Android Multimedia Sharing Application over NDN

Damian Coomes, Ashlesh Gawande
University of Memphis
{dmcoomes, agawande}@memphis.edu

Nicholas Gordon
University of Pittsburgh
nick.gordon@pitt.edu

Lan Wang
University of Memphis
lanwang@memphis.edu

ABSTRACT

Named Data Networking (NDN) thrives in use cases that require mobile peer-to-peer data sharing. To amplify interests in NDN research, we developed an NDN multimedia sharing application for mobile devices that mirrors the popular Snapchat app. Our research focuses on (1) creating a completely decentralized application, (2) exploring new trust models; and (3) using a new synchronization protocol that allows for synchronization of subsets of data. Our demo will show the basic flow of the application, from making friends to publishing and fetching files between Android devices.

CCS CONCEPTS

• **Networks** → *Social media networks*; *Mobile ad hoc networks*; • **Human-centered computing** → *Mobile phones*;

ACM Reference Format:
Damian Coomes, Ashlesh Gawande, Nicholas Gordon, and Lan Wang. 2018. Android Multimedia Sharing Application over NDN. In *Proceedings of ICN '18.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 PROBLEM AND MOTIVATION

The Named Data Networking (NDN) architecture has great potential in improving application performance especially in use cases where peer-to-peer data sharing is desirable. For example, nTorrent [4], an NDN version of BitTorrent, showed nearly a 50% reduction in network traffic and faster download speeds compared to BitTorrent. The NDN ChronoChat application [12] also showed faster synchronization with 40% of chat messages experiencing less than 20ms of delay compared to only 13% in a TCP/IP chat application. We believe that more research effort needs to be focused on developing NDN applications, particularly on mobile devices, for two important reasons. Firstly, from a research standpoint, the process of creating these applications will likely come with new challenges, which will highlight any potential deficits in the infrastructure. Secondly, providing a fully functioning mobile application will amplify interests in NDN research both in the current NDN community and outside the community.

In this work, we create an NDN application that closely mirrors Snapchat, a massively popular TCP/IP photo-sharing mobile application. This would provide a side by side comparison of NDN and TCP/IP and will ideally demonstrate the value of using NDN in modern day networking. We aim to explore the following research areas. First, we investigate the feasibility of developing a completely decentralized application. Removing the need to rely on a central entity to serve data and user information improves the application's mobility, security, and scalability. Second, due to the lack of a natural root identity trusted by all the users we would like to use an alternative trust model instead of the standard NDN hierarchical trust model for data authentication. Third, we evaluate how well Sync, which is the transport service in NDN, can support various types of data sharing needs. In particular, we use our PSync protocol to support the discovery of new friends, subscription of photos and stories among friends, and direct sharing of specific files.

2 PREVIOUS WORK

NFD has been ported to Android and used for a few applications [6]. NDNFit [10] collects health data on an Android device and publishes this data. ChronoChat allows users to form chatrooms to enable group messaging [7]. NDN-Whiteboard [3] enables users to share a "whiteboard" on which they can draw together in real time. Another application Now@ [2] implements a similar design to Twitter by allowing users to subscribe to multiple namespaces under which users can write posts.

3 DESIGN

We aim to achieve a fully decentralized design without a central database for one to check user names or find friends. Our design should also allow users to authenticate each other's data without relying on a few public trust anchors.

3.1 Finding Friends

In order to use the app, one needs to have friends with whom one can communicate. Without a central server that stores all the users in the system, each of the users will periodically advertise their names and pass around a list of all known users as shown in Figure 1. The application will collect such information and use it to avoid user name collisions and friend discovery. Given the potential size of the list as the application grows, nodes may decide to store only a portion of the user list based on certain constraints (e.g., only store users that are friends' friends).

3.2 Making Friends

Once the user has found someone he or she wants to be friends with, they can befriend one another in one of two ways. The first way can be done when the two users, Alice and Bob, are in the same physical location. The app generates a QR code containing the user's username and public key. Alice and Bob can become friends by scanning each other's QR codes, which registers the scanned user in their friends list. The other way to become friends is for

ICN '18, September 21–23, 2018, Boston, USA
© 2018 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

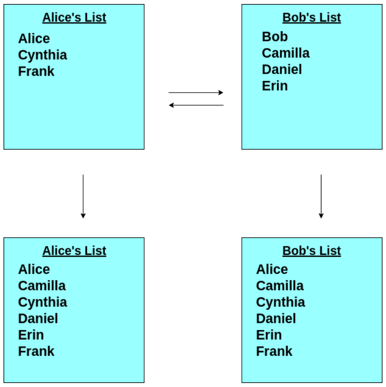

Figure 1: Users Alice and Bob exchange their user lists and update their copy with the differences.

Alice to send Bob a friend request in the form of an Interest. Bob can then choose to accept the invite and send a Data packet containing his public key. Bob will also send Alice a friend request to which Alice will respond by sending her own public key.

3.3 Sharing Data

Users can obtain new content from their friends and can share content with all or a subset of their friends. This is achieved by using a synchronization protocol which maintains a digest of the current state of each producer's published content. When a user produces new content, e.g., photos or files, the sync protocol will notify the user's friends of the new content. By default, all the friends can retrieve the content. However, the content can also be shared with a subset of friends of one's choosing. In this case, i.e., the user wants to restrict access to the content, the content will be encrypted and the notification message will contain a list of friends who can retrieve the decryption key which is encrypted with valid recipients' public keys. Although every friend will be notified of the new data, if a friend is not a listed recipient, the application will not fetch the new content. Even if the friend tries to retrieve the content, he/she cannot decrypt it. In addition, a user can subscribe to a friend's location feed – whenever the friend's location changes, this user will receive an update. In the future, we will add other types of feeds for users to subscribe to.

Our application uses PSync [11] (ported to Android) to support all the required sync functionality. More specifically, we use its full sync mode to support sharing of user lists and user content, and we use its partial sync mode to support subscriptions.

3.4 Authenticating Data

Since every data packet is signed by the original producer, the corresponding public key can be used to authenticate the data. This is the purpose of the exchange of keys when new friends are made. Doing so at the beginning removes the need to request certificates whenever a friend makes new content available. If the key pair ever changes, the application will notify the user's friends. The friends' apps will then retrieve the new public key and update their friend records. To authenticate a friend's public key, we will explore two options (a) a web-of-trust schema as proposed in [9] or (b) a block-chain based public key management system [8].

4 IMPLEMENTATION

We have developed a prototype implementation with basic functionality of making friends and sharing files/photos.

Making Friends. A QR code generator and a QR scanner, via the ZXing library [1], have been implemented for the purposes of facilitating in person friend registration and filename acquisition. Upon logging in as a new user, the app generates a key pair and encodes the user's chosen username and public key in the form of a QR code. When friends scan one another's QR code, the app creates a new file to store the user's public key. Then, the users can successfully retrieve data their friends have published.

Sharing Files and Photos. A file selection activity was created to allow users to publish files stored on their phone, packetizing them into individual data segments. Publishing a file also creates a QR code which encodes the data name. Other phones are able to fetch files by its name or by scanning the corresponding QR code. Users are also able to use the camera to take pictures, save them, and use a similar file selection activity to publish/fetch photos.

The jNDN library [5] is used for creating packets. This allows for Interests and Data to be named, packaged, sent, and received by Android phones. The application is assisted by the NFD-Android application which, while running, allows the application to interface with the network. When retrieving data, jNDN's SegmentFetcher is used to fetch all necessary data segments.

By using the public key that was exchanged upon friendship acquisition, a user can verify that the data segments are actually from their friend. If the stored friend's public key does not match the signature, the data packet is dropped, resulting in a timeout.

ACKNOWLEDGMENT

This work is supported by the National Science Foundation award CNS-1629769.

REFERENCES

[1] Agustin Delgado, et al. 2011. ZXing (Zebra Crossing) barcode scanning library for Java, Android. https://github.com/zxing/zxing
[2] Omar Aponte and Paulo Mendes. 2017. Now@ - Content Sharing Application over NDN. In *ACM ICN 2017*.
[3] Sumit Gouthaman, Peter Huang, and Peter Bankole. 2015. NDN-Whiteboard. https://github.com/named-data-mobile/apps-NDN-Whiteboard
[4] Spyridon Mastorakis, Alexander Afanasyev, Yingdi Yu, and Lixia Zhang. 2018. nTorrent: Peer-to-Peer File Sharing in Named Data Networking. In *ICCCN*.
[5] NDN Project Team. 2013. jNDN: A Named Data Networking client library for Java. https://github.com/named-data/jndn
[6] NDN Project Team. 2015. Android implementation of NFD. https://github.com/named-data-mobile/NFD-android
[7] Tyler Vernon Smith, Alexander Afanasyev, and Lixia Zhang. 2017. *ChronoChat on Android.* Technical Report. UCLA.
[8] Kan Yang, Jobin J. Sunny, and Lan Wang. 2018. Blockchain-based Decentralized Public Key Management for Named Data Networking. In *The International Conference on Computer Communications and Networks (ICCCN 2018)*.
[9] Yingdi Yu, Alexander Afanasyev, Zhenkai Zhu, and Lixia Zhang. 2014. *An Endorsement-base Key Management System for Decentralized NDN Chat Application.* Technical Report. UCLA.
[10] Haitao Zhang, Zhehao Wang, Christopher Scherb, Claudio Marxer, Jeff Burke, and Lixia Zhang. 2016. Sharing mHealth Data via Named Data Networking. In *ACM ICN 2016*. 142–147.
[11] M. Zhang, V. Lehman, and L. Wang. 2017. Scalable name-based data synchronization for named data networking. In *IEEE INFOCOM 2017*. 1–9.
[12] Zhenkai Zhu and Alexander Afanasyev. 2013. Let's ChronoSync: Decentralized Dataset State Synchronization in Named Data Networking. In *21st IEEE ICNP*.