Slides presented at CoXNet Workshop, Beijing, China, Nov. 2010.

Open Content Distribution using Data Lockers

November 4, 2010

Slides by Y. Richard Yang
Presented by Leo Chen

Richard Alimi, Y. Richard Yang, Leo Chen, Harry Liu, David Zhang, Zhihui Lv, Zhigang Huang, Haibin Song, Xiaohui Chen, Ye Wang, Akbar Rahman, David Bryan, Lili Qiu, Yin Zhang

Motivation

- Increasing content availability
  - Both user generated and publisher generated contents
- Increasing consumption of content
  - E.g., Akamai estimates: 1,296 Tbps [Akami]

Challenge

- Potential bottlenecks at
  - the uploader last (first) mile and/or
  - the middle mile
- Solution
  - Replication/service capability inside the networks

Existing Approaches and their Problems

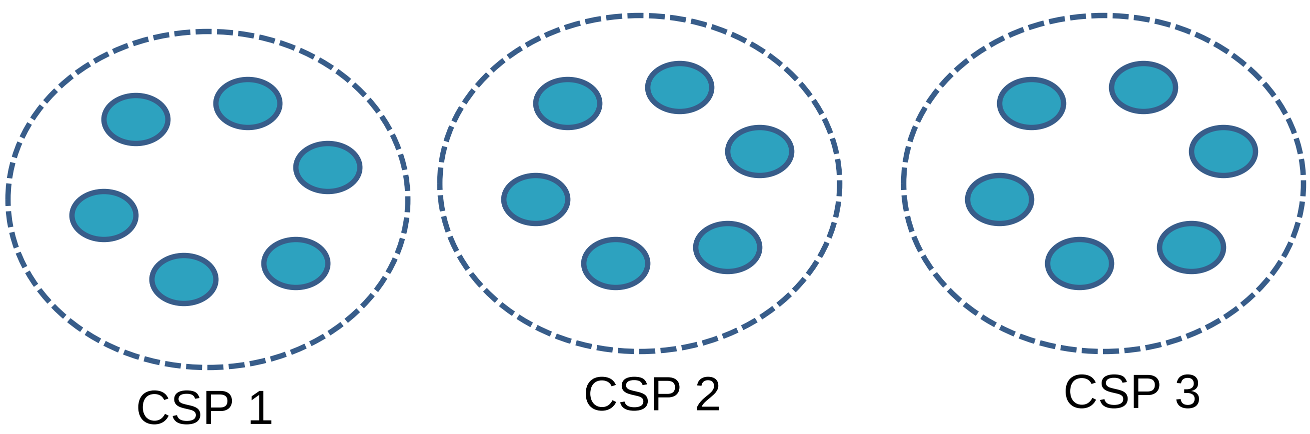
- IP multicast
  - Lacking in global deployment
- CDN
  - Closed systems with limited scopes
- P2P cache
  - Application specific

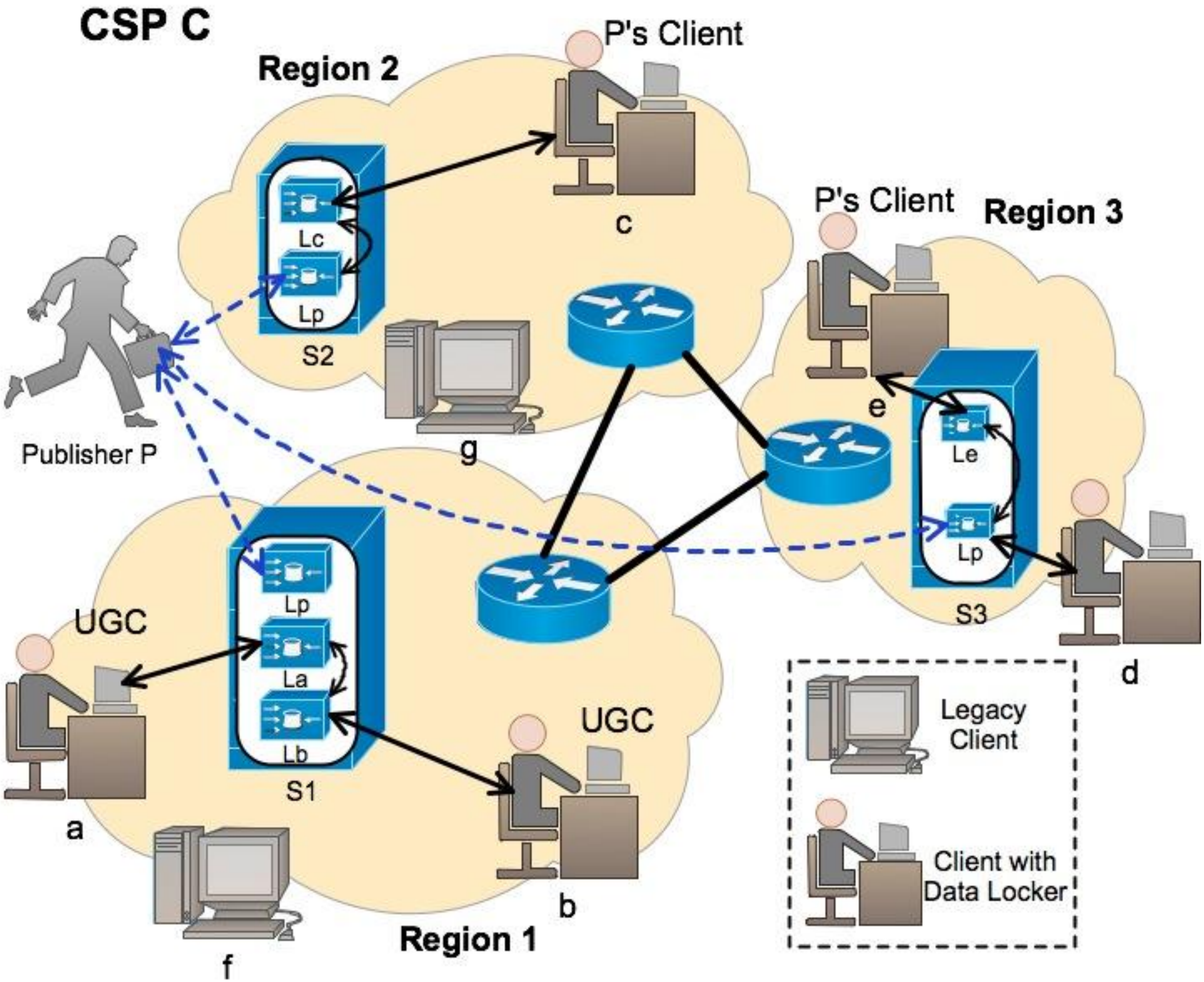
What are Data Lockers?

In-network storage/BW accounts, offered by multiple providers, accessible using a standard protocol, under application explicit control, to provide efficient content distribution.

Who Provide Data Lockers?

- A provider of data lockers is called a Content Delivery Storage Provider (CSP)
- An CSP can be an ISP, or a third party (e.g., cloud storage provider)




CSP C
Region 2
P's Client
c
Lc
Lp
S2
Publisher P
g
P's Client
Region 3
e
Le
Lp
S3
d
Lp
La
Lb
S1
UGC
a
UGC
b
f
Region 1
Legacy Client
Client with Data Locker

What should Data Lockers Provide?

Functions of a Content Delivery System

- Control
  - Content search/index & composition/authorization
  - Replication and request routing
    - Scaling, Efficiency (e.g., Proximity), Load Balancing and Reliability
- Data
  - Storage/transport

Architectural Spectrum

"Weak" Coupling:
Separation of Control
and Data.

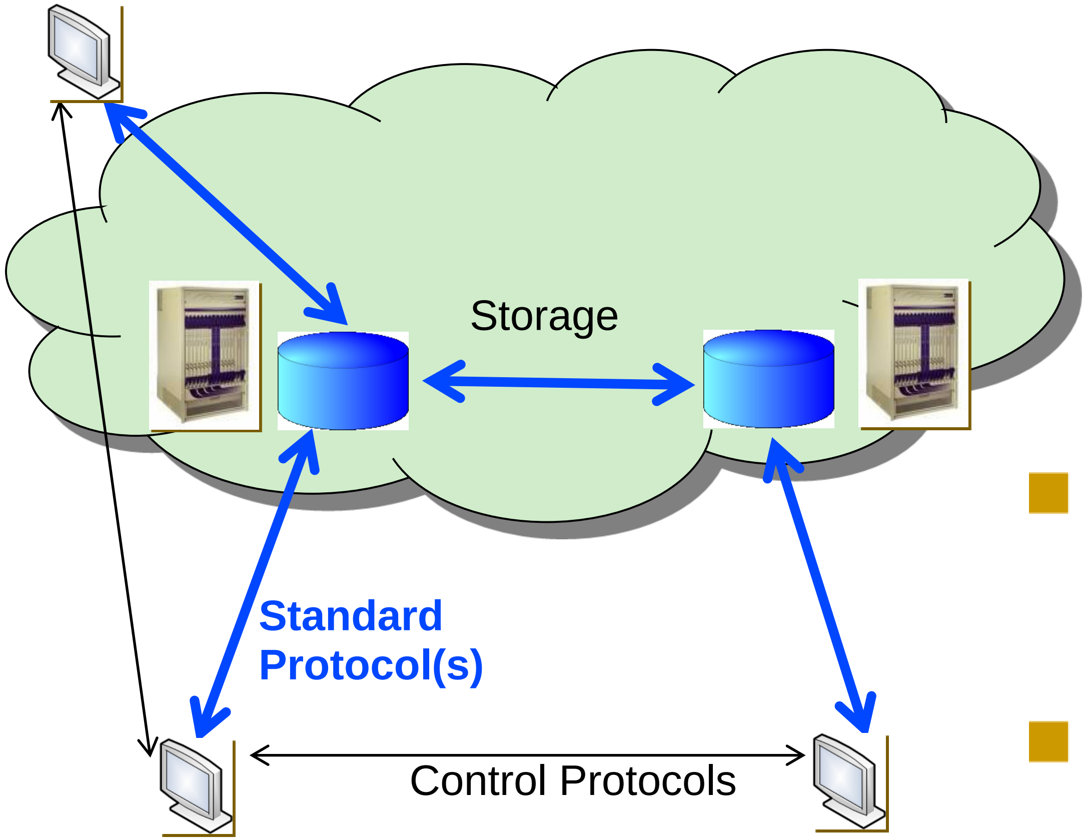
"Strong" Coupling:
Tight integration
of Control
functions into
in-network storage.

A potential extreme is virtual machines.
Not considered for now.

The Decoupled Design Principle

- The data-locker decouples control functions out of data functions as much as possible
  - Decoupled Control and Data Enroute (DeCaDE)
- Control functions implemented either by applications or separate control protocols

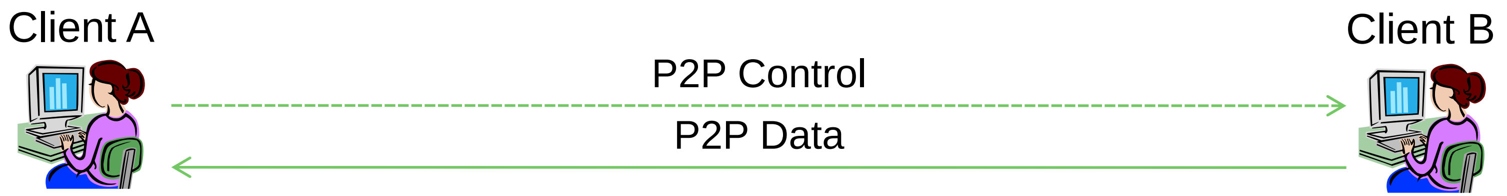
Overview



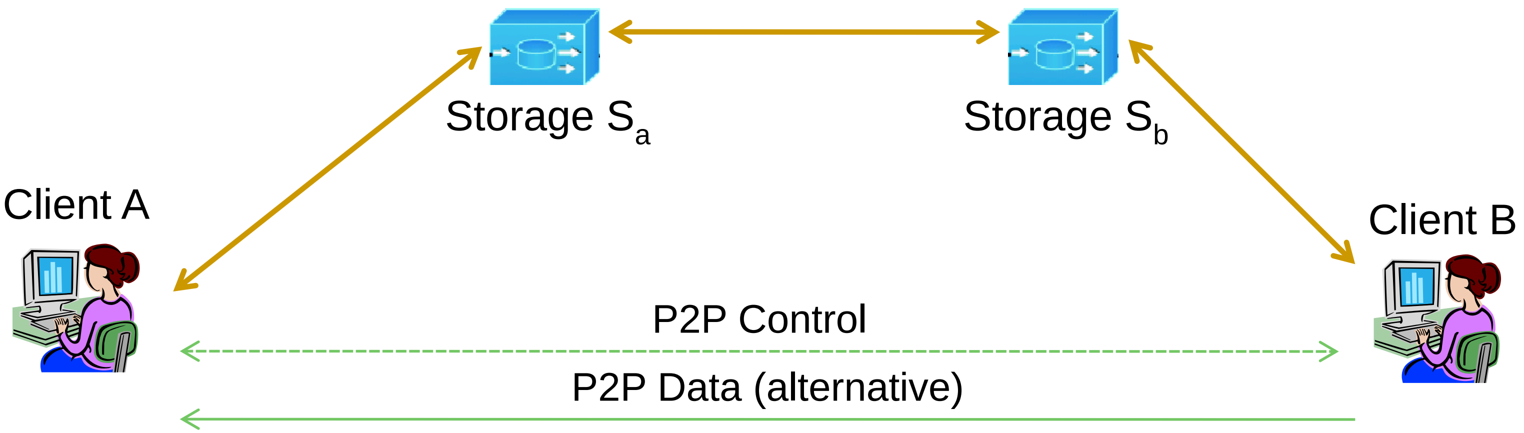
- Reduce production complexity and provide open access
- Integrate with application policies

Use Case: End-User Based Control

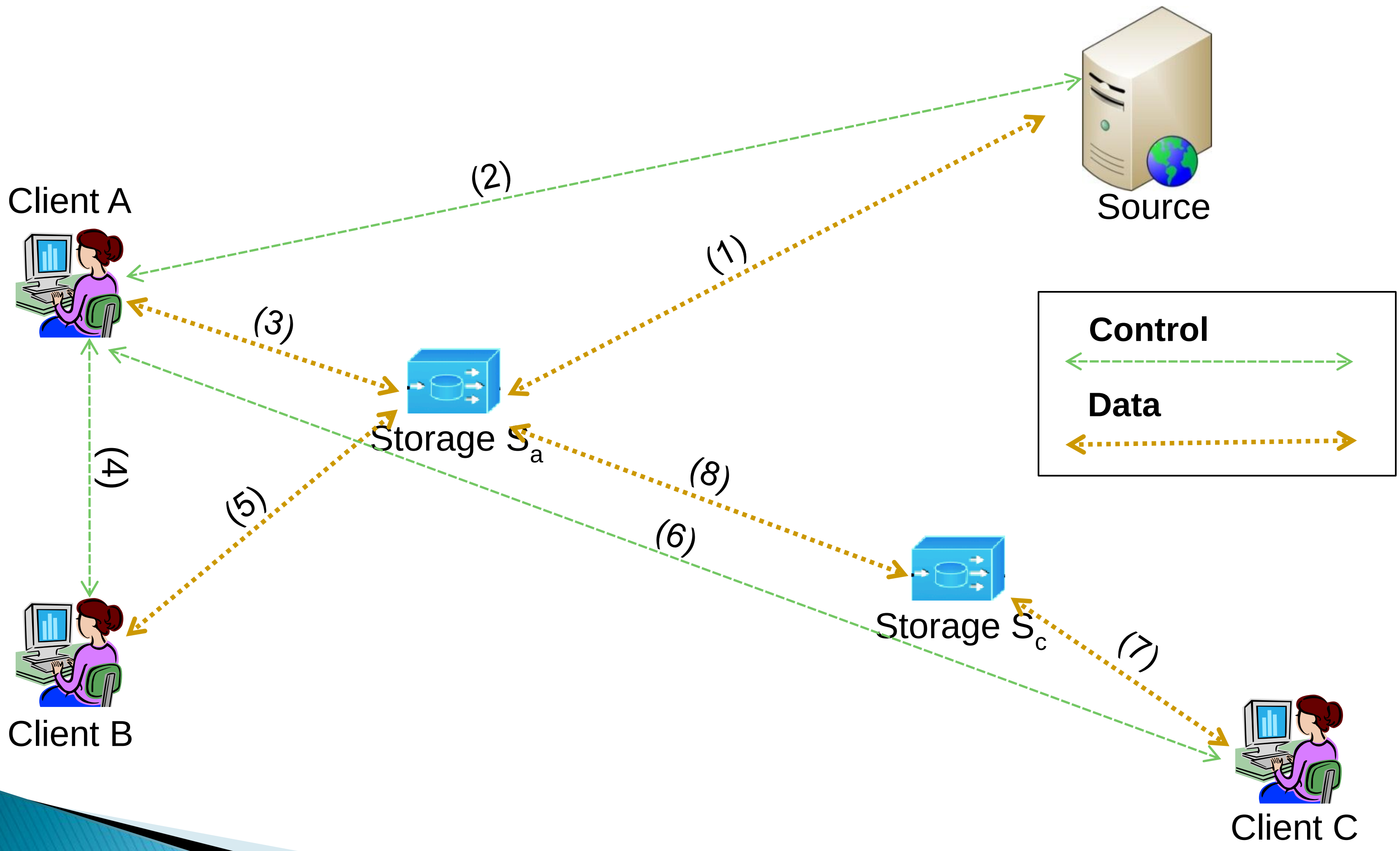
Native P2P Clients



Data Locker-enabled P2P Clients



Use Case: End-User Based Control

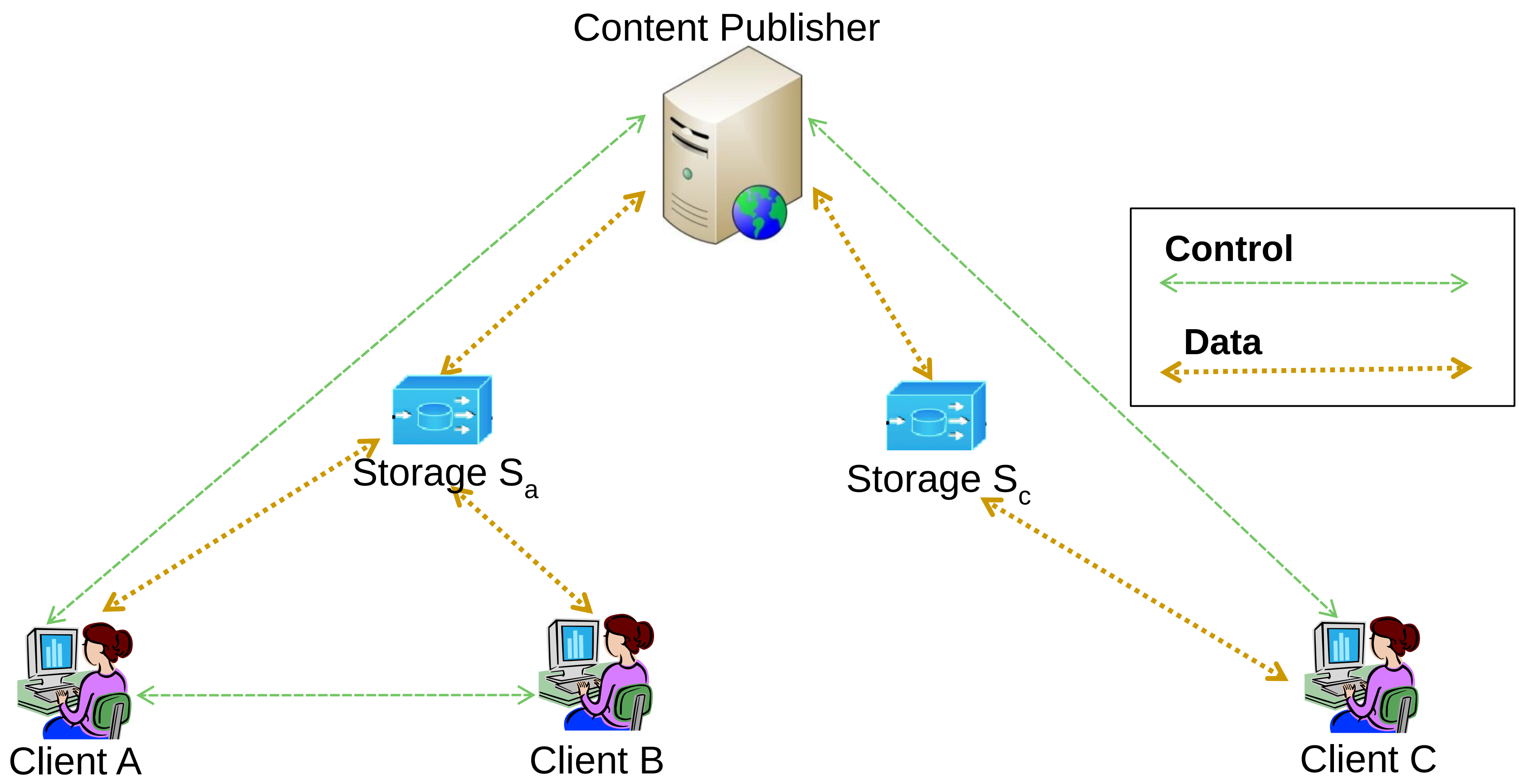


Can Data Lockers Address Challenges of P2P Based Control?

- Low network efficiency, in particular last mile
  - Upload from data locker account of a client, instead of client last mile

- Instability due to high churns
  - Upload from data locker account of a client, even when client goes offline

Use Case: Publisher Based Control



Potential Benefits of a Decoupled Architecture

- Separates the storage/bandwidth intensive (data) functions from the processing intensive (control) functions
  - Flexible/open/evolvable control platforms
  - Shared data infrastructure

- Decoupled architecture is not new, e.g.,
  - openflow
  - Google File System (GFS)

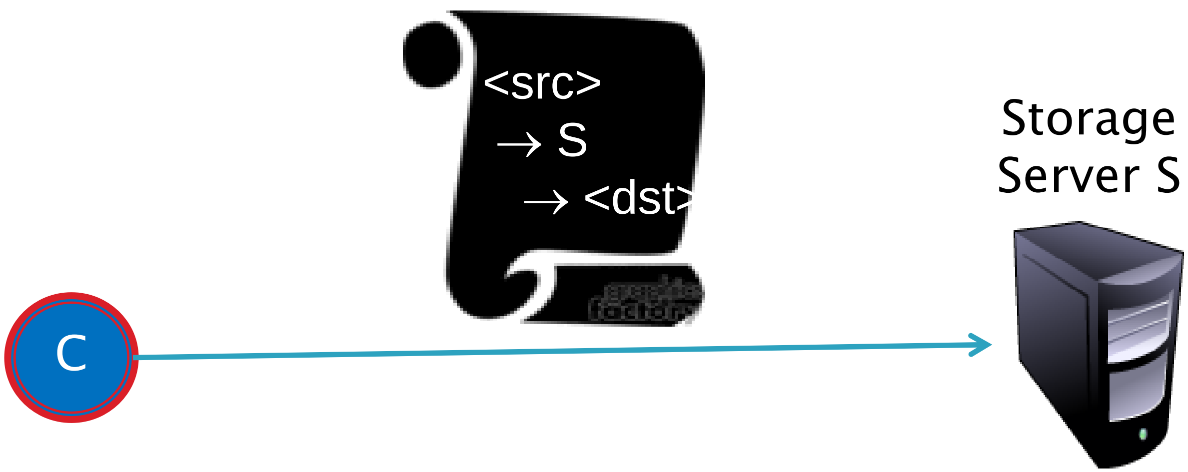
Design Details: What Does the Data Path Look Like?

Data Naming and Storage Model

- Data naming is an important problem
  "There are only two hard problems in Computer Science:
  cache invalidation and naming things."
  -- Phil Karlton
- There are many naming models,
  - E.g, Filename, URL, attribute, DONA
- Key assumption
  - Content distribution deals with immutable data
- For immutable data, no need to separate identifier and content
- Design:
  - Each account at a server provides a key-value store with self-certifying keys
  - ID=Key: <Hash_of_DataBlock>
  - Value: <DataBlock>

Read/Write Model: The Basic Distribution Primitive

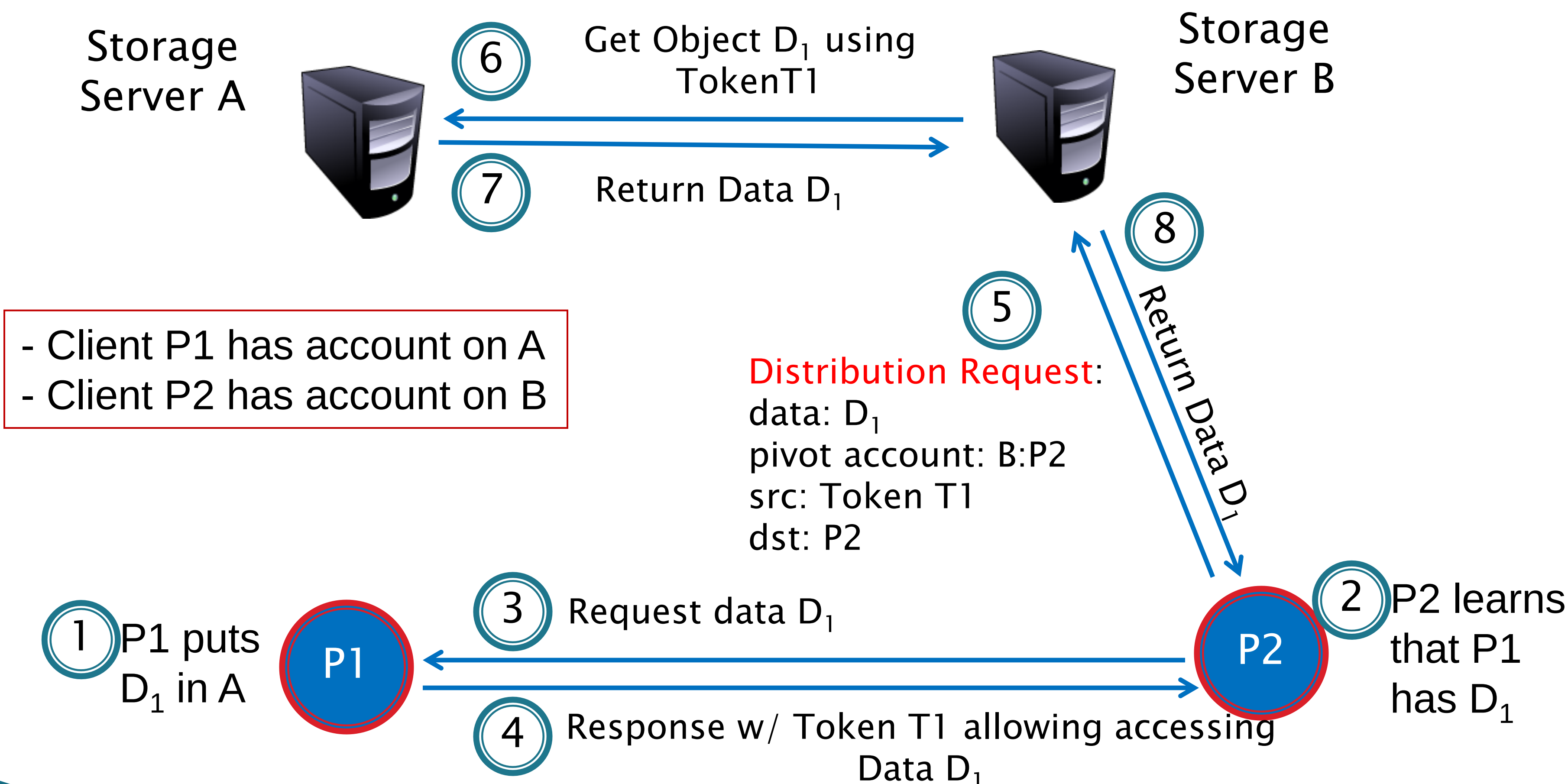
- A basic data command primitive is to indicate a data path



Read/Write Model: The Basic Distribution Primitive

- The data path primitive from Client C to server S specifies
    - <data id>,
    - a <src>,
    - an account on S <S:account>, and
    - a <dst>.
  - Interpretation
    - If <src> is null: it is a pure read to transfer data from <S:account> to <dst>
    - If <dst> is null: it is a pure write to store data from <src> to <S:account>
    - Otherwise, it is a distribution pipeline from <src> to <S:account> to <dst>

Example: Endpoint Controlled Data Flow



Write with Deduplication

```
if (no data across the server for <data id>)
   server S gets data from <src>;
else
    if (<src> is an account on same S)
                        // local deduplication
       link data, and ack succ else
    else
                        // remote BW deduplication
      sends challenge for verification
```

Note: Could move dedup out to app, but then fully implementing dedup requires cross application/session synchronization. Also, hood for content checking. <src> can be protected, can also be chained.

Resource Model

Two Major Architectural Components of Multipath Data-Oriented Content Distribution

Topology Management	Who connects to whom?
Chunk (Data) Scheduling	Who serves whom at what rates? Includes - A downloader requests from which uploaders - An uploader serves which downloaders at what rates

We can consider both components as conducting resource control on resources, including

- connection slots
- upload/download bandwidth
- storage capability

Why is Resource Control of BW/Connectivity Important?

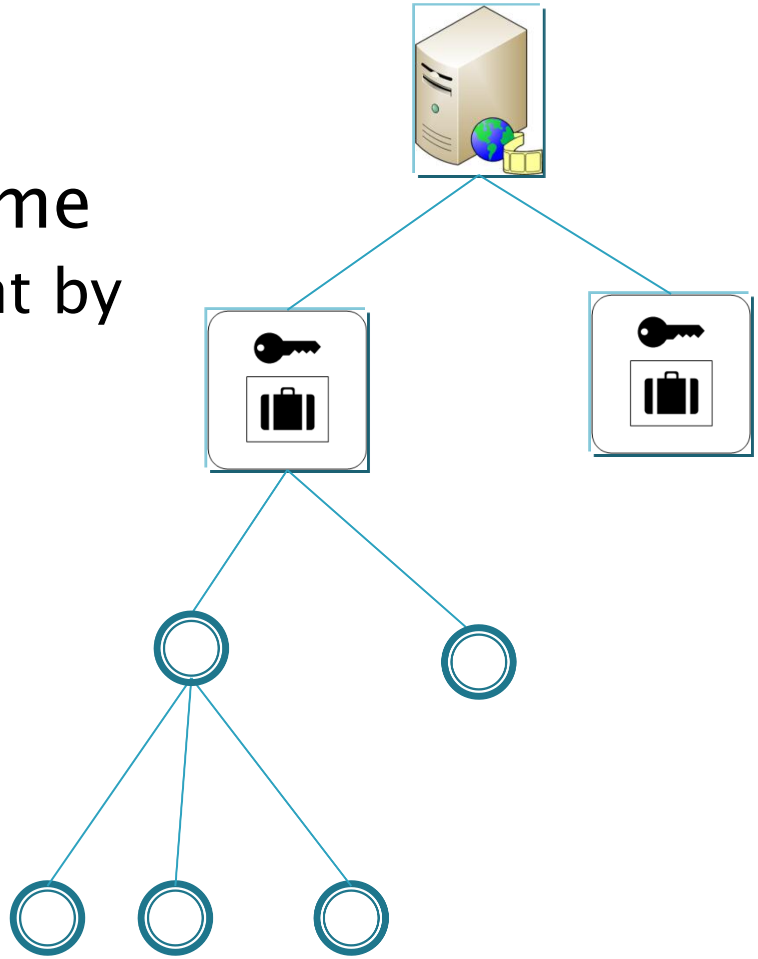
- Because BW resource control is fundamental for
  - Robustness against selfish behaviors
  - Robustness against attacks
  - Construction of efficient flow distribution patterns (in particular for streaming)

Example Flow Pattern: Live Streaming Feasibility Theorem

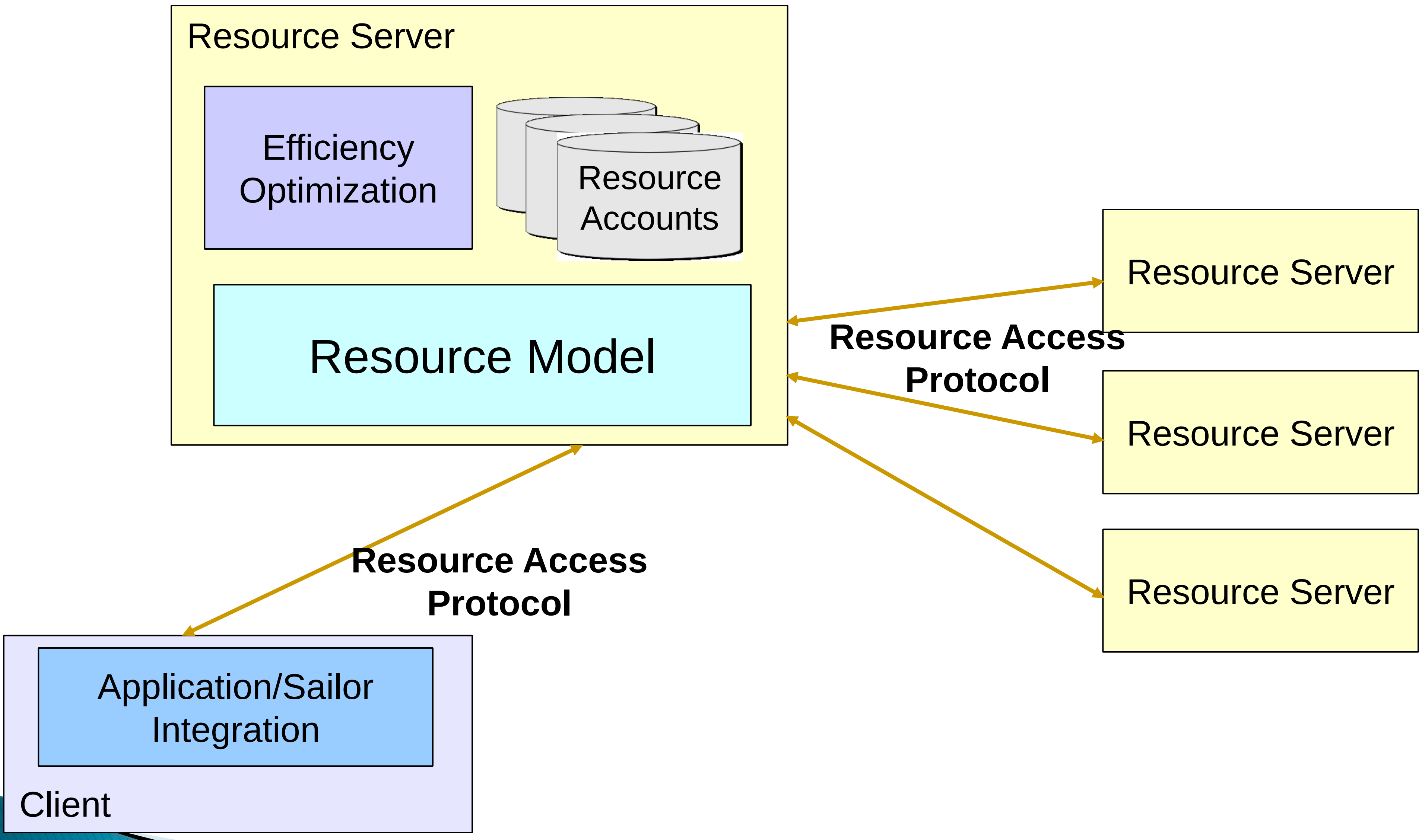
- The flow patterns depend on application types and can be the key “secret sauce” of different designers
- For live streaming
  - Assume that each peer u allocates capacity C_{uv} to a connected neighbor v
    - We call C_{uv} the link capacity of the link u to v
  - Constraints that $\{C_{uv}\}$ should satisfy:
    - Quota: sum of C_{uv} over all neighbors {v} of u should be less than the upload capacity of u
    - Flow Pattern: For any peer p, the maximum flow (minimum cut) from source s to destination peer p, under link capacity constraints, should be at least the streaming rate R

Data Locker Resource Model

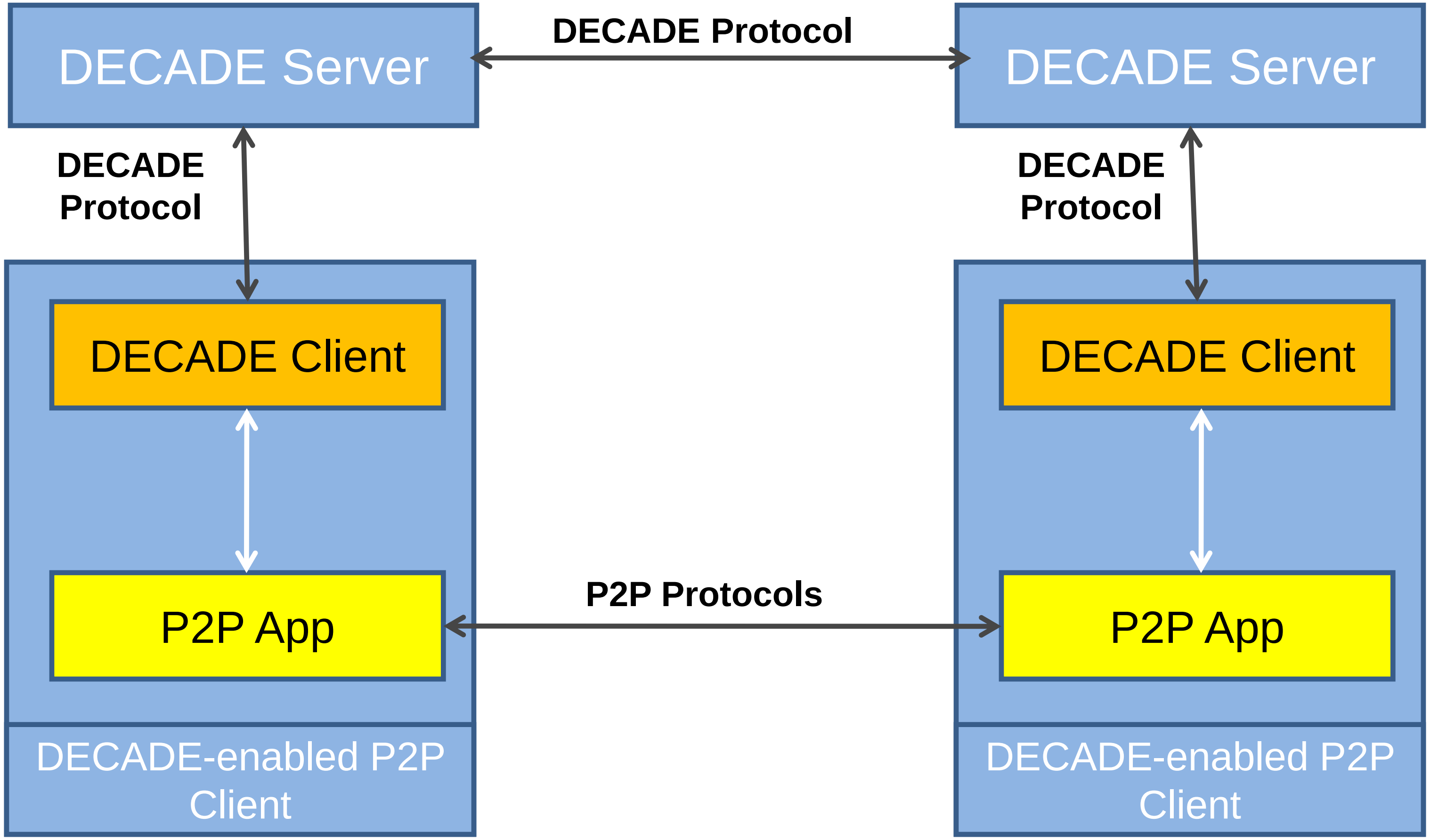
- A hierarchical, weighted resource partitioning scheme
  - Each user is assigned a weight by the data locker provider
  - A user configures weight assigned to each concurrent application
  - Each application controls the partition of resource among open connections



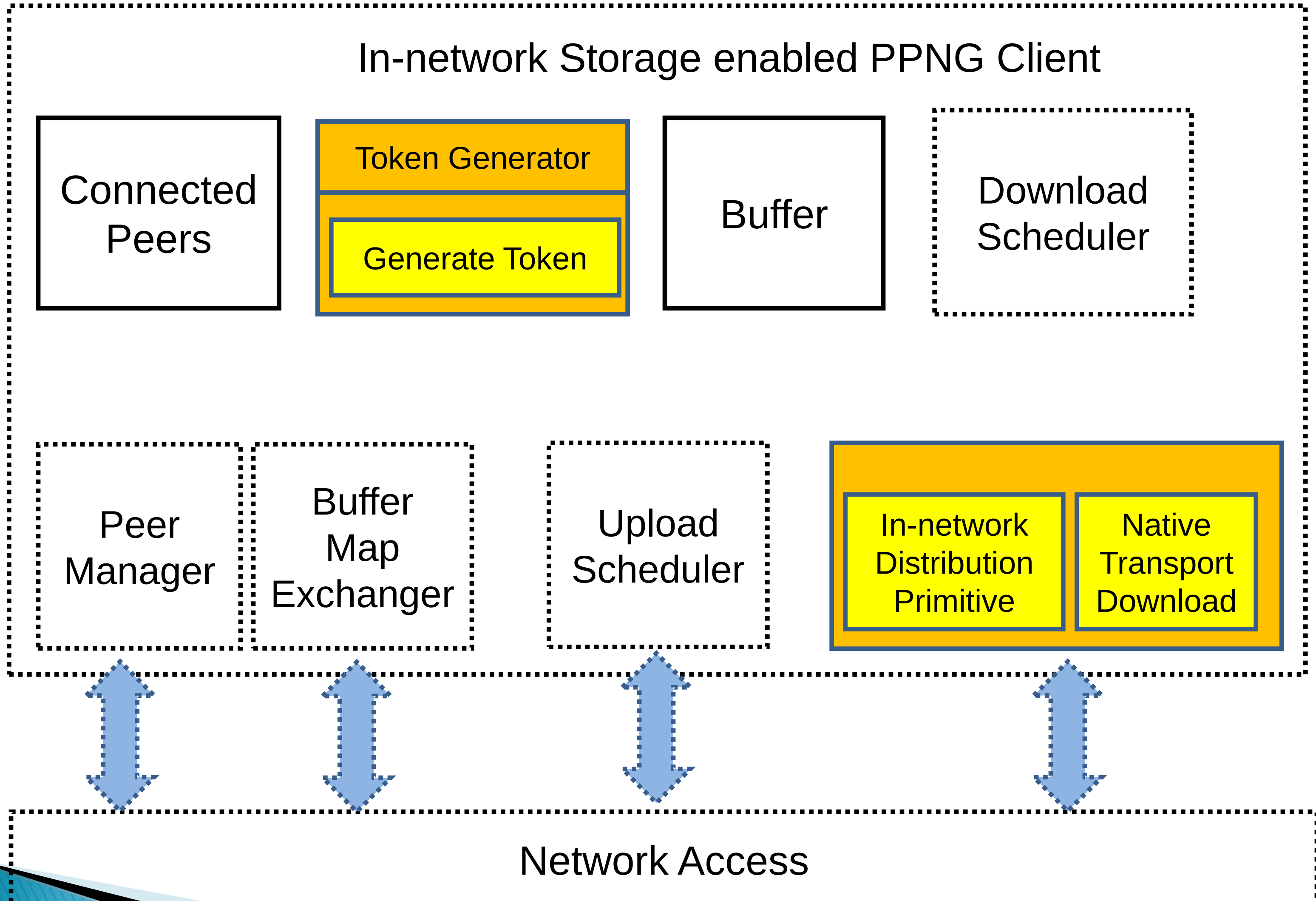
Evaluations


Resource Server
Efficiency Optimization
Resource Accounts
Resource Model
Resource Server
Resource Access Protocol
Resource Server
Resource Server
Resource Access Protocol
Application/Sailor Integration
Client

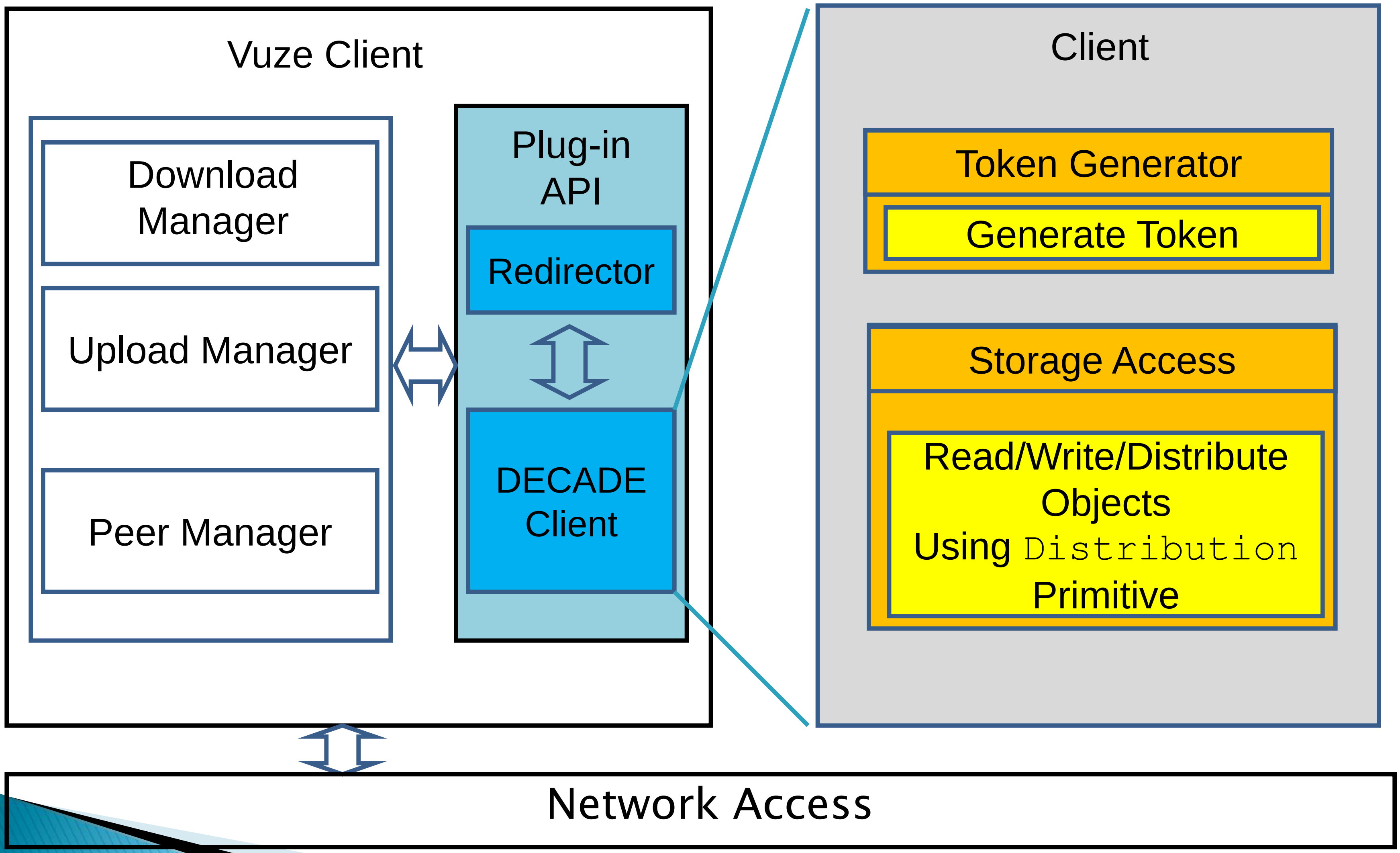
Integration Overview



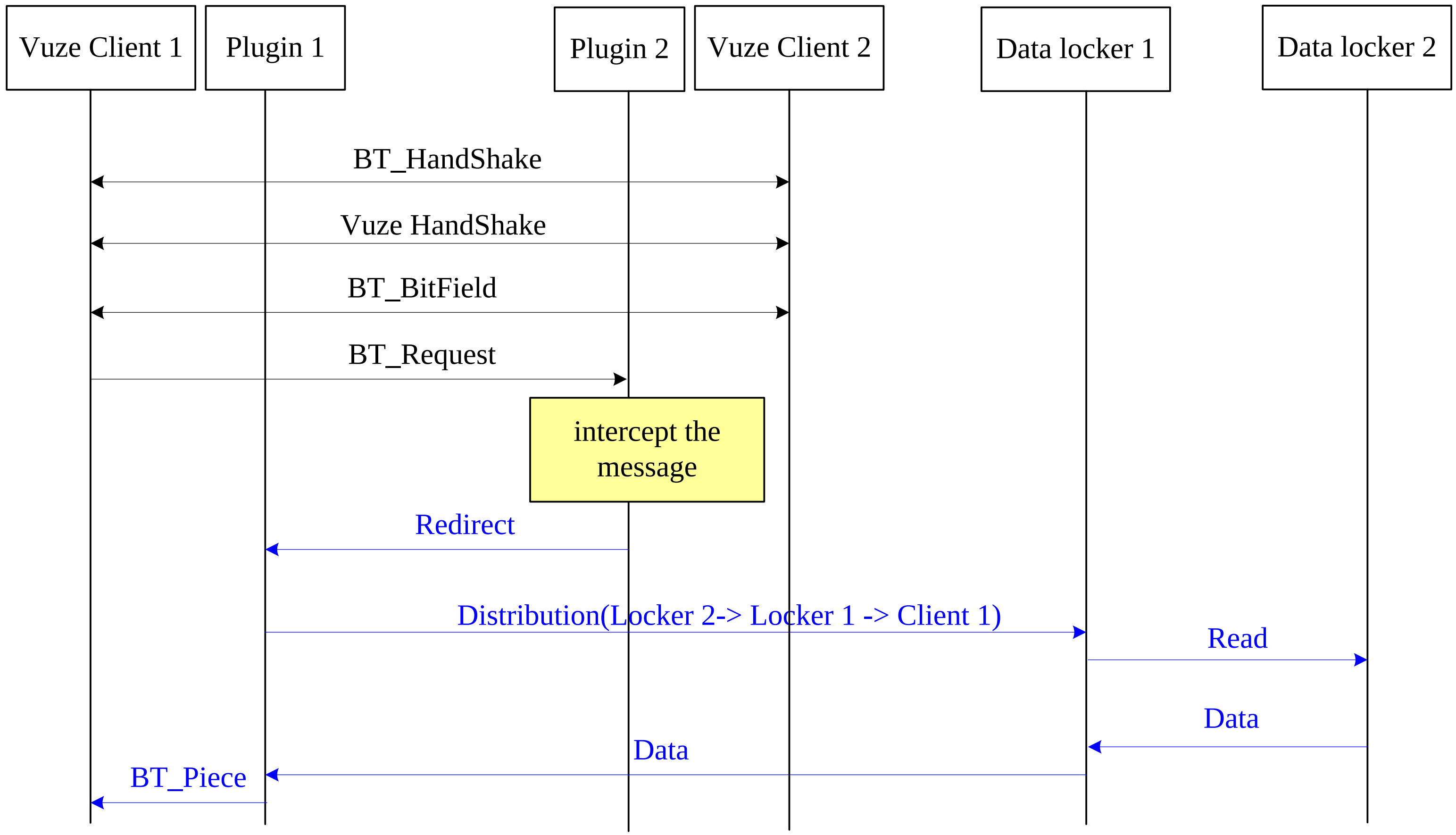
Integration Case 1: P2P Live Streaming



Integration Case 2: P2P File Sharing



Example: Data Request Flows of Vuze with Storage

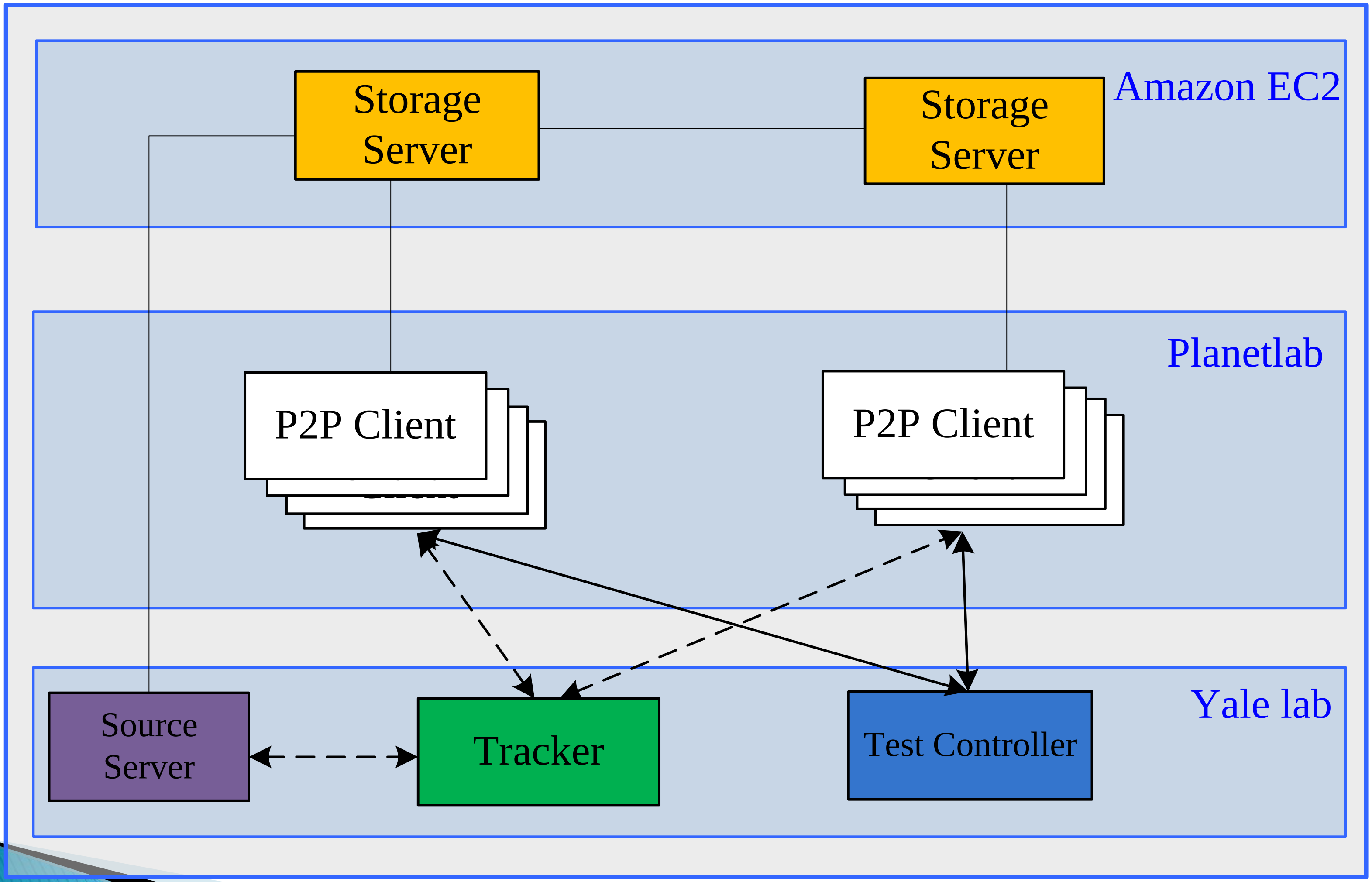


Benchmark Setting

- For performance comparison of Native and using Data Lockers, always consider two scenarios with **the same amount of total network bandwidth resource**

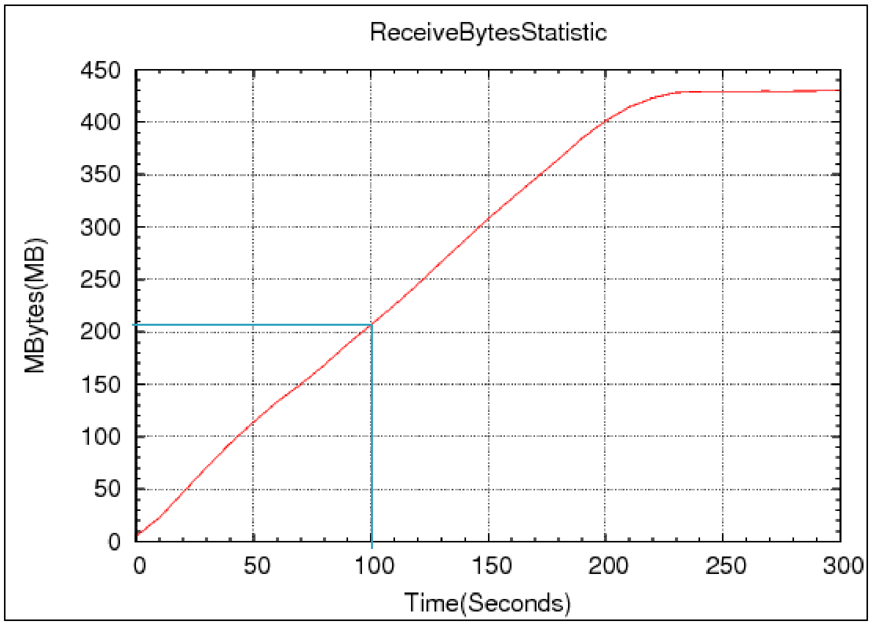
- **Evaluation on both file sharing and live streaming**

Experimental Setting

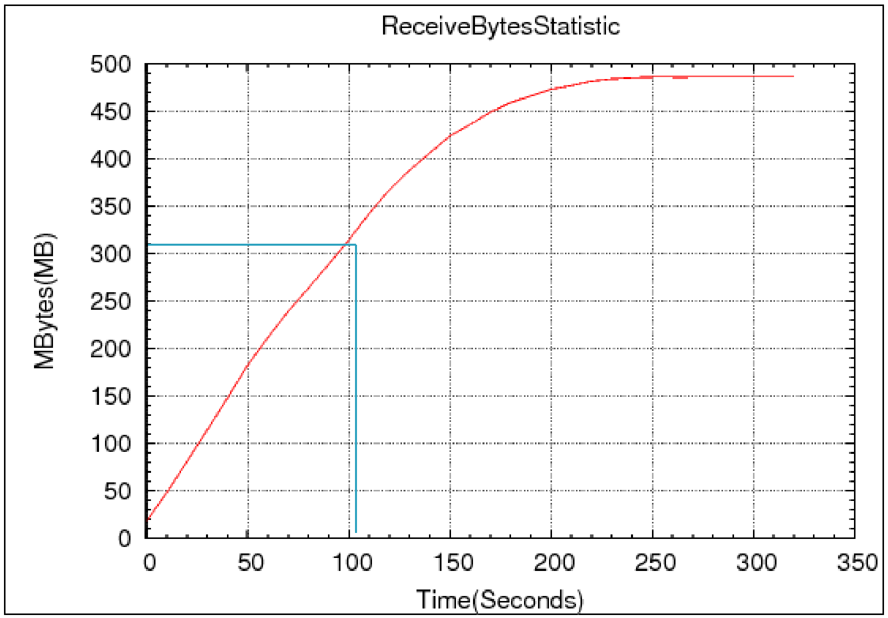


Evaluation: Non-Realtime Content

Integrating into Vuze/BitTorrent: Last Mile Download Traffic



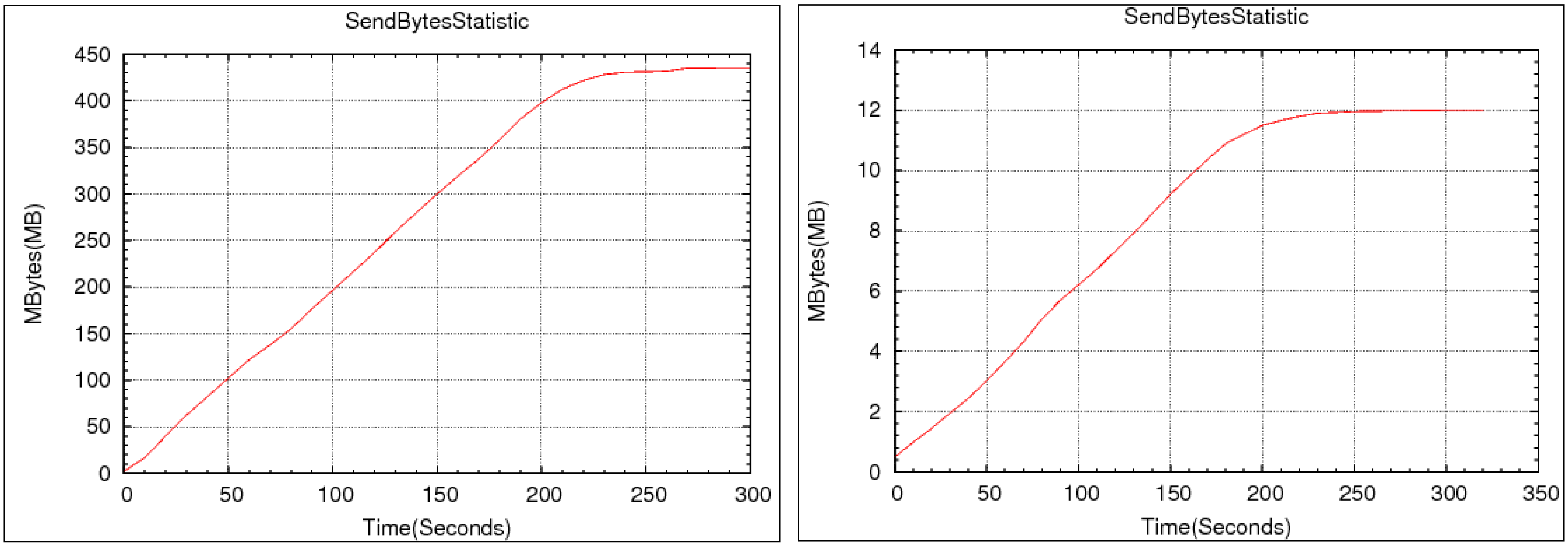
Native Vuze



Data Locker Vuze

- Note: The same total number of Vuze clients; but some Native Vuze clients could not finish downloading; the total download traffic of Native is lower

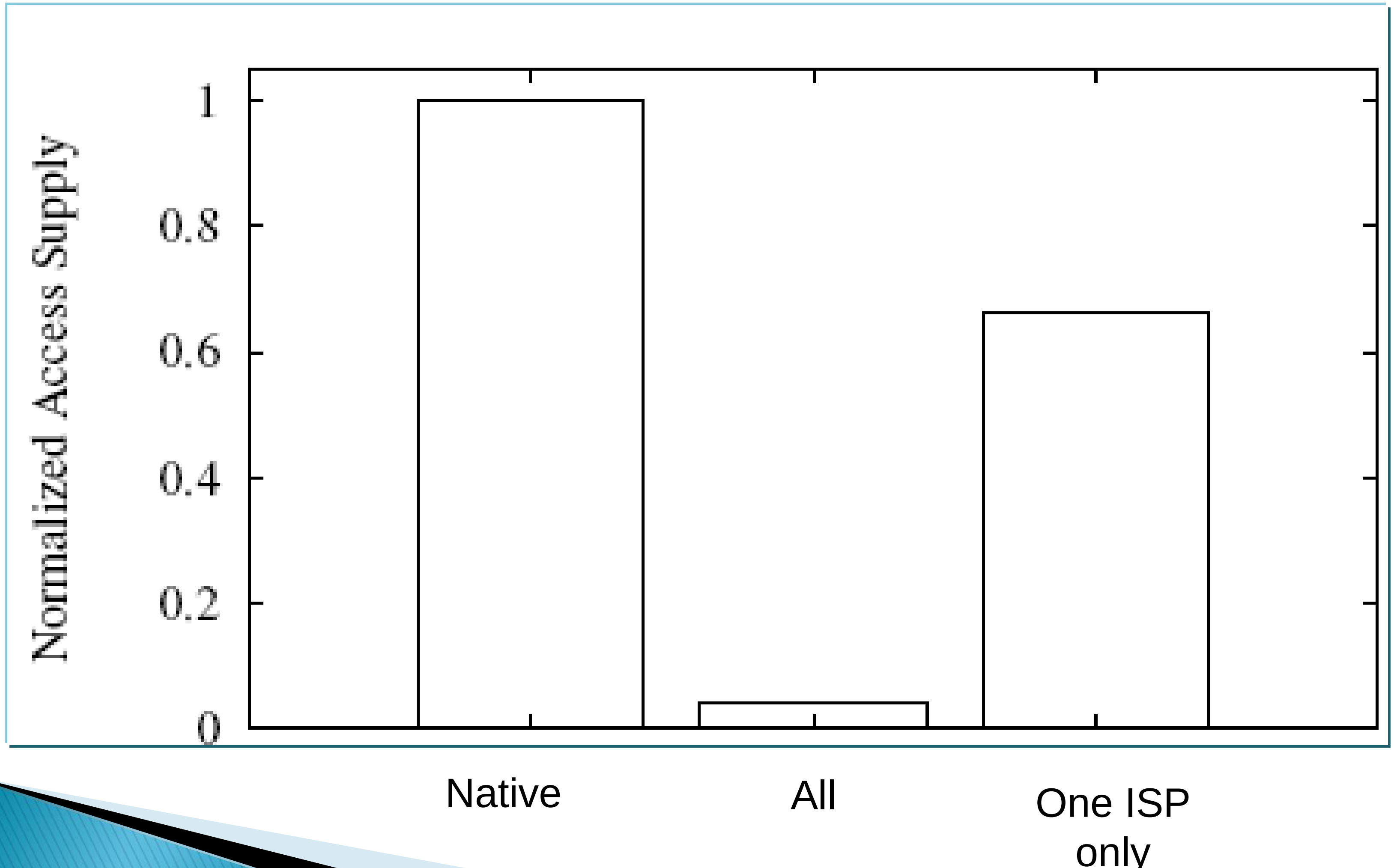
Integrating into Vuze/BitTorrent : Last Mile Upload Traffic



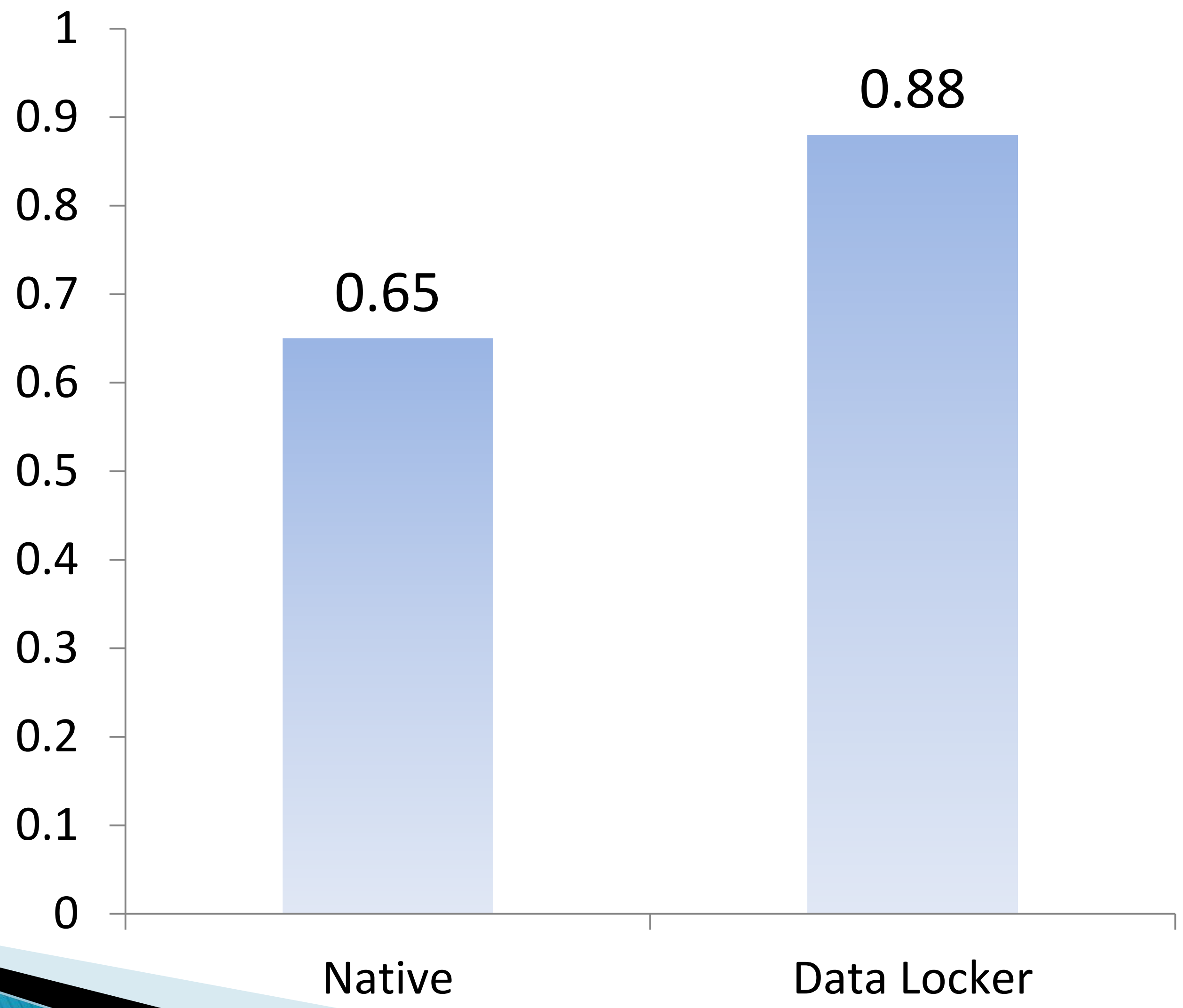
Native Vuze

Data Locker Vuze

Incremental Deployment: Last Mile Upload Traffic



Application Performance (Resource Efficiency)



Implication:

Speed up
= (0.88-0.65)/0.65
= 35%

Why Does Data Locker Gain on Efficiency?

- Statistical multiplexing gain
- Server deduplication to change the location of bottleneck
- Decoupled faster control cycle to speedup distribution
  - Not implemented by current prototype

Summary: P2P File Sharing

	Improvement with In-network Storage
Client upload volume	430 MB → 12 MB
System resource efficiency*	65% → 88% (35% speedup)

*System resource efficiency: fraction of total available upload capacity used

Evaluation: Live Streaming Content

Results: P2P Live Streaming

	Improvement with In-network Storage
Startup delay	At 80-percentile: reduced to 1/3 when no storage
Piece lost rate	About the same, at ≤ 0.02%
Average # of freezes	Reduced to 2/3 when no storage

Summary: Current Design Choices

- Storage Model
  - key-value store with self-certifying keys
- Read/Write
  - [RWDirect] with deduplication
- Resource Management
  - Work-conserving proportional allocation

Next Steps

- Presented preliminary design
- Pursuing the direction in IETF DECADE Working Group
- Participation welcome!

Backup Slides

Problem of P2P Cache

- *Poor documentation, ongoing protocol changes and rapid introduction of new features make P2P protocol support in caching system a constantly moving target.*

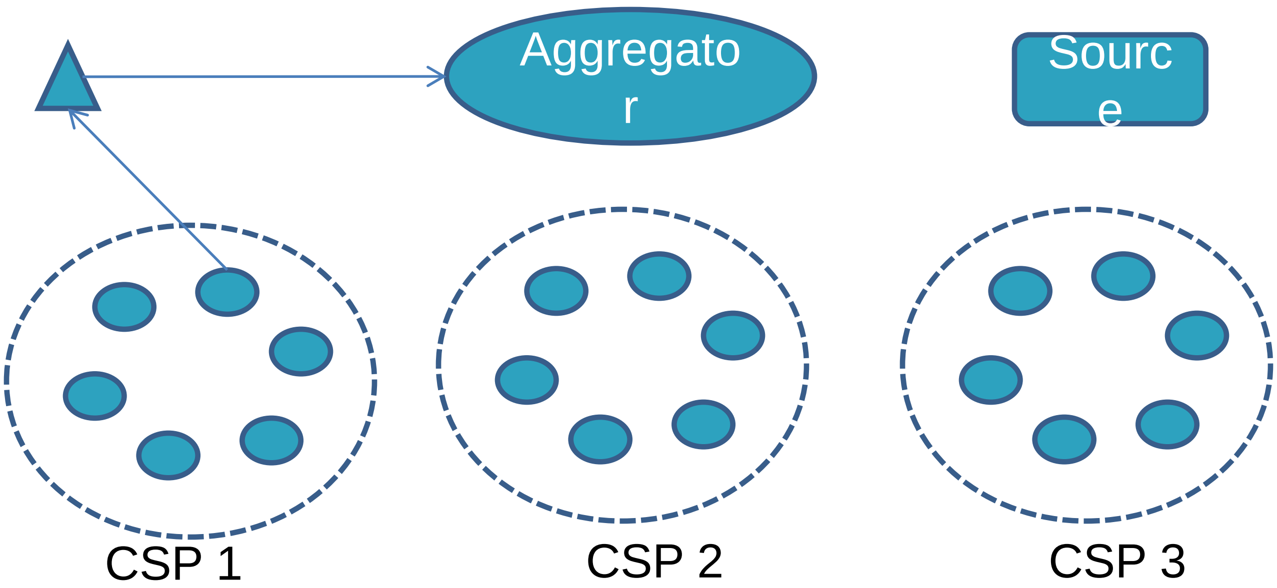
-- PeerApp

Why Data Lockers?

- Initially motivated by P2P CDN
  - Advantages
    - Highly-scalable
    - Robust
    - Space for innovation
      - Many novel techniques
      - Many players with novel ideas
  - Problems
    - Low network efficiency
    - High churns

Additional Use Cases

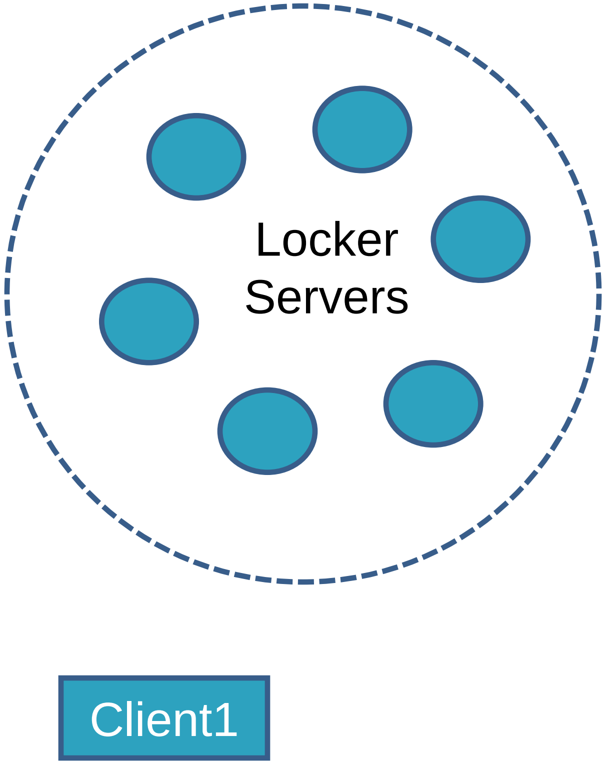
- [**Use Case II: Global CDN by Aggregation**; CDI] Can an aggregator build an Akamai-like global CDN utilizing multiple CSPs?



- [**Use Case III Video conference** (i.e., UGC)]
  - Can a video conferencing application (e.g., iPhone video) utilize CSPs to distribute video from one participant to multiple other participants?)

Design: Data Write

- Q: How to write into CSP?
- [DirectWrite]
  - Client writes into specific CSP server (cluster) store
    - Still allow DNS to direct to preferred server by CSP
  - Clients provide replication/ request routing
- [IndirectWrite]
  - ~~[RW-IndirectPull]~~
    - Client maintains a source
      - Publishes source location to CSP
    - CSP provides internal caching, replication and request routing among internal caches
    - CSP pulls from source when source data first requested
  - [RW-IndirectStaging]
    - Variation of [RW-Pull], client uploads to a staging service



Content Management

- Account holder can list keys in its own account, delete keys; keys have expiration time

- Not provided: listing of content at aggregation levels (management can have so)
  - Distributed indexing implement by Application
  - [Considering the possibility of Special account (e.g., public account)]

Why Resource Control: Robust Against Selfish Behaviors

- P2P systems depend on user contributions

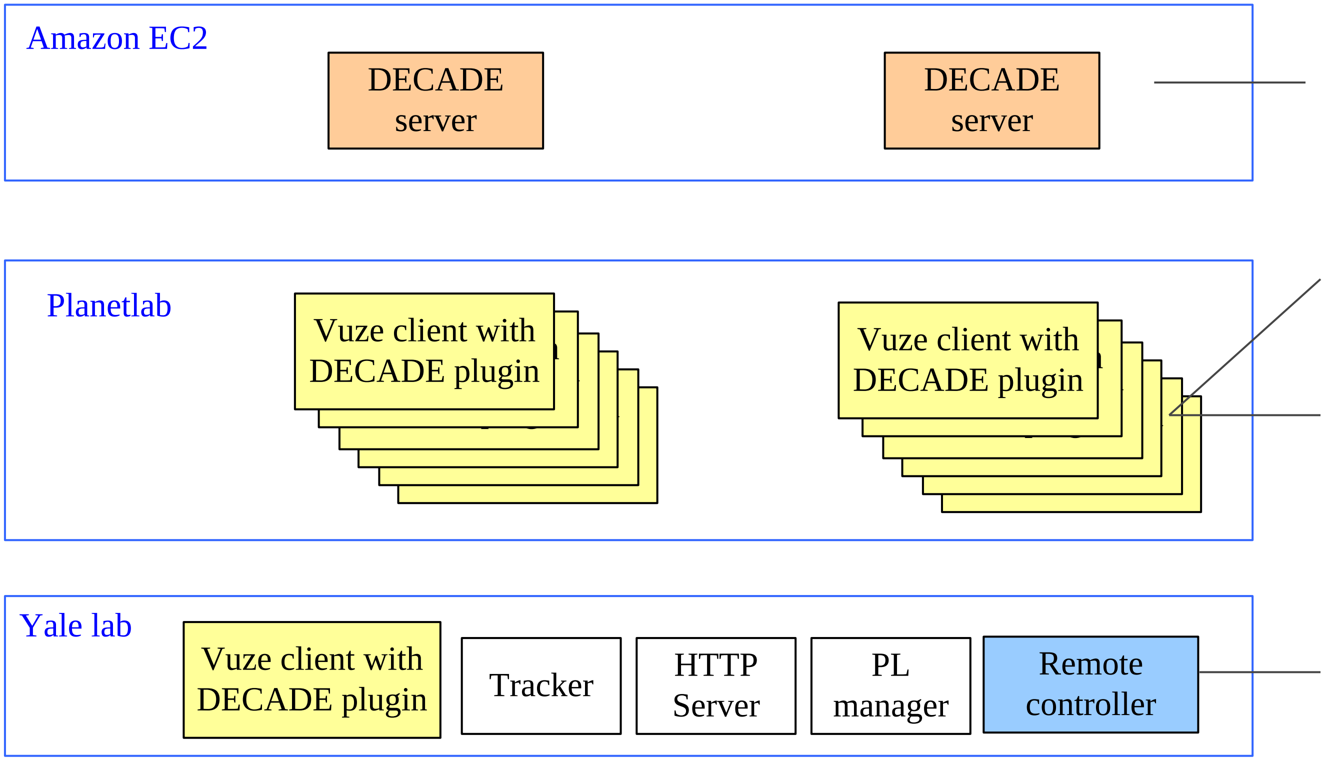
- Non-contributing users can be a serious problem
  - 70% of Gnutella users share no files and nearly 50% of all responses are returned by the top 1% of sharing hosts

- BW resource control is a major mechanism to design incentives and handle selfish behaviors
  - BitTorrent Tit-for-Tat
    - Attacked by BitTyrant
  - Provable Proportional Sharing [STOC'07; SIGCOMM'08]

Why Resource Control: Robust Against DoS Attacks

- A recent study [IMC'08] showed how to attack the Akamai streaming servers due to sharing of server bandwidth but no isolation
  - "We demonstrate that it is possible to impact arbitrary customers' streams in arbitrary network regions …" [IMC'08]

DECADE/Vuze Trial Setting



- DECADE server
- Vuze client: Open source P2P clients that can seed and/or download
- Decade plugin: Plugin to support DECADE function for Vuze.
- Remote controller: Shows all Vuze clients in UI and controls them to download the specific BitTorrent file, collects statistic data from Vuze clients.

- Remote controller cannot control Vuze clients to seed, so we use one Vuze client at Yale for seeding
- Tracker: Vuze client provides tracker capability, so we don't deploy our own tracker

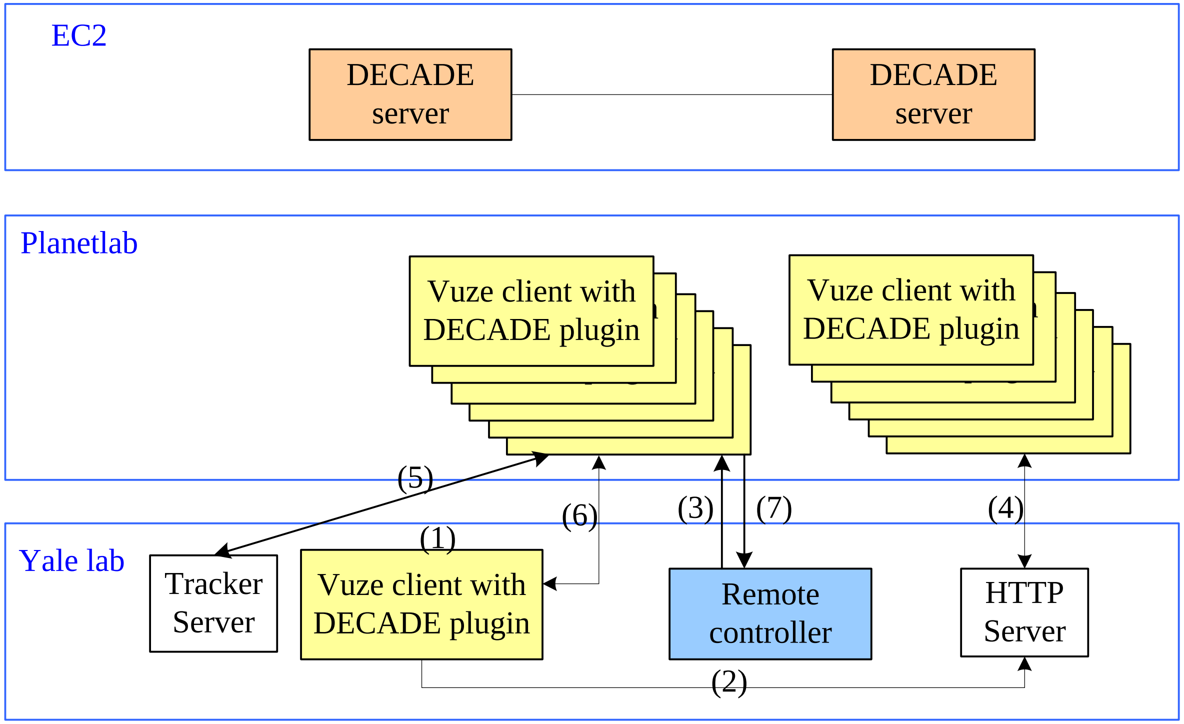
DECADE/Vuze Settings/Metrics

- Settings
  - 70 peers
  - Native: Peer Upload Capacity: 40 KBps
  - Data locker: Server Capacity: 40 KBps * 70
- Performance metrics
  - Client upload bandwidth
  - System resource efficiency: fraction of total network BW used

DECADE/Vuze Hardware and Software Specification

	Hardware Environment	Software Environment
DECADE server	• Server in EC2	• Ubuntu • DECADE server software version
HTTP server	• Server at Yale	• Windows 2003 Server • Tomcat
Tracker server	• Server at Yale	• Windows 2003 Server • Vuze client
Vuze client with DECADE plugin	• Download clients: Virtual machine at Planetlab • Seed client: at Yale	• Vuze client for Windows & Linux • JRE 1.6 • DECADE plugin software version
Remote controller	• Server at Yale	• Windows 2003 server • JRE 1.6 • Remote controller software version

Test Steps: Native Vuze



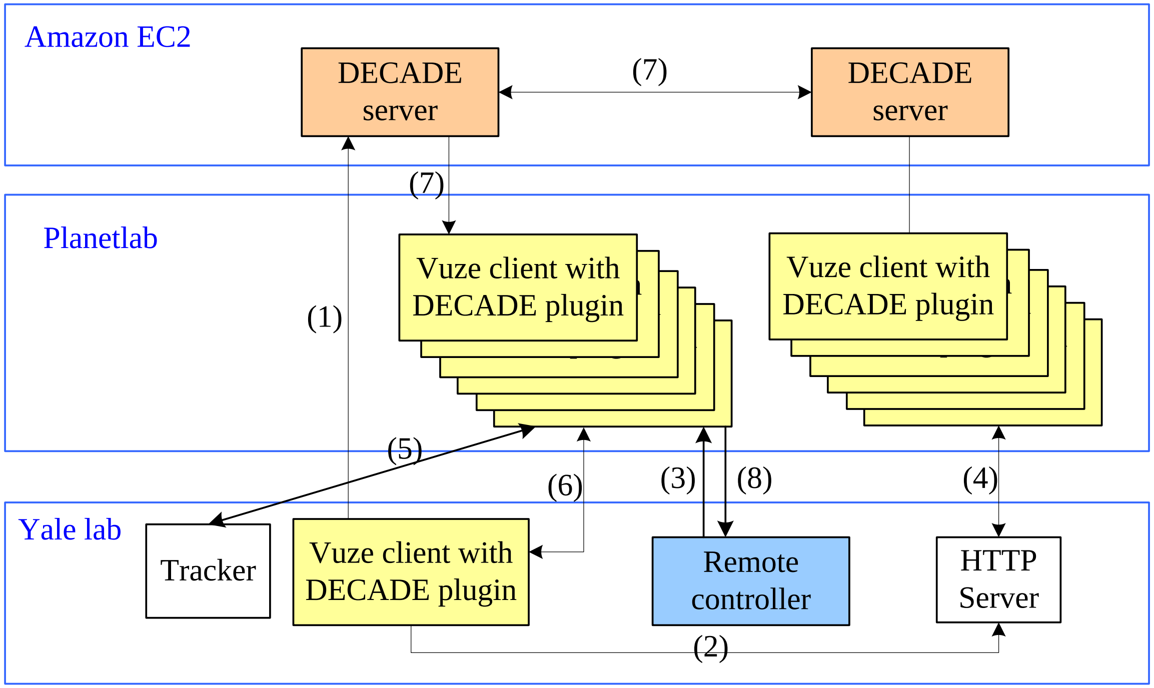
Test steps:

1. The Vuze client at Yale seeds
2. Manually upload the BitTorrent file to HTTP server
3. Remote controller starts up Vuze clients in PlanetLab
4. Vuze clients at Planetlab fetch BitTorrent file from HTTP server
5. Vuze clients at PlanetLab receives peer list from tracker server
6. Vuze clients at PlanetLab network send BT_Request to peers and get BT_Piece message from peers
7. All Vuze clients report statistics to remote controller

Precondition:

- Start remote controller and all Vuze clients
- Vuze clients register with remote controller; remote controller assigns the IP address of decade server

Test Steps: DECADE/Vuze



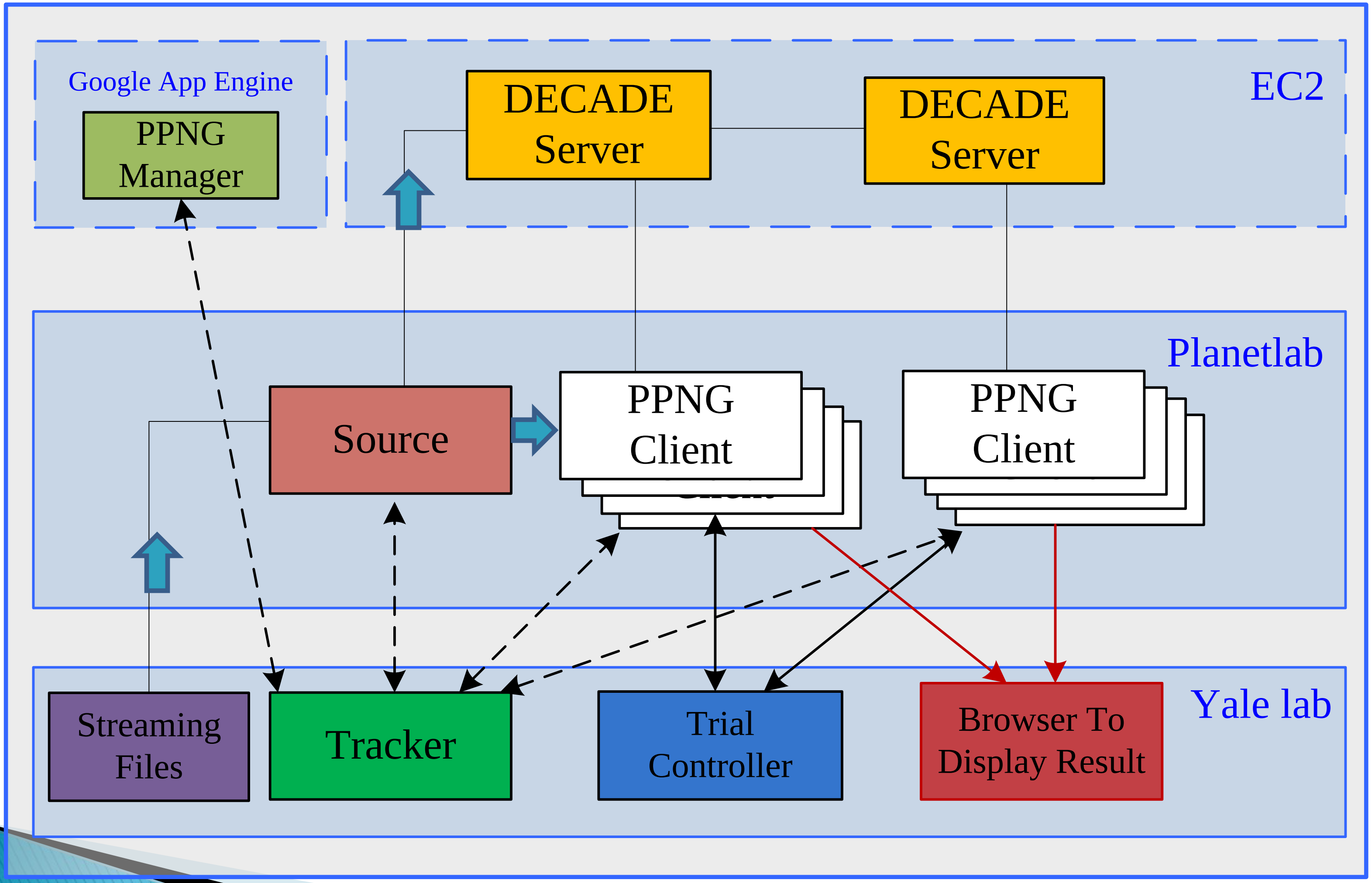
Precondition:

- Create DECADE server instances at EC2
- The rest is the same as Vuze/Native

Test steps:

1. The Vuze client at Yale seeds
2. Manually upload the BitTorrent file to HTTP server
3. Remote controller starts up Vuze clients at PlanetLab
4. Vuze clients at Planetlab fetch BitTorrent file from HTTP server
5. Vuze clients at PlanetLab receives peer list from tracker server
6. Vuze clients at PlanetLab send BT_request to peers and get Redirect messages
7. Vuze clients at PlanetLab download objects from DECADE servers
8. All Vuze clients report statistics to remote controller

DECADE/PPNG Trial Setting



DECADE/PPNG Setting

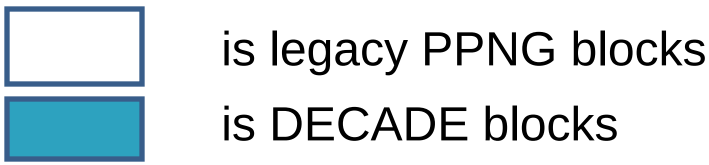
- Streaming Rate: 40 KBps
- Source Capacity: 200 KBps
- [Native] Peer upload capacity: 64 KBps
- [Data locker] Servers
  - 5 servers at different Amazon EC2 locations
  - Each server has capacity: 51.2 Mbps
  - P4P/ALTO Map to assign clients to close-by Data Locker servers

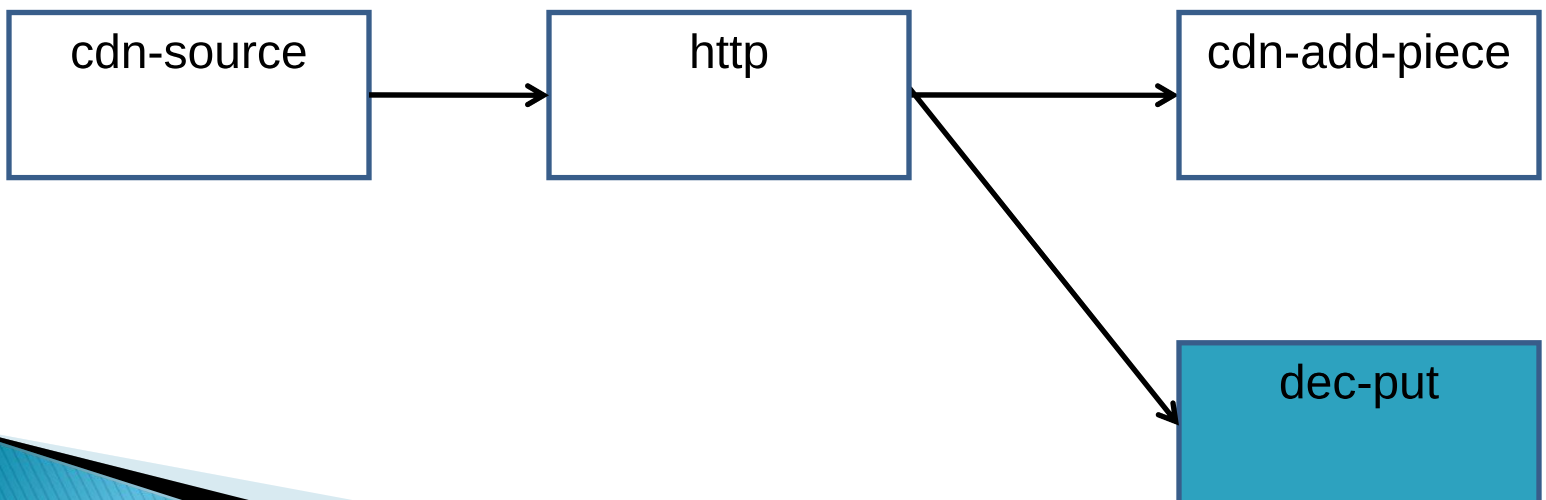
DECADE/PPNG Hardware and Software Specification

Components	Platform	Software
Decade Server	• Run in EC2, (US East, US West, EU, and Asia Pacific) • EC2 images	• OS: Ubuntu 10.04 • 3-party lib • EC2 control scripts
PPNG Client	• Run at PlanetLab	• PPNG Client • DECADE Client lib • 3-party lib
Tracker	• Run at Yale	• PPNG original tracker
Source	• Run at Yale	• PPNG original source • DECADE integration
Trial Controller	• Run at Yale	• Experiment control scripts • Log collecting scripts
Media Player Server	Run in PPNG Client	• Integrated in PPNG Client
GoogleMap Webpage	Run at Yale	• Google Map API • Runtime scripts
Online Statistic	Run at GoogleApp Engine	• Client log reporter • Log server

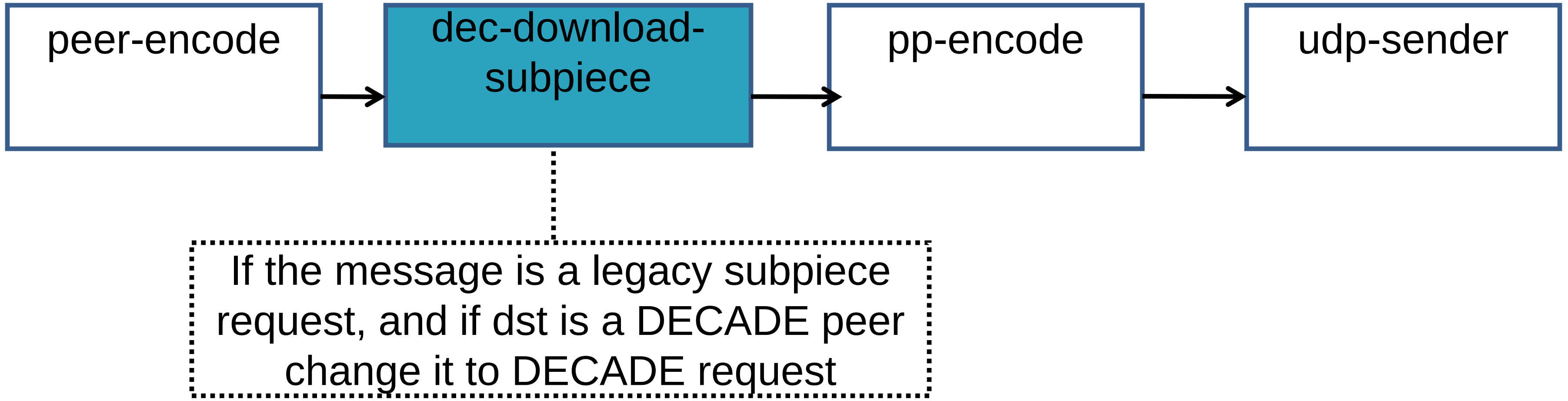
DECADE/PPNG Write/Put Block

Current version, only source explicitly put data into Data Locker

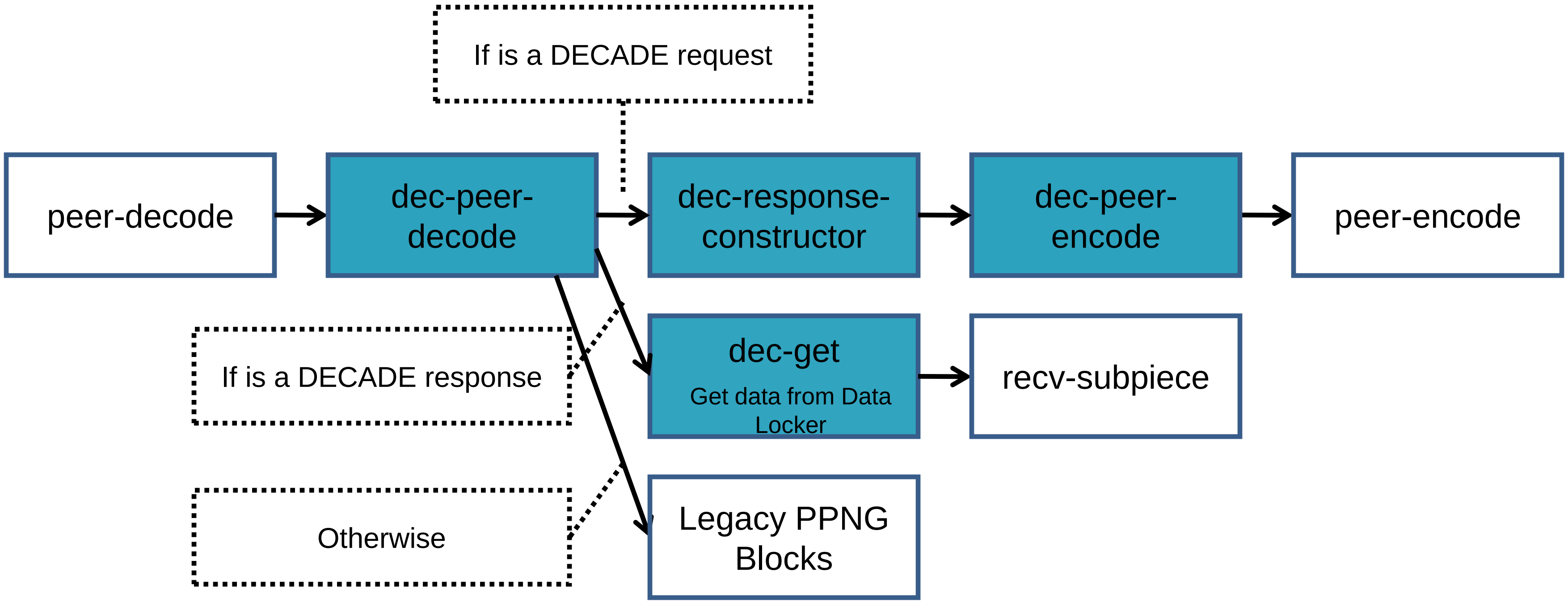




DECADE/PPNG Read/Request



DECADE/PPNG Response Block



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
    - $C_{a,b} = C^0_{a,b}$
  - else if (b supports DL but no locker)
    - $C_{ab} = C^0_{La,b}$
  - else // b supports DL and has locker Lb
    - $C_{ab} = C^0_{La,Lb}$

Resource Contention