Think Different
Exploring New Directions in Application Designs
a case study
How apps work today

• Client-server paradigm
• Servers: center of the world
  – Holding up all data
    • I upload my reading list to dropbox as a way to pass it from my laptop to my iPad
  – Linking up users with each other
    • Facebook
      – Even when you and your friends all in this room
Here is a simple application

• Multi-User text Chat (MUC)
Today’s implementation

• TCP pipes from each user to the centralized server
• The server mirrors back data to all users
Really the best we can do?

- Single point of failure?
- Traffic concentration?
  - e.g. send pictures to each other now and then
Really the best we can do?

• Single point of failure?
• Traffic concentration?
• Where may be the server?
• Who provide it?
How about a serverless MUC?

- NDN: retrieve data by name
- MUC: each user retrieve the data from all others in the same chatroom
  - Users can communicate directly within the vicinity
Building a Distributed MUC (D-MUC)

• First: get around the “non-networking” side of the work: borrowing from existing app
Building a working D-MUC

[Diagram showing network setup with nodes labeled Alice, Bob, and Ted, connected through LSD and NDN protocols]
D-MUC over NDN

• A simple idea: Alice sends Interest packets to Bob/Ted to get their data (and vice versa)
  – Not everyone typing all the time
  – seems lots of Interests going around

• A better idea
  – Each user assigns a seq# to each piece of his data
  – Alice just needs to know whether Bob/Ted’s seq# has changed
    • Then decides whether/what to retrieve
D-MUC: a simple design

- Data naming:
  
  /ndn/ucla.edu/alice/d-muc/westwood@ucla/335

- Compute a digest for the chatroom

- Ask around: do you have the same digest?
  
  /ndn/broadcast/d-muc/westwood@ucla/digest
The Basic Idea

- Synchronizing everyone’s view on the latest seq# of each user in the same room
  - If someone notices missing data, go fetch it
- How: each user broadcasts a SYNC Interest
  /ndn/broadcast/d-muc/westwood@ucla/digest
  - When Alice generates new data: reply with data packet (her new seq#)
  - This data packet satisfies everyone else’s SYNC Interest
    - They may then fetch the new data; Alice could also piggyback the data in her response
  - Everyone sends out a new SYNC Interest
    - the name reflects updated digest value
First User to A Room

• $T_0$: User Alice creates an empty digest tree
• $T_1$: Alice sends SYNC Interest with empty digest
  - Gets no reply: she is the only user in the room
• $T_2$: Alice adds her presence data to the room data set
  - Create a child node under the root for herself
  - Update the root digest
• $T_3$: Alice sends a SYNC Interest with the new digest
Keep Transaction Log

• Maintain the history of digest tree changes, together with the causes
Second User Shows Up

- $T_4$: Bob creates an empty digest tree
- $T_5$: Bob sends out SYNC Interest with “empty” digest, and receives SYNC Interest from Alice
  - Bob’s Interest reaching Alice: the digest differs, Alice knows why
- $T_6$: Alice replies with her seq# (her state)
- $T_7$: Bob receives data $\Rightarrow$ gets the same digest value
- $T_8$: Bob adds his presence add to the room data set
  - Create a node in the digest tree for himself
  - Update the root digest
  - Reply Alice’s Interest with his presence data name
- $T_9$: Alice updates her digest tree according to the data from Bob, update her digest, send new SYNC Interest
- $T_{10}$: Now Alice and Bob have the same view of room data set
Handle SYNC Interest: Normal Case

```
```

- Steady State’’
Handle SYNC Interest: Normal Case

• "Transient State"

PIT: /SYNC-prefix/digest_101

NDN

Alice

PIT: /SYNC-prefix/digest_101

PIT: /SYNC-prefix/digest_101

Bob

digest_101

Ted

digest_101
Handle SYNC Interest: Normal Case

• "Transient State"

Alice

DATA: /SYNC-prefix/digest_101

PIT: /SYNC-prefix/digest_101

Bob

digest_101

Ted

digest_101

NDN

PIT: /SYNC-prefix/digest_101

PIT: /SYNC-prefix/digest_101

3/19/12
Handle SYNC Interest: Normal Case

• "Steady State"

Alice

Bob

Ted

NDN

PIT: /SYNC-prefix/digest_102

PIT: /SYNC-prefix/digest_102

digest_102

digest_102

digest_102
Handle SYNC Interest: Timed Wait

- In the above example, Ted may have received the new SYNC Interest before Alice’s reply reaches him
  - Ted should wait before react to the new SYNC Interest with digest 102
  - After Alice’s reply arrives, Ted’s root digest also becomes 102
Exchange Digests

• Use a SYNC Interest to carry to digest
  – SYNC Interests reach every participant

• Each participant keeps one outstanding SYNC Interest, i.e. send a SYNC Interest only when
  – SYNC Interest gets satisfied
  – SYNC Interest expires
Handle SYNC Interest: Use Log

- When receiving an Interest with the digest value different from one’s own root digest
  - First check if such digest exists in the transaction log
  - If so, reply with all participant states that have changed since that digest value
  - If not, timed wait
State Reconciliation

• There are cases that above steps do not work
  – Participants do not recognize each other’s signatures
    • E.g. network partition

• Solution
  – Reply Interest with one’s knowledge about the current states of all participants
Exchange States

• Reply the SYNC Interest with their knowledge of ```states'``'

• After receiving such state packet, compare and update the digest tree if the state learned are more recent

<table>
<thead>
<tr>
<th>/SYNC-name-prefix/Digest/States</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice's Name Prefix</td>
<td>195</td>
</tr>
<tr>
<td>Bob's Name Prefix</td>
<td>37</td>
</tr>
<tr>
<td>Ted's Name Prefix</td>
<td>320</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Signing Signature</td>
<td></td>
</tr>
</tbody>
</table>
DISCUSSIONS
A new building block for distributed applications: SYNC

• MUC is a simple distributed app
  – But representative for a class of distributed apps

• All distributed applications are built on top of client-server model

• THINK DIFFERENT: is that really true?
  – SYNC is a simple way to inform all players of the overall system state
  – Each player makes its own action decisions
Getting the SYNC interest to everyone

• “need broadcast, but you can’t do it in any non-trivial scale”

• THINK DIFFERENT: is that really true?
  – We used to think that all files have to be served out of some centralized powerful server boxes
  – Now we know that P2P can do much better
“Data Push Apps”

• “NDN only supports data pulling, but lots apps need data push”

• MUC could be viewed as a “push-app”
  – Each user pushes his/her new text to everyone else
    • How do you know they get them all?
      – Reliable multicast?
        » ACK implosion?

• THINK DIFFERENT: MUC – really a much simpler problem of letting everyone request whatever data as needed
  – If we just let them know what’s out there
What to take away

• More exciting time is yet to come
• Apply NDN approach to problems that TCP/IP has a hard time to fit-in
  – Some new exciting front: HHDs, IoTs
• THINK DIFFERENT

D-MUC source: git://github.com/bcy/muc.git
(an older version; SYNC-based implementation is coming)