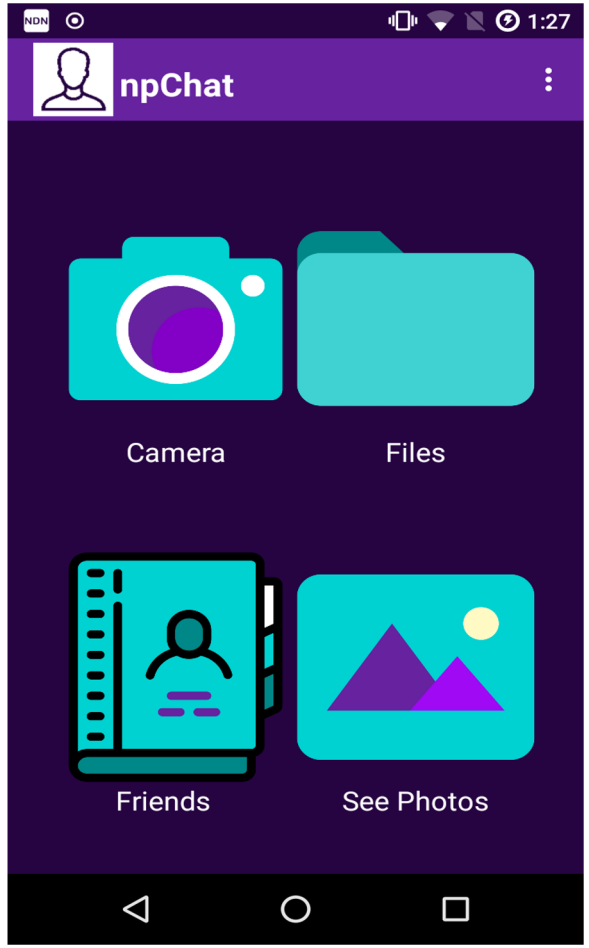
npChat Application Design and Pub-Sub

Lan Wang, Jeremy Clark, Ashlesh Gawande
University of Memphis
ACM ICN 2020 Tutorial

npChat: Decentralized Social Media App [1]

- Share multimedia with friends
- Discover new users to add to their friends list
- Establish trust based on real-life models (meeting in person, organizational, mutual friends)
- Control who has access to their data



[1] A. Gawande, J. Clark, D. Coomes, L. Wang, "Decentralized and Secure Multimedia Sharing Application over Named Data Networking," in Proceedings of ACM Conference on Information-Centric Networking, Sept. 2019

Design Requirements

No central entity

No single user directory

No special infrastructure

No single trust anchor

User control of data

Design Elements

- Name data
- Discover npChat users and become friends
- Publish and fetch data, including certificates, and validate data
- Encrypt and decrypt data
- Learn about new data (PSync's pub-sub API)

Name Space

No single application namespace. Each user uses his/her own namespace.

Alice

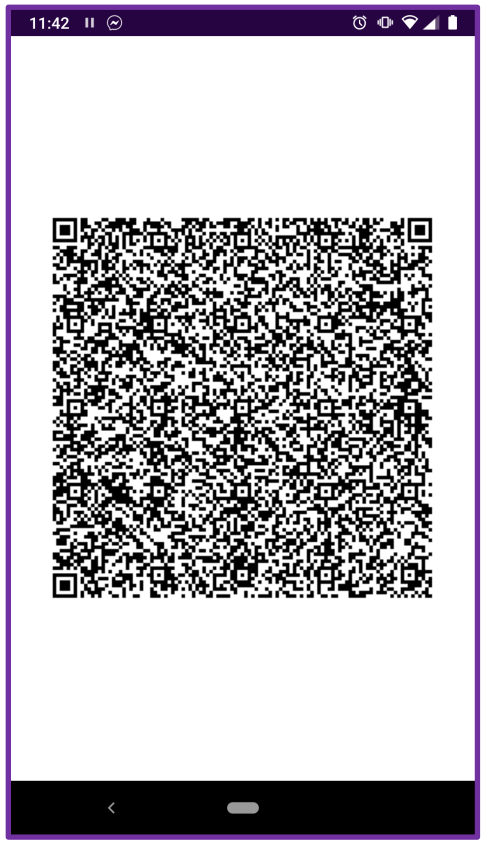
/AliceDoe/npChat/alicedoe123

Bob

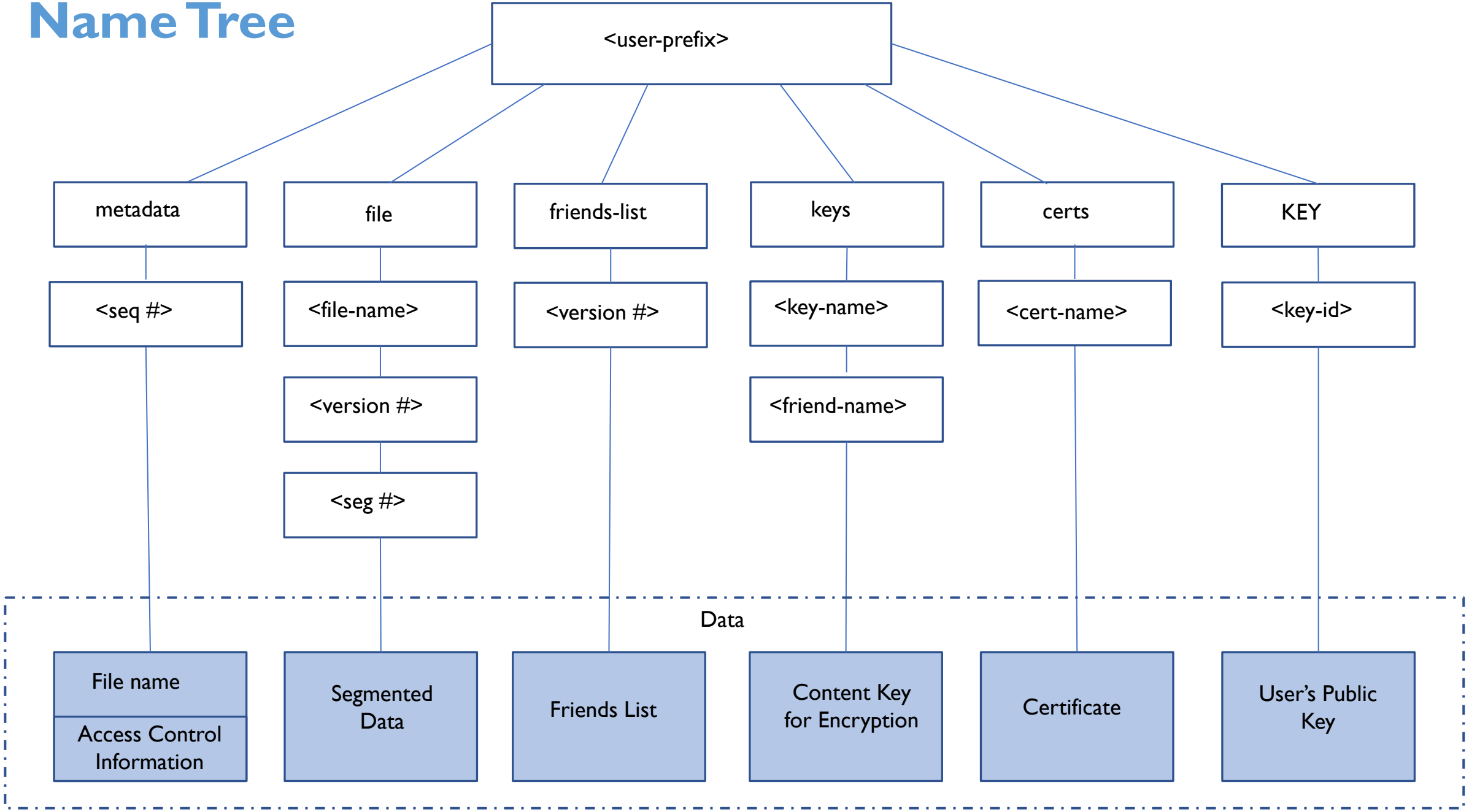
/edu/memphis/BobSmith/npChat/bobsmith321

Public/Private Key as Security Identity

- npChat generates a public/private key pair under user's name space
  - Alice's npChat name space: /AliceDoe/npChat/alicedoe123/
  - Public key name: /AliceDoe/npChat/alicedoe123/KEY/<key-id>
- The public key is self-signed and encoded into a QR code.
- The user can obtain a certificate of his/her public key from
  - a trust anchor associated with the user's name space
  - any friend
- Users obtain each other's certificate to establish trust relationship.



Name Tree


<user-prefix>
metadata
file
friends-list
keys
certs
KEY
<seq #>
<file-name>
<version #>
<key-name>
<cert-name>
<key-id>
<version #>
<seg #>
<friend-name>
Data
File name
Access Control Information
Segmented Data
Friends List
Content Key for Encryption
Certificate
User's Public Key

Trust Model

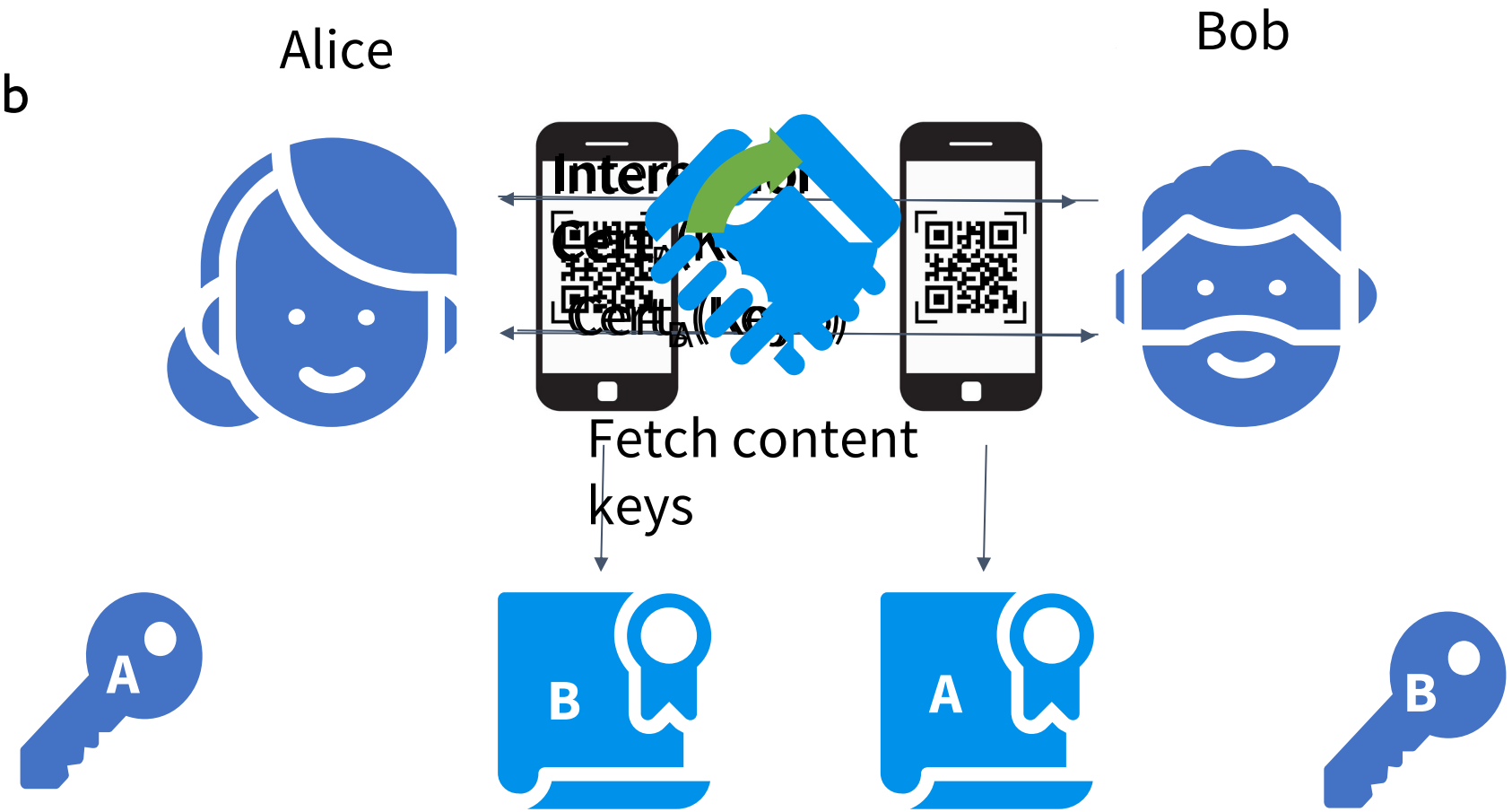
Meeting in Person

Hierarchical/
Same Organization

Mutual Friends

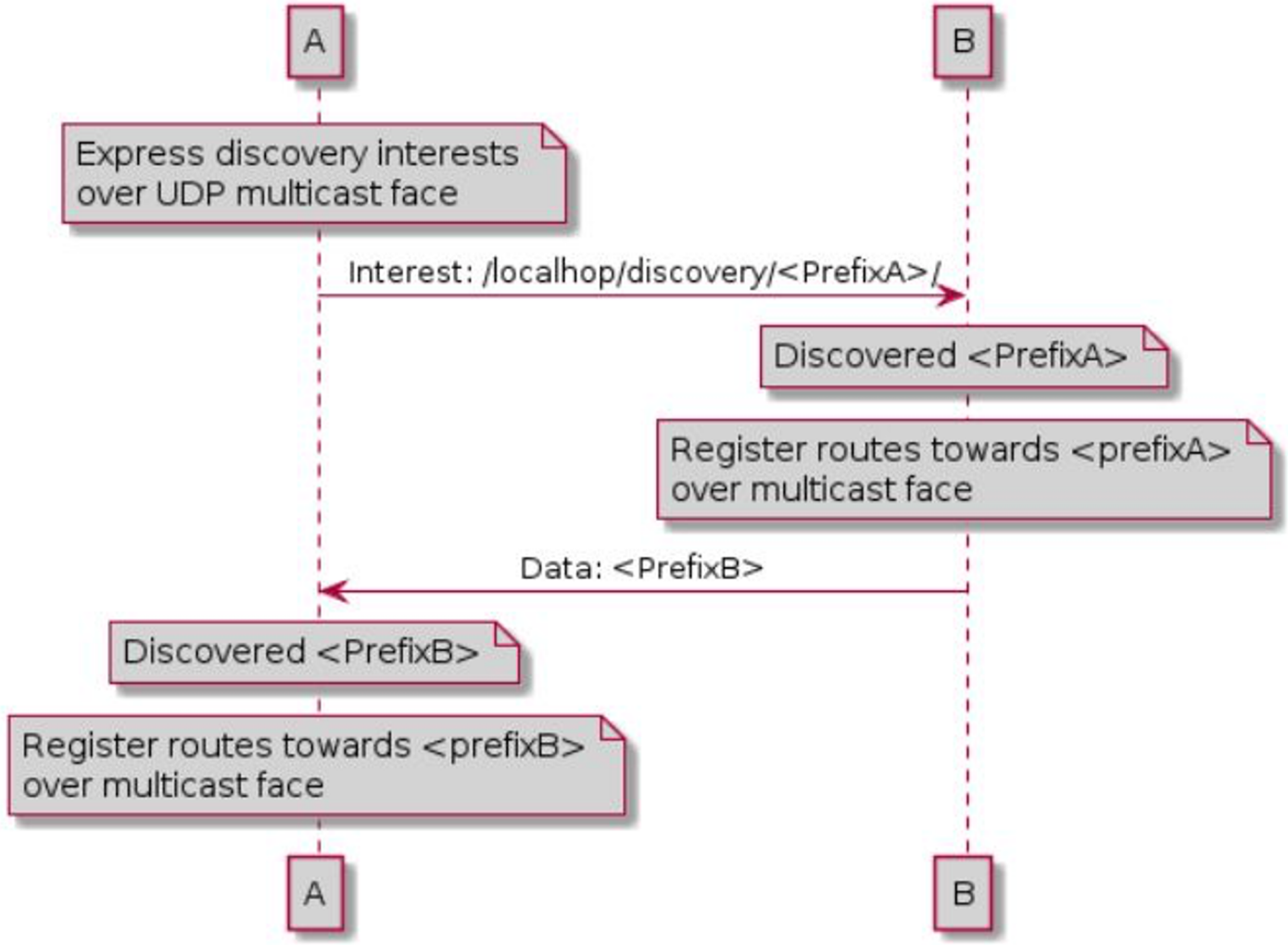
Becoming Friends via Scanning Certificates

Alice and Bob meet in person and want to become friends on npChat.



Discovering Users on Local Network through Multicast

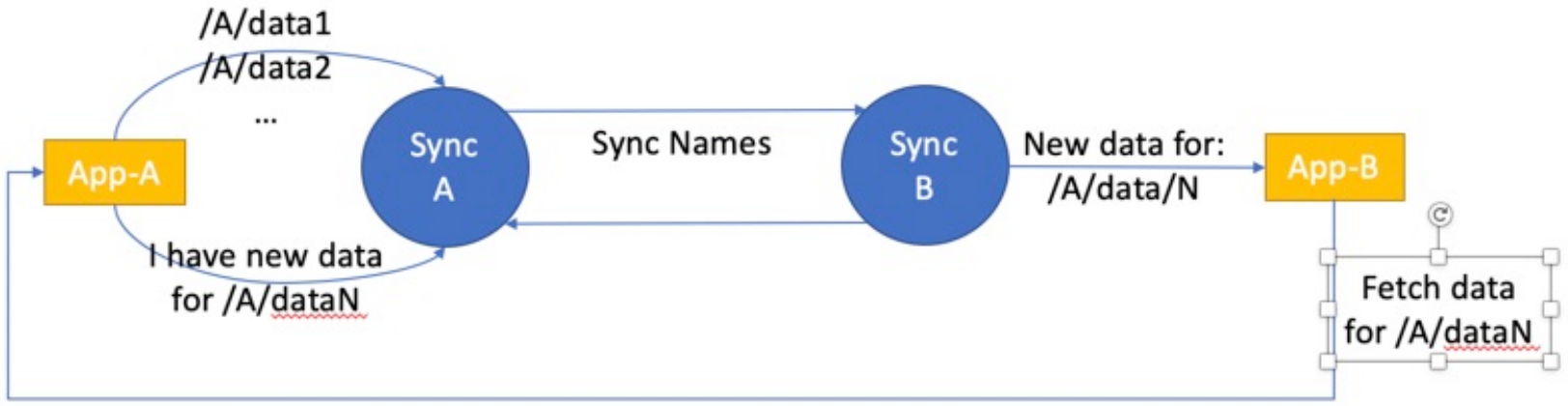
npChat discovers currently active users in the local network via multicast discovery interests.



Publishing and Subscribing to Feeds

npChat uses PSync's pub-sub API [2] to publish and subscribe to friends' feeds.

- Invertible Bloom Filter (IBF) encodes names published by a producer.
- Full Sync API and Partial Sync (Pub-Sub) API



Each npChat instance subscribes to three prefixes for each friend:

- /metadata – information about newly published content
- /friends – user's friends list
- /keys – content keys used to encrypt media intended for that user's friends.

[2] M. Zhang, V. Lehman, L. Wang, "Scalable Name-based Data Synchronization for Named Data Networking," in Proceedings of the IEEE INFOCOM 2017, May 2017

PSync Pub-Sub Consumer

Bob as consumer of Alice's feeds:

```
consumer = new PSync.Consumer(aliceSyncPrefix, helloDataCallback,
          syncDataCallback, bFilCount, falsePositive);
```

On hello data callback, Bob subscribes to Alice's feeds (prefixes) and begins sending sync interest.

```
for (String prefix : prefixes) {
     consumer.addSubscription(prefix);
}
consumer.sendSyncInterest();
```

On sync data callback, consumer handles according to app specification, e.g., fetch metadata, friends list, keys.

PSync Pub-Sub Producer

Alice creates a producer:

```
producer = new PSync.PartialProducer(expectedNumEntries, aliceSyncPrefix,
                                     “/AliceDoe/npChat/alicedoe123”,
                                     helloReplyFreshness, syncReplyFreshness);
```

When Alice wants to share a new photo /AliceDoe/npChat/alicedoe123/file/mycat, she creates a metadata object that contains the new photo’s name and which friends can access the photo.

```
producer.publishName(“/AliceDoe/npChat/alicedoe123/metadata”, 56);
```

This publishes /AliceDoe/npChat/alicedoe123/metadata/56.

Subscribed consumers, e.g., Bob, get notification of the new metadata name, retrieve the metadata, and then retrieve the photo and the corresponding content key.

Publishing a Photo

Alice publishes a photo with **NDN-CNL:**

```
Namespace objectPrefix("/AliceDoe/npChat/alicedoe123/file/mycat", &keyChain);
objectPrefix.setFace (&face, const ptr_lib::shared_ptr<const Name>&prefix);
SegmentedObjectHandler().setObject(objectPrefix, Blob(signed_certificate));
```

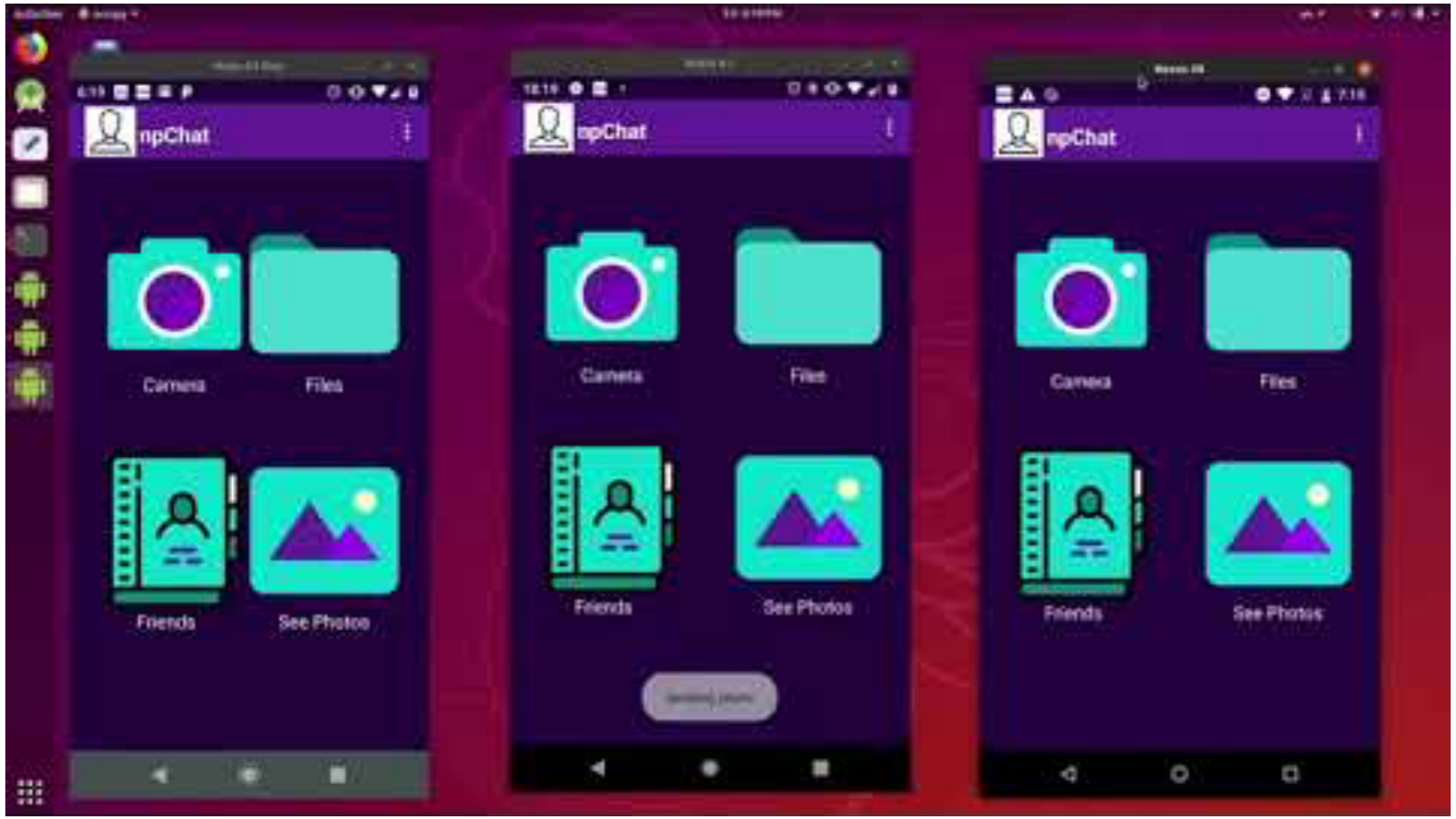
Fetching and Validating Data

Bob fetches Alice's photo with **NDN-CNL**.

```
Namespace objectPrefix("/AliceDoe/npChat/alicedoe123/file/mycat");
objectPrefix.setFace(&face);
auto onObject = [&](const ptr_lib::shared_ptr<ContentMetaInfoObject>&
     contentMetaInfo, Namespace& objectNamespace)
                {
                  //Handle received photo
               };
SegmentedObjectHandler(&objectPrefix, onObject).objectNeeded();
```

Demo

https://www.youtube.com/watch?v=KsfKVGUGDUY



Ashlesh Jeremy Nexus

Thanks!

Questions?

lanwang@memphis.edu

https://github.com/named-data-mobile/ndn-photo-app