SOCIETY FOR SCIENCE AND EDUCATION
UNITED KINGDOM

TNC TRANSACTIONS ON NETWORKS AND COMMUNICATIONS

VOLUME 4, ISSUE 4
ISSN: 2054 -7420

Serverless Mobile Multiuser Chat Application Based on ChronoSync for Named Data Networking

Yudi Andrean Phanama[1], Fransiskus. Astha Ekadiyanto[2] and Riri Fitri Sari[3]
Department of Electrical Engineering, Faculty of Engineering, University of Indonesia. Kampus Baru UI Depok 16424, Indonesia
yudi.andrean@ui.ac.id; astha.ekadiyanto@ui.ac.id; riri@ui.ac.id

ABSTRACT

Client-server application model is prone to single point of failure on the server, thus making distributed model a good choice against the problem. Named Data Networking (NDN) has been a prospecting architecture to enable the future internet, putting the focus of the network to 'what' data to fetch instead of 'where' to send the data. NDN can provide communication for many networking devices, including mobile devices.

In this paper, we present DChronoChat, an implementation of ChronoSync synchronization protocol to provide a distributed, serverless mobile multiuser chat application on NDN network. We implemented DChronoChat using jNDN 0.10 library on Android devices with real NDN Forwarding Daemon (NFD) for Android device. We show through our implementation that ChronoSync can provide synchronization in real implementation of mobile multiuser chat application.

Keywords: Named Data Networking; Distributed Application; Serverless; Mobile Multiuser Chat.

1 Introduction

Named Data Networking (NDN) is a novel Internet architecture which relies on the exchange of 2 types of datagram, i.e. Interest and Data. Interest is a request for a Data packet of the requested name, and Data is the reply packet of an Interest. The model of communication is "data centric", as every communication is a request for Data, and a reply in form of Data. This communication model leads to one-to-one balance of Interest-Data packet, every 'one Interest' is satisfied with exactly 'one Data'. NDN's stateful forwarding plane [1] provides smarter middle nodes and inherited multicast delivery from its architecture. Its embedded security demands every data in the network to be secured with signatures [2]. As a novel architecture, NDN still needs a continual validation and evaluation in real life implementations.

Distributed applications are designed to settle out the problem with centralized client-server application, a single point of failure. NDN's architecture brings a new challenge in creating distributed applications. NDN inherently supports caching and multicast, which brings its own potential to support applications which use networking. Multiuser chat application can be delivered using either centralized client-server model [3] or distributed model [4]. In the distributed model, every application entity acts as the server and the client simultaneously, which gives the potential to keep the service running without a single point

DOI: 10.14738/tnc.44.2179
Publication Date: 26th August 2016
URL: http://dx.doi.org/10.14738/tnc.44.2179

of failure like in the client-server model. In the implementation, distributed application can be deployed in either stationary devices or in mobile devices, with their own constraint and requirements.

A crucial thing that needs to be handled in chat application is chat dataset synchronization across all the user. ChronoSync [5] is a synchronization protocol designed to synchronize anything over NDN. ChronoSync can be used to sync chat data, file sharing, joint editing, etc. Through the set of simulation in [6], it has proven to be effective and efficient in the synchronization of dataset changes of the simulated application.

In this work we designed and implemented DChronoChat, an Android mobile multiuser chat application with ChronoSync protocol to synchronize the dataset of each application entity. We implemented DChronoChat using jNDN 0.10 [5], an NDN Java library. We then evaluated DChronoChat on our real NDN scenario, overlaid on top of IP network with Android devices and NDN Forwarding Daemon (NFD) [7] running on the devices.

The rest of this paper is organized as follow: Section 2 explains how NDN provides network connection to get data, and how ChronoSync works. Section 3 describes the design and implementation of DChronoChat with ChronoSync protocol. Section 4 presents our experiment result and analysis. Section 5 discusses open problems and our future works. Finally, Section 6 concludes the paper.

2 Named Data Networking

2.1 NDN Forwarding and Routing

NDN brings the use of two types of network datagrams, Interest and Data packet. Interest is the request for Data, and each 'one Data' satisfies 'one Interest' sent by the receivers. Each Interests and Data is bound to the name, which represents the data needed by consumers. When a receiver wants to request specific data, the receiver fetches the data by sending Interest packet containing the name of the data. The role of the NDN network nodes is to forward the Interest to the data source, and forwarding the data in form of Data packet by the data name back to the receiver, following the exact path the Interest passed in the network. If the network cannot forward the Interest to the Data source because it does not have the prefix of that name, network will notify the receiver using NACK packet. Each network node does the forwarding using NFD [8], which decides to what interface an Interest of a name prefix will be forwarded to.

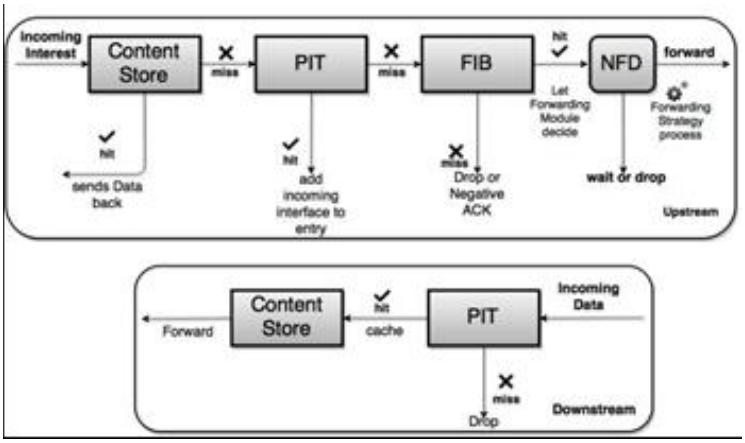

Figure 1. Forwarding Mechanism in NDN

NFD determines whether, and where to forward packets using the information provided in three data structures: Content Store which acts like a cache; Pending Interest Table (PIT) which list every unsatisfied Interests and their incoming interfaces; and Forwarding Information Base which lists the name prefixes and their outgoing interfaces [9]. Along with those three data structures, NFD uses "forwarding strategies" with different methods to forward packets. This forwarding mechanism gives NDN a stateful forwarding plane [6], enabling routers to measure the performance of different outgoing interfaces, detects link failure quickly, and tries different forwarding path after failures. Figure 1 describes the forwarding pipeline of NDN, upstream flow of Interest forwarding and downstream Data forwarding. First, the incoming Interest will be checked in the Content Store. If it is a cache hit, the Data will be sent back to the receiver, if it is not, it will be passed to the PIT. In the PIT check, if the name of Interest matches any in the PIT, the incoming interface will be added to the PIT entry, and the Interest will be dropped, due to the fact that the same Interest has been requested by previous node. If it does not match, it will then be checked into the FIB. In the FIB, if the name does not match or the router does not have the route to the Data. The Interest will be dropped and NACK packet will notify the receiver to find other routes. If the Interest name match the FIB, the Interest will be passed to NFD, which decides the forwarding.

Unlike IP, which leaves the forwarding plane unintelligent, giving all the control of routing and forwarding to the routing plane, routing in NDN is used only to bootstrap the Forwarding Information Base of NDN [7], while the forwarding decisions are given to the NFD on the forwarding plane. Routing protocols in NDN fills the entries of Routing Information Base (RIB) which will then be used by NFD to fill the FIB (depending on the NFD policy). The routing protocol used in NDN is a general purpose link state routing called NDN Link State Routing (NLSR) [8], which exchanges the link state information in 'Interest-Data' communication manner. NLSR builds the routing table by establishing the neighborhood with the adjacent nodes and disseminates the routing information to bootstrap the FIB. Forwarding is then decided by the NFD, depending on the forwarding strategy used. Forwarding strategies enable the intelligence to make decision of when, where, and whether to forward Interests. There are 5 default forwarding strategies in NFD, i.e. Best Route, Multicast, Client Control, NCC, and Access Router Strategy.

For development purposes as of NDN version 0.4.1 [7], NDN can be run and implemented as an overlay network on the current IP protocol. NDN is stated as a 'universal overlay network' which means that any protocol that runs over IP, including the IP itself can be run over NDN, and NDN can be run over IP. To implement NDN over IP, one must have ndn-cxx [10] and NFD [9] installed in the IP device.

2.2 ChronoSync and ChronoChat

ChronoSync [5] [11] is designed to support the NDN's implementation of distributed applications such as group chat, file sharing system, file editing, etc. by providing a synchronization service for the applications. ChronoSync uses NDN's features such as multicasting and smart forwarding to efficiently synchronize the knowledge of the dataset state among distributed users..

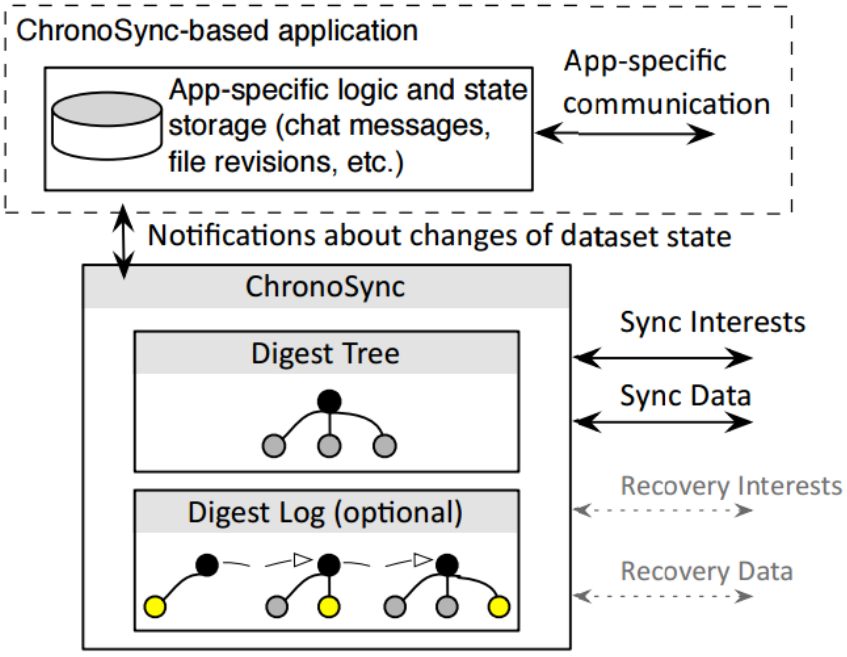

Figure 2. Application with ChronoSync [6]

In ChronoSync-implementing application, there are two interdependent modules as depicted in Figure 2. ChronoSync module synchronizes the knowledge of the users across the network, and application module gives the action according to the change of dataset detected by ChronoSync. To use ChronoSync, application just need to provide the naming scheme and rule, then let ChronoSync do the synchronization. ChronoSync summarizes the dataset state in the form of Merkle Tree [12] and exchanges the state digest through NDN network to detect any different state digest.

For the purpose of detecting the difference in dataset state, ChronoSync module of each application sends sync interest which contains the state digest maintained in the root of the digest tree, placed in the Name section of that sync interest. To know which changes was done by whom, ChronoSync also provides digest log which stores the changes of every state digest in form of key-value pair. With the digest tree and digest log, ChronoSync can reply the sync interest with the data containing the changes.

An implementation of ChronoSync is ChronoChat [6], a C++ based desktop multiuser chat application running on NDN with NFD-cxx. In ChronoChat, there are two naming schemes used, one for broadcast prefix for sync data, and the other to show the chat data of each user. For example, /ndn/broadcast/ChronoChat/TestRoom shows a naming scheme for a sync data broadcast for TestRoom, and /ndn/clab/yudiandrean/ChronoChat/TestRoom/82 shows the name of the data with sequence number 82 under user prefix yudiandrean in chatroom TestRoom. The data produced by users are sequentially named under the user chat prefix.

2.3 NLSR Routing

NDN Link State Routing [8] [13] is an intra-domain routing protocol for NDN. The protocol disseminates routes to name prefixes instead of IP prefixes in common IP LSR protocol. The routing protocol uses ChronoSync to synchronize the LSA information in the network. NLSR does not support inter-domain routing, and is implemented for single domain use.

3 Design and Implementation

DChronoChat uses jNDN 0.10 library provided by NDN project to implement the native functions of NDN on Android. We use Android Studio [14] as the IDE to develop the application. In this section we describe DChronoChat application design. We first describe the architecture of the application, the naming scheme for the application, illustrate the flow process of DChronoChat, and then describe the heartbeat mechanism in the application.

3.1 Software Architecture

DChronoChat is inspired by ChronoChat. There are 3 main operating modules in DChronoChat: application module, ChronoSync module, and Android NFD. DChronoChat uses ChronoSync module to synchronize the knowledge of dataset state in form of digest tree. The application uses Android NFD to interact with the network using NDN protocol. The architecture of DChronoChat is depicted in Figure 3.

The application provides data structure to store the messages needed to present to the user. The messages are fetched by the application to the data sources, according to the sync data provided by ChronoSync module. ChronoSync sends sync interest and retrieves sync data containing the names of the data to be fetched. The application can then fetch the data based on the data names provided by ChronoSync.

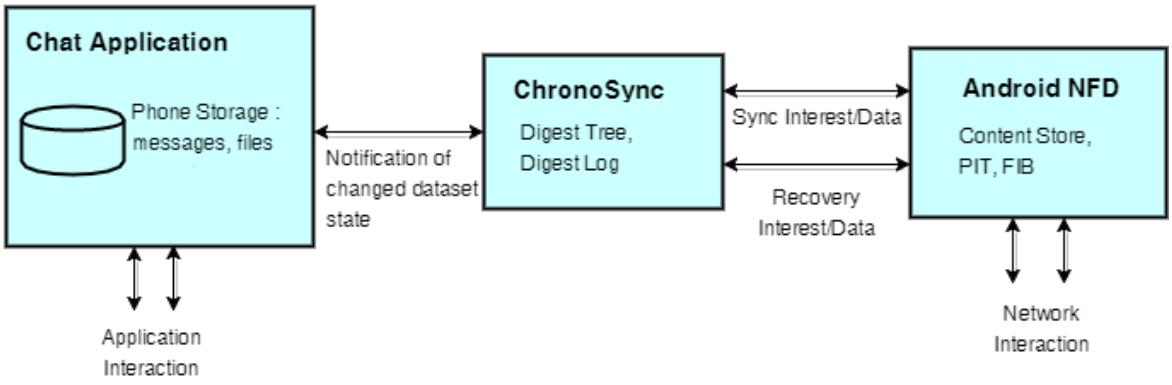

Figure 3. DChronoChat's architecture

3.2 Naming Scheme

Naming is important in any NDN application [15]. The name is carried in Interest and Data packet to identify data, and is used by the NDN network to determine where to forward Interest. We use two types of naming, broadcast prefix and user chat data prefix. The naming scheme is shown in Figure 4.

```
/ndn/broadcast/ChronoChat/TestRoom/gjs3123..
(a)

/ndn/ui/yudi/ChronoChat/TestRoom/1467070123/12
(b)
```

Figure 4. Naming schemes for DChronoChat

Figure 4 (a) shows an example of application broadcast prefix to deliver the sync interest for getting sync data back throughout the network in a multicast manner. The prefix /ndn/broadcast/ChronoChat

/TestRoom represents the broadcast prefix of the chat room, and the prefix /gjs3123.../ represents the latest digest of the interest sender. Every user of DChronoChat will register the broadcast prefix to the designated router/hub so that the broadcast message can be delivered. The router/hub of the chat itself also need to manually register broadcast prefixes to all the users, as Android NFD 0.4.1 [16] still does not support auto prefix propagation. With the presence of auto prefix propagator for Android NFD, the manual registering of broadcast and data prefixes is not needed.

In Figure 4 (b), an example of a user's data prefix is shown. The prefix is routable so that interests can be forwarded to the data producers, or in this case the users. Prefix /ndn/ui/clab/yudiandrean/ represents the route to the user, /ChronoChat/TestRoom/ represents the application prefix and chat room, /1467070123/ represents the session number used in handling group roster to identify the session identifier, and finally /12/ prefix represent the sequence number of the data from the user.

3.3 Activity Diagram

To represent the flow of process in DChronoChat, we used activity diagram as depicted by Figure 5. The receiver first will register its broadcast prefix and chat data prefix to the designated hub as the process of logging in into the application. Then the user will start the synchronization process by sending the default sync interest with the same digest to the broadcast prefix. This will result in a stable state, where the state digests are in steady state, the same state digests on all users. When the first user changes the dataset digest by sending a message, the user will satisfy the outstanding interest with the corresponding sync data, giving the information of what changes, and who caused the changes to the other users. The other users then sends the chat data interest to get the real chat data from the sending user.

3.4 Heartbeat

To notify the other users about the alive status of a user, we implemented a heartbeat mechanism by sending HELLO messages to the other users. The heartbeat is sent every 60 seconds. It is also used by the application to notify other users about the presence of a joining user. A joining user will firstly send HELLO message to other users. The roster management mechanism of every user will then manage the HELLO message as a join request to the chat room.

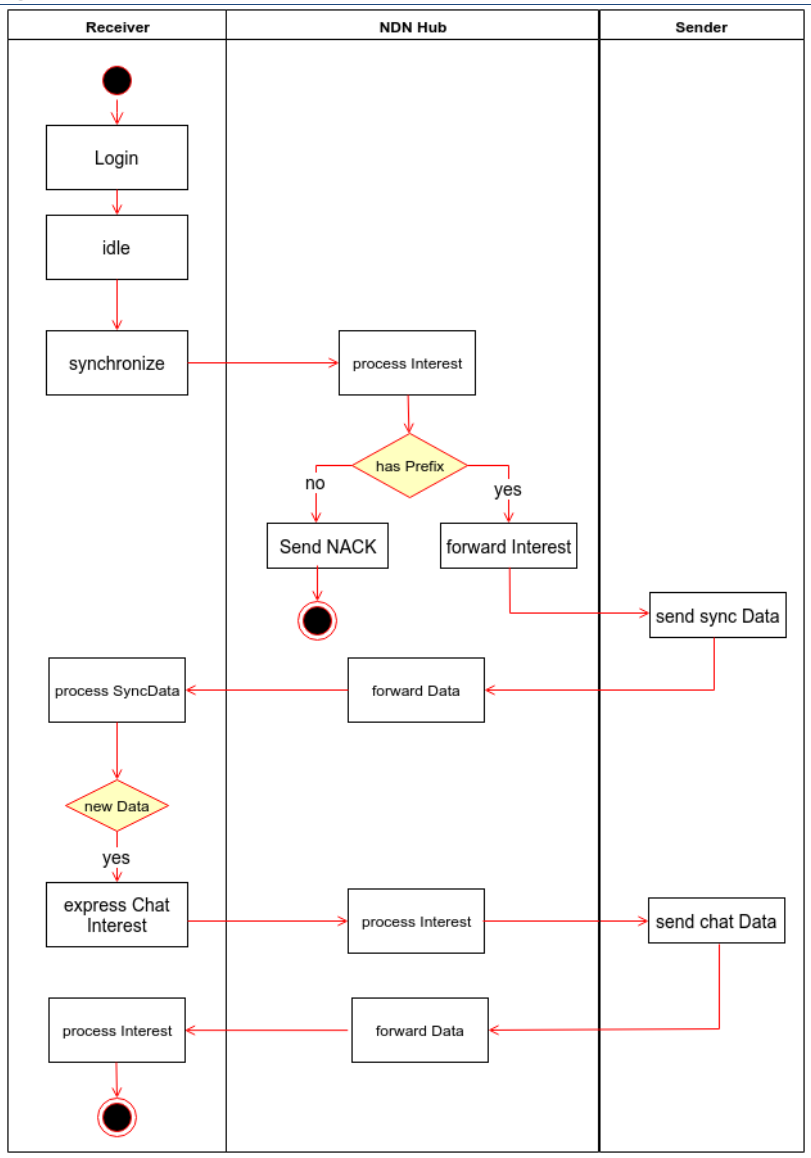

Figure 5. DChronoChat activity diagram

4 Evaluation and Analysis

We evaluated DChronoChat and NDN in providing serverless multiuser chat service through several scenarios with 2-10 users. We used real Android devices running on API Level 22 [17] and Android-x86 [18] virtual machines. The aim of the evaluation is to validate the implemented ChronoSync protocol in providing sync service, and to evaluate NDN's routing protocol, NLSR in providing a real implementation of chat service over NDN with multiple hubs/routers.

4.1 Scenario 1 – 2 Users with 1 Hub

This scenario represents two users chatting in a chat room. This may represent a model of private chat between two users. In this test scenario, two different users designate the same hub as their router. We used real Android devices TCP connection across all devices. Each user sends 100, 200, and 300 messages. The result of this scenario is presented in Figure 6.

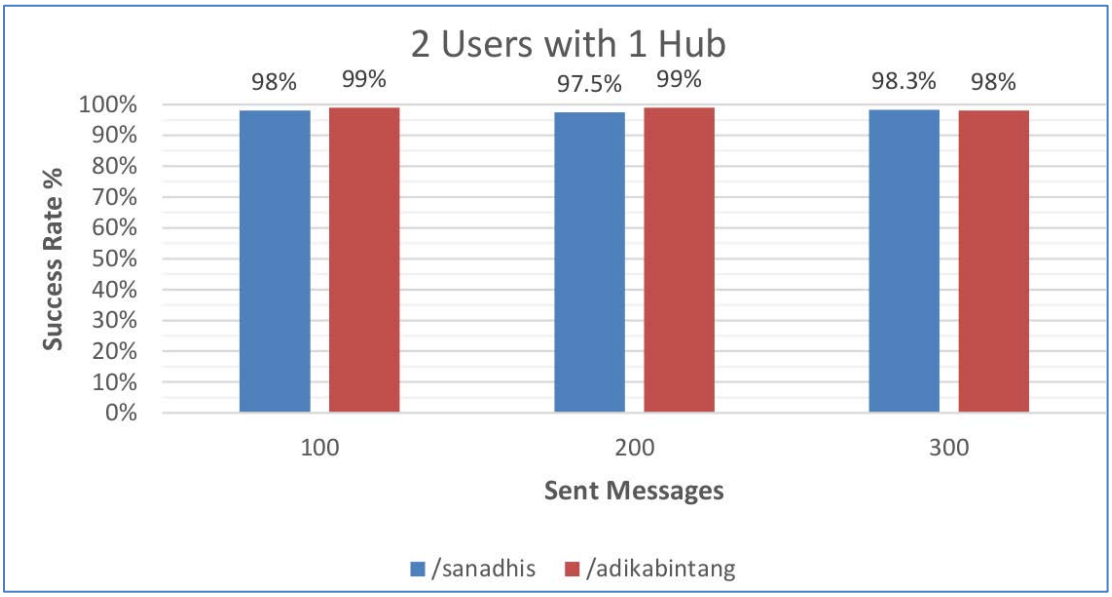

Figure 6. Success Rate of 2 Users with 1 Hub

The result shows the success rate of up to 99% in the experiment runs. This shows that success rate will stay steady as the number of messages sent increases. It may be because of the heartbeat mechanism, when the user fetches message or sends message on the same time with heartbeat sequence publishing, causing the Android NFD to handle only one of the event. Investigation on Android NFD may be done to improve the NFD.

4.2 Scenario 2 – 3 Users with 1 Hub

This scenario represents three users chatting in a chat room. This may represent a model of group chat between more than two users. In this test scenario, three different users designate the same hub as their router. We used TCP connection across all devices. The results of this scenario is presented in Figure 7.

Identical to Scenario 1 results, the success rate stays steady as the number of message sent increases. However, the success rate is slightly lower than the session with only two users, with up to 95.7% message successfully fetched. This is because of more network load that needs to be done by the devices. Android devices manages its own network resources on kernel level with battery constraint, which might affect the performance of Android NFD on real Android devices.

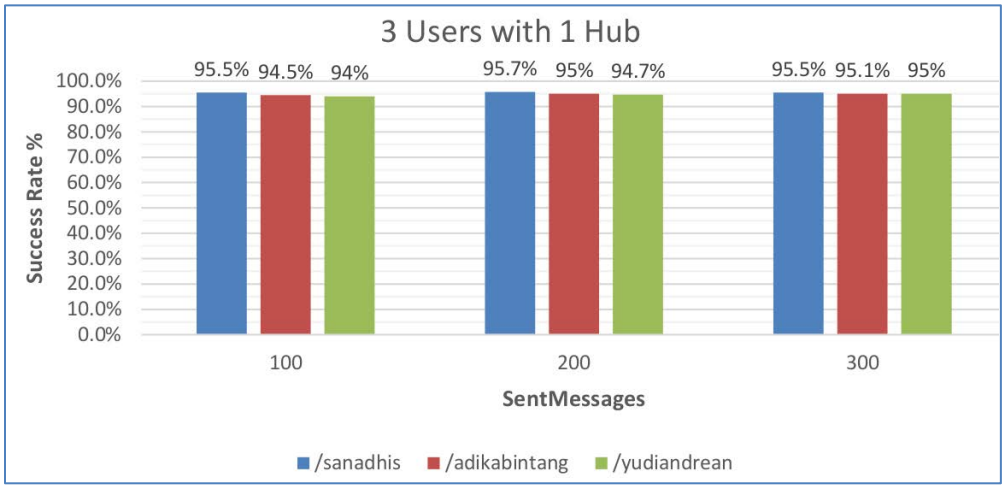

Figure 7. Success Rate of 3 Users with 1 Hub

4.3 Scenario 3 – 2, 3, 5, and 10 Users with 1 Hub

In this scenario, we aim to evaluate the message delivery performance result of the implementation. We created four runs of real experiment on our testbed of 10 Android-x86 virtual machines running on QEMU environment with 2, 3, 5, and 10 users on each run respectively. We used virtual machines to avoid the energy constraint of Android devices, to see the trend of the implementation handling more users. The results of the experimentation are presented in Figure 8 and Figure 9.

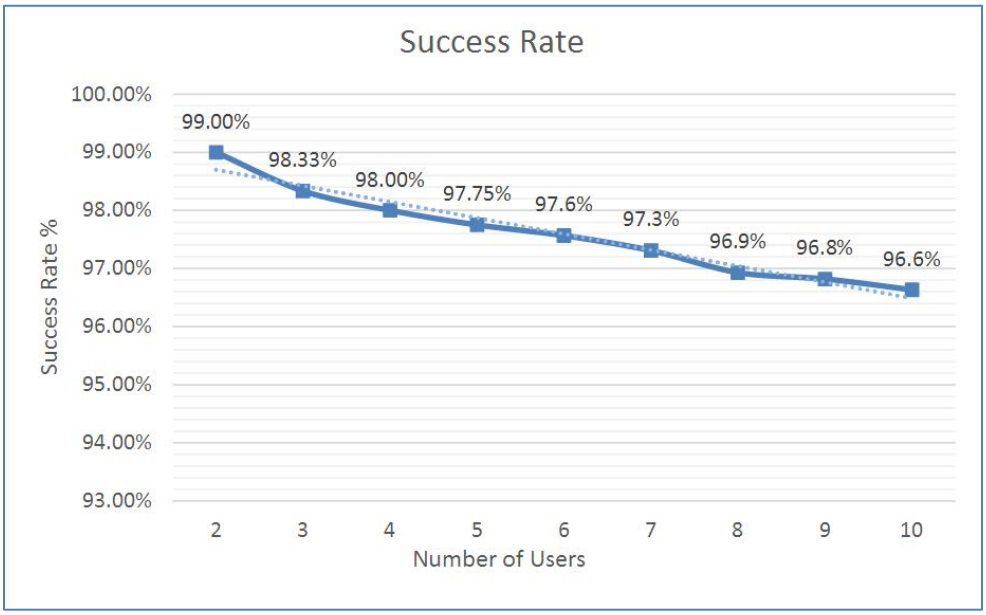

Figure 8. Success Rate of Scenario 3

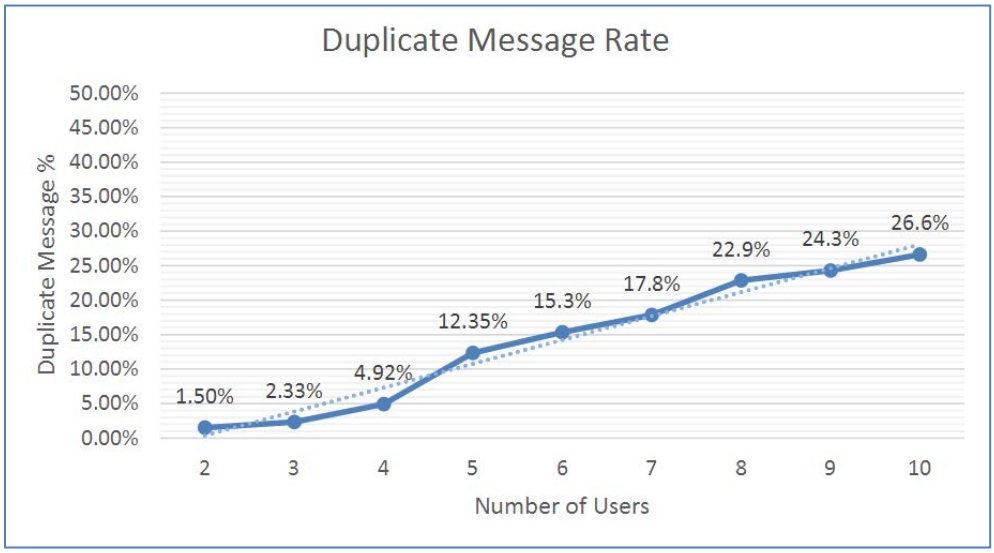

Figure 9. Duplicate Message Rate of Scenario 3

From the result, we show that the success rate decreases slightly as the number of user increases. Similar to the previous scenarios, this may be the result of the heartbeat mechanism. We also measured the percentage of the duplicate message packets fetched by every user in the chat room. The duplicate messages are not shown to the user interface as the roster management handles it, but it loads the network with bogus packets. The rate increases as the number of users increases greatly from 2.3% to 12.4% as the number of users changes from three participants to five participants. This might be in cause of more broadcast messages unhandled by the implementation. Android NFD does not support auto-prefix propagation as of version 0.4 [16], which demands the hub to manually register broadcast and application prefixes to the designated Android NFD. We implemented the manual register with a shorter prefix which may cause more broadcast messages.

4.4 Scenario 4 – 3 Users with 4 Hub Implementing NLSR

This scenario represents three users in a chat room. This represent a model of group chat between more than two users. In this test scenario, three different users designate different hubs as their router. The hubs implements NLSR routing protocol to disseminate the knowledge of forwarding route to all the other hubs. We used TCP connection across all mobile devices and UDP for NLSR operations. The results of this scenario is presented in Table 3.

The result shows similar success rate with Scenario 2, with the average success rate of 96.2%. This shows that NLSR is capable to provide a scalable network environment for DChronoChat and similar implementations. NLSR also has the potential to provide an NDN inter-domain network for the implementation if the protocol is improved further.

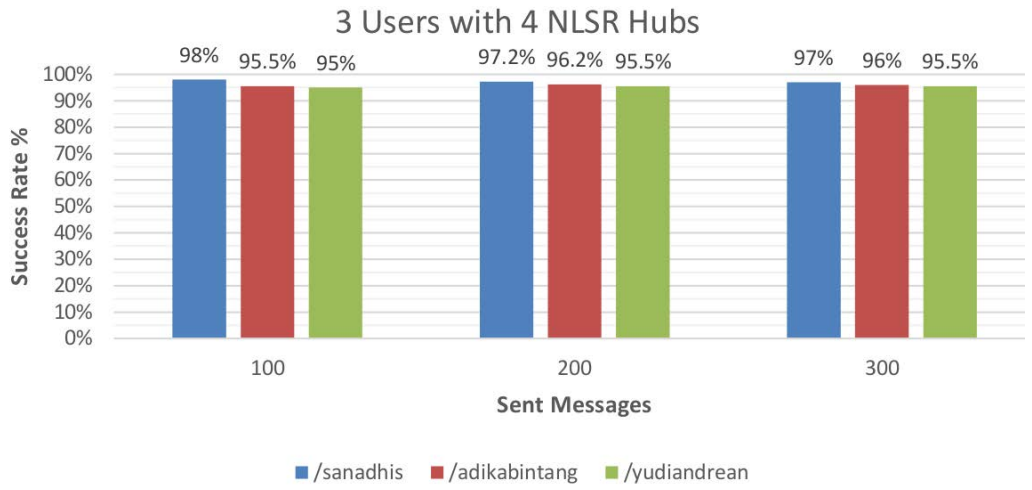

Figure 10. Success Rate of 3 Users with 4 NLSR Hubs

5 Future Works

Our implementation shows that NDN is capable of providing network for group chat implementation with text messages without the use of servers in the same network. NDN's implementations such as FileSync [19] and ChronoShare [20] enable NDN users to exchange files across the network, and also NDNRTC [21] to provide real time video conference for users. Our future implementation includes building a mobile chat platform capable of exchanging files and enabling video chats over the same integrated applicatin on NDN network.

6 Conclusion

In this paper, we designed and implemented DChronoChat, a mobile serverless multiuser chat application using ChronoSync in real NDN implementation over IP network, and evaluated it under overlay IP network. We also evaluated the application to work with many NDN hubs with NLSR routing protocol to disseminate routing information. From our evaluation, we show that DChronoChat is able to deliver serverless multiuser chat service over NDN with the success rate of up to 99% in two users implementation, and up to 96.6% in the implementation of ten users. We also show that NLSR can provide routing dissemination for DChronoChat to use under many hubs scenario, with the delivery rate of up to 98%.

REFERENCES

[1]. C. Yi, A. Afanasyev, I. Moiseenko, L. Wang, B. Zhang, and L. Zhang, "A case for stateful forwarding plane," *Comput. Commun.*, vol. 36, no. 7, pp. 779–791, Apr. 2013.

[2]. L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, kc claffy, P. Crowley, C. Papadopoulos, L. Wang, and B. Zhang, "Named data networking," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 3, pp. 66–73, Jul. 2014.

[3]. Berson, Alex. *Client/server architecture*. McGraw-Hill, Inc., 1996.

[4]. Umar, A. and Fraser, C.W., 1993. *Distributed computing: a practical synthesis of networks, client-server systems, distributed applications, and open systems*. Prentice-Hall, Inc...

[5]. Z. Zhu and A. Afanasyev, "Let's ChronoSync: Decentralized dataset state synchronization in Named Data Networking," in Proceedings of the 21st IEEE International Conference on Network Protocols (ICNP 2013), pp. 1–10.

[6]. Z. Zhu, C. Bian, A. Afanasyev, V. Jacobson, and L. Zhang, "Chronos: Serverless Multi-User Chat Over NDN." NDN Technical Report NDN-0008. Revision 1: October 2012.

[7]. NDN Project, "NFD Developer's Guide - Named Data Networking (NDN)," *Named Data Networking (NDN) Technical Report NDN-0021*. Revision 5: Oct 2015.

[8]. A. K. M. M. Hoque, S. O. Amin, A. Alyyan, B. Zhang, L. Zhang, and L. Wang, "NLSR: named-data link state routing protocol," The 3rd ACM SIGCOMM Workshop on Information-Centric Networking, August 2013, p. 15.

[9]. NDN Project, "Overview (jndn 0.12 API)." [Online]. Available: http://named-data.net/doc/0.4.1/jNDN/. [Accessed: 14-Jun-2016].

[10]. NDN Project, "ndn-cxx: NDN C++ library with eXperimental eXtensions — ndn-cxx: NDN C++ library with eXperimental eXtensions 0.4.1-6-g4f512fb documentation." [Online]. Available: http://named-data.net/doc/ndn-cxx/current/. [Accessed: 13-Jun-2016].

[11]. NDN Project, "GitHub - named-data/ChronoSync: sync library for multiuser realtime applications for NDN." [Online]. Available: https://github.com/named-data/ChronoSync. [Accessed: 27-Jun-2016].

[12]. R. C. Merkle, "A Digital Signature Based on a Conventional Encryption Function," in *Advances in Cryptology — CRYPTO '87*, vol. 293, C. Pomerance, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1988, pp. 369–378.

[13]. "A Secure Link State Routing Protocol for NDN - Named Data Networking (NDN)," *Named Data Networking (NDN) Technical Report NDN-0037*. Revision 1: January 2016.

[14]. Wolfson, M. and Felker, D., 2013. *Android Developer Tools Essentials: Android Studio to Zipalign*. " O'Reilly Media, Inc.".

[15]. NDN Project, "NDN Technical Memo: Naming Conventions", NDN Technical Report NDN-0022, Revision 1: July 21, 2014.

[16]. NDN Project, "Overview - NFD-android - NDN project issue tracking system." [Online]. Available: https://redmine.named-data.net/projects/nfd-android. [Accessed: 13-Jun-2016].

[17]. Android Developers, 2015. *Android Lollipop*. [Online]. Available: https://developer.android.com/about/versions/lollipop.html. [Accessed: 13-Jun-2016].

[18]. Android-x86 Project, "Android-x86 - Porting Android to x86." [Online]. Available: http://www.android-x86.org/. [Accessed: 13-Jun-2016].

[19]. NDN Project, "FileSync/NDN: Peer-to-Peer File Sync over Named Data Networking - Named Data Networking (NDN)," *Named Data Networking (NDN) Technical Report NDN-0012*. Revision 1: March 2013.

[20]. A. Afanasyev, Z. Zhu, Y. Yu, L. Wang, and L. Zhang, "The Story of ChronoShare, or How NDN Brought Distributed Secure File Sharing Back," IEEE MASS 2015 Workshop on Content-Centric Networks 2015, pp. 525–530.

[21]. P. Gusev and J. Burke, "NDN-RTC: Real-Time Videoconferencing over Named Data Networking," ICN '15 Proceedings of the 2nd International Conference on Information-Centric Networking 2015, pp. 117–126.