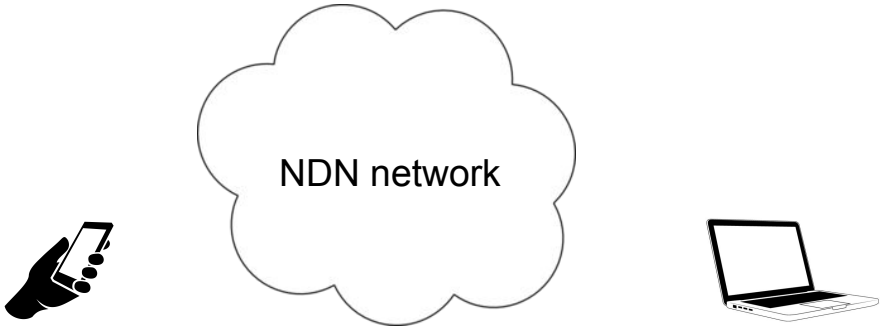
Firefly: A globally scalable message bus with NDN at the edge

Jeff Thompson, Peter Gusev, Haitao Zhang
5th NDN hackathon, 12/17/2017

Needs

- Motivations
  - Sync and repo are still research topics and implementations are in early stages.
  - They cannot provide perfect support for applications.



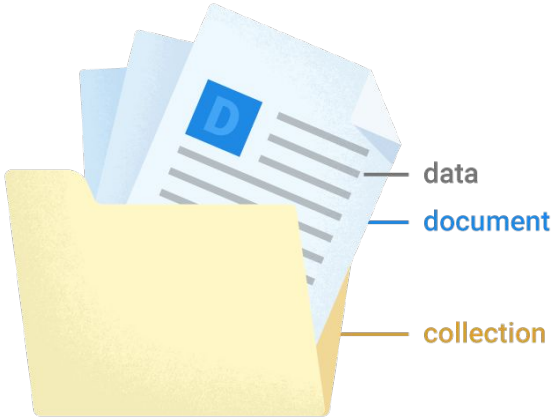
- Goal: Use Google firestore to emulate a salable, robust, disruption-tolerant and real time NDN database

Approaches

- How to store NDN Interest and Data in Firestore
  - Design mapping between Firebase and NDN namespaces and data formats
- How to enable NDN communication between NDN applications and Firestore
  - Develop bridge process for linking a local NDN network with a Firestore
- How to sync data
  - Design a sync mechanism to sync data between Firestore and applications
- Design and implement a sample application

Stored Interest and Data in Firestore

- Firestore: document database
  - Structure: Collection - document - subcollection - document
  - Collections contains a list of documents
  - Documents contains a list of collections and fields
  - Fields are key-value pairs



- Design: struct Interest and Data as a collection-document tree
  - Map each name component to a collection/document
  - Store an Interest or a Data packet as a key-value
  - An example:
    - /ndn/ndnchat/<user>/session-id/seq → /ndn/_/ndnchat/_/<user>/_/session-id/_/seq/_

Communication between Firestore and Applications

- Firestore: push model, clients listen to changes
- Design:
  - Use firestore push model to implement FireflyFace
    - Emulate NDN pull model
    - Emulate NDN Face methods to communicate with Firestore
  - FireflyFace.registerPrefix() implementation
    - Listen to Firestore documents storing Interest, get notified about new Interests
    - When an interest is received, produce the Data packet and call putData
  - FireflyFace.putData() implementation
    - Add the Data packet to Firestore document for the Data name
    - Remove Interests that it satisfies from Firestore
  - FireflyFace.expressInterest implementation
    - Fetch a matching Data packet from Firestore if it already exists
    - Insert it, with expressTime and lifetime, into Firestore document for this Interest
    - Listen to Firestore data documents; if necessary, add child documents

Sync Data

- Basic Design: emulate vector sync
  - Sync namespace: /sync/firefly/fireflychat/<vector>
  - Each member in the chat maintains its own field in the document. That's a vector:
    - /ndn/ndnchat/haitao/session-id: 19
    - /ndn/ndnchat/peter/session-id: 21
    - /ndn/ndnchat/jeffT/session-id: 30
  - All members in the chat will get notified about the update of the document
- Record history in order: versioned vectors
  - Sync namespace: /sync/firefly/fireflychat/<version>/<vector>
  - Each member in the chat maintains a field in the document; however, instead of changing the existing document/vector, create a new document/vector whenever a member needs to update its field.

Benefits: Compared with Alternatives

- Alternatives: current NDN repo and sync protocols
- Benefits:
  - Repo:
    - Scalable: work with global network, could handle large dataset.
    - Robust: Firestore is a business product, which has less bugs and work better
    - Real time
  - Sync:
    - Record history records in order
  - Enable NDN application developers to
    - focus on applications implementation without worrying about network problems.
    - They can switch to real NDN network after application development is done.

Achieved:

- Interest and Data Storage Schema in Firestore
- Library
  - Firefly face: provides the same APIs and functions as the normal NDN face
  - Firefly sync: provides similar APIs and functions as VectorSync
- A sample application: NDNChat

Link & Demo

- Link: https://github.com/5th-ndn-hackathon/firefly
- Demo:
  - Users join and leave chat room
  - Users chat with each other
  - Get history messages