Multi-party Synchronization Part 1

NDN Tutorial – ACM ICN 2015
September 30, 2015

Hila Ben Abraham
Washington University in St. Louis

Goals

Understand the concept of Sync in NDN.

Sync is a way to implement a reliable communication in NDN, but it gives us more than that.

Applications can benefit by using Sync.

Outline

Concept and Motivation:

- What is sync?
- Why to sync?

Solution space:

- The history of Sync in ICN:
  - CCNx Sync
  - iSync
  - ChronoSync

Applications and Sync

Future Work & Open questions.

CONCEPT & MOTIVATION

What is Sync? – The Traditional Definition

The goal of sync is to reconcile the set-difference of a set shared among multiple parties.

S_a = {pic1,pic2,pic3}

S_b={pic1,pic2,pic4}

S_c={pic2,pic3}

Sync

S_a = {pic1,pic2,pic3, pic4}

S_b={pic1,pic2,pic3, ,pic4}

S_c={pic1,pic2,pic3, pic4}

Motivation

Set reconciliation is a common application paradigm used by many of today's applications.

- Dropbox style file sharing
- Real-time chat
- Ad-hoc environments (e.g., vehicular networking)
- Routing protocols.
- Etc.

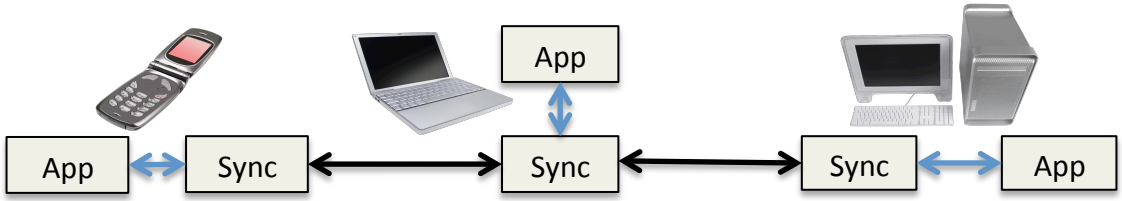
In data centric architecture, sync is more than just a primitive of data synchronization. Sync is a way to implement and maintain a reliable communication.

Motivation (cont.)

In data centric architecture, reliable multi-point communication can be achieved by synchronizing a shared dataset.

- Define application-specific data units.
- Synchronize the namespaces of those data units.

➔ Sync is a way to implement data-oriented communication protocol.

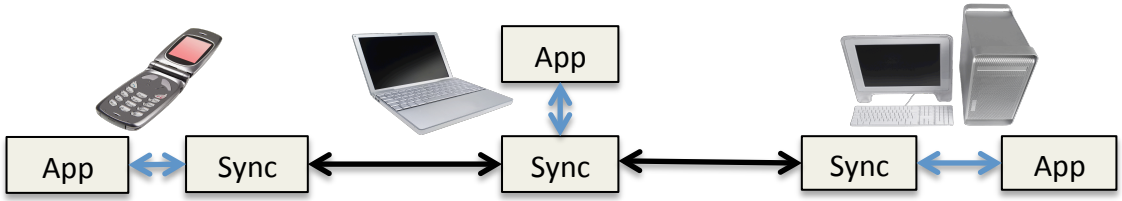


Therefore, we replace the communication problem with a synchronization problem.

Motivation (cont.)

We gain additional benefits by transforming the communication problem into a synchronization problem:

- Multi-point communication.
- A participant doesn't have to stay online all the time.
- No need in mechanism to start or terminate the communication.



SOLUTION SPACE

Breaking Down the Problem

Sync consists of the following tasks:

1. Understanding whether a set is up-to-date or out-of-date.

S_a = {pic1,pic2,pic3}

S_b={pic1,pic2}

S_c={pic2,pic3}

Breaking Down the Problem

Sync consists of the following tasks:

1. Understanding whether a set is up-to-date or out-of-date.
2. Finding the set difference.

S_a = {pic1,pic2,pic3}
S_delta_a = {}

S_b={pic1,pic2}
S_delta_b={pic3}

S_c={pic2,pic3}
S_delta_c={pic1}

Breaking Down the Problem

Sync consists of the following tasks:

1. Understanding whether a set is up-to-date or out-of-date.
2. Finding the set difference.
3. Retrieving the missing items.

S_a = {pic1,pic2,pic3}

S_b={pic1,pic2, pic3}

S_c={pic1,pic2,pic3}

Existing Work

CCNx Sync:

- https://www.ccnx.org/releases/latest/doc/technical/SynchronizationProtocol.html

ChronoSync:

- http://named-data.net/publications/chronosync/

iSync:

- http://named-data.net/publications/synchronizing_namespaces_invertible_bloom_filters/

The three protocols follow a similar high-level communication pattern to satisfy the first task:

- Use a digest to represent the dataset status.
- Each party multicasts periodic interest consists of its dataset state.

But are slightly different in the way they address the second & third tasks:

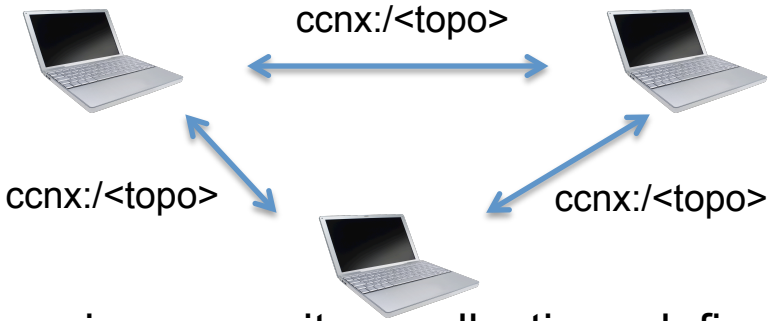
- Their namespace design.
- The data structures used to represent the synchronized dataset.
- Their message flow.

CCNx Sync

The first implementation of sync in ICN.

CCNx Sync was cited by the “Custodian-Based Information Sharing” paper as the supporting protocol of its secured sharing system.

- http://named-data.net/wp-content/uploads/06231277.pdf



The protocol synchronizes repository collections defined by the user.

- A collection is defined by routable prefix, and a name.
- All the participant nodes are required to define the same collection.

CCNx Sync Limitations

CCNx Sync operates only on content that was previously added to the CCNx repository.

It doesn't provide API to applications.

It retrieves all the collection's content as part of its set-reconciliation protocol.

It doesn't scale well with the number of changes.

- Requires multiple interest-data exchanges to reconcile large set-difference.

iSync

iSync addresses the last limitation of CCNx Sync by following the data structure suggested in the paper: “What's the difference?: efficient set reconciliation without prior context”

- http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.220.6282&rep=rep1&type=pdf

iSync optimizes the set-reconciliation part of CCNx Sync by utilizing a 2-level IBF structure to represent the synchronized datasets.

- IBF is an extension of Bloom filter. This not only keeps a record of data, but allows you to add, delete and make a list of the data you have stored.
- The set-difference of two IBFs can be obtained by XORing their corresponding IBFs

iSync scales up with the number of updates by exchanging a single IBF.

ChronoSync

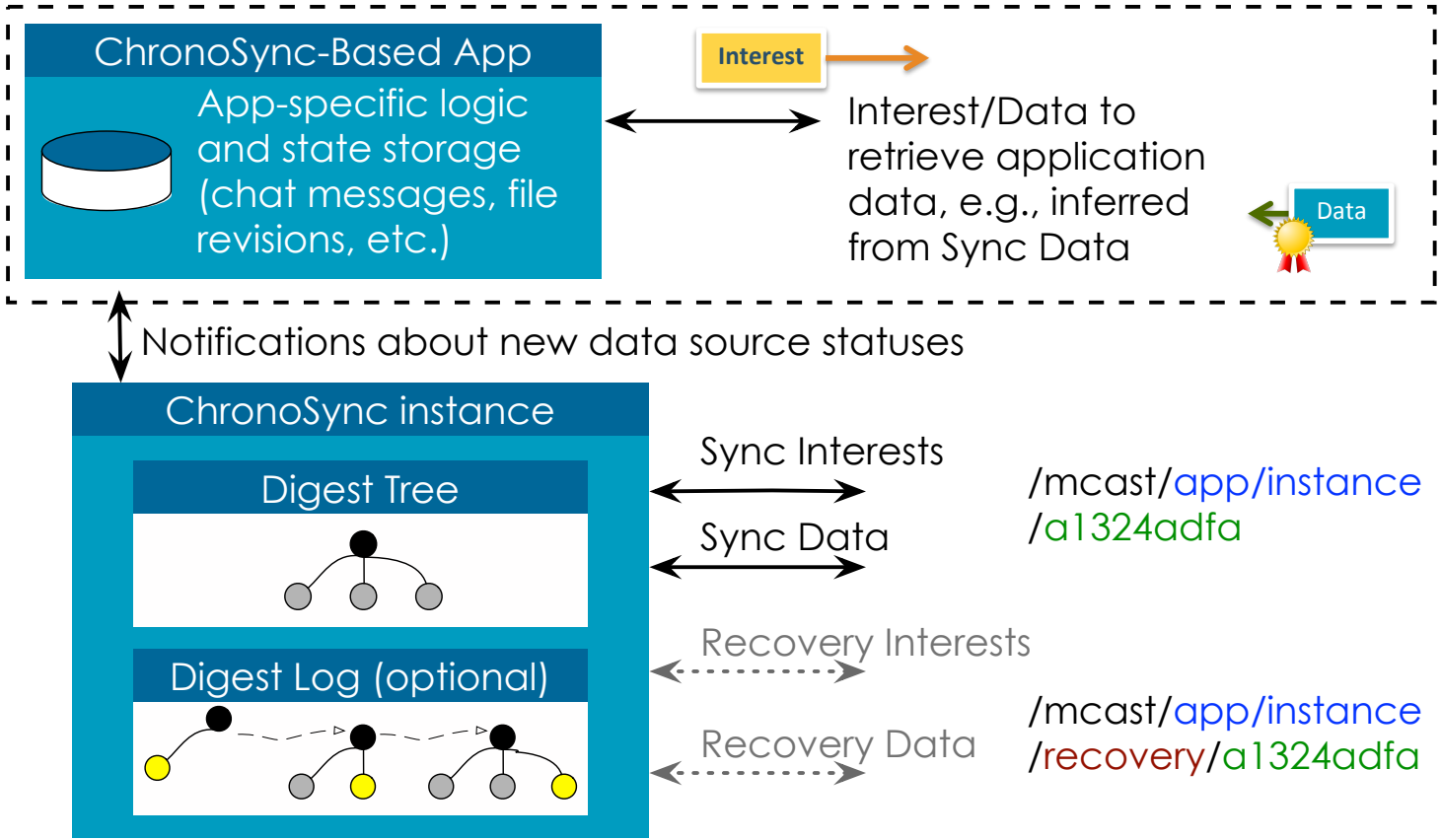
ChronoSync addresses CCNx Sync limitations by making the following improvements:

- Provides Sync as a service by bringing the knowledge about the dataset up-to-date, and leaving the application to decide whether to fetch it.
- Uses 2 specialized namespaces:
  - Sync data name to represent the dataset status:

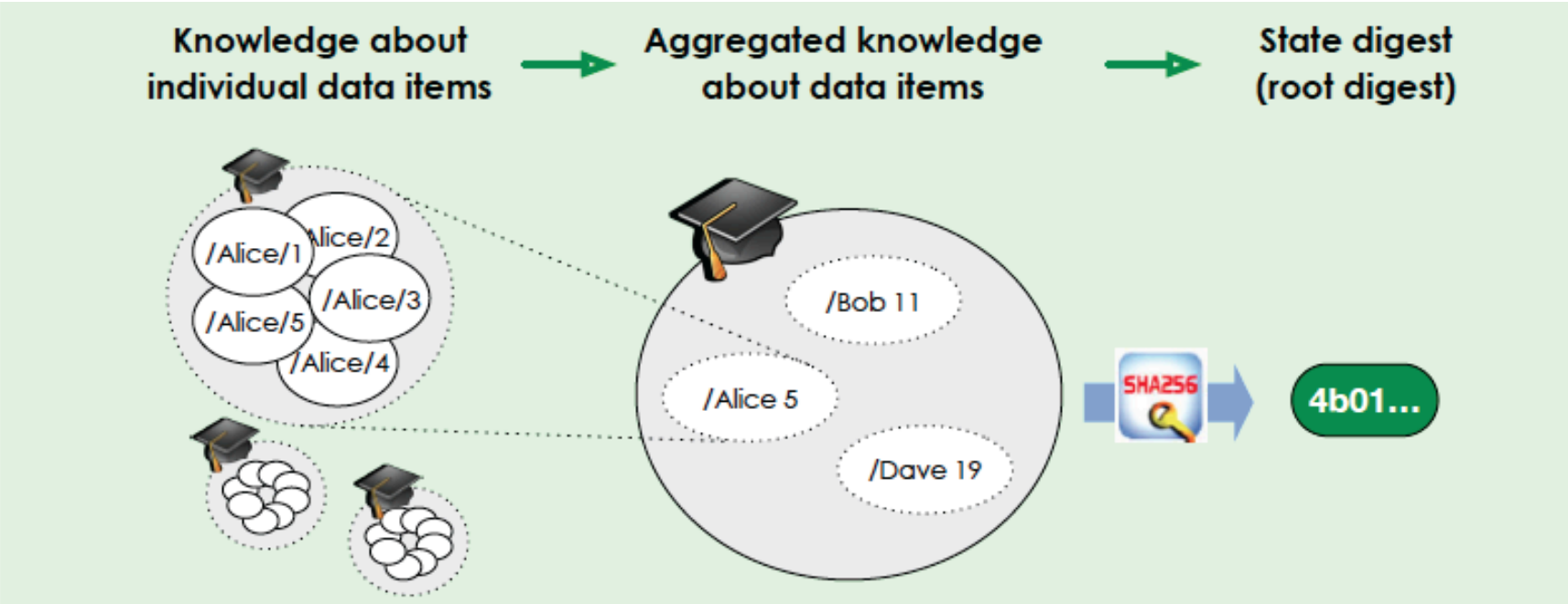
    Multicast_prefix/ApplicationName/digest
  - Application data name:

    Participant_prefix/AppName/msg_seq
- Changing the semantic of the multicast interest to fetch changes instead of notify them.
- Provides an API for applications built on top of NDN codebase.

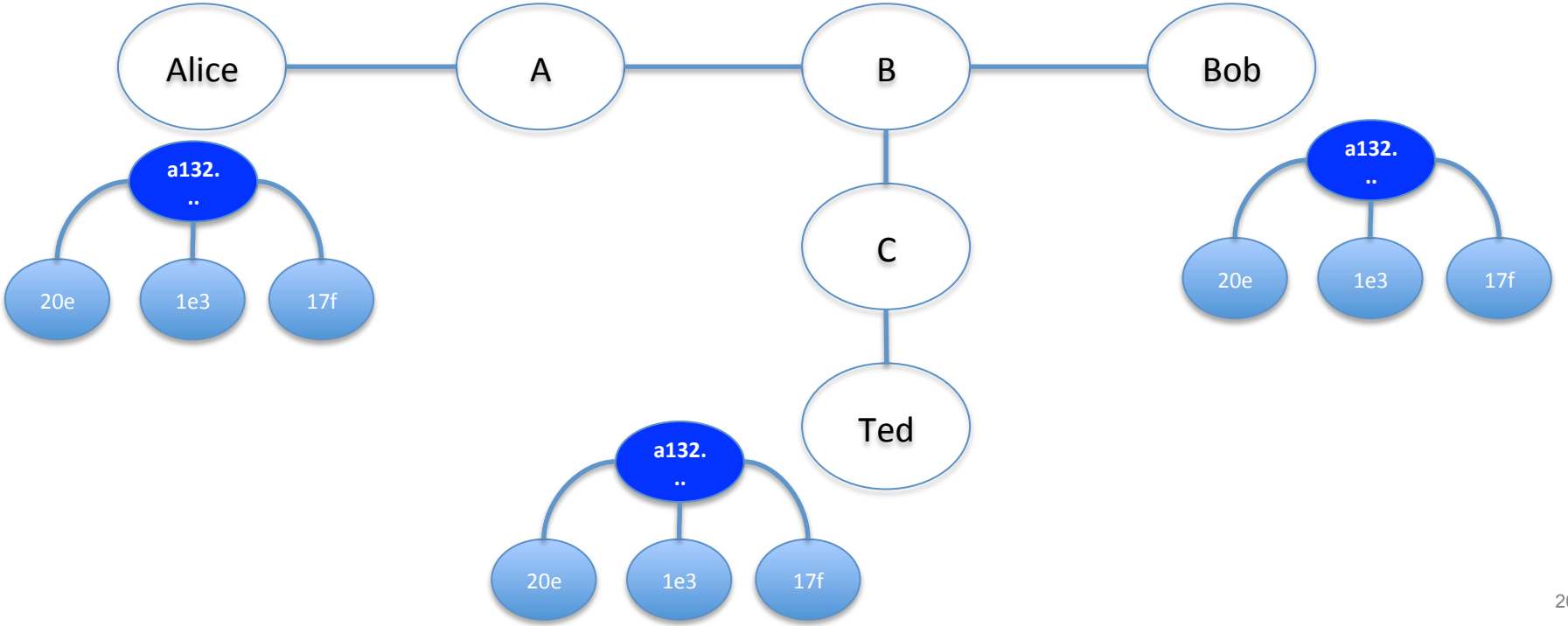
ChronoSync-Based App Design Overview



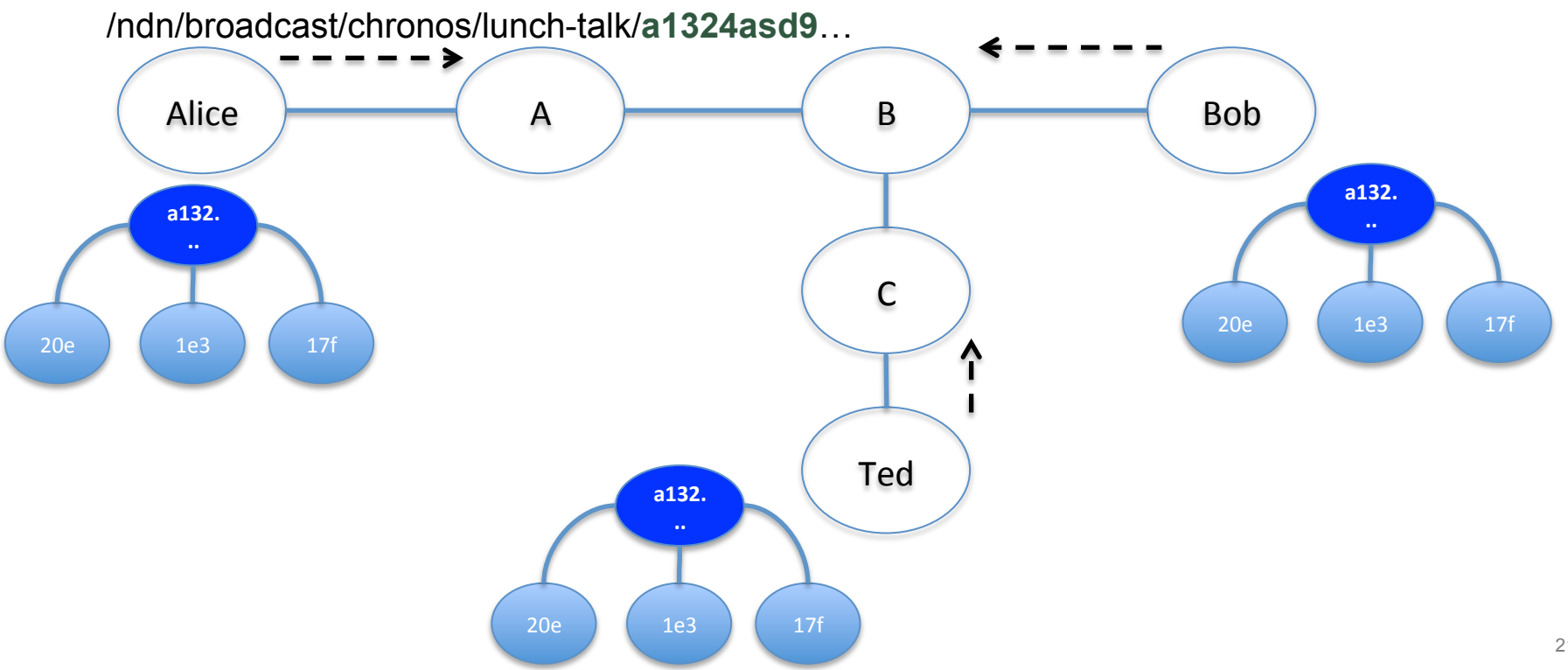
State of a Dataset



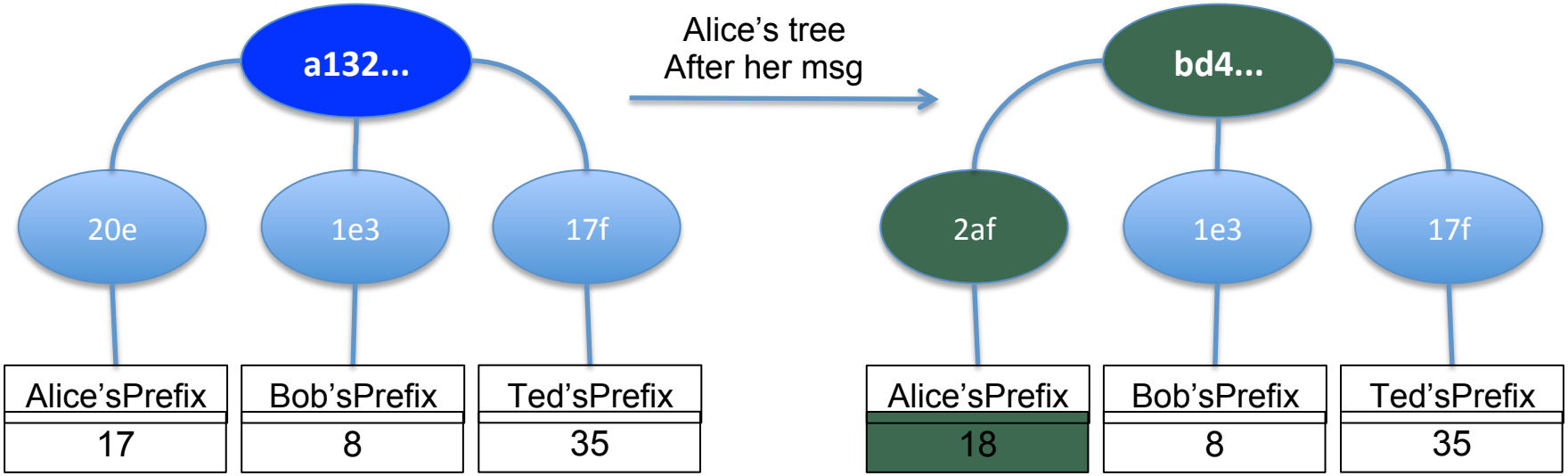
How does it work – a chat room example



How does it work – a chat room example



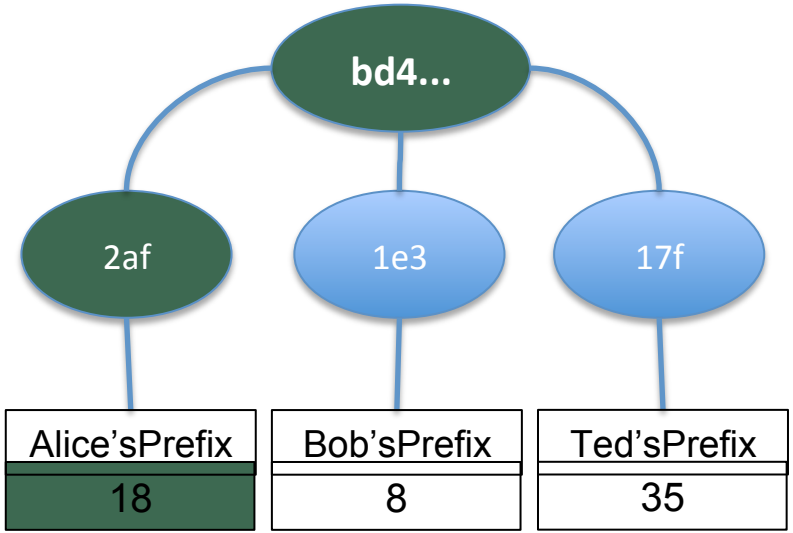
How does it work – a chat room example



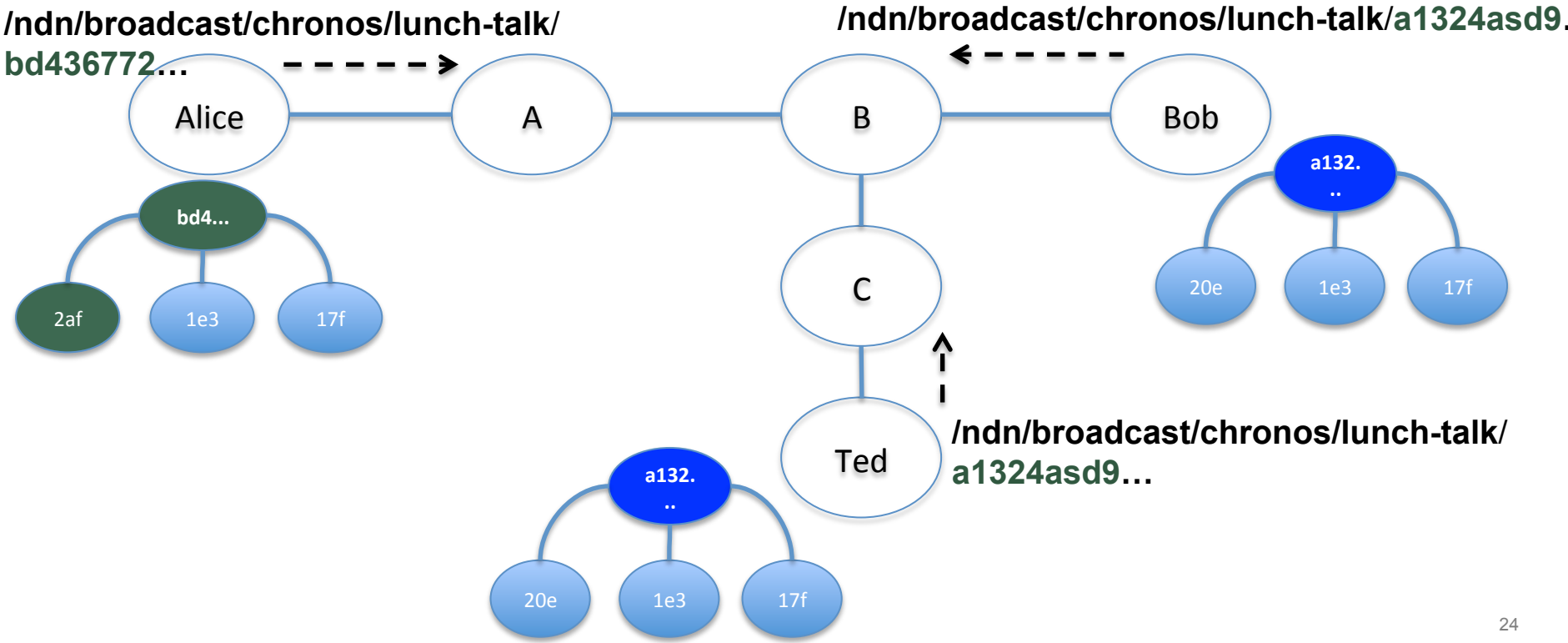
How does it work – a chat room example

Digest Log

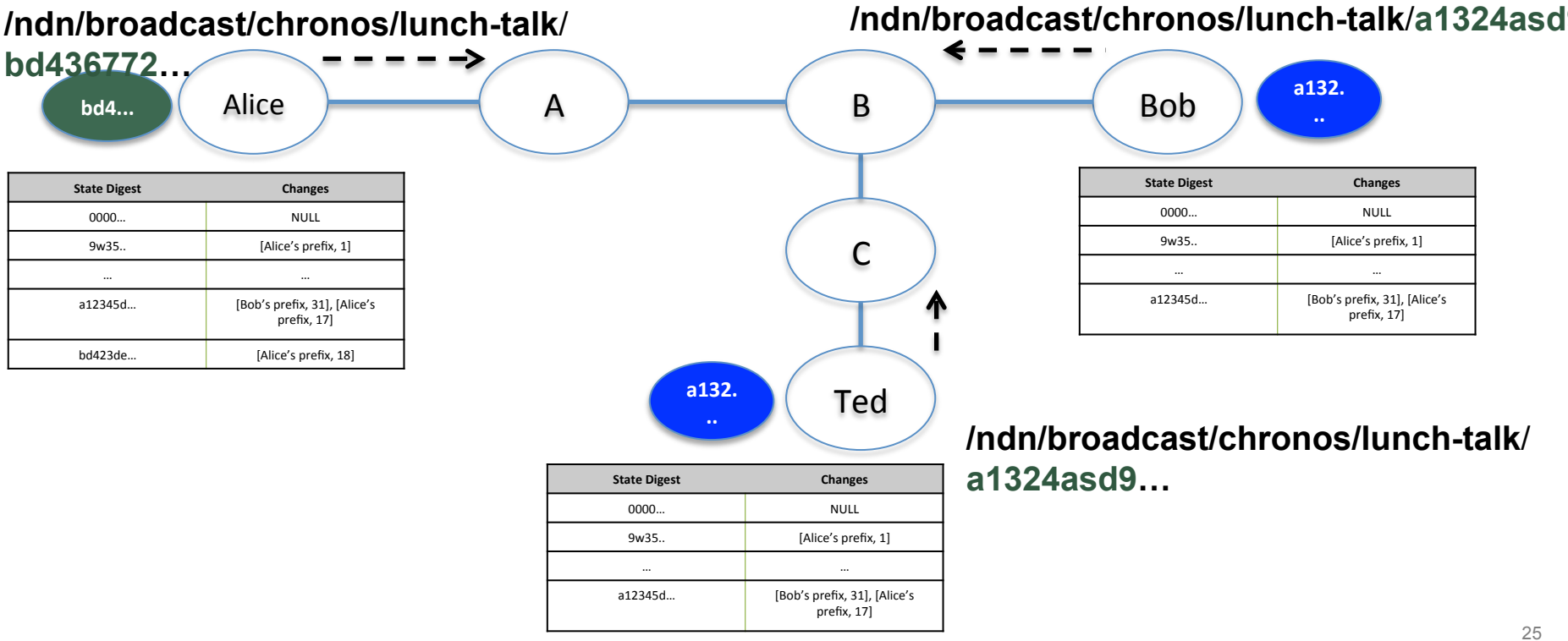
State Digest	Changes
0000…	NULL
9w35..	[Alice's prefix, 1]
…	…
a12345d…	[Bob's prefix, 31], [Alice's prefix, 17]
bd423de…	**[Alice's prefix, 18]**



How does it work – a chat room example



How does it work – a chat room example

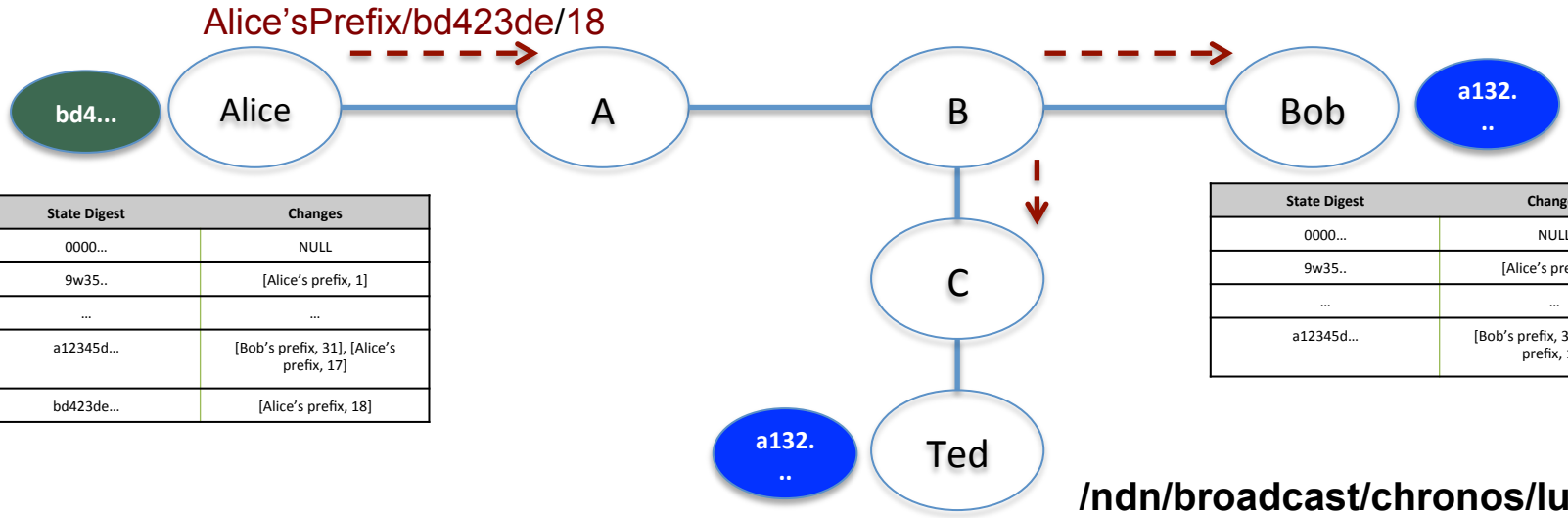


State Digest	Changes
0000…	NULL
9w35..	[Alice's prefix, 1]
…	…
a12345d…	[Bob's prefix, 31], [Alice's prefix, 17]
bd423de…	[Alice's prefix, 18]

State Digest	Changes
0000…	NULL
9w35..	[Alice's prefix, 1]
…	…
a12345d…	[Bob's prefix, 31], [Alice's prefix, 17]

State Digest	Changes
0000…	NULL
9w35..	[Alice's prefix, 1]
…	…
a12345d…	[Bob's prefix, 31], [Alice's prefix, 17]

How does it work – a chat room example



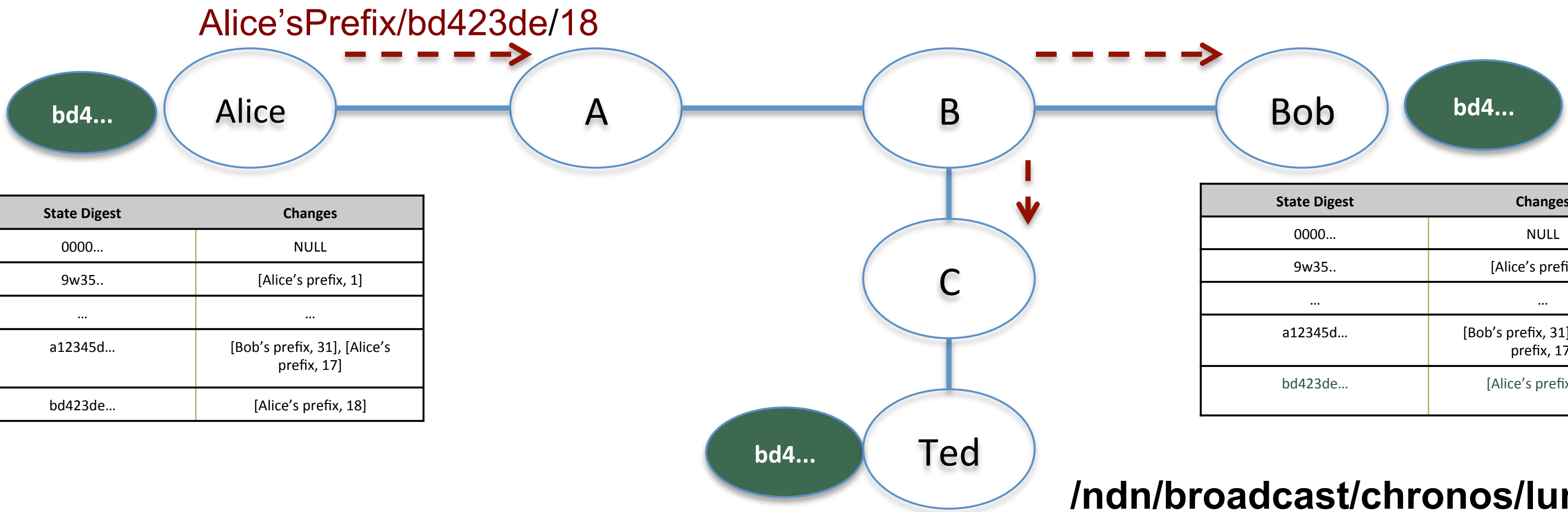
State Digest	Changes
0000…	NULL
9w35..	[Alice's prefix, 1]
…	…
a12345d…	[Bob's prefix, 31], [Alice's prefix, 17]
bd423de…	[Alice's prefix, 18]

State Digest	Changes
0000…	NULL
9w35..	[Alice's prefix, 1]
…	…
a12345d…	[Bob's prefix, 31], [Alice's prefix, 17]

State Digest	Changes
0000…	NULL
9w35..	[Alice's prefix, 1]
…	…
a12345d…	[Bob's prefix, 31], [Alice's prefix, 17]

/ndn/broadcast/chronos/lunch-talk/
a1324asd9…

How does it work – a chat room example



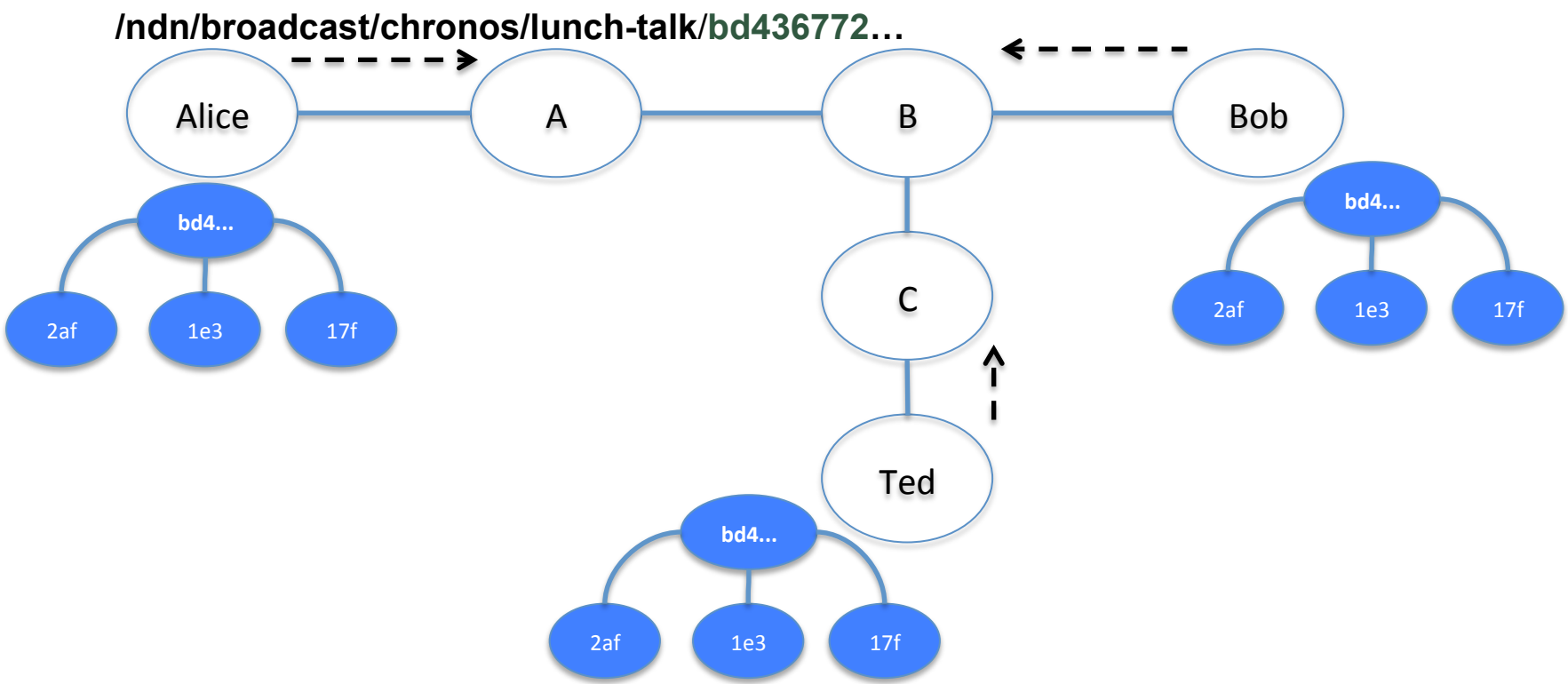
State Digest	Changes
0000…	NULL
9w35..	[Alice's prefix, 1]
…	…
a12345d…	[Bob's prefix, 31], [Alice's prefix, 17]
bd423de…	[Alice's prefix, 18]

State Digest	Changes
0000…	NULL
9w35..	[Alice's prefix, 1]
…	…
a12345d…	[Bob's prefix, 31], [Alice's prefix, 17]
bd423de…	[Alice's prefix, 18]

State Digest	Changes
0000…	NULL
9w35..	[Alice's prefix, 1]
…	…
a12345d…	[Bob's prefix, 31], [Alice's prefix, 17]
bd423de…	[Alice's prefix, 18]

/ndn/broadcast/chronos/lunch-talk/ a1324asd9…

How does it work – a chat room example



Revisit the Sync Problem

ChronoSync solves the three tasks:

1. Understanding whether a set is up-to-date or out-of-date.
   - By exchanging the dataset digest.
2. Finding the set difference.
   - By looking up the received digest in the digest log.
3. Retrieving the missing items.
   - By sending the set-difference found in the digest log in the response data packet.

Existing Work Summary

We discussed the three exiting sync implementations:

- CCNx Sync and iSync are implemented in the CCNx codebase.
- iSync is mostly focused on the second task, efficient set-reconciliation using Invertible Bloom filter.
- ChronoSync is currently the only Sync implementation in the NDN codebase, and it provides synchronization as a service for NDN applications.

SYNC & APPLICATIONS

Apps & Sync (Cont.)

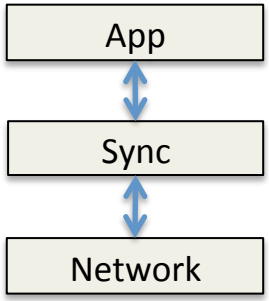
Sync implements a reliable communication and introduces new benefits to NDN applications:

- Simplified interest-data exchange flow.
- Recovery from a link failure.
- No need to establish persistent connection - a participate can join or leave.
- Can speed up the application development time, and simplify the code complexity.

Apps & Sync

Sync operates in between an application and the network, it keeps the application knowledge up-to-date, but leaves the application to decide whether to fetch the content.

Therefore, sync can be viewed as a service that provides a layer of abstraction in the NDN stack.



Can applications evolve by using sync?

Design Questions

Define data units for sync.

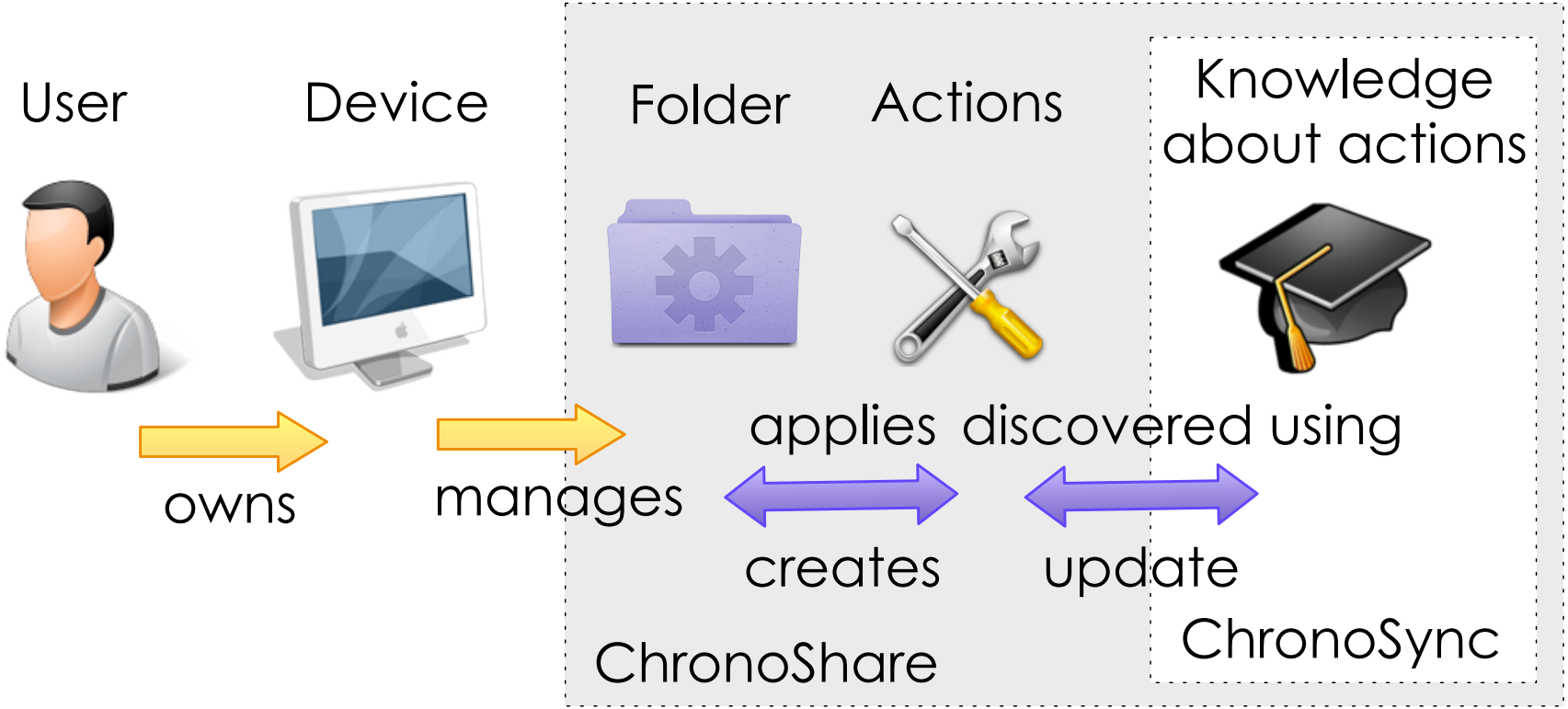
- Choose the application namespace?
  - A critical question to any NDN application.

How to respond to to sync notifications?

- Fetch the data once it's ready?
- Wait for subsequence data to arrive?

What's the relationship between the user, app and sync?

ChronoShare: Peer-to-Peer File Sharing



Existing Applications & Sync

- Current applications that directly use ChronoSync are:
  - ChronoShare (http://named-data.net/publications/story_of_chronoshare/)
    - Dropbox style application.
    - ChronoSync Synchronizes the actions made in a directory.
    - The app decides whether and when to fetch the content.
  - NLSR (http://named-data.net/wp-content/uploads/2013/07/nlsr-final.pdf)
    - The NDN routing protocol.
    - ChronoSync synchronizes the 'Link State Advertisement' (LSA) information.
    - The app decides on the preferred link.
  - ChronoChat (https://github.com/named-data/ChronoChat)
    - A chat application.
    - ChonoSync synchronizes the messages of a chat room.
    - The applications decides when and if to present the messages.

Existing Applications & Sync (cont.)

tutorial app

ndn-atmos (https://github.com/named-data/ndn-atmos)

ndncon (https://github.com/remap/ndncon)

ndnfit (http://redmine.named-data.net/projects/ndnfit)

NDN Android game apps:
https://github.com/dchimeraan/ndn-hangman, https://github.com/ohnonoho/photoSharing, https://github.com/sumitgouthaman/NDNWhiteboard

chronochat-js (https://github.com/named-data/ChronoChat-js)

FUTURE WORK

Open Questions & Future work

How to deliver sync interest to all the participants?

Security: How to protect the protocol from “bogus” sync interests.

Better support frequent and simultaneous updates.

- In-progress work done by ChronoSync team attempts to better support simultaneous updates by separating new data discovery from existing data synchronization by using additional state.

Optimizations:

- Reconcile the differences using minimal number of interest-data exchanges.
- Optimize the recovery process, or whenever the set-difference is too big.

Takeaway Points

We are interested in Sync because it's a way to implement a reliable communication.

We get additional benefits by implementing sync as a data-oriented communication protocol.

We can break down the sync problem into three tasks: 1) Understanding whether a set is up-to-date or out-of-date. 2) finding the set-difference. 3) retrieving the missing items.

ChronoSync implements sync on top of the current NDN codebase, and it can be used by applications as a sync service.

There are interesting open questions and research opportunities!

Thank you!

Questions?