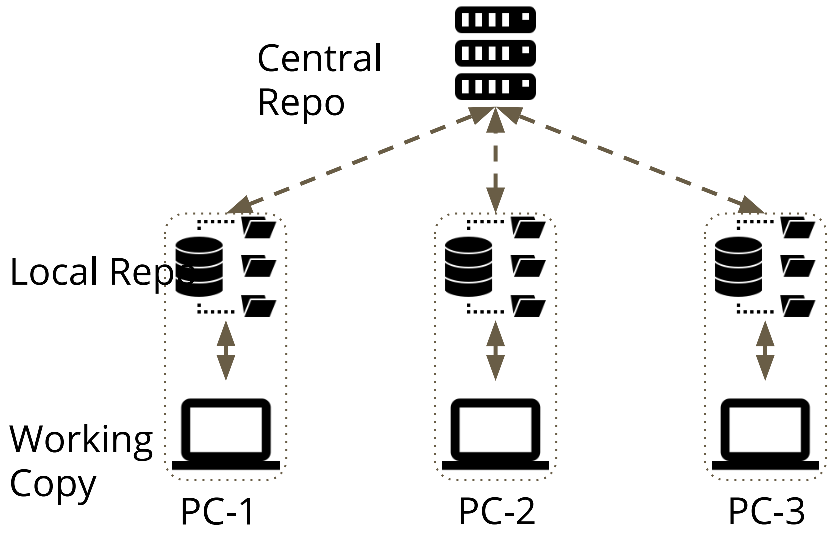
GitSync

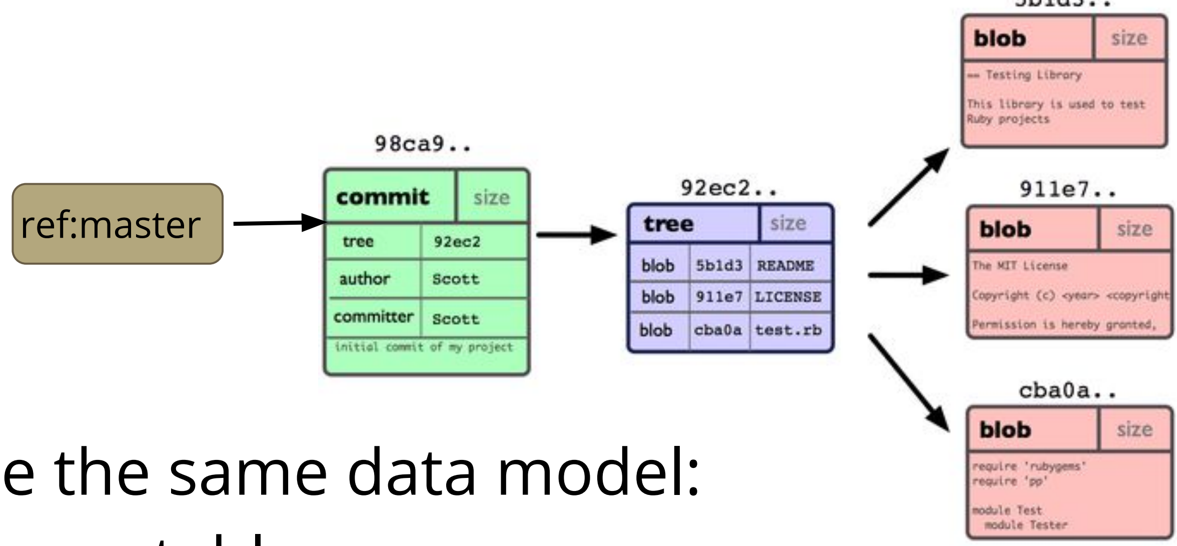
NDN Community Meeting
September 5, 2019

How people use Git today



- Git is a **decentralized** protocol
  - Everyone saves the entire history of changes
  - No special node that acts as server
- Today, Git is deployed in a **centralized** manner
  - The central server is always online and reachable
  - Security is in the cloud

Decentralizing Git: running over NDN



- Git's and NDN share the same data model:
  - Git objects are immutable
    - Only *ref pointers* (e.g. "master") are mutable
  - Git objects are identified by name (SHA1)
- Git over NDN can remove the need for centralized server
  - Support asynchronous communication
  - Localized trust
  - Make good use of available edge connectivity

Problems to solve to decentralize Git

- Synchronization among multiple peers
- Conflict resolution to keep consistency
- System security
  - Authentication
  - Authorization

Design: synchronization between peers

Make the system survive network partition:

- Need individual nodes act as peers
- Multiple peers need to synchronize

Solution:

- Sync peers with per-repo State Vector Sync
  - Sync <*branch_ref : timestamp*> pairs
    - e.g. `{<master:t1>, <stable:t2>, <dev:t3>}`
  - When timestamp updated, ask for new ref value, perform tree walk, and fetch objects

Design: consistency with per-branch custodian

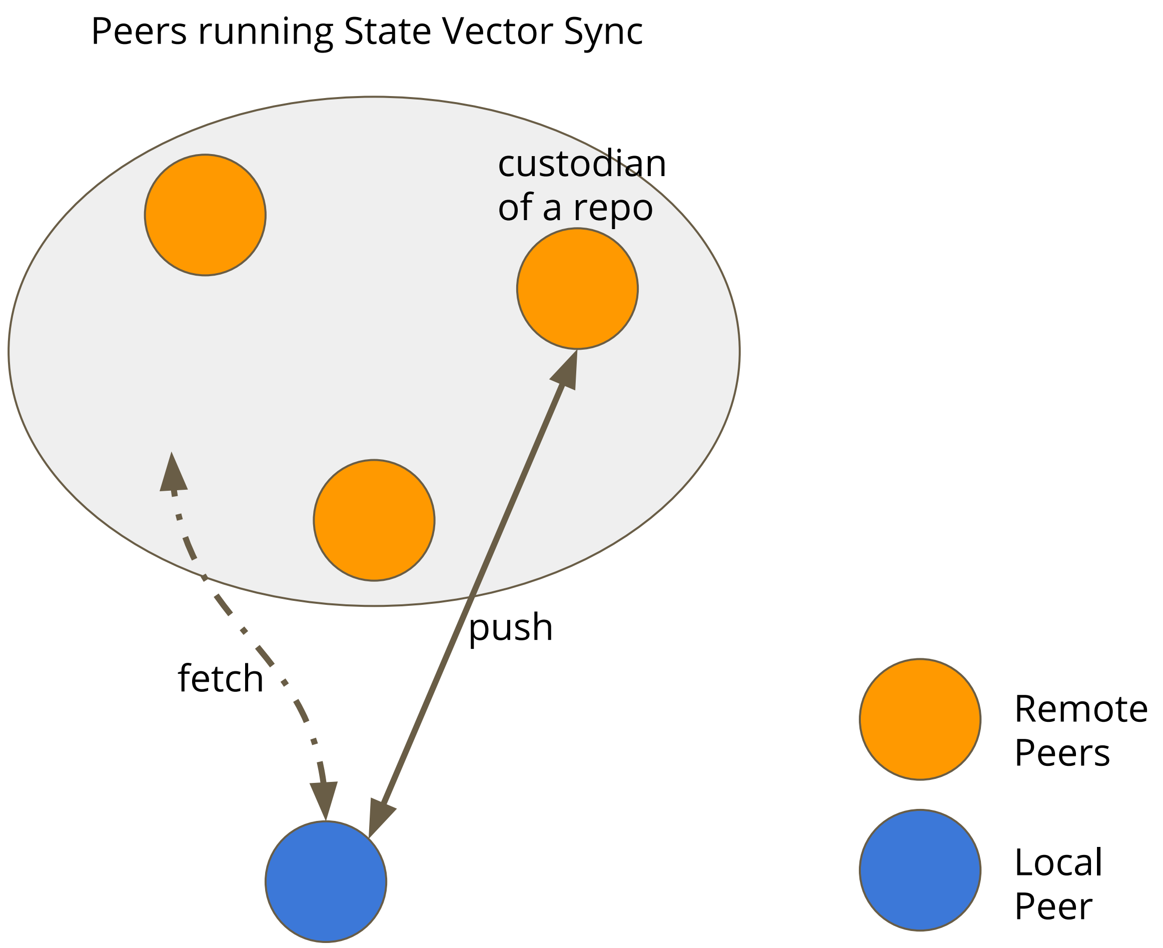
Store a copy on each peer:

- Simultaneously push to different peers -> Inconsistency
- How to do fine-grained branch management

Solution:

- Assign each branch a **custodian**
  - Push must go through the custodian to obtain a signature
    - (Objects can be fetched from anywhere)
  - When some peers can’t be reached, can still push to the remaining peers

Design: consistency with per-branch custodian



Design: security

Authentication

- Objects named by SHA-1, can’t be malformed
- Ref for each branch need to be authenticated
    - Schematized trust can be used

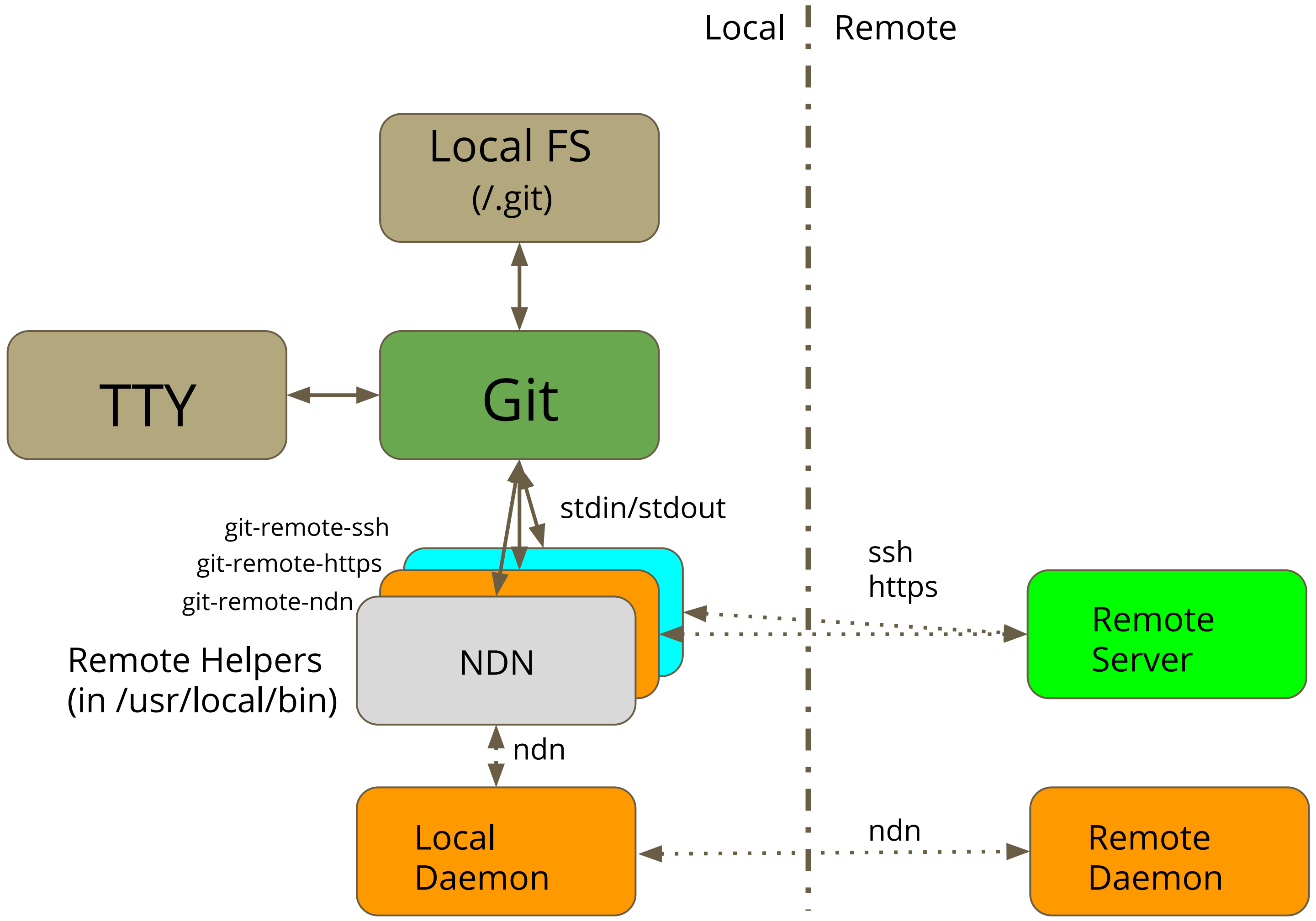
$$\begin{tikzcd}
\text{/lab/KEY} \arrow[d, bend right, "\text{signs}"'] \\
\text{/lab/zhaoning/KEY} \arrow[d, bend right, "\text{signs}"'] \\
\text{/git/repo\_foo/refs/master/1567550827}
\end{tikzcd}$$

Design: security

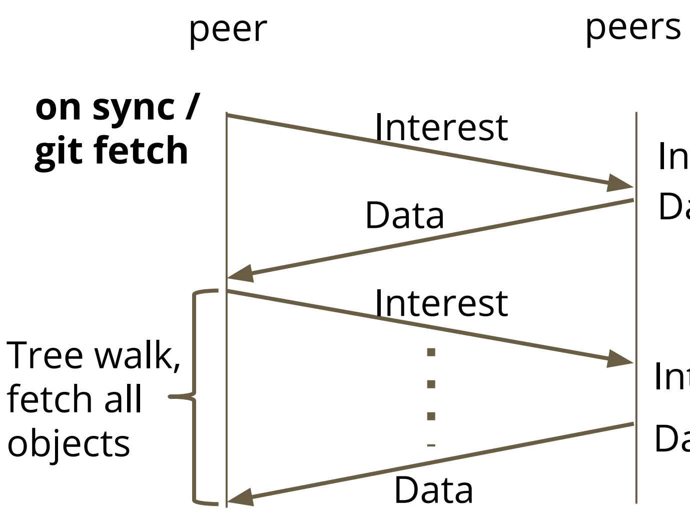
Authorization: who can push to a branch?

- Policy should be determined locally by the custodian, e.g.
  - Each developer has his/her own branch, which he/she can push to
  - Only senior developers can push to the master branch

Implementation: daemon & NDN remote helper



Implementation: fetch over NDN



1. Ask for latest ref

Interests: `/git/<repo>/refs/<branch>/<timestamp>`

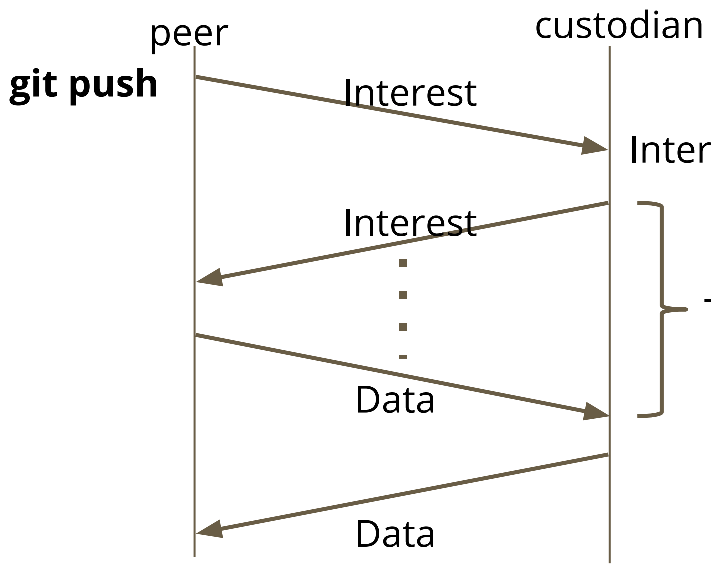
Data: ref

2. Fetch objects

Interests: `/git/<repo>/objects/<hash>`

Data: objects

Implementation: push over NDN



1. Request for push

Interest: `/<custodian_name>/push/<repo>/<branch>/<param_hash>`

2. Fetch objects

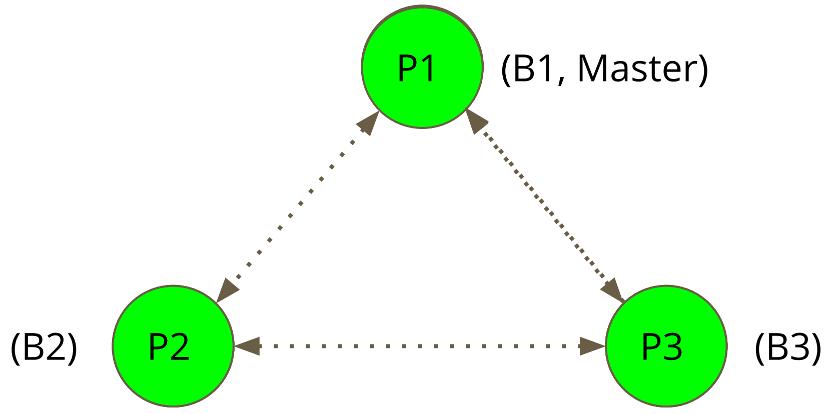
Tree walk, fetch all objects

3. Ack push

- Success
- Fail
- Pending

Example

Branch	B1	B2	B3	M
Ref *	1	2	~~3~~ 5	~~3~~ 5



B1	B2	B3	M
1	2	~~3~~ 5	~~3~~ 5

B1	B2	B3	M
1	2	~~3~~ 5	~~3~~ 5

Step 1: All peers in sync

Step 2: P3 push to branch B3
(e.g. "git push gitsync B3")

Step 3: P1, P2 sync with P3 on branch B3, and fetch missing objects

Step 4: P3 decide to push the changes to master branch M. It sends an interest for push to custodian P1
(e.g. "git push gitsync master")

Step 5: P2, P3 sync with P1 on branch master. There's no need to fetch objects here, because they're already fetched at step 3.

* ref's values are SHA1 hashes. Here they're represented as integers for simplicity

Conclusion

Decentralized deployment of Git over NDN

- No single point of failure
- Survive unstable connections
- Partition

Run State Vector Sync across peers

- Sync ref pointers, then traverse graph, fetch objects

Per-branch custodian to control push access

- No simultaneous push to different peers -> consistency
- Fine-grained branch management

Current code status & plan

Code status:

- Available on Github
- Implemented with PyNDN
- https://github.com/JonnyKong/GitSync

Next step:

- Auto build, CI & CD
- Access control
- Distributed Code Review

Thank You

Xinyu Ma xinyu.ma@cs.ucla.edu

Zhaoning Kong jonnykong@cs.ucla.edu