2016 IEEE 22nd International Conference on Parallel and Distributed Systems

A Peer-to-Peer File Sharing System over Named Data Networking

Jian Shi, Xuewei Piao, Lihua Li and Yunbo Xun
School of Electronics and Computer Engineering (SECE)
Peking University
Email: 1501213960@sz.pku.edu.cn

Kai Lei*
School of Electronics and Computer Engineering (SECE)
Peking University
Corresponding Author: leik@pkusz.edu.cn

***Abstract*—Named Data Networking (NDN), a promising Future Internet Architecture design, requires new experimental applications to demonstrate its performance and feasibility. Through designing, implementing, and evaluating NDNMaze, i.e., an NDN version of a widely deployed peer-to-peer file-sharing application called IPMaze, we find that NDNMaze has a simpler system architecture with improved performance and flexibility. The innovative messaging mechanism and distributed hash tables (DHT) in NDNMaze simplified the implementations of key system components such as user management, nearest neighbor discovery, file discovery and distributions. To systematically evaluate the performance, we simulate both versions with NS-3 simulator, and collect a broad range of performance metrics including hop count, data request latency, data request efficiency, and network transmission efficiency. Our experimental results show that NDNMaze achieves better performance than IPMaze due to the NDN's advantages in content-centric data distribution and sharing. Our work sheds light for distributed application design in NDN.**

***Keywords*-Named Data Networking; peer-to-peer; IPMaze; NDNMaze; distributed application design.**

I. Introduction

Named Data Networking (NDN) uses names of content, rather than addresses of end hosts, to route Data packets [1]. As one of the future Internet architecture proposals [2]–[5], focus of the NDN project has recently moved from architecture design and protocol development to the testing and evaluation of prototype systems in real network environments. The promise of NDN as the architecture of the future Internet essentially depends on whether this new architecture can effectively and efficiently support different needs from a broad range of applications such as video streaming, file sharing, and massive content delivering (CDN).

Peer-to-peer (P2P) file sharing is one of the most popular TCP/IP applications which generates tremendous data volume traversing the Internet [6], but there are still some challenges, such as network address translation (NAT) issues, bandwidth redundancy, and unnecessary data retransmission, etc. We managed to use quite a few technologies, methods and tools, including NAT traverse, mirror servers and smart downloading strategies, to resolve these challenges, however, making the architecture even more sophisticated. All these difficulties are fundamentally caused by the original design of TCP/IP, which was not targeted for internet content sharing. To our great joy, since NDN has some inherent properties such as content oriented characteristics, in-network caching and messaging mechanisms, the system architecture of NDNMaze becomes less complicated and more straight forward.

This paper designs and implements an NDN-based P2P file sharing system, named NDNMaze. While designing and implementing NDNMaze, we leverage the inherent properties of NDN and compare the scalability and performance of NDN-based system, i.e., NDNMaze, and TCP/IP-based system, i.e., IPMaze (see section II-B), over a range of experimental scenarios. Comparing with six different types of servers in IPMaze, the NDNMaze system only requires three types of light weighted servers including control servers, user servers, and index servers for user management, file indexing and searching. In addition, considering the vast amount of files in peer-to-peer systems, we introduce distributed hash tables (DHT) for effectively managing and searching the indexing of files and NDNMaze users. The ultimate goal of this study is to make a concrete step towards understanding the feasibility and scalability of NDN architecture and provide a good reference for other application developers.

The main contributions of this paper are as follows:

- We design and implement a peer-to-peer file sharing system in NDN architecture, and shed light on the challenges and experiences of migrating existing TCP/IP applications to NDN.
- Our experience of developing NDNMaze shows that NDN significantly simplifies the overall system architecture of NDNMaze which becomes more serverless and has a less complicated message mechanism.
- We provide distributed hash tables and extensible super node methods to solve scalability issue and make the functions work more efficiently.
- We implement NDNMaze in a real testbed and ndnSIM, and compare the efficency of ndnSIM based NDNMaze and NS-3 simulator based IPMaze.

The remainder of this paper is organized as follows. Section II provides an overview of NDN architecture and IPMaze, and Section III introduces the system design of NDNMaze. Section IV describes the user management and

1521-9097/16 $31.00 © 2016 IEEE
DOI 10.1109/ICPADS.2016.28

neighbor discovery in NDNMaze, while Section V is devoted to NDNMaze file indexing and searching. Section VI evaluates the performance of NDNMaze and IPMaze with a variety of metrics. Section VII discusses related work, while Section VIII concludes this paper and outlines our future work.

II. Overview of NDN and IPMaze

In this section we first briefly describe NDN networks and point out the key difference between NDN and today's TCP/IP networks. Subsequently we present an overview of IPMaze, a widely deployed peer-to-peer file sharing system with an average of 30 thousand active nodes at any given moment. The primary goal of developing NDNMaze is to support IPMaze over NDN networks and to explore the benefits of NDN in-network caching for effective file sharing.

A. Named Data Networking (NDN)

In NDN, data communications are driven by the consumers who express interest of obtaining the named Data [1]. The communication mode leads to two fundamental packet types in NDN: `Interest` and `Data`. Figure 1 presents a schematic diagram of the core NDN packet forwarding engine, which consists of three key data structures: Forwarding Information Base (FIB), Content Store (CS), and Pending Interest Table (PIT).

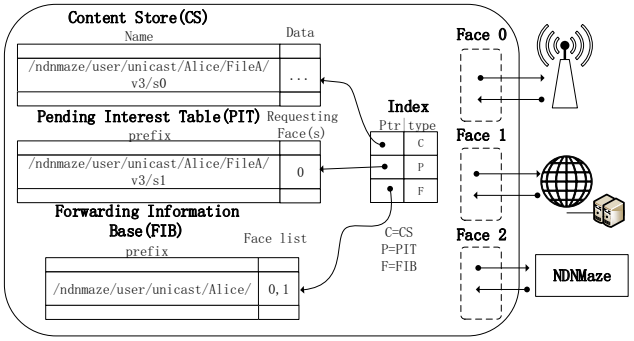

Figure 1. The core NDN packet forwarding engine.

To request the data of interest, a consumer first sends out an Interest packet, which carries a name that uniquely identifies the desired data, e.g., `/ndnmaze/user/unicast/Alice /FileA`. Upon receiving the Interest, a NDN router remembers the incoming interface from which the request comes in, and forwards the Interest packet by looking up the corresponding name in its FIB table which has been populated by a name-based routing protocol. Once the Interest packet reaches a node with the requested data, a Data packet is sent back carrying both the name and the data content. The returning path traversed by the Data packet is exactly the same as the path travelled by the Interest, since NDN routers forwarding the Data packet back to the same interfaces from which the Interest comes in.

NDN routers cache both Interest and Data with an expiration time period. When multiple Interests for the same named data are received from downstream routers or consumers, only the first Interest is forwarded to upstream routers or the producers, also referred to as data source. NDN routers store the Interests in their local PIT tables, where each entry contains the name of the Interest and a set of interfaces from which the Interest has been received. When Data packets arrive, a NDN router first finds the PIT entry via longest prefix matching and forwards the data to all the interfaces listed in the PIT entry. Then the router deletes the corresponding PIT entry after data forwarding and caches the Data in the `Content Store` for serving future Interest requests of the same named data. The Content Store is essentially the buffer memory of the router, and runs a replacement policy for effective in-network caching.

It is important to note that during the entire process, Interest packets are routed towards data producers purely based on the names embedded inside the Interest packets, while Data packets are returned based on the state information of the corresponding Interest maintained by NDN routers between the consumer and producer. In other words, the packet forwarding is based entirely on the *name*, which is fundamentally different from the location-based forwarding on today's TCP/IP networks, i.e., IP addresses of the destinations in TCP/IP packets. More importantly, the in-network caching mechanism via `Content Store` reuses cached Data packets, thus reducing unnecessary transmission of duplicate Data packets.

B. IPMaze

IPMaze [7] is a popular peer to peer file sharing system in China Education and Research Network(CERNET), which is similar to Internet2 network of United States.IPMaze follows a hybrid structure in its design and has evolved into a sophisticated and robust system after years of improvement. At the user side, the system includes a browser/server (B/S) user client with a GUI and a client daemon on each peer node. At the server side, the system includes a variety of central servers, such as *user servers*, *heartbeat servers*, *status servers*, *directory servers*, *indexing servers*, and *searching servers*, for user management, file indexing and file search. Figure 2 illustrates the system architecture of IPMaze and its key components.

The functions of user servers are user registration, authentication and management, while heartbeat servers maintain the online information, such as IP address and ports of all peer nodes, which periodically report their current statuses to heartbeat servers. The status servers collect the statuses of all the peer nodes from heartbeat servers, build optimized

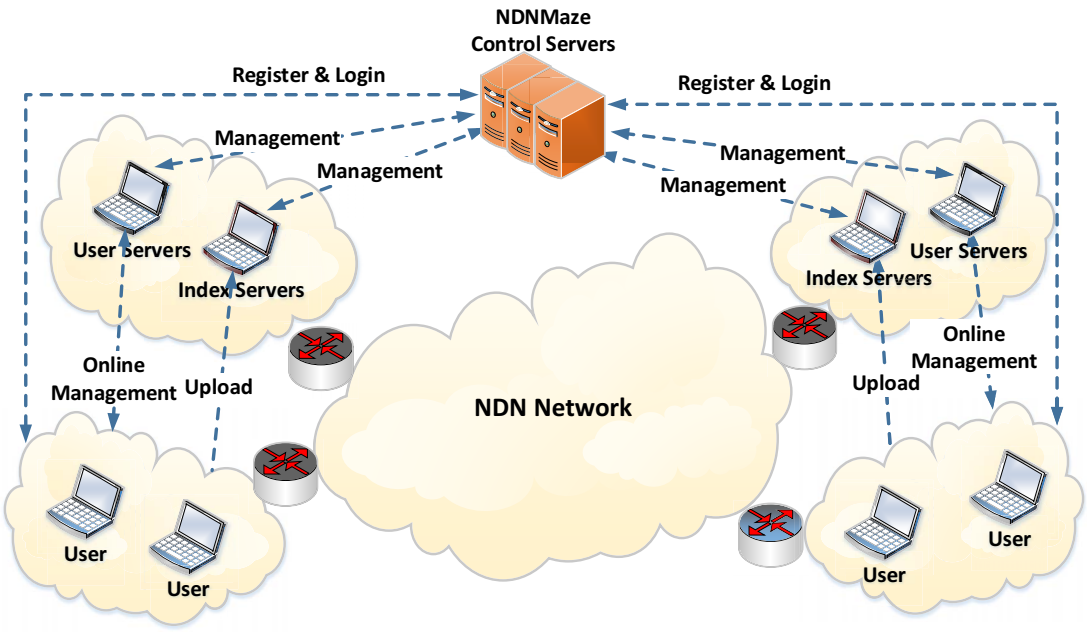

Figure 3. The system architecture of NDNMaze

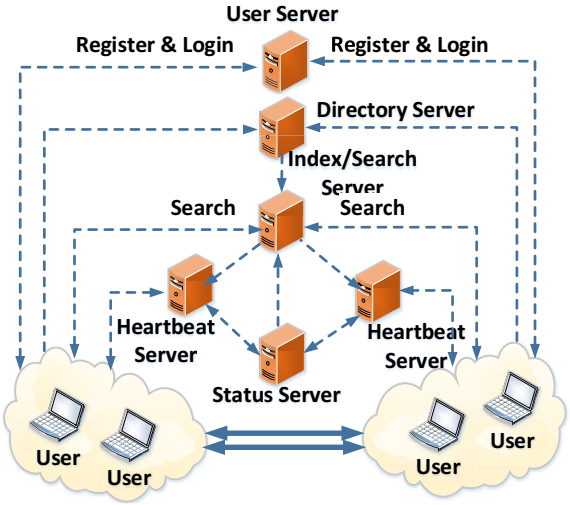

Figure 2. The system architecture of IPMaze.

connections between peers, forward messages, and assist in searching and downloading among peers. The registered users are able to share their files via uploading meta data of the shared directories to directory servers. At the same time, indexing servers build real-time file indexes using hash maps in memory. As the files and data blocks are identified by their MD5 values, both keyword search and MD5 search are supported by search servers. Finally each peer node could search or browse other peers' shared directories for locating and downloading the interested files.

III. NDNMaze System Design

In this section, we first present an overview of NDN system architecture and describe its core components. Subsequently, we explain how we design the naming convention for effectively forwarding and routing `Interest` packets and `Data` packets of NDNMaze.

A. System Architecture

Figure 3 illustrates the system architecture of NDNMaze which consists of NDNMaze users, *user servers*, *index servers* and *control servers*. The only servers centrally managed by our proposed NDNMaze system are control servers, while the remaining two kinds of servers are dynamically selected peer nodes in NDNMaze. Due to the inherent property of NDN in-network caching, not only the number of different types of servers is reduced, but also the functionality of corresponding servers is simplified, thus creating a concise architecture of NDNMaze and facilitating the management of servers and users. The key challenges of design NDNMaze lie in the naming convention for Interests and Data packets for facilitating the complicated functions and interactions of NDNMaze, and the scalability of servers for supporting millions of NDNMaze.

To provide a peer-to-peer based file sharing platform in NDN, we develop NDNMaze with two major functions: user management and file management, and build user servers and index servers to deliver these two key functions, respectively. Similar to the scalable NDN-based conferencing (SNC) architecture [8], NDNMaze is designed with a

distributed architecture with a number of user servers and index servers via selecting super peer nodes with sufficient computing, bandwidth, networking and storage resources. The user servers are responsible for managing login and logout processes of the users and their online statuses, while the index servers allow all users to search the files of interests and to identify the peers for downloading.

To join NDNMaze system, a new user, say `Alice`, will first register with one of control servers. Upon successful registration, `Alice` will search and register a user server, which will immediately start to manage `Alice's` online status. Alice uploads the names of the files she is willing to share to the index servers, which provides the indexing and searching services for these files. If another NDNMaze user searches one of the files Alice shares, the user will send an Interest request to get the information on the user who owns the file. To prevent the overloading of the servers, the control servers will continuously monitor the system usage of each user and index server, and dynamically choose additional user nodes as new servers.

B. Naming Convention

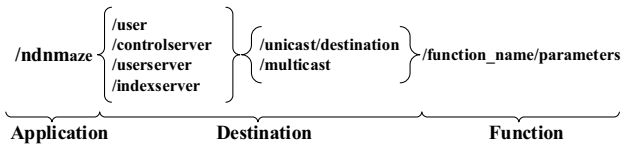

Figure 4. The naming convention.

Naming is one of the key aspects in NDN architecture and a challenging task for designing all NDN-based applications. As described earlier, the communications between NDNMaze users and servers and among NDNMaze servers form the majority of data flows and control flows in NDNMaze. In addition, some NDNMaze Interests are multicasted to multiple destinations, while other Interests are forwarded to a specific destination via the unicast mode.

In NDNMaze we design the naming convention based on three key components: application, destination, function, as illustrated in Figure 4. The first component carries the name of the application for specifying the application that `Interest` and `Data` packets belong to, while the second component tells the role of the destination, e.g., NDNMaze user, control server, user server or index server and combines with /unicast/, or /multicast/ to specify the mode of transmission. The last component describes the function and its parameters for specifying which functions `Interests` and `Data` packets are used for in NDNMaze system. In the next two sections we will describe the detailed use of naming convention in NDNMaze user management and NDNMaze file indexing and searching.

IV. NDNMaze User Management

In this section we describe the user management of NDNMaze, in particularly explain user login procedure, online and offline statuses management, and neighbor discovery for efficient file transfers.

A. User Login Procedure

The user login procedure in NDNMaze consists of two steps (Figure 5). First, a user, say Alice, sends a named Interest, i.e., `/ndnmaze/controlserver/unicast/login /Alice/entryptedpassword`, to control server with her user name and encrypted password which was created during the initial user registration process. Upon successfully verifying the user name and the password carried in the Interest, the control server returns a Data packet that includes a login success message and more importantly a unique login identification number (uniqueID) signed and encrypted by the control server.

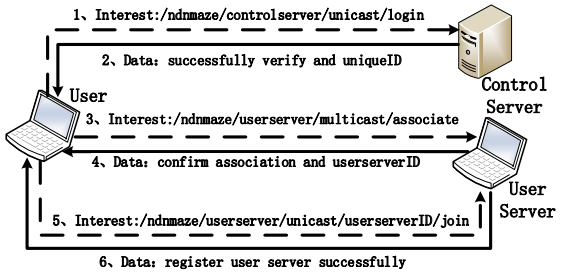

Figure 5. User login and online association.

After Alice receives the login success Data packet, she first extracts uniqueID and then multicasts a named Interest, i.e., `/ndnmaze/userserver/multicast/associate /Alice/uniqueID`, to associate with one of the user servers. The user server, which first receives this multicast Interest due to a small end-to-end delay, replies a Data packet to Alice for confirming with the association between Alice and itself. After receiving the Data packet from this user server, Alice send a Interest, i.e., `/ndnmaze/userserver/unicast/userserverID /join/Alice/uniqueID` with her username and basic account information. Finally the user server extracts these information, adds Alice into its online user list, update Alice's status as online, records the starting timestamp of Alice's online status and return Data packet for confirming with Alice register user server successfully.

B. User Status Management

To let the user server continuously keep track of the online status of a user in NDNMaze, the user,

say Alice, periodically issues a heartbeat Interest, i.e., `/ndnmaze/userserver/unicast/userserverID/heartbeat/Alice/uniqueID/timestamp`, to the user server managing her. The user server, after receiving the heartbeat signal, sends back a Data packet meanwhile representing that the user server is active and updates the active duration of Alice based on the timestamp carried in the Interest.

In NDNMaze, users normally exit gracefully with the logout step, but could also become offline due to external reasons such as network or system failures. In the first scenario, the user sends a logout Interest, i.e., `/ndnmaze/userserver/unicast/userserverID/logout/Alice/uniqueID/timestamp`, to the user server for informing the logout action. As a result, the user server changes the user status as offline and updates the online user list. For the second scenario, the user server will automatically discover the offline status, after it fails to receive any regular heartbeat Interests of the user.

NDNMaze typically prefers the most resource-rich user node, often referred as super node in peer-to-peer systems, for serving user servers. However, it is not surprising to observe the failures of user servers during long term system operations. To handle the failures of user server, NDNMaze requests each user node to actively confirm the failure via issuing an *echo* Interest if the node has not received the Data packet corresponding to the heartbeat Interest from user server after the scheduled time window. The user node, after confirming the failure of the user server, re-multicasts the same named Interest, i.e., `/ndnmaze/userserver/multicast/associate /user/uniqueID`, to associate with a different user server. (If there is no user server available for the user, control servers will choose a super node to serve the user as a user server.)

C. Neighbor Discovery

File sharing or content distribution applications often prefer data requesters and data sources in local network, e.g., campus networks, due to shorter network path, smaller end-to-end delays, and less abundant bandwidth. Since the overhead of user list maintainence is high, a user should only maintain the closest neighbors' information. Thus we build NDNMaze with a novel neighbor discovery feature for each NDNMaze user to locate the neighboring nodes with similar interests, which likely host files which the user would like to download.

The first step of neighbor discovery is to broadcast an Interest `/ndnmaze/user/broadcast/neighbor discovery/Alice` with an expiration timestamp, similar to time-to-live (TTL) field in IP datagrams, for preventing the Interest from traveling too far away. The NDNMaze nodes, receiving such neighboring discovery Interests, immediately return an Interest `/ndnmaze/user/unicast/Alice /nerghborconfirm` confirming their existence. To remove nodes with large end-to-end delays, the user will use a stopwatch, e.g., 1 second, to indicate when to stop reading the Interests. When Alice receives those return Interests from neighbor user, say Bob, Alice send an Interest `/ndnmaze/user/unicast/Bob/filelist` to Bob, and Bob returns a Data packet to Alice. In addition, the data packet also carries the file list of each neighbor that essentially represents its interest on certain files.

Assuming that the order of Data packets returned from different neighbors captures the end-to-end delay between the node and the neighbor. The second step of neighbor discovery is to find the top 10 neighbors which share the most similarity with the user, from all the neighboring nodes and to recommend them to the node. The similarity measure between the user u and a given neighbor v is based on the similarity of the files they own. Specifically, let f_{ui} denote the value of the file i of the user u. f_{ui} equals to 1 if the user u has the file i. f_{ui} equals to 0 if the user u does not have the file i.

$$f_{ui} = \begin{cases} 1, & \text{the user } u \text{ has file i,} \\ 0, & \text{the user } u \text{ does not have file i} \end{cases} \tag{1}$$

And the same definition to f_{vj}. Then the similarity $s_{u,v}$ between u and v becomes:

$$s_{u,v} = \sum_i (f_{ui} \times f_{vi}) \tag{2}$$

After calculating the similarity scores with all the neighbors, the user u will select the top 10 nodes with the highest similarity scores as *effective neighbors*, which not only have short end-to-end delays but also share strong file interests with the user u. Thus, the user will always prefer to scan the file list of these effective neighbors for downloading files if the neighbors happen to host the files the user is interested in.

V. NDNMaze File Indexing and Searching

In this section we first describe how we develop a DHT-based file indexing and searching technique for NDNMaze. Subsequently we describe the detailed processes of file indexing and file searching in NDNMaze.

A. Distributed Hash Tables (DHT) for NDNMaze

Considering the sheer number of files in NDNMaze, we adopt distributed hash tables (DHT), a widely used decentralized distributed system for managing massive *key-value* pairs, to manage the file indexing. Specifically, we use a tree-structure DHT in which every node in the tree names itself a unique bit-sequence based on its position in the binary tree. Any node in the binary tree could represent an index server that provides the indexing and searching functions in NDNMaze. Thus for a given file, we use a hash function,

e.g., MD5, to produce a unique bit-sequence essentially mapping to a specific leaf node which will be responsible for maintaining the metadata and the data sources of this particular file.

Figure 6 illustrates an example of DHT mapping between files and index servers in NDNMaze. An index server, a node in the binary DHT tree, manages all leaf nodes of the subtree with the index server node as the root, thus provides indexing services for files that are mapped to these leaf nodes via the hash function. Thus, the name of the index server is the shared common prefix of the names of all the leaf nodes under the subtree.

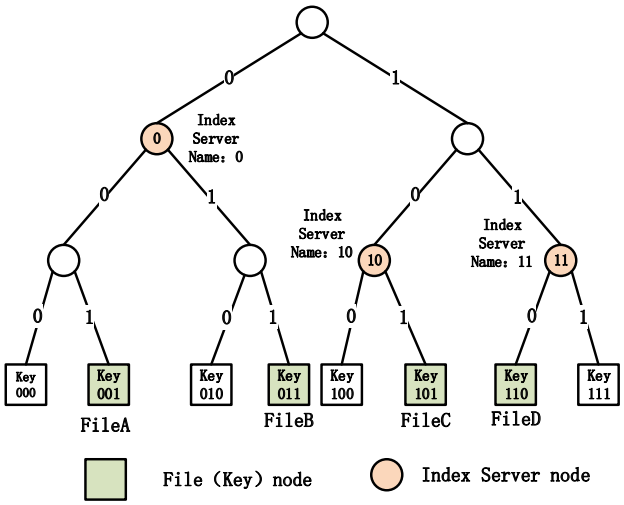

Figure 6. A simple example of NDN-DHT

A major benefit of DHT lies in its flexibility and resilience to handle an extremely large number of *key-value* pairs and to allow continuous node participation, departures and failures. In NDNMaze, the responsibility of managing index servers falls on control servers which actively monitor the healthy statues and indexing and searching load of all index servers.

When index servers are overloaded due to node failures or dramatic load increase, control servers will automatically locate alternative super nodes to become new index servers. Correspondingly, the tree-based DHT will be re-balanced to reflect the changes of index servers to ensure a balanced mapping between files and index servers.

B. File Indexing

During the initial login process, each NDNMaze user will synchronize with control servers for requesting an updated list of index servers. In addition, when the user has updated the shared files, e.g., adding a new file, removing an old file, or modifying an existing file, and user also resynchronizes with control servers for updated index servers. For each shared file, the user will use the same hash function as used in DHT to generate a bit-sequence, and finds the leaf node and the corresponding index server node in the binary DHT tree. Through repeating the same process for all the files, the user could classify the files based on the mapped index servers, and create a file cluster for each index server.

Next for each file cluster, as shown in Figure 7, the user issues an Interest request, i.e., `/ndnmaze/indexserver/unicast/indexserverID/upload/user`, to the corresponding index server. The server, upon receiving the request, immediately sends a new Interest request to the user for requesting the metadata for each file in the cluster. Finally, the user returns a Data packet that carries these meta data for the index server to build the indexes for these files.

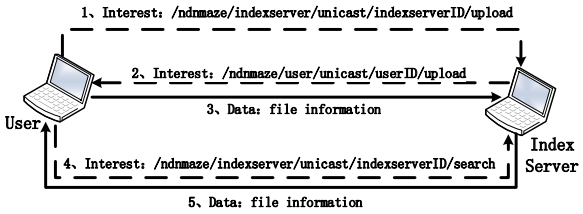

Figure 7. File indexing and searching.

Unlike always-on dedicated servers, NDNMaze users could change the online and offline statuses in a very random fashion. Thus in addition to maintain the indexing of the files, each index server also queries user servers on the online statuses of the owners of these files. If the status of an owner becomes offline, the index server will mark the corresponding files as *unavailable* and *unsearchable*.

C. File Searching and Delivery

NDNMaze adopts a DHT-based structure to map files and index servers, thus significantly reducing the search load in NDNMaze. If a user is interested in a given file, the user first uses the hash functions based on the file name to locate the index server. Then the user sends an Interest request to the index server (Figure 7), which looks up the file from the local file indexing and return a Data packet carrying the meta data of the file as well as the information of all the online users who own the particular file.

After the user receives the information of the server and data sources from the index server, she first searches whether one or more data sources are also in the list of all neighbors. If the search is successful, the user will select one neighbor and issue an Interest request to the neighbor for downloading the file of interest. Otherwise, the user randomly selects one data source and issues an Interest request. When the Interest arrives at the data source or one of intermediate routers which happens to have the same file in its local `Content`

`Store`, the Data packet with this file will be immediately sent back to the user.

VI. System Evaluations

In this section, we first describe the experiment setup and environment, and subsequently we present the evaluation results of our proposed peer-to-peer file sharing system over NDN network and compare a variety of performance metrics with IPMaze, i.e., the counterpart system in TCP/IP networks via extensive experiments.

A. Experiment Setup and Environment

Our performance evaluations on NDNMaze and IPMaze are based on simulations on the widely used NS-3 simulator that supports both NDN and TCP/IP simulations, thus allowing us to compare the performance metrics, such as hop count, data request latency, data request efficiency, and network transmission efficiency of these systems under different network architecture. To demonstrate the operational feasibility of our proposed NDNMaze system, we have implemented the entire NDNMaze system with NDNx package as well as supporting a user-friendly GUI interface. The system supports all the functions described in the previous section, and runs on either Linux or Windows platforms [1]. It had been tested on a real testbed, which consists of many NDN nodes. Since we could only build a small network of NDN, we could only test the feasibility of the system, but not evaluate the efficiency of the system on the real testbed.

Based on Waxman model [10], we use the topology generator BRITE [11] to generate a random network with 100 nodes as routers and 200 communication links among these routers. This random network has the minimum, average and maximum degrees of 2, 4, and 11, respectively. In addition, we use a uniform distribution to set the link bandwidth in the range of [10kbps, 2Mbps] and the link delay in the range of [10ms, 200ms]. In each simulation scenario, we randomly select 10 nodes as download sources which has the same 10MB file for sharing. To study the impact of the system performance, we run the experiments with a wide range of users varying from 5 to 50.

B. Performance Evaluation

1) Hop Count: In peer-to-peer system, *hop count* is a simple yet effective metric to approximate end-to-end performance between data request node and data source node. In both NDNMaze and IPMaze, data request node is a regular peer node, while data source node is a regular peer node in IPMaze, but could be either a peer node or an intermediate router in NDN due to the in-network caching property of NDN architecture.

Figure 8 illustrates the average numbers of hop counts, which is defined as the number of hops between data request node and data source node, for NDNMaze and IPMaze with varying number of users. As shown in Figure 8, the average hop count in NDNMaze is significant lower than the average hop count in IPMaze, especially as the number of user increases.

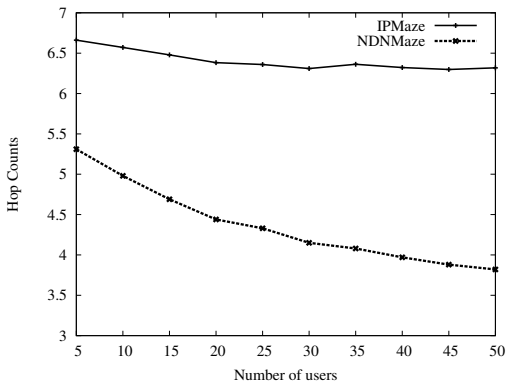

Figure 8. The average numbers of hop counts for NDNMaze and IPMaze

The difference in hop count between NDNMaze and IPMaze can be explained by the following three reasons. First, IPMaze users first search the desired file from the center servers and then download the file from one of the data sources, thus resulting in a stable number of hop counts even as the number of users increase. Second, in NDN networks each peer node communicates with other nodes through multiple paths due to NDN's broadcast advantage, which increases the overlapping probability of multiple Interest packets from different NDNMaze users. Third, our proposed neighboring discover scheme for efficient file sharing also reduces the hop count since the scheme increases the probability of requesting files from nearby nodes that are able to satisfy Interest requests.

2) Data Request Latency: Next we study the data request latency which measures the round-trip time between the data request initiated from the request node and file download completion at the same node. As shown in Figure 9, the average data request latency of NDNMaze is much smaller than that of IPMaze. As the number of users increases, the average data request latency of IPMaze increases, while the average data request latency of NDNMaze remains steady.

Our in-depth examination finds that in IPMaze, the increase of users leads to additional load on the center servers, thus increasing the time for searching the desired files. However, in NDNMaze even as the number of users increases, the delay of data request remains unchanged. Due to in-network caching and multi-path forwarding, the hops in which NDNMaze users get the requested data are reduced, thus reducing the average time latency in which NDNMaze users get the request data. In addition, when there is more than enough cached data in NDN networks, the time latency tends to be stable.

[1]The Windows version of NDNMaze is built under Cygwin [9]

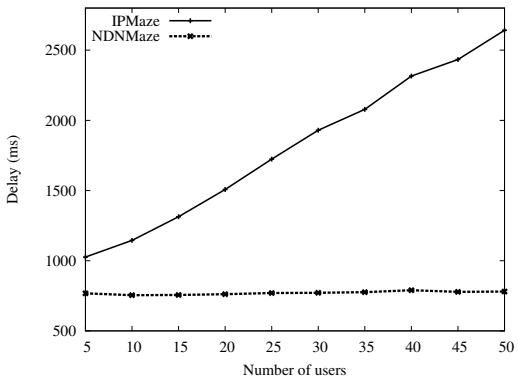

Figure 9. Data request latency of NDNMaze and IPMaze

3) Network Transmission Efficiency: The final performance metric we study is network transmission efficiency, which evaluates the ratio between effective network transfer load and the overall network load. Let e and t denote effective network transfer load and the overall network load, then the network transmission efficiency f becomes $f = \frac{e}{t}$. Figure 10 shows network transmission efficiency of NDNMaze and IPMaze with varying number of users. Clearly, NDNMaze achieves a significant higher than IPMaze.

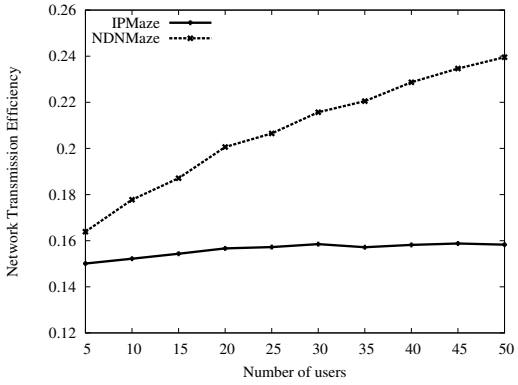

Figure 10. Network transmission efficiency of NDNMaze and IPMaze

The root causes of such difference are three-fold. First, the overall network load of NDNMaze is lower than that of IPMaze since the traffic redundancy increases exponentially [12] in IPMaze as a result of increasing data requesters and the network size. Second, the in-network caching in NDN architecture eliminate the repeated transmissions of data packets on the same file, as the later arrivals of NDN Interests for the same data can be satisfied by the cached Data packets along the path. In other words, in NDNMaze each piece of data is transmitted over a link no more than once. Third, for the both NDNMaze and IPMaze, the effective network transfer load is the same. Thus, the network transmission efficiency is inversely proportional to the overall network load, which explains why NDNMaze has a higher efficiency than IPMaze.

In summary, our experimental results demonstrate NDN-based peer-to-peer file sharing applications could explore the inherent advantages of NDN architecture and achieve better end-to-end performance. Compared with IPMaze in TCP/IP networks, NDNMaze not only reduces the latency on data request, but also effectively increases the network transmission efficiency and the Data request efficiency thanks to the cached data in intermediate NDN routers that satisfy later arrivals of the Interest requests on the same file.

As a final remark, note that although NDNMaze takes the inherent property of NDN in-network caching which IPMaze does not have, even if we take ISPs' TCP/IP cache for P2P into consideration (which improves IPMaze performance to some extent), the simulation results will still convince NDNMaze's superiorities since the former one is based on hop-by-hop while the latter one is from end to end. Besides, a fact has been provided in [13] that since the performance penalty of using NDN vs. TCP is around 20%, while the performance gain from sharing is integer multiples, there is a net performance win from using NDN even when sharing ratios/hit rates are low.

VII. Related Work

The research on NDN is extensive and generally spans two directions. The first one is supplementing NDN with mechanisms aiming to improve NDN architecture. The second one is investigating the feasibility, validity and performance of NDN. Our research follows the latter direction.

There is a fundamental difference between our paper and the above research works: the latter one concentrates on comparing NDN with other architectures or network protocols and purely considering NDNx [14], an open-source of NDN software prototype developed by UCLA[2], without any consideration to real application environment while the former one focuses dedicated attention on comparing the performance of NDN and IP with respect to P2P file sharing application environment.

To the best of our knowledge, several studies have referred to applying NDN in P2P networks, and a few primitive prototype of distributed applications have also been implemented over NDN to prove the validity of the new trend which transforms data into a first-class entity. Some of them are: VoCCN [15], which exemplifies how real-time voice applications are implemented using NDN; NDNPurple [16], which supports existing Instant Messaging applications running over NDN; ACT [17], which is a distributed audio conference tool using Named Data Networking approaches and MUC [18], [19], which is a distributed multi-user text chat application built over NDN.

[2]UCLA's project on the NDN direction is called NDN, hence the UCLA open source package is dubbed NDNx. NDN's project website: http://www.named-data.net/

VIII. Conclusions and Future Work

This paper designs and implements a peer-to-peer file sharing system over NDN. Leveraging the advantages of NDN architecture such as in-network caching and messaging mechanisms, we develop the NDNMaze system with a decentralized architecture and a simple yet scalable naming scheme, and integrate key system functionalities based on data-centric semantics. Our proposed NDNMaze system provides new insights on how to migrate existing TCP/IP applications to NDN architecture.

One thing worth mentioning is that, the different versions of IPMaze have inconsistent compatibility issues when IPMaze protocol upgrades. This kind of headache recurs frequently when applications need to upgrade in TCP/IP. Due to the longest prefix match in NDN, NDNMaze only defines the naming convention, and appends new components to the prefix, which brings better forward or backward compatibility to NDNMaze. The properties of NDN give a chance for users to be independent to applications and platforms, while the applications could still hold the features of themselves. Synthesizing all above, moving applications from TCP/IP to NDN provides a good idea for users and developers and bring a new experience in Internet.

Our future work focuses on the deployment and system evaluations of NDNMaze in a real network environment.We also plan to introduce a social network feature in NDNMaze to facilitate friends making and resource finding for further reducing the file search and download time.

References

[1] L. Zhang, D. Estrin, J. Burke, V. Jacobson, J. Thornton, D. Smetters, B. Zhang, G. Tsudik, D. Massey, C. Papadopoulos *et al.*, "Named data networking (ndn) project," *Relatório Técnico NDN-0001, Xerox Palo Alto Research Center-PARC*, 2010.

[2] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman, "A survey of information-centric networking," *Communications Magazine, IEEE*, vol. 50, no. 7, pp. 26–36, 2012.

[3] C. Dannewitz, "Netinf: An information-centric design for the future internet," in *Proc. 3rd GI/ITG KuVS Workshop on The Future Internet*, 2009.

[4] S. Tarkoma, M. Ain, and K. Visala, "The publish/subscribe internet routing paradigm (psirp): Designing the future internet architecture." in *Future Internet Assembly*, 2009, pp. 102–111.

[5] T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, and I. Stoica, "A data-oriented (and beyond) network architecture," in *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 4. ACM, 2007, pp. 181–192.

[6] "Top applications (bytes) for subinterface: Sd-nap traffic," in CA/DA workload analysis of SD-NAP data. Available: http://www.caida.org/research/traffic-analysis/byapplication/sdnap/, Tech. Rep., 2002.

[7] Q. Lian, Z. Zhang, M. Yang, B. Zhao, Y. Dai, and X. Li, "An empirical study of collusion behavior in the maze p2p file-sharing system," in *Distributed Computing Systems, 2007. ICDCS'07. 27th International Conference on*. IEEE, 2007, pp. 56–56.

[8] D. Fesehaye and J. Wei, "Snc: Scalable named data networking (ndn)-based conferencing architecture." *11th Annual IEEE Consumer Communications and Networking Conference (CCNC2014) - Special Session on Information Centric Networking. Las Vegas, USA. January 2014.*, 2014.

[9] J. Racine, "The cygwin tools: a gnu toolkit for windows," *JOURNAL of Applied Econometrics*, vol. 15, no. 3, pp. 331–341, 2000.

[10] B. M. Waxman, "Routing of multipoint connections," *Selected Areas in Communications, IEEE JOURNAL on*, vol. 6, no. 9, pp. 1617–1622, 1988.

[11] A. Medina, A. Lakhina, I. Matta, and J. Byers, "Brite: An approach to universal topology generation," in *Modeling, Analysis and Simulation of Computer and Telecommunication Systems, 2001. Proceedings. Ninth International Symposium on*. IEEE, 2001, pp. 346–353.

[12] R. Schollmeier and G. Schollmeier, "Why peer-to-peer (p2p) does scale: An analysis of p2p traffic patterns," in *Peer-to-Peer Computing, 2002.(P2P 2002). Proceedings. Second International Conference on*. IEEE, 2002, pp. 112–119.

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

[14] Project ndnx website. [Online]. Available: https://github.com/named-data/ndnx

[15] V. Jacobson, D. Smetters, N. Briggs, M. Plass, P. Stewart, J. Thornton, and R. Braynard, "Voccn: voice-over content-centric networks," in *Proceedings of the 2009 workshop on Re-architecting the internet*. ACM, 2009, pp. 1–6.

[16] J. Wang, E. Osterweil, C. Peng, R. Wakikawa, L. Zhang, C. Li, and P. Cheng, "Implementing instant messaging using named data," in *Proceedings of the Sixth Asian Internet Engineering Conference*. ACM, 2010, pp. 40–47.

[17] Z. Zhu, S. Wang, X. Yang, V. Jacobson, and L. Zhang, "Act: audio conference tool over named data networking," in *Proceedings of the ACM SIGCOMM workshop on Information-centric networking*. ACM, 2011, pp. 68–73.

[18] Z. Zhu, C. Bian, and L. Zhang, "Xmpp over named data networking: Design doc," 2012.

[19] Z. Zhu and A. Afanasyev, "Let's chronosync: Decentralized dataset state synchronization in named data networking," *Proceeding of the 21st IEEE International Conference on Network Protocols (ICNP 2013), Goettingen, Germany, October 2013.*, 2013.