PythonRepo: Persistent In-Network Storage for Named Data Networking

Tianyuan Yu*, Zhaoning Kong†, Xinyu Ma*, Lan Wang‡, Lixia Zhang*

*University of California, Los Angeles
†Purdue University
‡The University of Memphis
Web Objects Today

- Web: fetching URI/URL identified objects
- Today’s realization: running over secure connections between clients and cloud servers
  - Constrained by TCP/IP’s point-to-point packet delivery by IP addresses

![ TLS Channel Diagram ]

datatracker.ietf.org/meeting/118/agenda?

datatracker.ietf.org/meeting/118/agenda

Cloudflare
Named Data Networking and Data-Centric applications

- **Named Data Networking**: fetch data by app-named data
  - Producers *publish* semantically named, secured data objects
    - *Sync*, NDN’s transport protocol, informs users in the same app of latest data production
  - Consumers *request* desired data by data names
  - Network *retrieves* data requests by name

- **Building data-centric apps over NDN**: greatly simplifies systems, offers more functionalities

![Diagram of data transmission](datatracker.ietf.org/meeting/118/agenda)
A New Building Block for Data-Centric Applications

- **Secure Web Objects (SWO)**[^1]
  - URI-like namespace to identify each object
  - Encrypting and signing each object
- **Users meet at rendezvous place to exchange SWO by URI-like identifiers**
- **Enabling decentralized applications**
  - May use, but not depend on, cloud resources

An Example SWO App: Workspace

- A decentralized, collaborative web app: [https://ndn-workspace.web.app/](https://ndn-workspace.web.app/)
  - Users identified by names
  - Mutual authentication among peers
  - Users exchanging SWO publications over NDN connectivity
  - Supporting asynchronous collaboration with users coming and going offline
    - Newly connected users fetch latest SWO stored in other users browsers

/MeetAgenda/alice@example.com/seq=2
Content: 0x4a53f…..
Signed by
/MeetAgenda/alice@example.com/KEY

/MeetAgenda/aws-jane23/seq=6

Web of Trust Authentication

/MeetAgenda/github-bob23/seq=3
An Example SWO App: Workspace

- A decentralized, collaborative web app: [https://ndn-workspace.web.app/](https://ndn-workspace.web.app/)
  - Users identified by names
  - Mutual authentication among peers
  - Users exchanging SWO publications over NDN connectivity
  - Supporting asynchronous collaboration with users coming and going offline
    - Newly connected users fetch latest SWO stored in other users browsers
Enabling Asynchronous SWO Exchanges in All Situations

- If any users are online at any time: a newly connected user gets latest updates
- To assure SWO availability at all times: need *in-network storage* to keep all SWO available at all times
  - In-network storage can be provided by the network service provider[^2]

[^2]: “Mobile Data Repositories at the Edge”, 2018 USENIX Workshop on Hot Topics in Edge Computing
PythonRepo: In-Network Data Storage

- Managed storage for semantically named and secured data chunks
- Design goals
  - Basic operations: insertion, deletion
  - User authentication and authorization
    - Accepting insertion or deletion request from authenticated and authorized users

/att/alice  

Authorized Data Insertion and Deletion

/att/repo
Inserting Data into Repo

- Telling the network who is inserting what data to which repo
- Repo fetching the data to be served

“Inserting /meetings/alice@example.com/seq=2 to /att/repo”

Command signed by /att/alice/KEY

Telling the network who is inserting what data to which repo

Fetching data by name
Systematic Security Policy Definition by Semantic names

- Repo authenticates command by verifying Alice key
- Policy enforcement: AT&T users can sign ATT repo insertion commands

Naming convention enabled schematized security policies based on names
Data Deletion

- As simple as different keyword
- Policy enforcement: AT&T users can sign ATT repo deletion commands

"Deleting /meetings/alice@example.com/seq=2 from /att/repo"

Command signed by /att/alice/KEY
Putting Pieces Together: Data Production and Storage

- Alice joins the AT&T network and workspace “MeetAgenda”
- Alice produces SWO when nobody is online, and sends an AT&T repo insertion command
  - Signed by her AT&T key
- Repo verifies the command, fetching SWO by name
- Before Alice goes offline

```
“Inserting /MeetAgenda/alice@example.com/seq=2 to /att/repo”
```

```
/MeetAgenda/alice@example.com/seq=2
Content: 0x4a53f.....
Signed by /MeetAgenda/alice@example.com/KEY
```

```
/att/repo
```
Putting Pieces Together: Asynchronous Communication

- Bob comes online, fetches Alice SWO by name
- Bob validates Alice SWO and decrypts content
  - Using Alice’s workspace key

In-network storage is generic to all apps and data types
End-to-end user data security independent from connectivity

"Inserting /MeetAgenda/alice@example.com/seq=2 to /att/repo"
PythonRepo Implementation

- Python Implementation: [https://github.com/UCLA-IRL/ndn-python-repo](https://github.com/UCLA-IRL/ndn-python-repo)
- Experimental deployment over NDN Testbed
  - A global NDN network, overlay on IP
  - Supporting decentralized applications with asynchronous communication
PythonRepo Implementation

- Used in multiple projects
  - Hydra: a federated file storage\textsuperscript{[3]}
  - mGuard: a named based access control system for mobile health\textsuperscript{[4]}
  - Workspace: ongoing

\textsuperscript{[3]} "Hydra: A Scalable Decentralized P2P Storage Federation for Large Scientific Datasets", 2024 IEEE ICNC
\textsuperscript{[4]} "Building a Secure mHealth Data Sharing Infrastructure over NDN", 2022 ACM ICN
PythonRepo vs. Cloud Storage

PythonRepo
- **Generality**
  - Works for all NDN applications as a network service
  - Maintained by the ISP
- **Agnostic to application security**
  - Data are signed by application producers.
  - Data are encrypted by users.
  - Repo is not trusted as part of the application.

Cloud Storages (like Google CloudStorage)
- **Application-specific**
  - Per application per instance (like Gmail)
- **Security on the Cloud, controlled by the Cloud**
  - Users authenticated by cloud
  - Controlled by the cloud
Takeaways

- **PythonRepo** leverages NDN’s basic design block of named, secured data
  - Command as named data, securing command with name-based security policies
  - Future work: implement support for distributed repos
    - Using NDN Sync to keep all repo instances synchronized
- **Repo: user-controlled In-network storage**
  - Storage as part of network service to support data-centric, asynchronous apps\[^5\]
    - The cost can be covered in similar way to network connectivity charges
  - Data authenticity and access control are controlled by end users
- **In-network** storage enables the development of decentralized apps
  - Workspace as an illustrative example

\[^5\] "The Role of Data Repositories in Named Data Networking", 2019 IEEE ICC RAFNET Workshop