DECADE Strawman Proposal

Richard Alimi, Hongqiang Liu,
David Zhang (PPLive), Roger Zhang
(China Telecom), Y. Richard Yang

Laboratory of Networked Systems

Yale University

Relation to DECADE

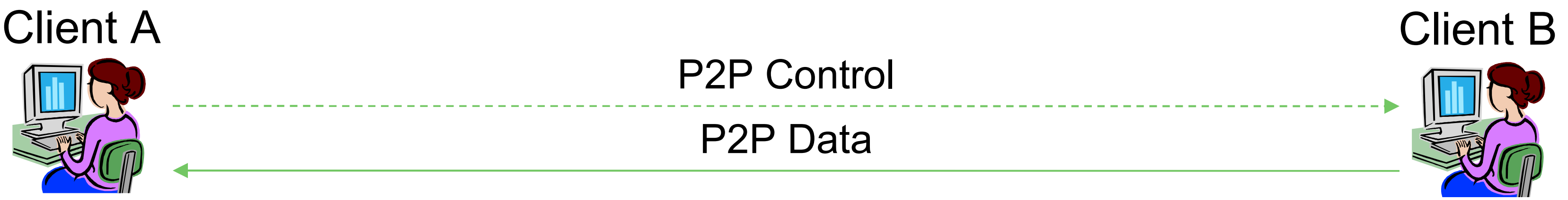
- Research project at Yale Laboratory of Networked Systems
- Just one possible solution architecture for the DECADE problem statement

Overall Operational Model

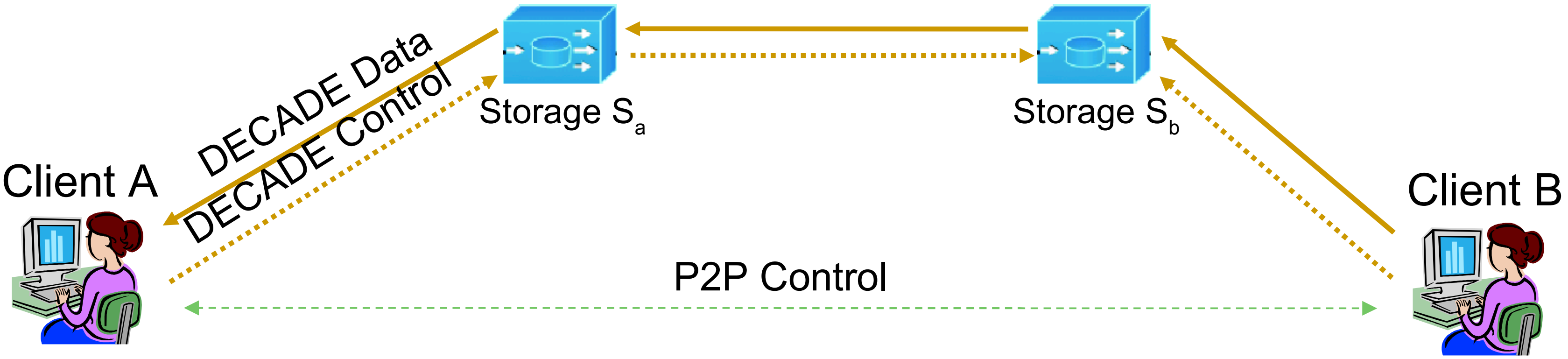
- *Service Provider* provides multiple *storage servers*
- Data locker server hosts multiple *storage accounts*
- *User* gets storage account(s) on storage servers
  - User may be an end user or a content publisher
- Users' *P2P applications* retrieve/store objects (chunks) using storage servers

Example Operation

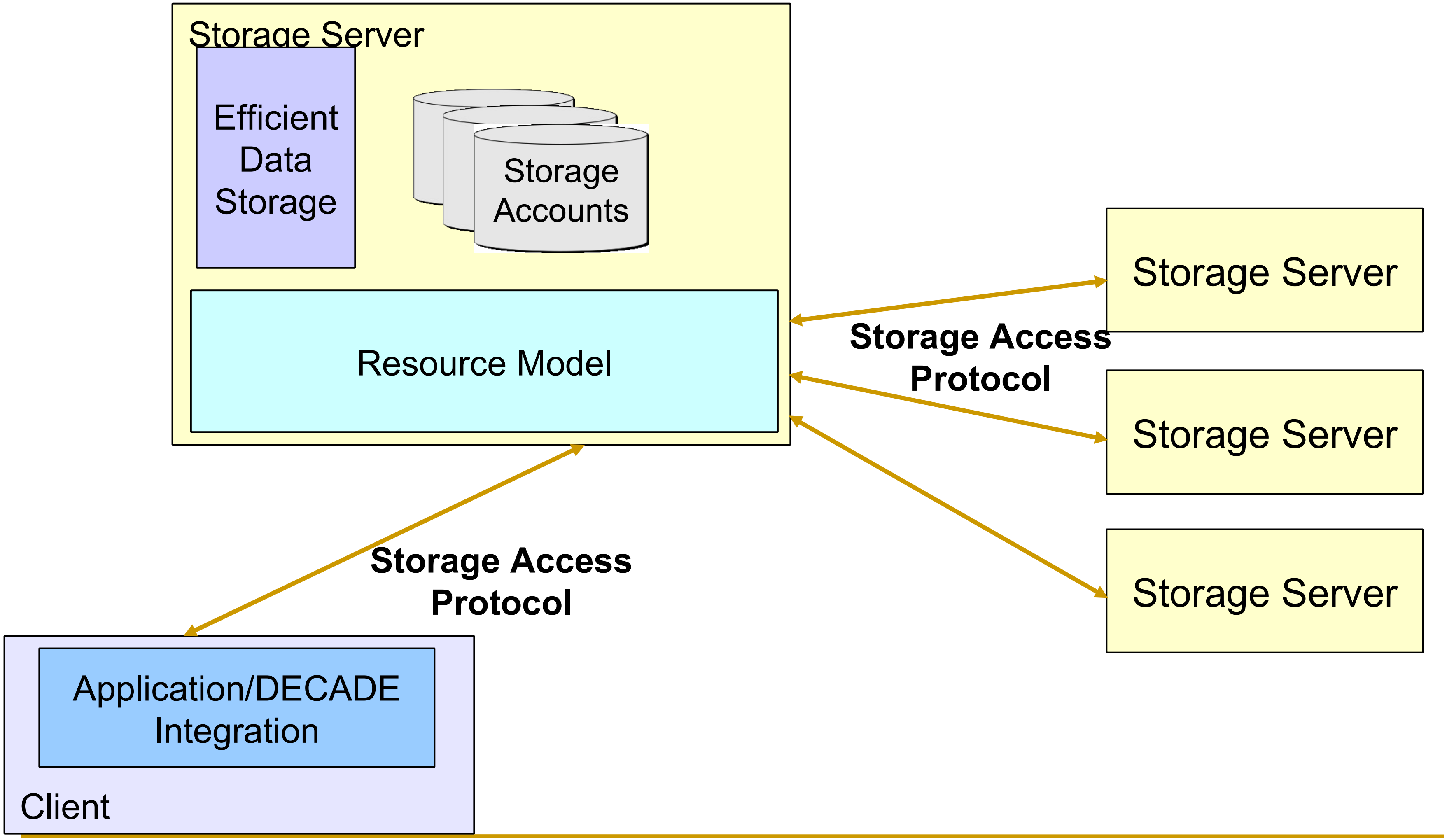
Native BitTorrent Clients



DECADE-enabled BitTorrent Clients

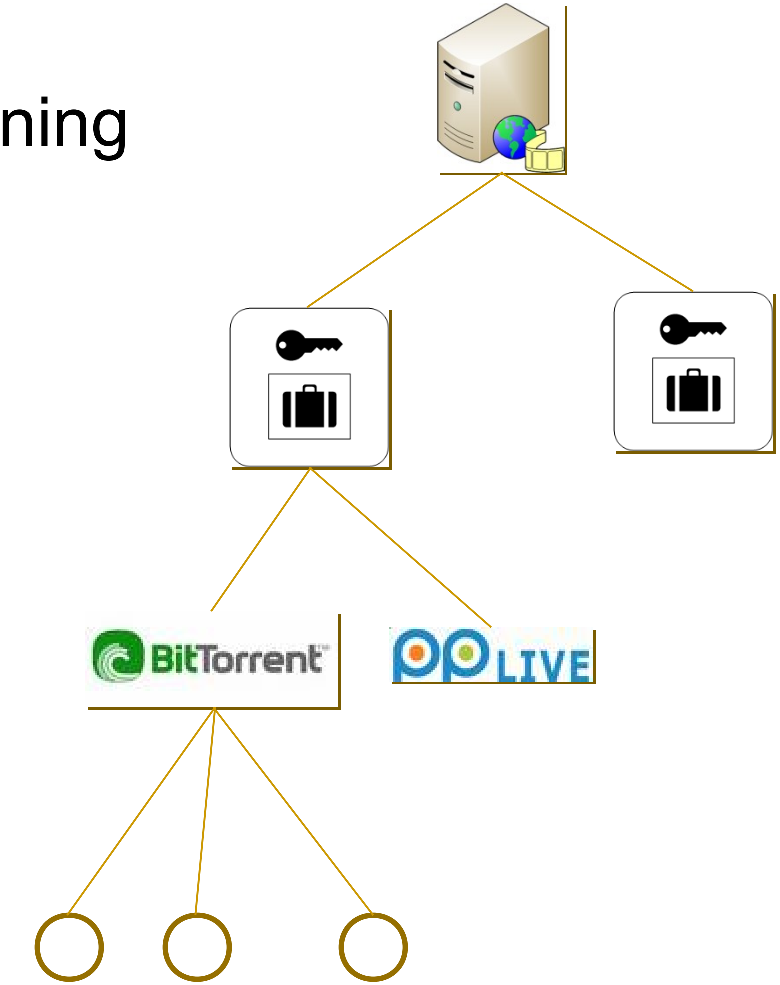


System Architecture



Storage Server Resource Model

- Hierarchical, weighted partitioning
  - Each user assigned a weight by storage provider
  - User configures weight assigned to each application
  - Application controls the partition of resources among open connections (if applicable)
- Resources
  - Bandwidth, storage, open network connections



Access Protocol

- General Approach
  - Storage Server simplicity
    - Scale to many users
  - Reduce resource management messaging

- Components
  - Data Interface
    - Get, store, inter-server communication
  - Management Interface
    - Manage resources in own server

Access Protocol Requirements

- End-to-end Control
  - Users decide (independently) when to use storage
  - Explicit authorization for each item
- Concurrent transfers
  - Upload/download to/from multiple peers
- Low latency data transmission
  - Reduce delay due to passing data though lockers

Authorization using Tokens

- Capability tokens encode
  - Authorization
  - Resource allocation
- Generated and managed by clients
  - Shared key with own storage server
  - Tokens passed via P2P application protocol

Access Protocol: Data Interface

- **`store`**
  - Store object in data locker
  - In: `AppID`, `ObjID`, `ObjData`, `Token`
  - Out: `ErrCode`
- **`get`**
  - Retrieve object from data locker
  - In: `AppID`, `ObjID`, `Token`
  - Out: `ObjData`, `ErrCode`

Access Protocol: Data Interface (cont'd)

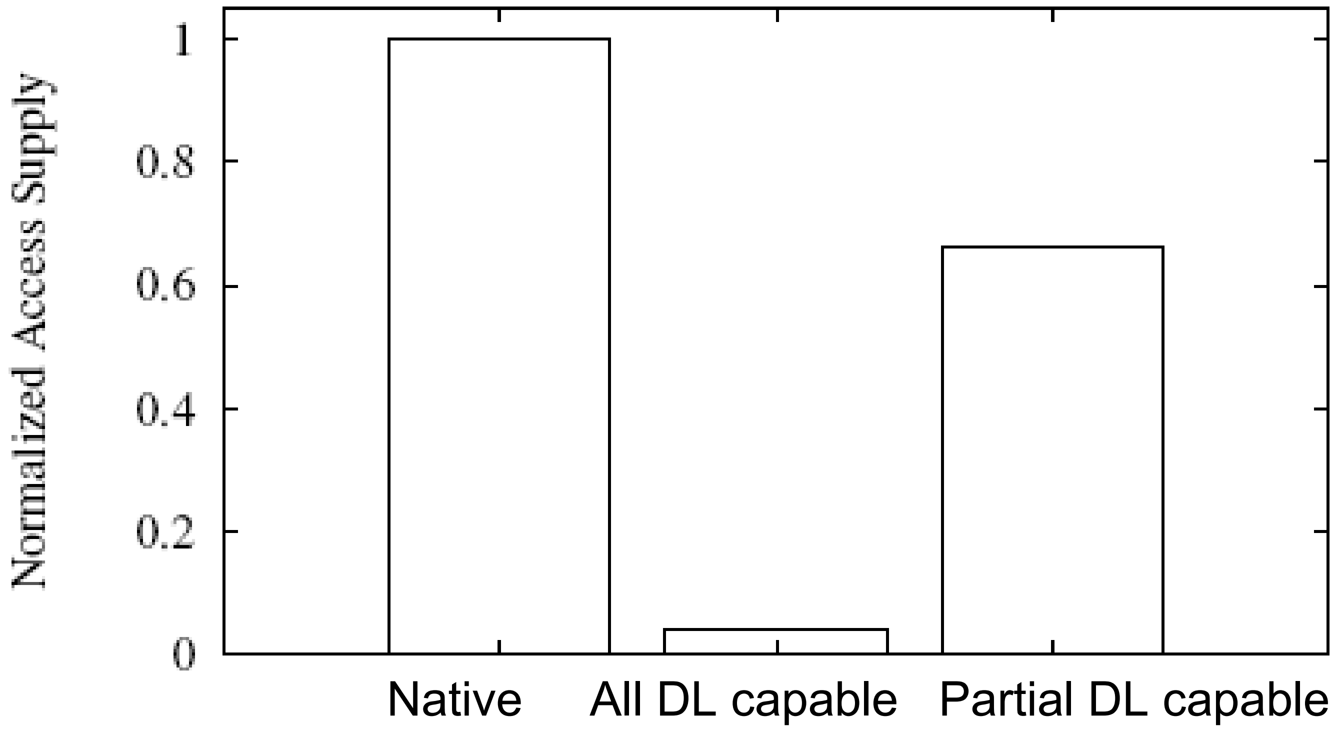
- **`get`** (overloaded)
  - Retrieve object from remote storage server and store into own account
  - In: `AppID`, `ObjID`, `Token`, `RemoteAppID`, `RemoteToken`
  - Out: `ObjData`, `ErrCode`

Thank you!

Backup Slides

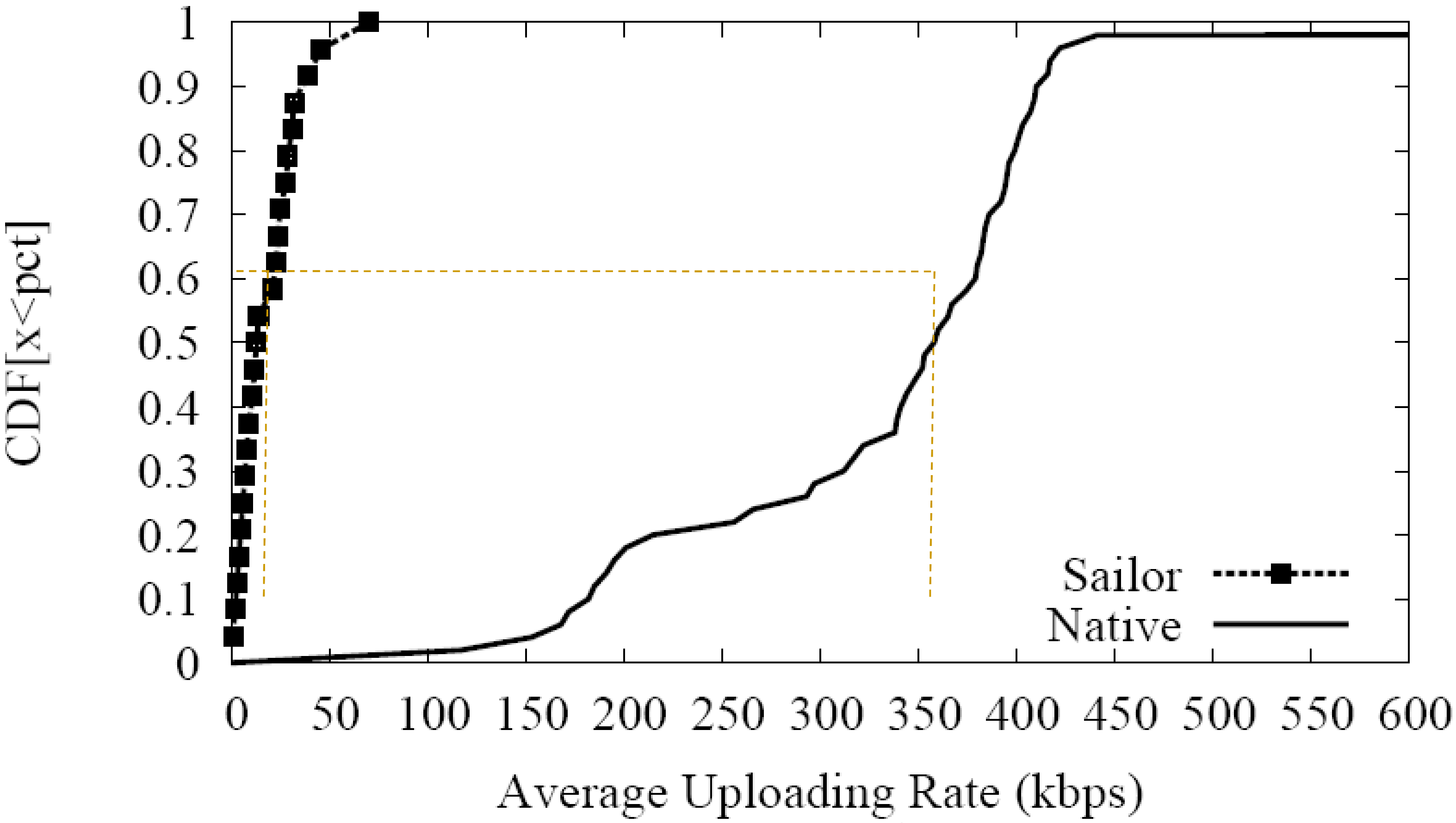
Preliminary Evaluation: Bittorrent

All clients inside an ISP have locker accounts

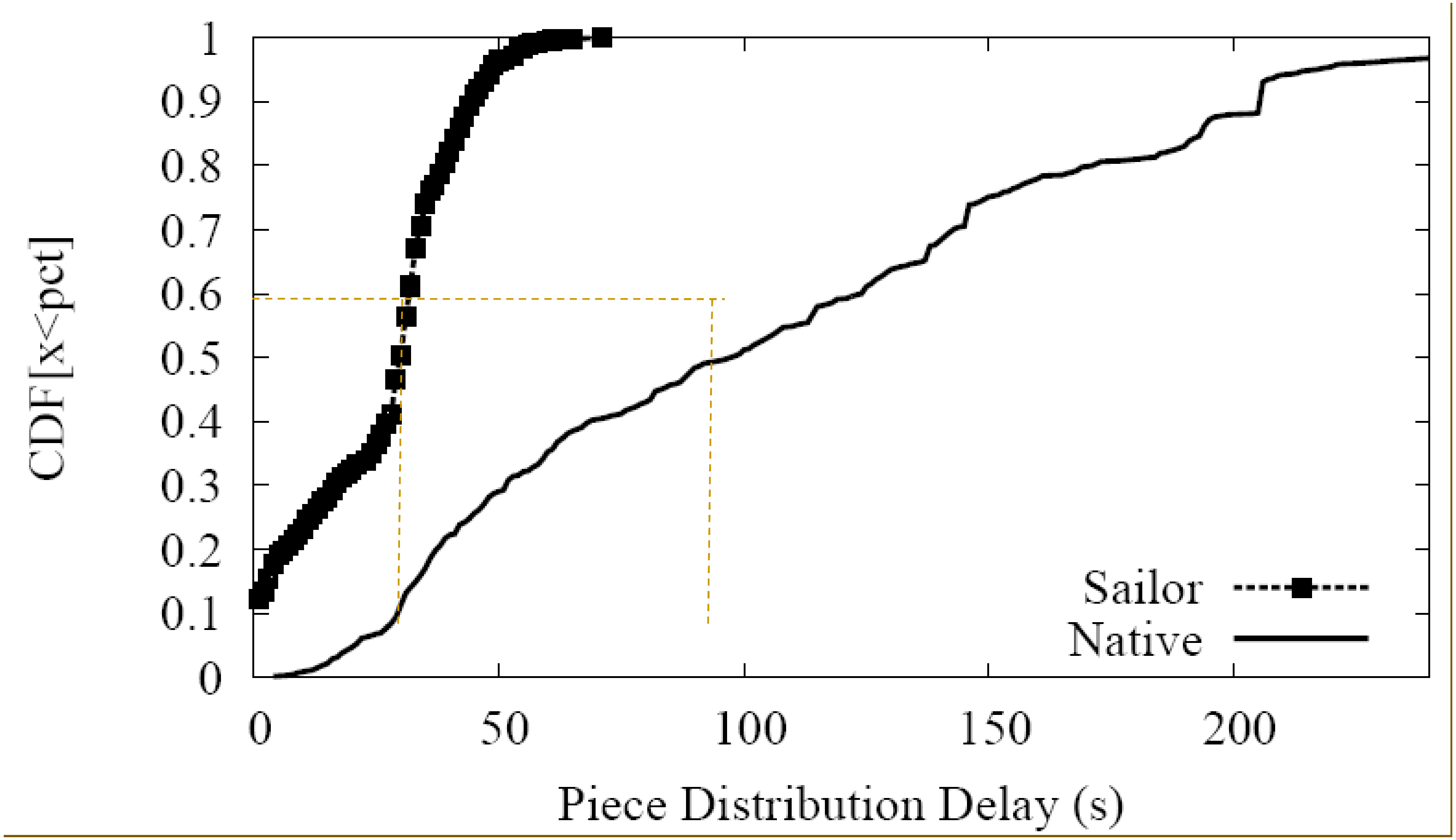


(a) Access Supply

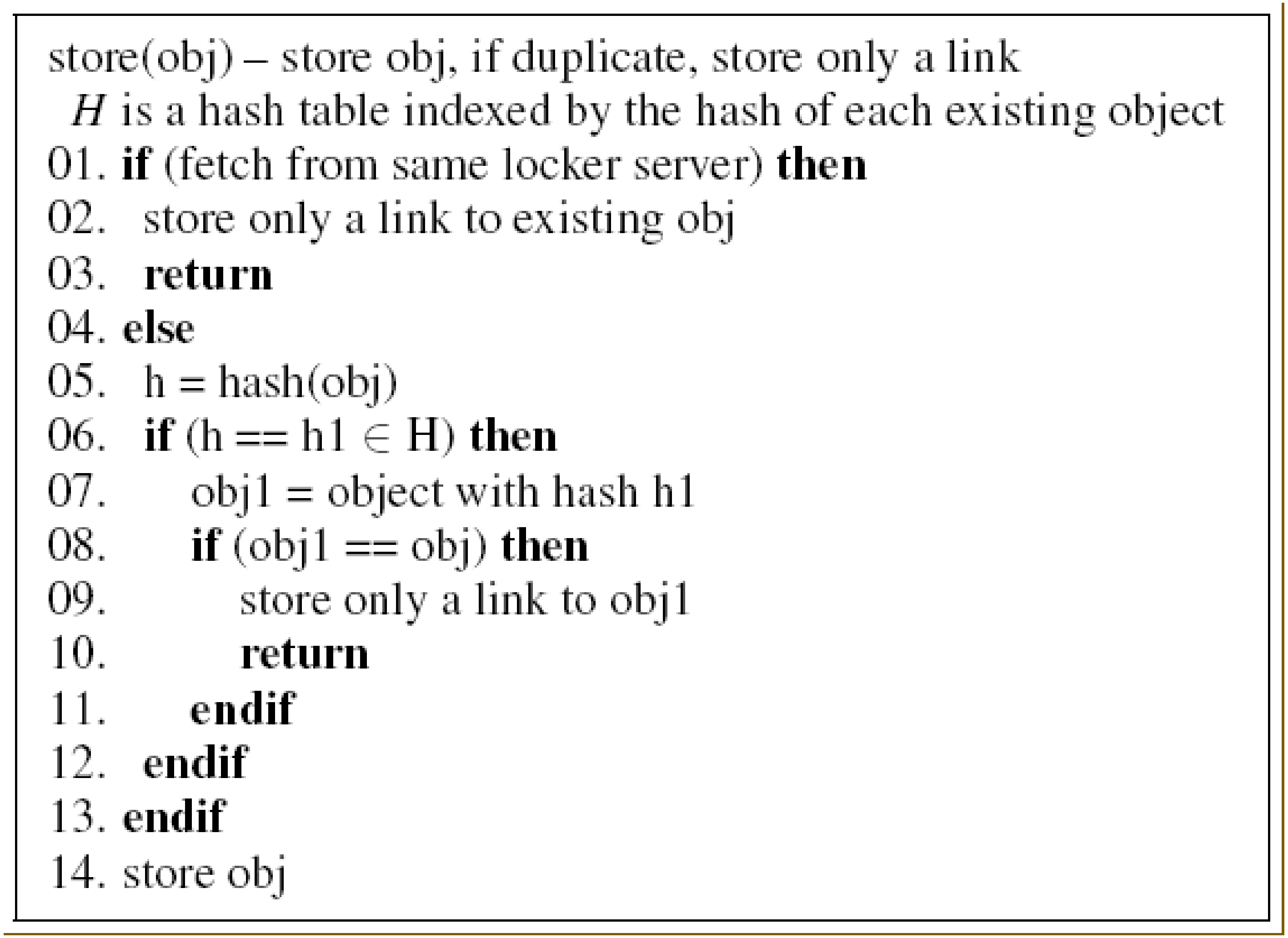
Preliminary Evaluation: PPLive



Preliminary Evaluation: P2P Streaming App.



Efficient Locker Data Storage

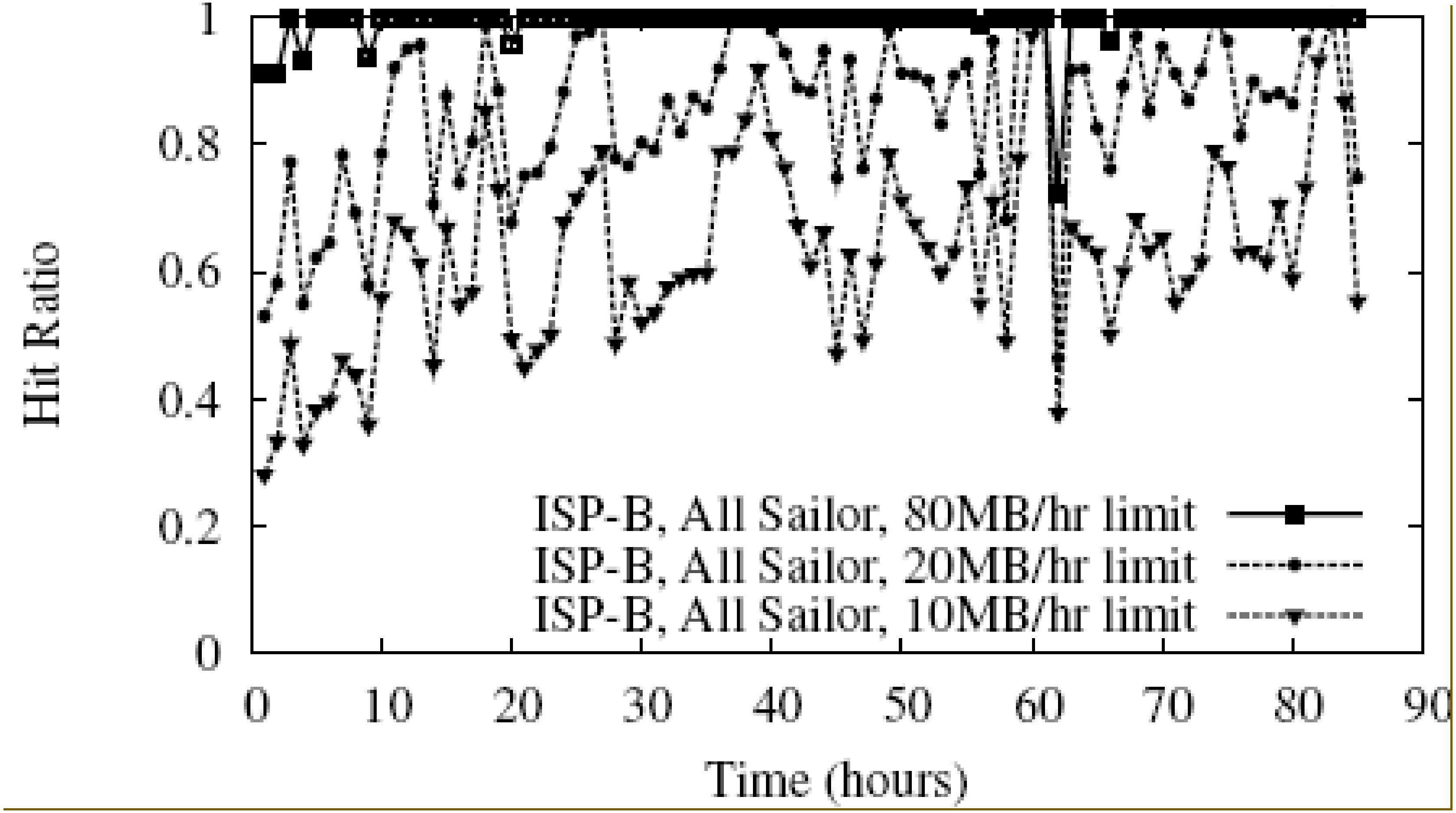


```
store(obj) – store obj, if duplicate, store only a link
 $H$ is a hash table indexed by the hash of each existing object
if (fetch from same locker server) then
    store only a link to existing obj
    return
else
    h = hash(obj)
    if ($h == h1 \in H$) then
        obj1 = object with hash h1
        if (obj1 == obj) then
            store only a link to obj1
            return
        endif
    endif
endif
store obj
```

Data Locker/P4P(ALTO) Integration

- Client a with locker La needs to select peers
- Consider peer b
  - Let $C^0_{a,b}$ be the cost from a to b
- Three cases
  - If b is a legacy peer
    - $C_{a,b} \leftarrow C^0_{a,b}$
  - else if (b supports DL but no locker account)
    - $C_{ab} \leftarrow C^0_{La,b}$
  - else // b supports DL and has locker Lb
    - $C_{ab} \leftarrow C^0_{La,Lb}$

Preliminary Evaluation: Bittorrent


Hit Ratio
Time (hours)
ISP-B, All Sailor, 80MB/hr limit
ISP-B, All Sailor, 20MB/hr limit
ISP-B, All Sailor, 10MB/hr limit