniques was carried out with the intention of obtaining a subset of the data set that can be a replica of the whole data set. In this regard the graph sampling techniques from [39] where the authors explore the performance of various algorithms such as Breadth First Search (BFS), Metropolis-Hashing Random Walk (MHRW) and Frontier Sampling (FS) was observed.

Two further verify the relevance of the sampling techniques for sampling the Wikipedia data set two kinds of sampling were performed on the Wikipedia data set. The first sampling technique was performed by randomly choosing a thousand categories and the other sampling technique was performed using MHRW technique. The evaluation was carried out with five different samples from each of the techniques and the results were compared with the Wikipedia full data set.

Fig. 7.3 is a plot of the number of ancestors that every category has in the Wikipedia data set. A BFS was carried out to identify the number of ancestors for each of the categories in the data set. The Fig. 7.3a shows a plot of the number of ancestors that every category is associated with in the full data set whereas the Fig. 7.3b and Fig. 7.3c is a plot of the number of ancestors for each of the categories in the five different samples for the MHRW and Random sampling techniques.

Fig. 7.4 is a plot of the number of descendants for every category in the Wikipedia data set. Similar to the above case of ancestors BFS was performed to identify the number of ancestors for each of the categories in the data set. The Fig. 7.4a shows a plot of the number of descendants for every category in the full data set whereas the Fig. 7.4b and Fig. 7.4c is a plot of the number of descendants for each of the categories in the five different samples for the MHRW and Random sampling techniques.

Additionally, a workload was generated by firstly assigning subscriptions for each categories using the Zipf distribution. This was followed by a process where the every category was assigned the number of Interests equal to its assigned number of subscriptions and additionally all of the descendants of this category were also assigned with this number of Interests along with the number of Interests equal to the number of subscriptions for the descendants. The Fig. 7.5 shows a plot of the number of Interests generated for each of the categories in the Wikipedia data set. The Fig. 7.5a shows a plot of the number of requests generated for every category in the full data set whereas the Fig. 7.5b and Fig. 7.5c is a plot of the number of requests generated for each of the categories in the five different samples for the MHRW and Random sampling techniques.

It was observed from the Fig. 7.3, Fig. 7.4, Fig. 7.5 that both the the sampling techniques were almost similar to the original data set. Hence we chose a random sample for performing a small scale evaluation. This was performed with a subset of the data for a category with 1,772 descendants to study the benefit of ORICE in a smaller scale. The evaluation obtained 4,495 names from the category relationships and identified 13,913 articles that are related to these categories.
7.2. SMALL SCALE EVALUATION

In the first evaluation, it was assumed that the publishers do not know all the names of a category due to the lack of an object resolution service. To receive all the related data, a consumer has to subscribe to all the names that belong to the categories he/she is interested in. Although there is no way a consumer can acquire the complete list of names without an object resolution service, this assumption was made to calculate the amount of states needed to ensure complete data delivery.

According to [40], the popularity on the articles follow a Zipf distribution and the parameter $\beta$ is between 0.5 and 0.8. Due to the lack of category popularity from the Wiki dump, it was assumed that category popularity should follow Zipf distribution similar to article popularity. Therefore as part of the evaluation $1.7 \times 10^6$ category interests were generated accordingly with $-1 \leq \beta \leq 1$. Two types of category rankings have been considered – names/category ranking (a category with more names will have more consumers interested in it when $\beta > 0$), and a random category ranking.

Fig. 7.6 shows the number of states needed to ensure complete data delivery with different $\beta$. For the names/category ranking, the number of states ranges from 2.6 million to 21 million. It consumes 15-119 times more states than the situation with ORICE, where each “interest” on a category will only need 1 entry in the network. To get a statistically significant result on random-ranking Zipf distribution, the evaluation included a run of each distribution 100 times. The average amount of states in the network and the 95% confidential interval is shown in the plot. It is seen that the states consumed by random ranking is similar to that of an even distribution ($\beta=0$). Around 4.5 million states are required in the network (~5.3 times as the ORICE case).

Then an evaluation of the message loss rate was carried with the condition that neither publishers nor subscribers get a view of the whole name space. Here, it was assumed, the interests in the categories follow a Zipf distribution on names/category rank with $\beta=0.5$. The evaluation was
Figure 7.7: Message loss rate on different % of known CDs.
7.3 Large Scale Analysis

To emphasize the need for ORICE in ICN a comparison was performed with the benefits offered with ORICE for ICN. The evaluation is extended to the whole data set to see the states needed for a complete data delivery in a realistic environment. In current ICN solutions, with a plain CD layout as described above if a user is interested in receiving all the information related to a topic say “Comics” then the user has to subscribe to all the CDs associated with this topic i.e., “/Comics”, “/Comics/Futurama”, etc. Additionally this list could be large (approximately 3e51 CDs) and the user may not have the information regarding all the CDs associated with this topic. Similarly, when a publisher has to publish content related to this topic, they have to publish to all of the CDs associated with this topic; again one has to remember that there is no system in place from where the publisher can receive the list of all 3e51 CDs associated with this topic. Due to the absence of object resolution system in this environment, the chances of mismatch are high leading to increased subscription and loss of information as it has been already seen in the above section with the small scale evaluation. But the effects are multiplied with the whole data set.

Figure 7.8: # of states needed for complete data delivery with $1 \times 10^9$ interests.
With ORICE on the other hand the above mentioned mismatch can be eliminated and it can contribute towards exploiting the benefits offered by ICN. Since ORICE maintains hierarchical information about the categories, a user interested in receiving information regarding for example a topic “Comics” can just subscribe to the highest level of the category which in this example is “/Comics”. When a publisher has some content to publish, they will query the object resolution system to get the list of CDs associated with this topic and will publish the content to all the CDs. This effectively reduces the number of subscriptions the user has to make and eliminates the possibility of information loss.

As part of the large scale evaluation $10^9$ interests were generated to the $10^6$ categories using different $\beta$. The result is shown in Fig. 7.8. Due to the complicated structure in the full data set, the number of names per category becomes much larger compared to the small-scale data set. As a result, the states needed in the network becomes much larger. It requires at least $\sim3.8 \times 10^{11}$ states and can reach up to $3.4 \times 10^{15}$ states (3.4 million times as large as the interest count). Note that the amount of states grows linearly with the amount of interests in a same distribution. With the help of ORICE, the network only needs to keep $10^9$ states disregarding the interest distribution. This can allow the network to be much more scalable and efficient.
Chapter 8

Conclusion and Future Work

This work studied the importance of object resolution services in a network like ICN. It analyzed
the requirements of an ideal object resolution system for ICN and then proposed the design of
ORICE, an architecture that meets the requirements for a resolution service and provides the
foundation for exploiting ICNs. We also discussed the rationale behind the design choices that
have been made in this work for ORICE mainly the separation of logic between the application
and network layer to render network simple and separation of name space management from
object resolution to support service diversity. This work also implemented a prototype of ORICE
with ICN as the underlying network to demonstrate the need and the feasibility of the design.
Through an evaluation on the Wikipedia data set, this work showed that ORICE can reduce the
amount of states needed in the network and the message loss rate during data delivery.

8.1 Future Work

As part of the future work, this work will continue with the study of the two different naming
schemes namely Hierarchically structured names and Flat names in ICN. The study is intend
to identify the benefits and limitation with the two types of names by observing their impact in
ICN through evaluation. The work also would consider more detailed observation of sampling
techniques and identify a technique that can be applied to the large data set for carrying out the
evaluation. The prototype is currently built on ICN, it is desired to extend this by building it on
multiple different ICN solutions and allow the object resolution systems to communicate within
this cross platform architecture to provide better suggestions to users.
Bibliography


