WayFinder:Navigating and Sharing Information in a Decentralized World

Christopher Peery, Matias Cuenca, Richard P. Martin, Thu D. Nguyen

Department of Computer Science,
Rutgers University

http://www.panic-lab.rutgers.edu/Research/planetp

Background

Different goals for information-storage systems:

- A store of binary objects for general purpose use:
  - E.g. Unix FFS, LFS, NetApps …
- Records & relations describable using relational algebra:
  - E.g., Oracle, DB2, MySQL, Illustra …
- Global publishing and sharing:
  - E.g, Web, Napster (P2P), …
- Group-level sharing:
  - E.g. Wayfinder, Notes, groove …

Range of operations: consistency, durability, atomicity semantics.

Wayfinder initially targeting group-level sharing

- Migration path to a more generally usable storage service.

Technology trends increasing importance of sharing & publishing

Motivation

Two technology trends are fundamentally changing the computing landscape

- Increasing network connectivity (my dad is on the net)
  - Complex and dynamic sharing patterns
- Increasing performance/cost-size ratio
  - Multiple computing devices per person

Users must manage information across multiple domains of sharing across multiple devices

- Network connectivity increasing but not ubiquitous
  - Also unreliable: Isabel knocked out Thu's cable connection from home for 3 days
- Invariably, devices are used as caches of data for disconnected operation

Goals

Explore a file system that will ease the emerging data management problem in a medium-sized (100's-1000's) group context

Want to: share information (publish)

Read the paper I put out there

Want to find information published by others

Where's the paper by so-and-so on topic X?

Want storage function too! (I want my cake and eat it too)

E.g., don't manage local HTTP space separately from FS space

Want to remove the burden of information management across devices

Don't force users to remember where the latest is

Additional constraints:

Users have multiple devices

Lessons from the Web

Decentralized control for sharing

Complex and dynamic sharing patterns impossible to impose centralized control

Relax semantics to allow scale

Give up strict atomicity, durability, high availability.

E.g. namespace is partitioned: normal FS => stop, FSCK

Web => view whatever portion of namespace is currently reachable

Need both directory-based *and* content-based addressing

Directories: Yahoo, Dmoz, etc.

Content search: Google, Ask Jeeves, etc.

WayFinder Abstractions

Merged local FS trees into a single global namespace

- Compare to Web, NFS "graft" model

First class content addressing

- Semantic directories

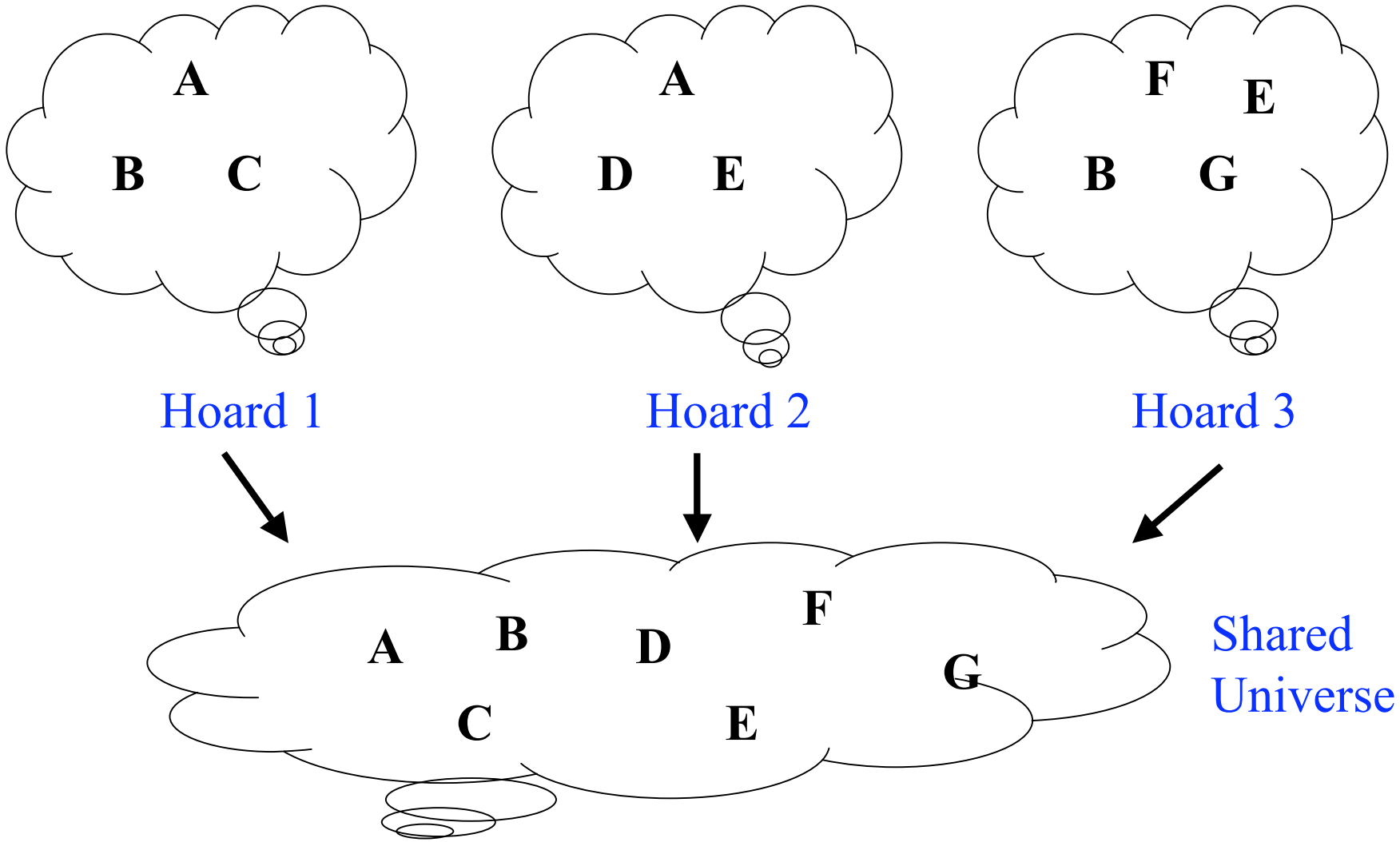
Probabilistic durability and availability of files

- Allows system to scale back junk as function of free space
- When you have too much space, you keep lots of junk

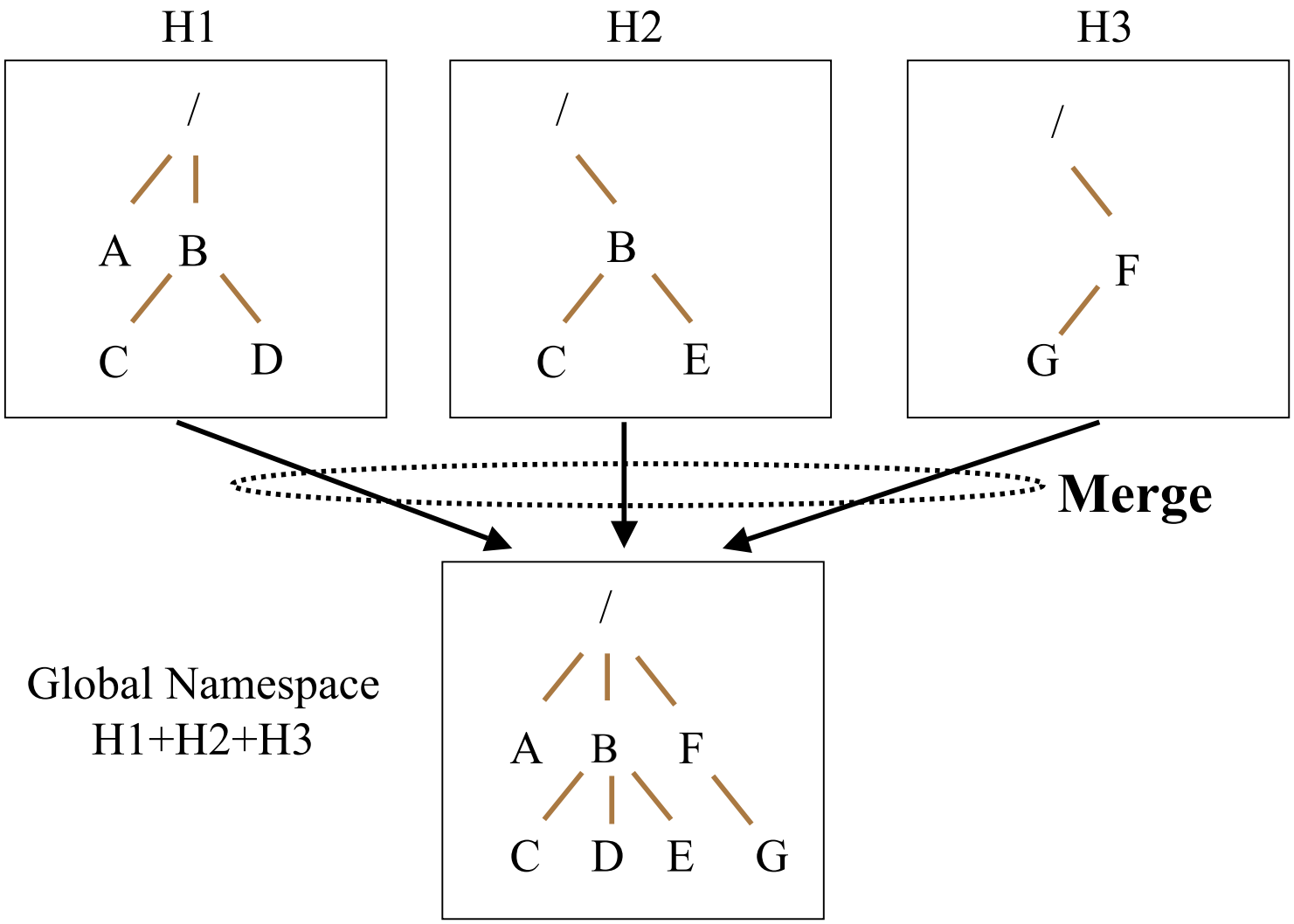
Group-Wise Hoarding Model

- Allow users to specify a set of devices and content
- This content is actively synchronized across the set devices

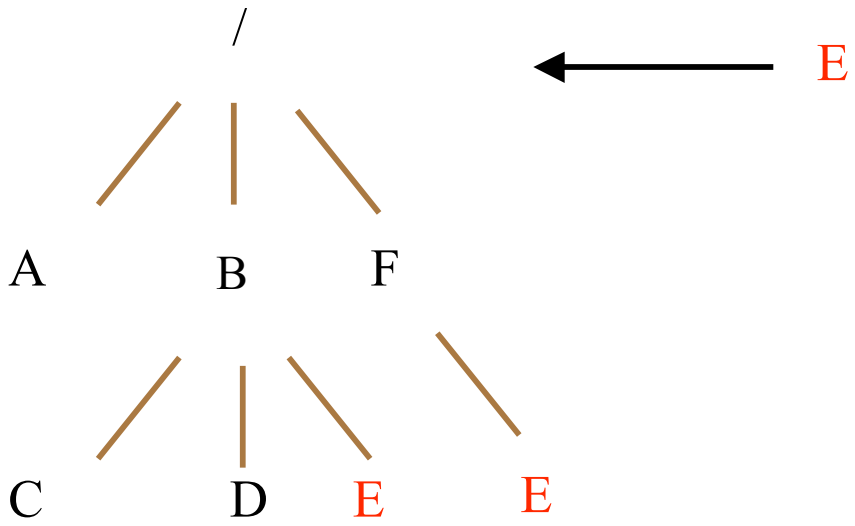
Group Sharing



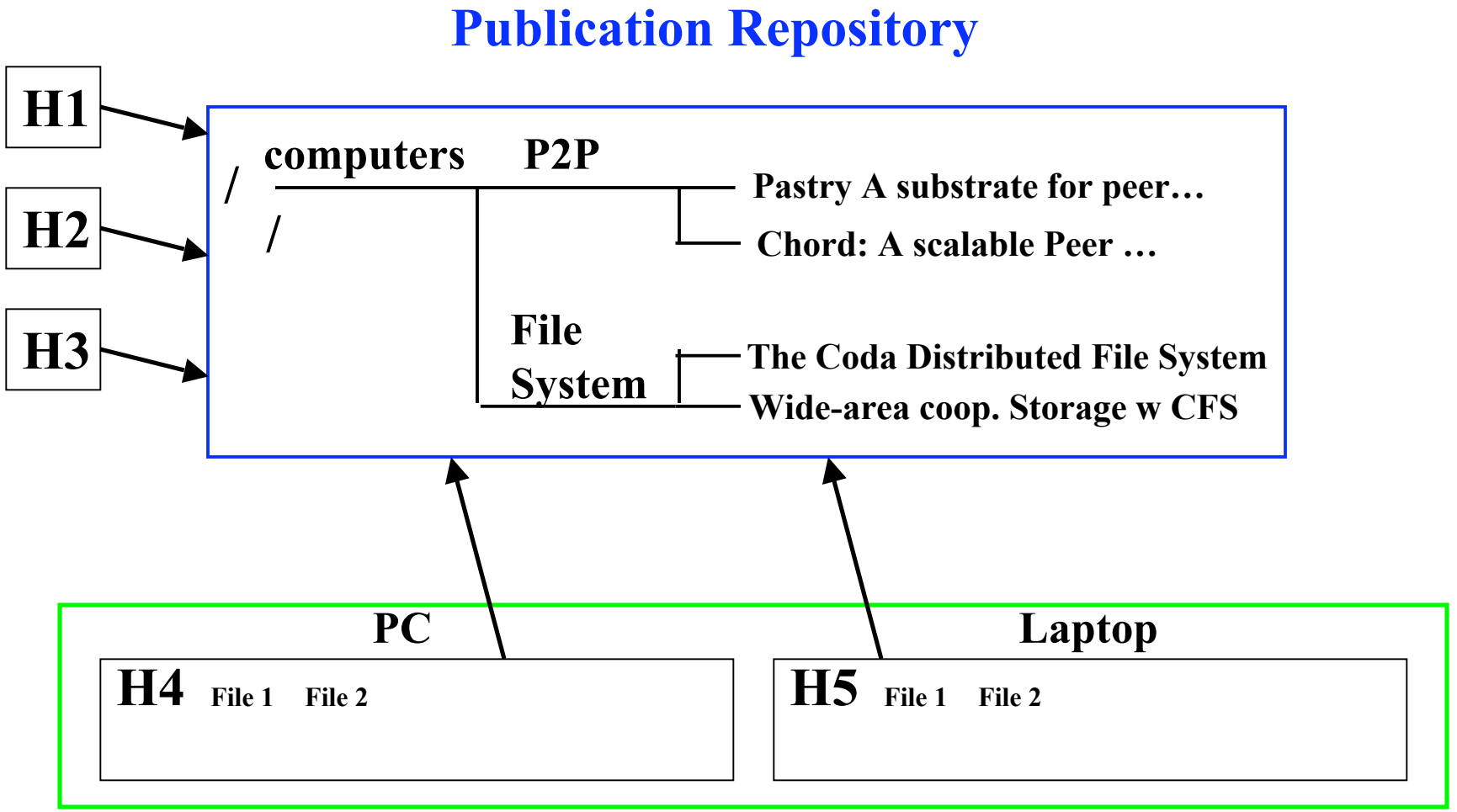
Namespace Model



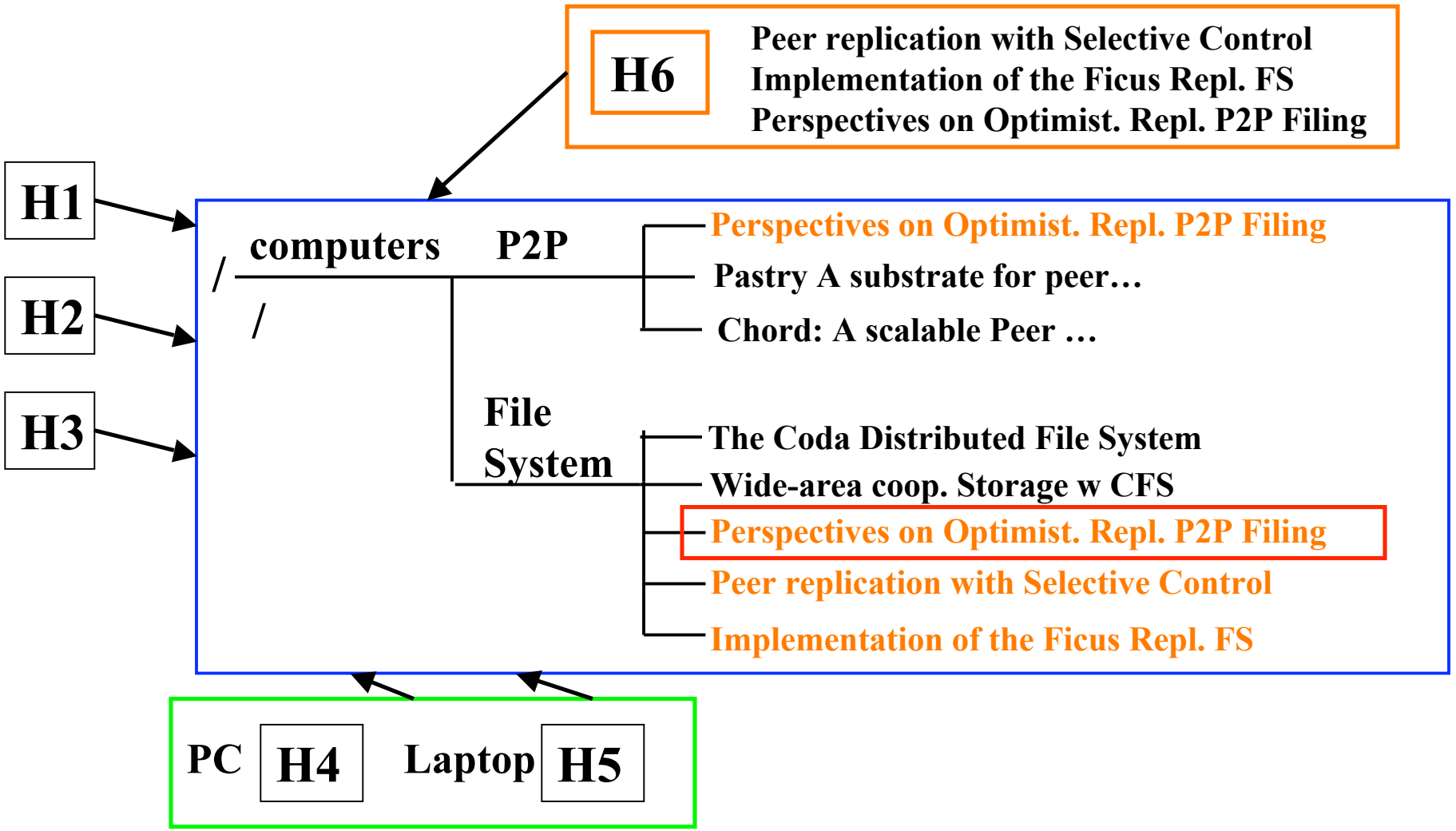
"Automatic" Content-Based Organization



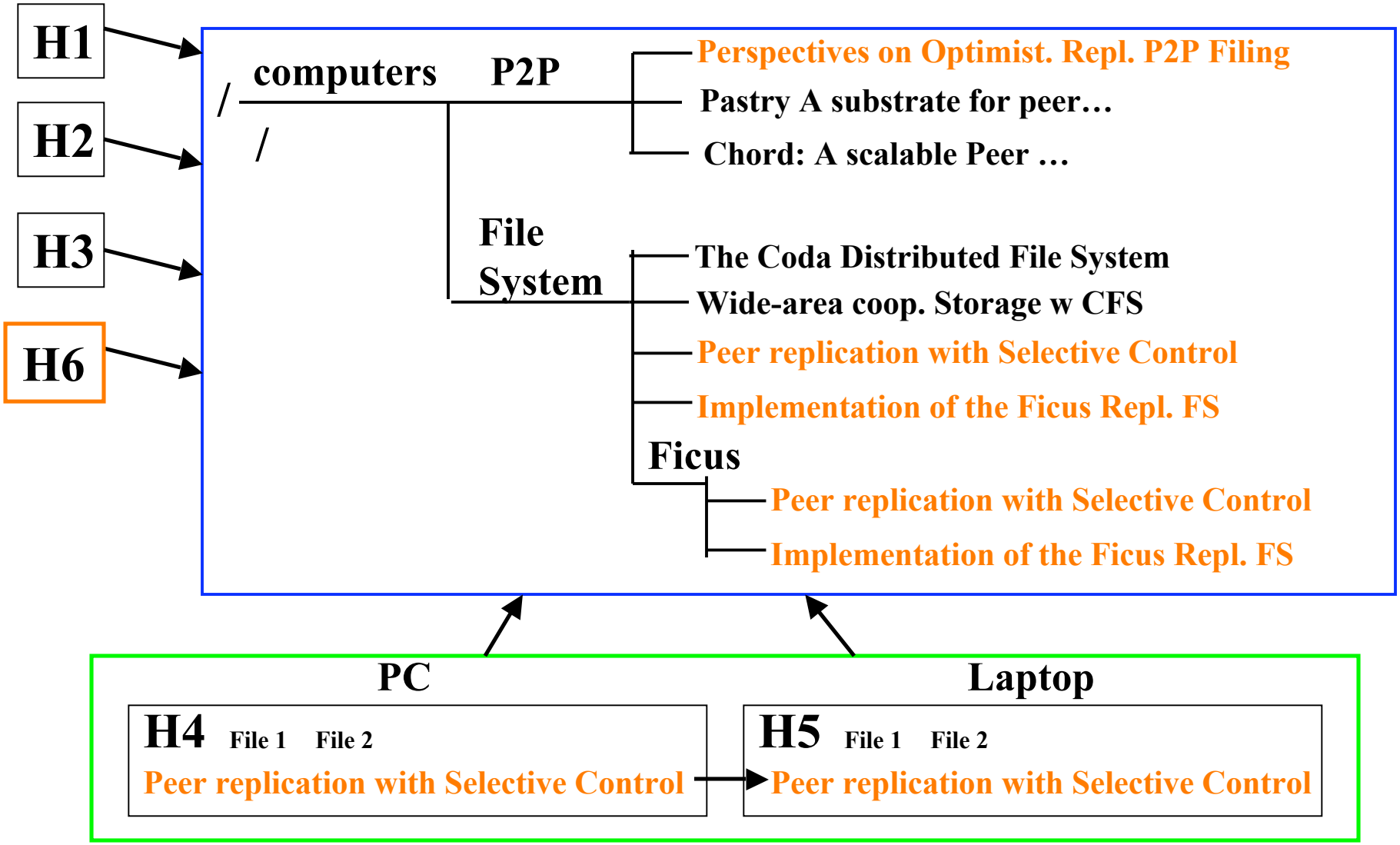
Motivating Example



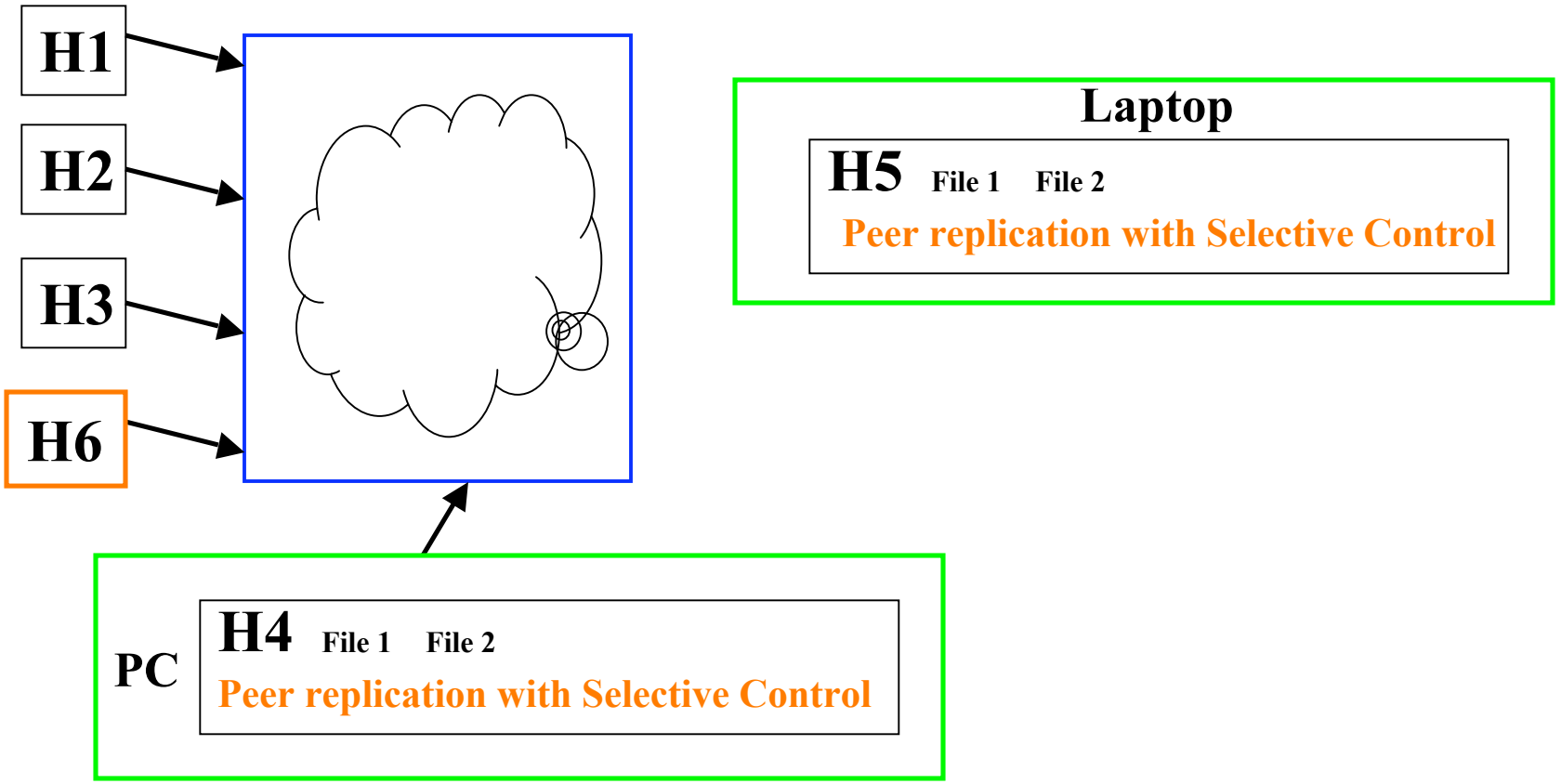
Motivating Example



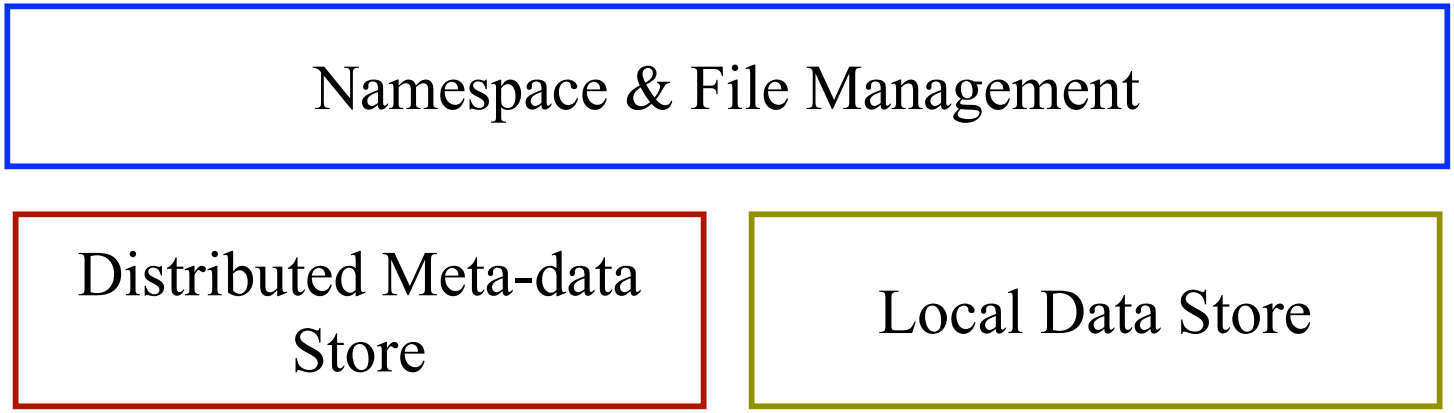
Motivating Example



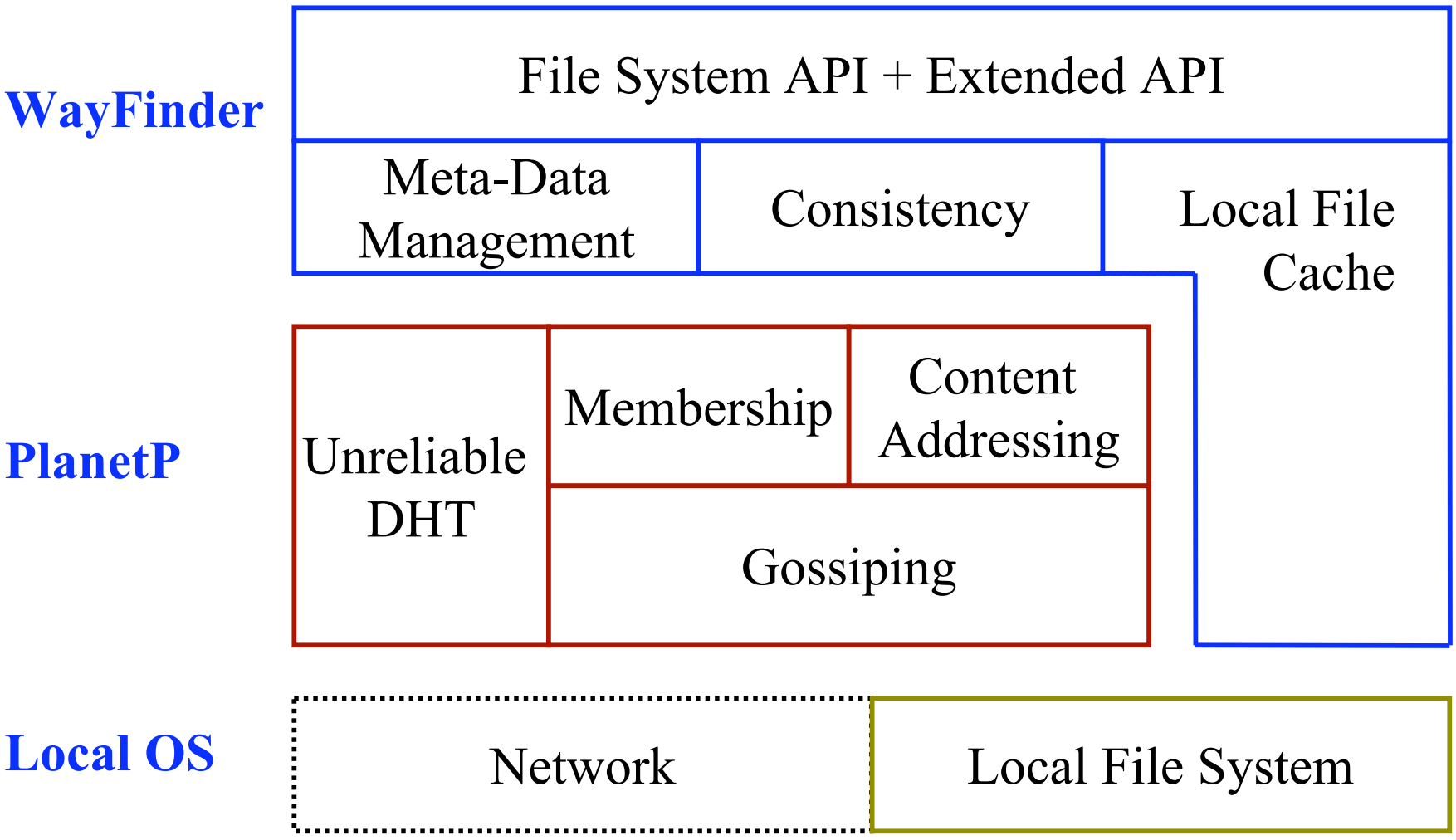
Motivating Example



High-Level Architecture



High-Level Architecture



PlanetP

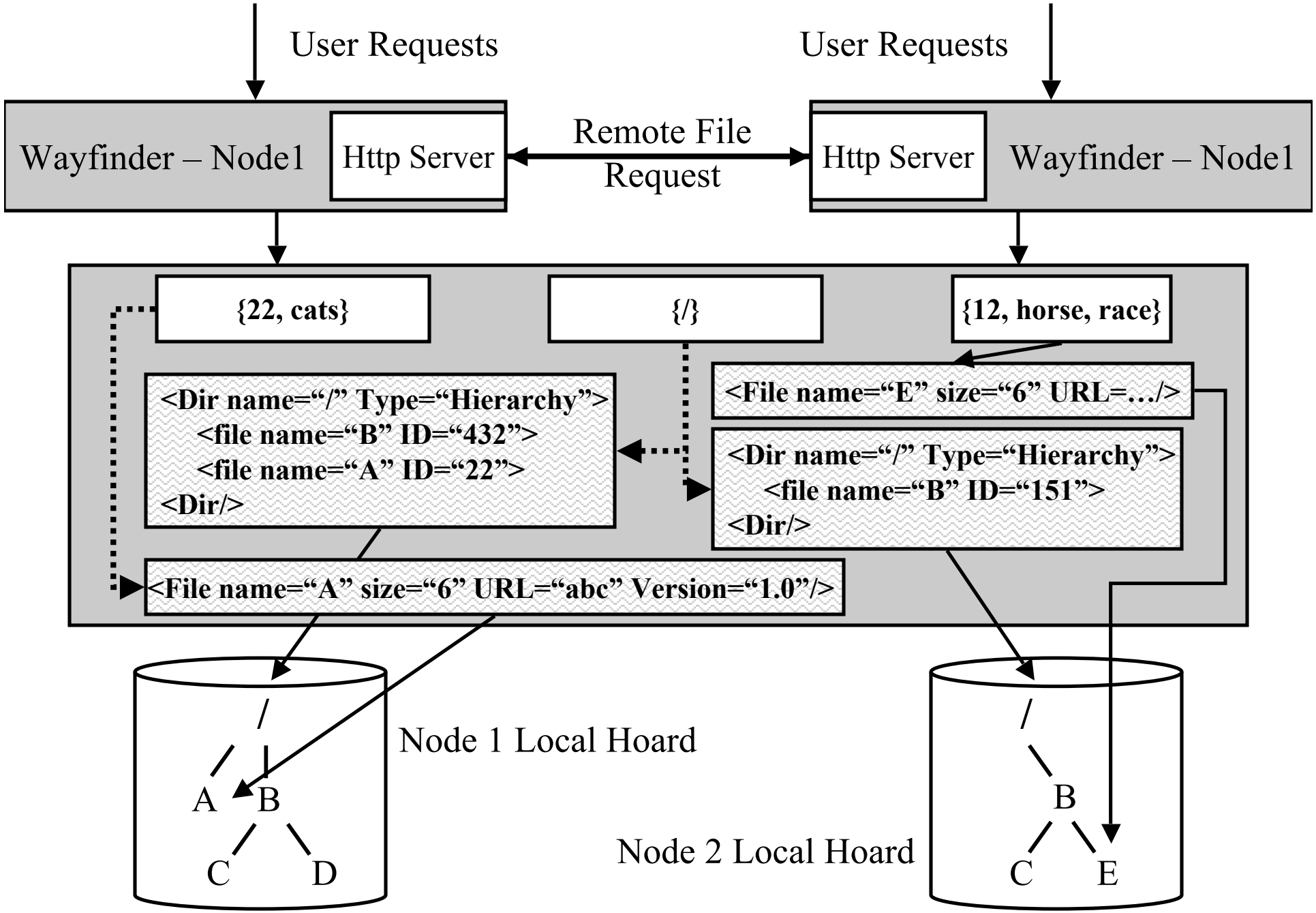
Infrastructure for building content addressable information sharing P2P systems

Major components

- DHT: key-based distributed object look-up similar to CHORD
- Global Membership directory: who's currently on-line
- Global/local index: efficient content search
  - Global index: 1 Bloom filter for each hoard giving an approximate "t□ h" mapping
  - Local index: normal inverted index
- Global data structures kept loosely synchronized using gossiping

Publish/subscribe usage model

- Shared objects mostly take form of XML snippets


User Requests
User Requests
Wayfinder – Node1
Http Server
Remote File Request
Http Server
Wayfinder – Node1
{22, cats}
{/}
{12, horse, race}
<Dir name="/" Type="Hierarchy">
<file name="B" ID="432">
<file name="A" ID="22">
<Dir/>
<File name="E" size="6" URL=…/>
<Dir name="/" Type="Hierarchy">
<file name="B" ID="151">
<Dir/>
<File name="A" size="6" URL="abc" Version="1.0"/>
/
A
B
C
D
Node 1 Local Hoard
Node 2 Local Hoard
/
B
C
E

Namespace Construction

Creating a accurate directory or file state may be expensive

- Worst case you may need to contact the entire community
- E.g., constructing the view for "/" or a the state of popular file

Cache views and file in PlanetP's DHT

- The first node that browses a directory will create a view the hard way but caches the view for fast subsequent accesses
  - Same for files
- Processed state is stored
- Cached views are discarded periodically

DHT only used to store soft state

- DHT "impossible" to maintain in face of unreliable nodes & network
- E.g., group of 1000 sharing 100 GB stored in DHT with Gnutella observed availability => 4GB data movement per node per day

Semantic Directories

Semantic Directories provide content-based organization

- They are directories whose names are treated as content queries
- Populated by files whose waynodes are returned as results
  - The scope of a query is defined as the files located in the parent directory
  - May be nested to provide a simple conjunctive query language
  - e.g. /computer/P2P computer AND P2P

They may be used as normal directories

- The contents may be altered by removing or adding files
- Semantic directories are re-evaluated periodically (or when requested explicitly by the user)

Provide an easy means for adding and removing structure based on incoming/outgoing content

File Access

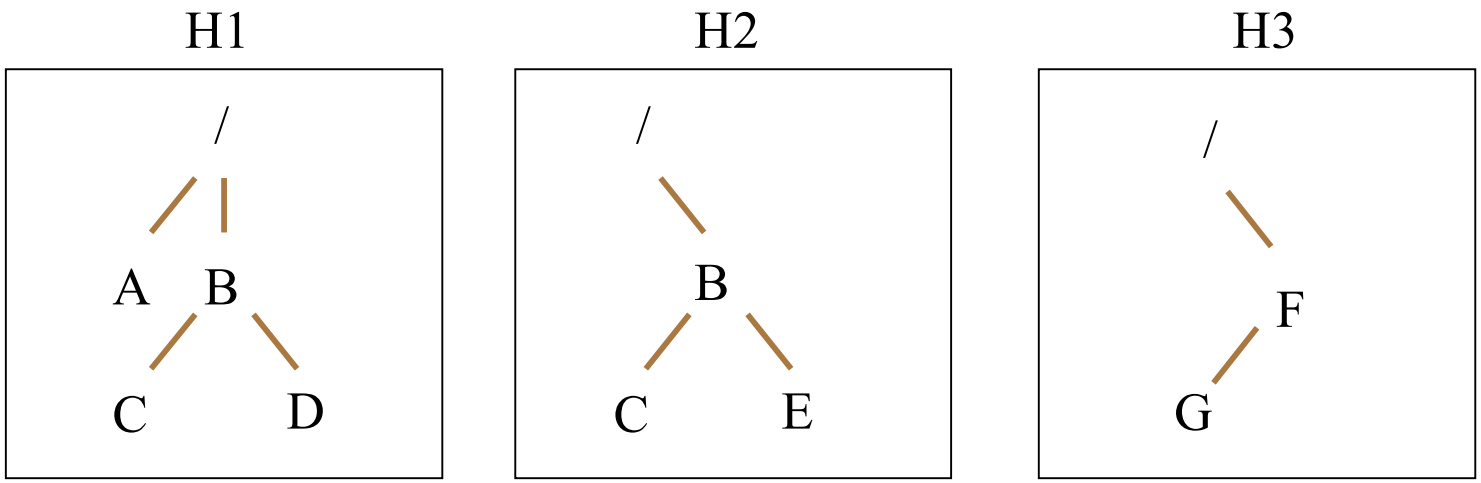
Accessing a file *F* requires a local copy of *F*

- Find a replica of the latest version and make a local copy
  - Querying PlanetP for waynode using file ID
  - Choose a waynode with latest version and retrieve using URL
  - The file's location is mirrored in local namespace (hoard)
- The local copy is republished as an additional replica

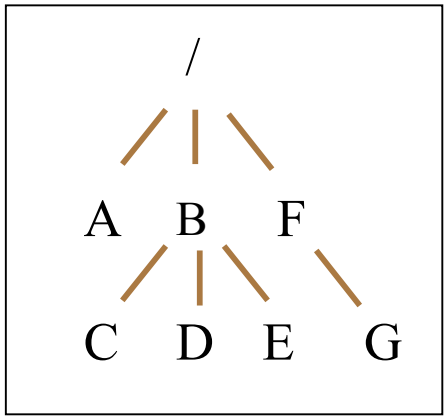
Updates

- Open-for-write/close creates a new version
  - Unique version identified by <node id, number>
- Writes encoded as diffs for efficient propagation
  - Can roll forward and backward

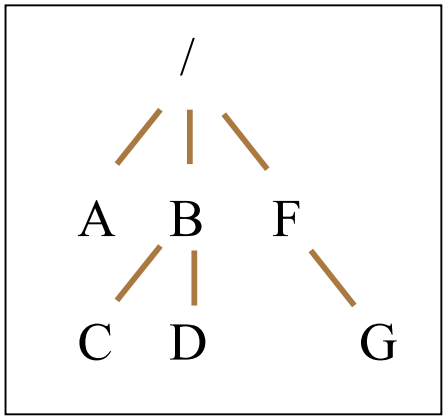
Partitioned & Disconnected Operation

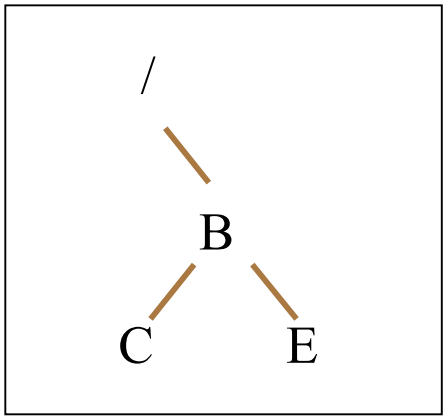


Global Namespace
H1+H2+H3



Partitioned Operation





Consistency Model

Single copy availability / Eventual Consistency

Wayfinder can not ensure that the latest version can be found

- Information maybe off-line
- Notification may be delayed by gossiping/rumoring
- Can even happen during when the full community is on-line

This is a problem for any system supporting partitioned operation

Wayfinder will attempt to find most to up-to-date data

Cached data can reduce window of vulnerability

- Will warn user if conflict detected at creation time
- Update cached views on changes if they exist

Consistency Model - Files

Files

Concurrent writes will lead to version conflict

Automatically resolved using a deterministic (but arbitrary) order of conflicting versions

User can choose to unroll and resolve conflict

The resolved version of the conflict becomes a new version

Directories

Name collisions are ok: each file uniquely identified by file ID

When hoarding an existing file, the hoard replica will inherit the existing ID

When creating a new file, assign new ID

Name conflicts resolved by differentiating on the IDs

Availability Model

Probabilistic availability model

- Each Node may independently make decision concerning files

Allow user to specify desired availability for files

- Try to achieve desired availability using autonomous replication (SRDS 2003)
  - Envision specifying coarse availability levels for directory trees
  - Can increase availability by introducing server-like hoards

Content may be *unavailable* because

- Hoards holding the desired content is off-line
- The last replica of a file is evicted
- Warn user when there is not enough space for desired availability

Current Availability Model

The ultimate goal is to have as much the namespace, as possible, visible at all times

The current model is based on an approach presented in SRDS

- Assumes connectivity to a single large community
- Node availability being with respect to this community

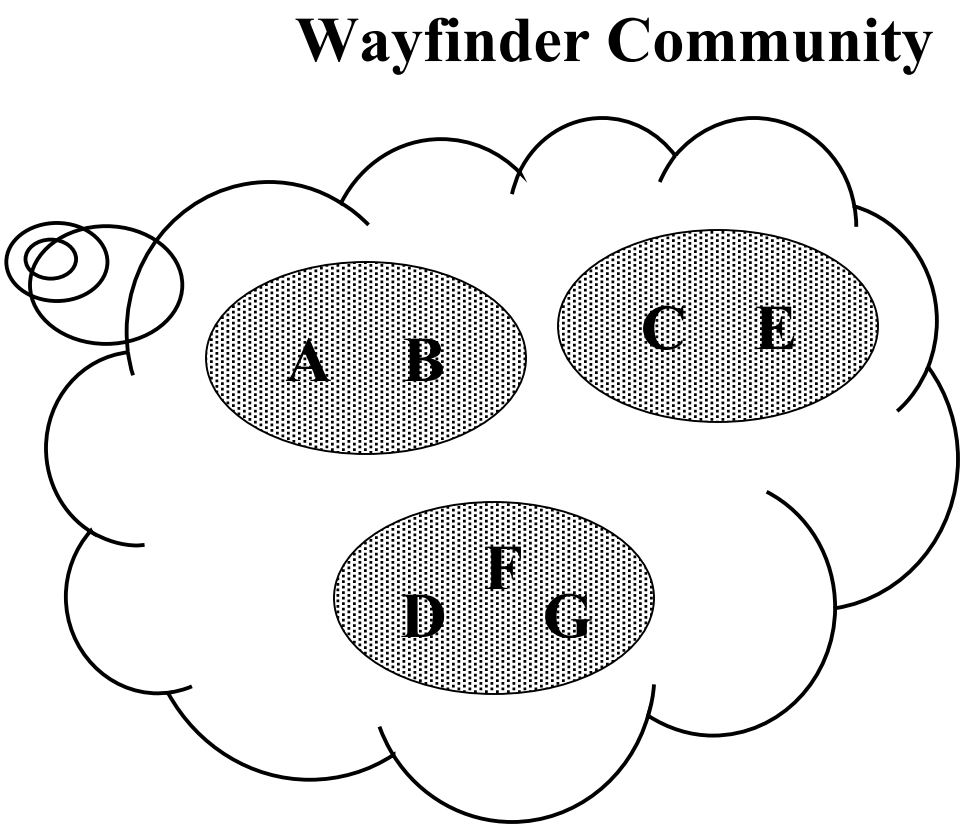
Wayfinder may not have this single large community

- Designate a subset of the community as a Core
- Track nodes availability with respect to this core, and replicate accordingly

There are problems with this approach

- The idea of the Core is counter-intuitive to wayfinder
- Availability measures will be may be very pessimistic
- Really does not get at the point of availability in wayfinder

Availability Model



We want to consider partitioned operation

Availability has been traditionally been achieve through hoarding/replication

Challenges:

Mobility between clusters

Insufficient Space

Node-Centric Availability measure

Unified Availability Model

Evaluation Plan

We've currently have a finished a working prototype

- Exports the NFS v2.0 interface + extended RMI API

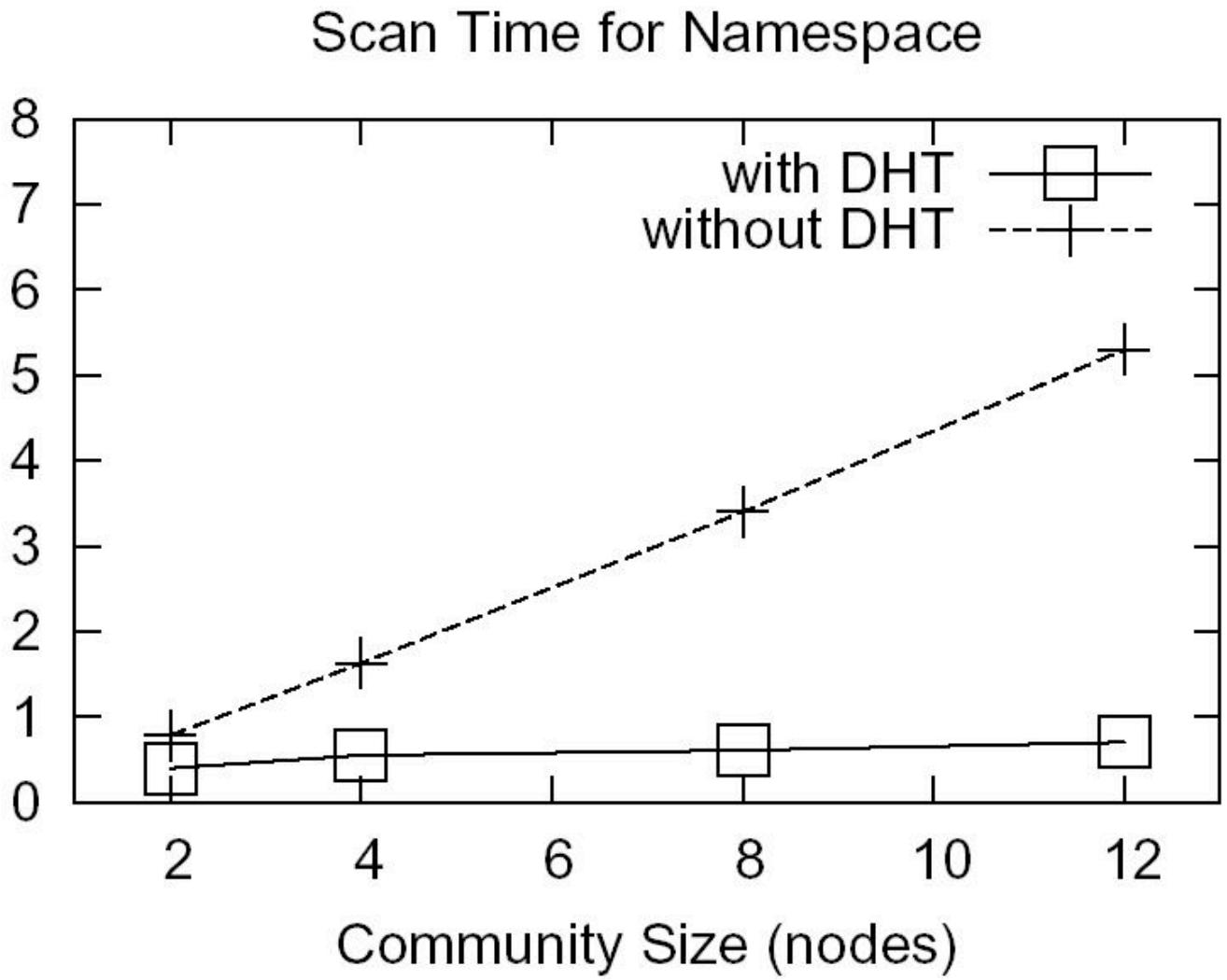
Evaluation:

- The cost of browsing using the global index and the DHT
  - The DHT should provide a relatively constant cost for directories as the community grows
- Determine the cost of running Wayfinder to access local files
- Measure the effect of DHT failures on performance
- The additional amount of network traffic sent because we use PlanetP

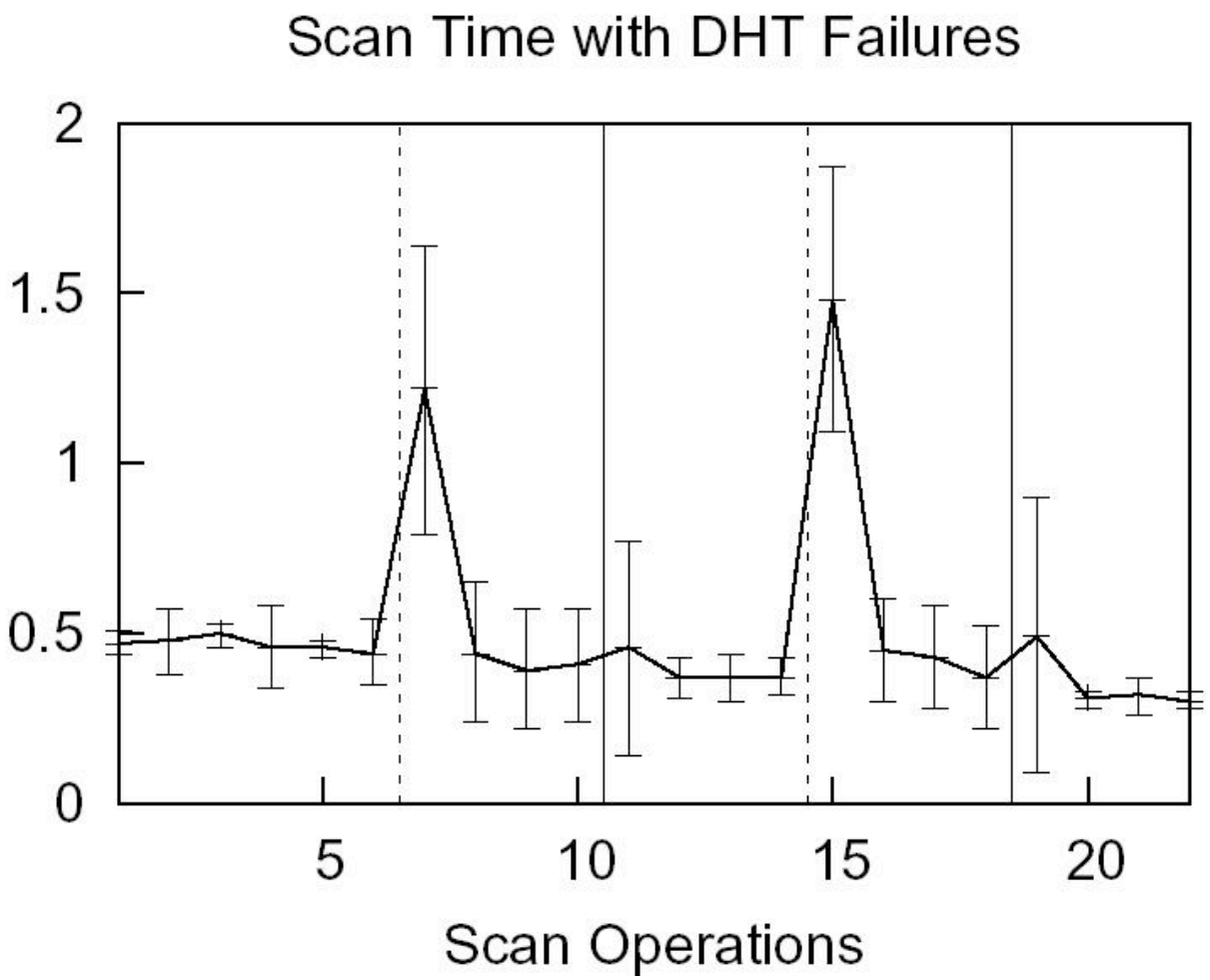
MAB

Modified Andrew Benchmark				
	Linux NFS	JNFSD	Wayfinder: 1 Node	Wayfinder: Worst Case
Ph. 1	0.02 s	0.04 s	0.04 s	0.10 s
Ph. 2	0.18 s	0.37 s	0.82 s	1.51 s
Ph. 3	1.03 s	0.82 s	0.85 s	1.08 s
Ph. 4	0.84 s	1.58 s	1.64 s	1.82 s
Ph. 5	2.09 s	3.13 s	3.30 s	3.49 s
Total	4.16 s	5.94 s	6.65 s	8.01 s

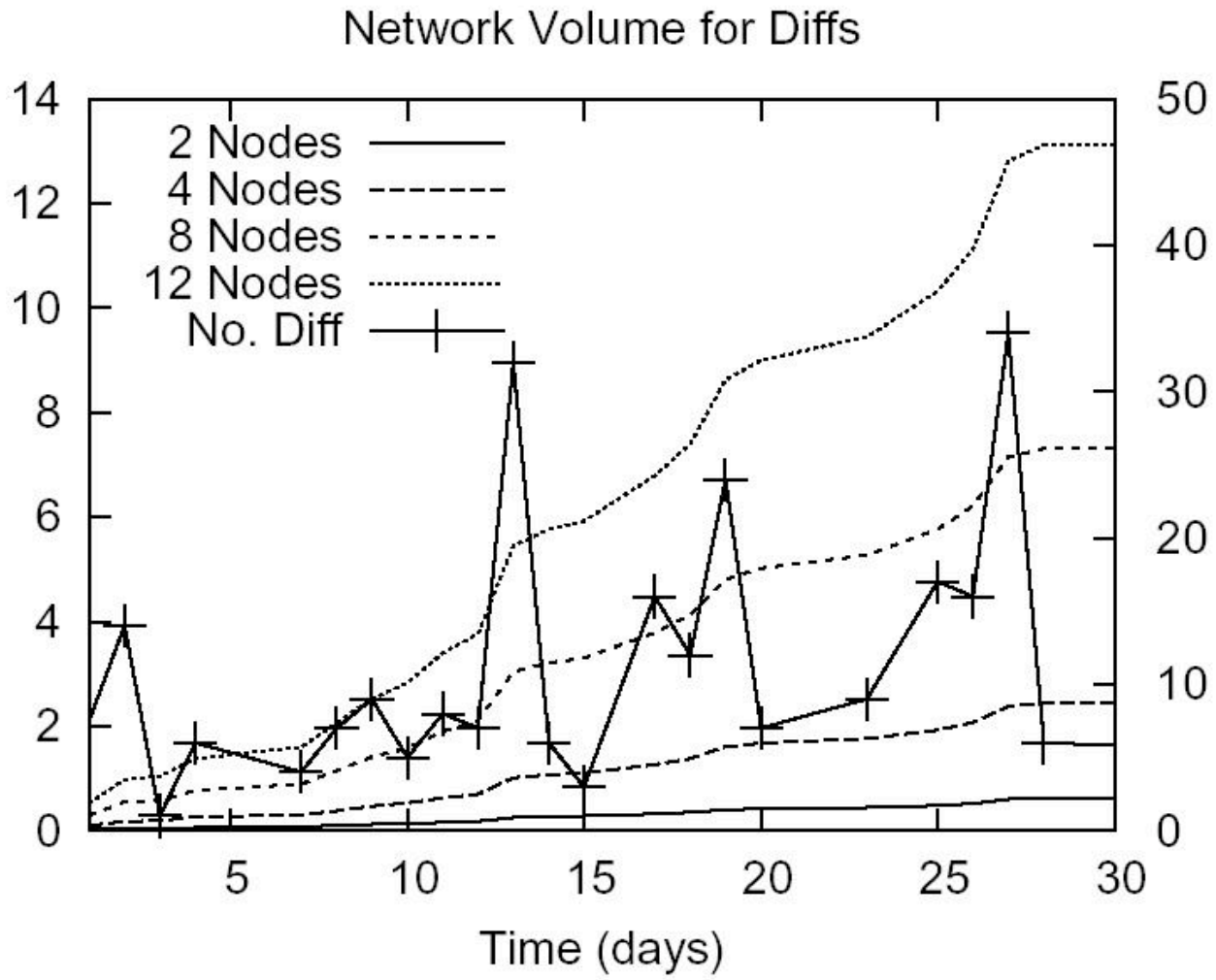
Scan Time


Scan Time for Namespace
with DHT
without DHT
0
1
2
3
4
5
6
7
8
2
4
6
8
10
12
Community Size (nodes)

DHT Robustness



Diff Network Volume



Future Work

Group-based hoarding

Availability Model

Actually use WayFinder

- PANIC publication repository
- PANIC workgroup sharing

Attempt to answer following questions:

1) Are users actually willing to categorizing importance of files?

2) Can users actually deal with probabilistic availability model?

3) Will users perceive WayFinder as the Google of group-based file sharing?

4) Can Wayfinder deliver useable availability and performance?

Security

The security model will seek to control writes

- Read access means you have a hoarded file
- Very hard to do revocation for read access
- For same reason, once write permission is given, granted for life

Attempt to control the application of diffs

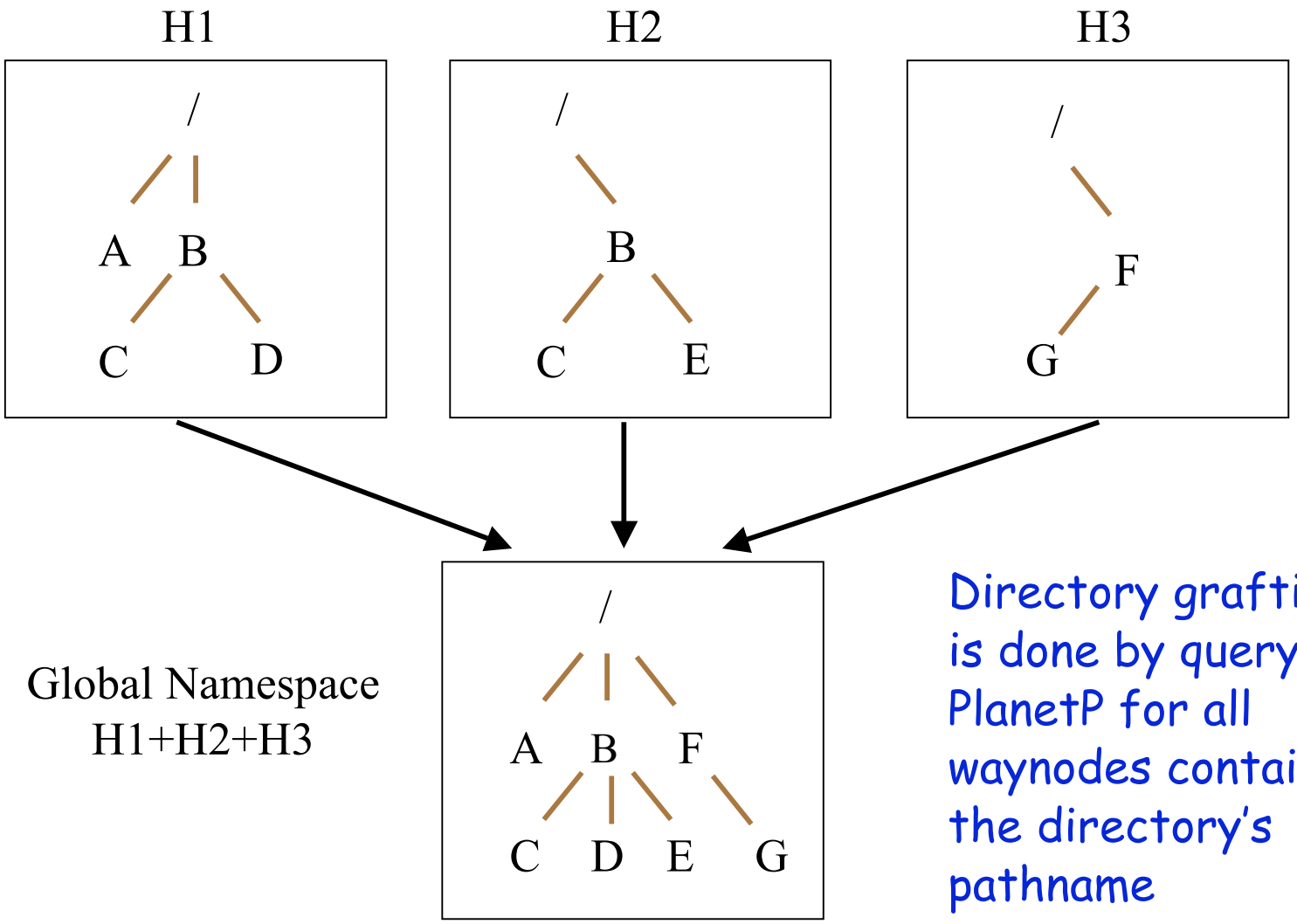
Any node can publish a diff

Only diffs from permitted nodes will be applied to files

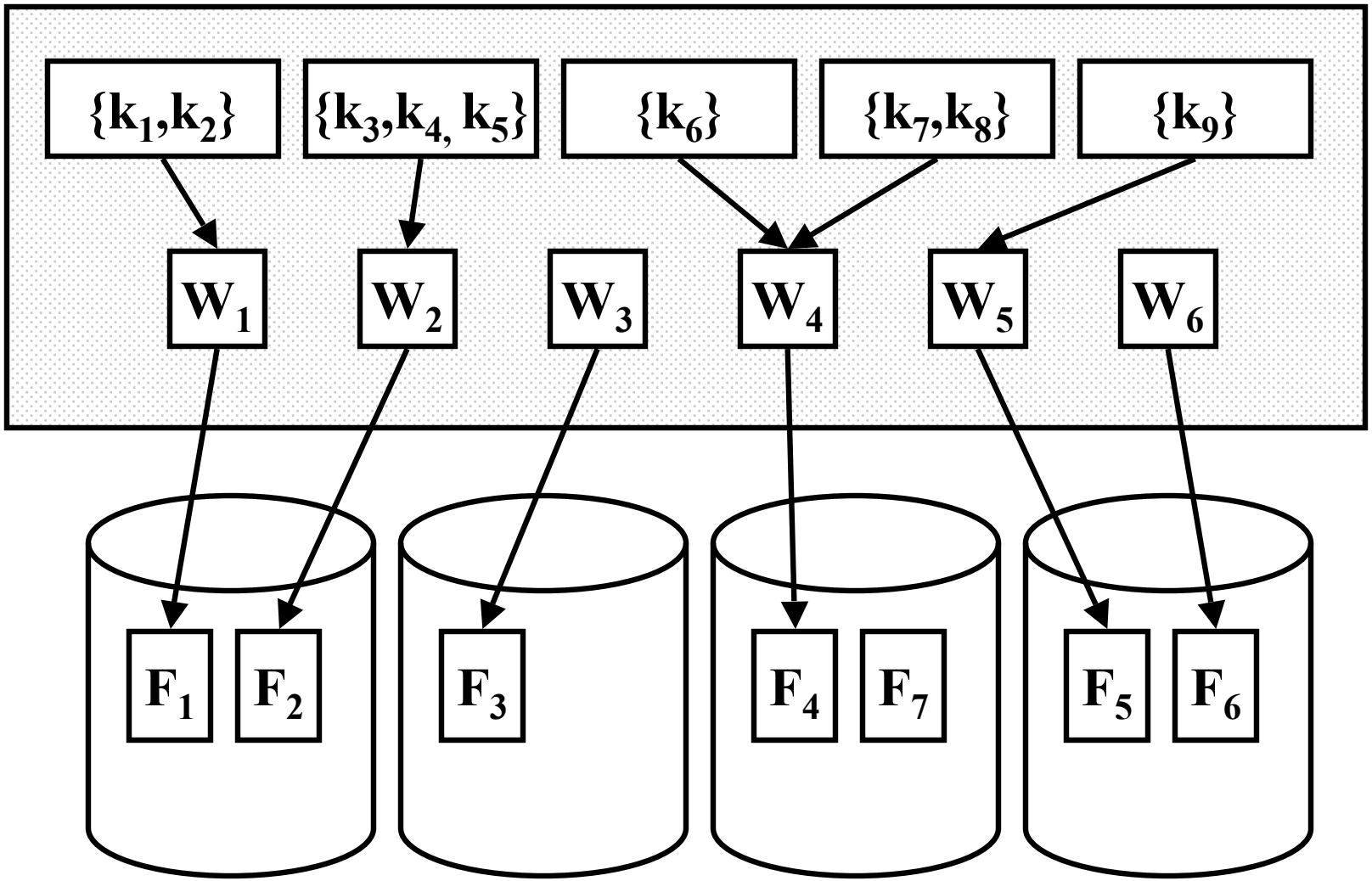
Security Framework

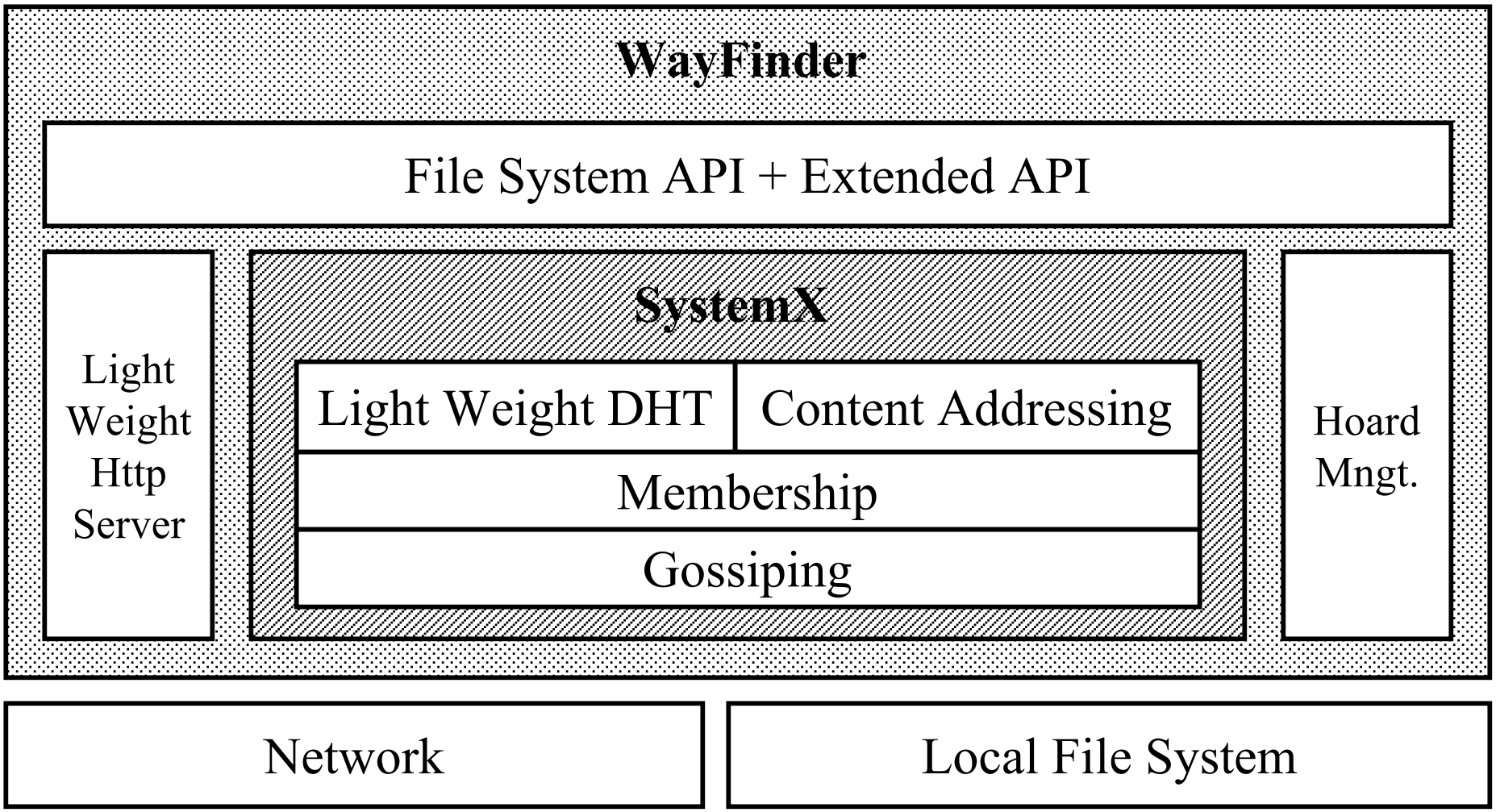
- Based on ACL and signatures.
- Files are "owned" by a user's key or a group key
- All diffs are signed and verfied against ACL.
- Anyone with write permission may alter ACL

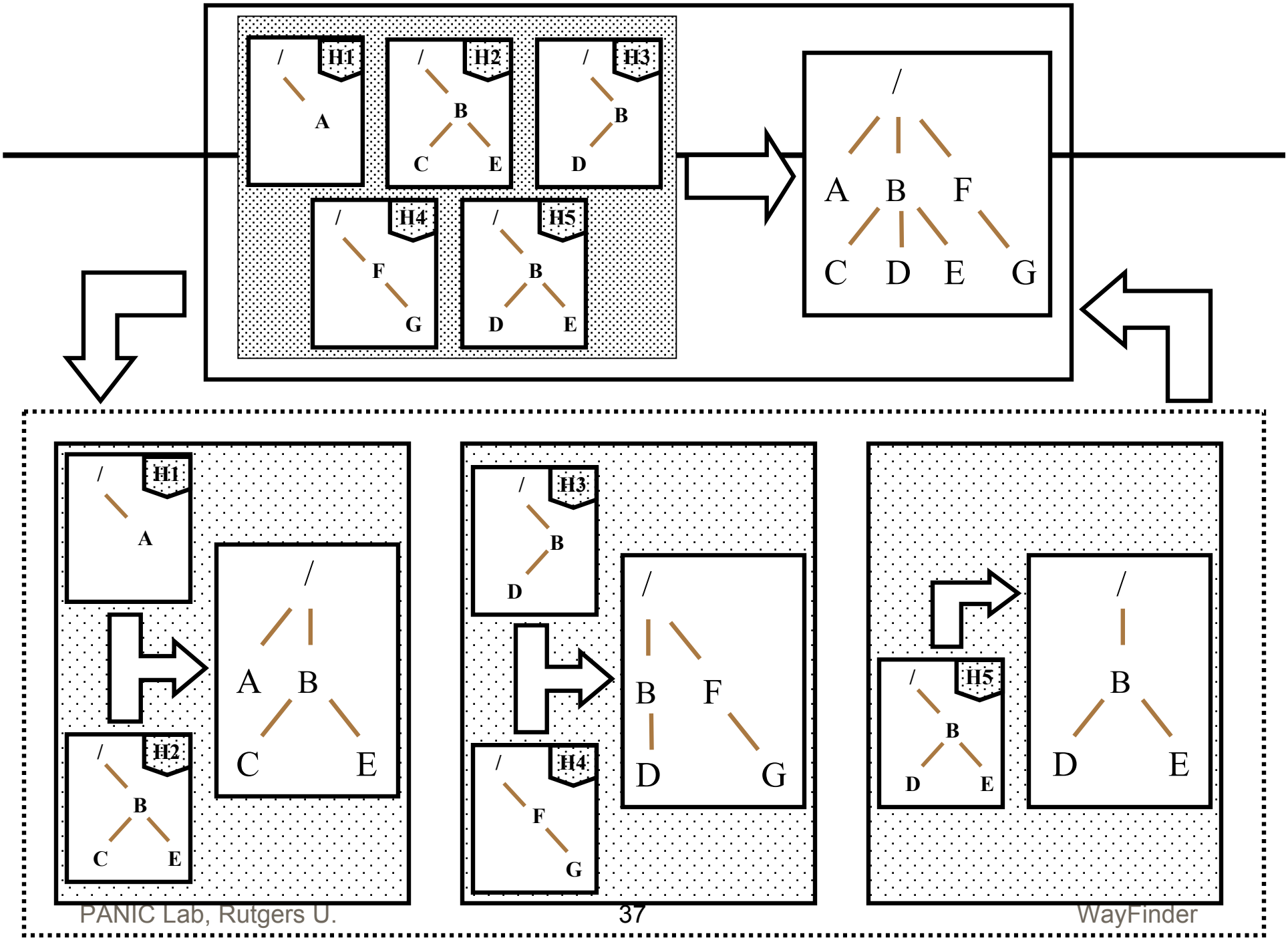
Directories REMOVE???????



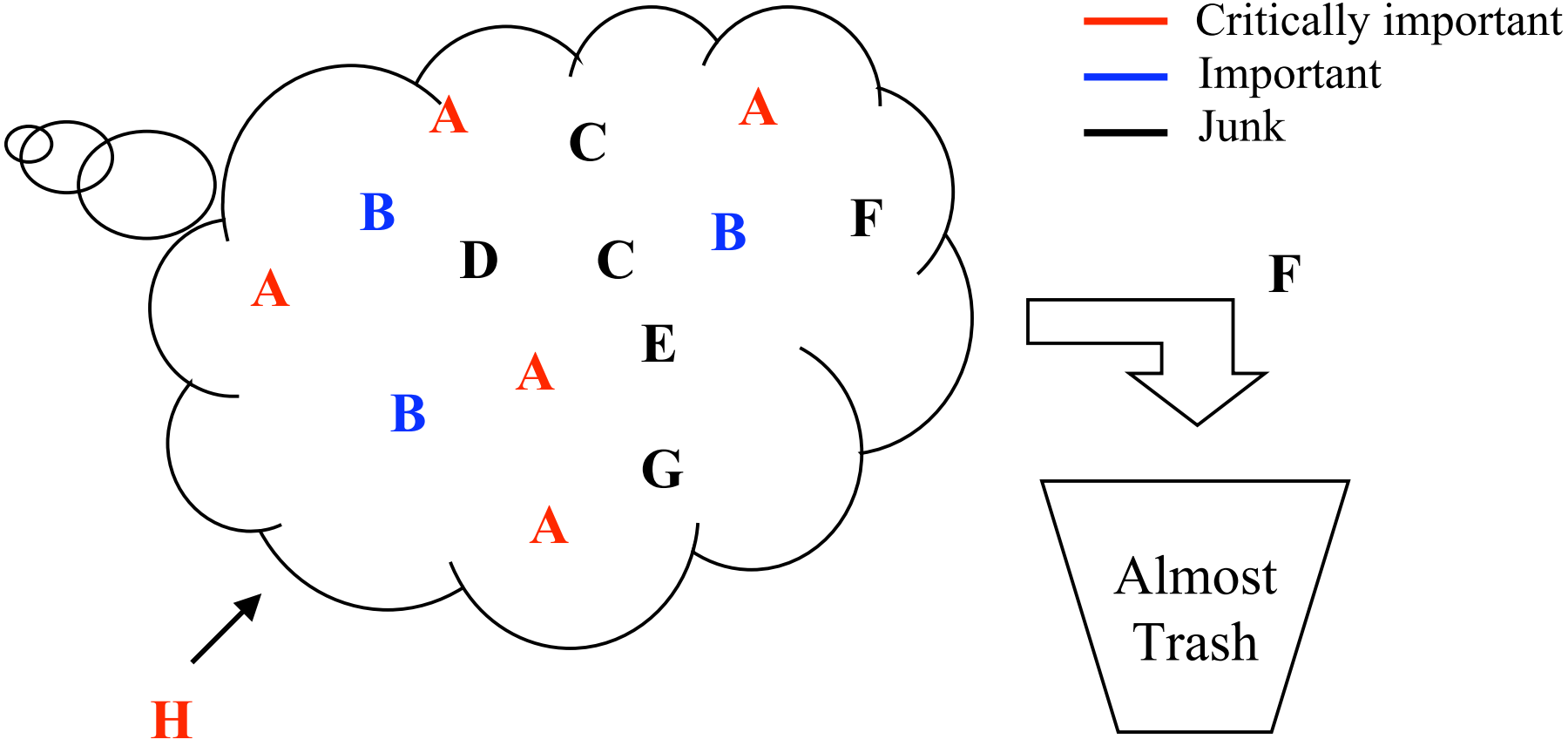
Directory grafting is done by querying PlanetP for all waynodes containing the directory's pathname


{k1,k2}
{k3,k4, k5}
{k6}
{k7,k8}
{k9}
W1
W2
W3
W4
W5
W6
F1
F2
F3
F4
F7
F5
F6


WayFinder
File System API + Extended API
Light Weight Http Server
SystemX
Light Weight DHT
Content Addressing
Membership
Gossiping
Hoard Mngt.
Network
Local File System


/
H1
A
/
H2
B
C
E
/
H3
B
D
/
H4
F
G
/
H5
B
D
E
/
A
B
F
C
D
E
G
/
H1
A
/
A
B
C
E
/
H2
B
C
E
/
H3
B
D
/
B
F
D
G
/
H4
F
G
/
H5
B
D
E
/
B
D
E

Availability Model



Files

Each file replica is described by a unique meta-inode (called a waynode)

Contains a globally unique file ID, version vector, content hash, and location (and an ACL)

Version vectors used to to keep track of changes

Encoded in XML for portability and ease of debugging

Waynodes are then published to PlanetP

Unique keys of the file are also published for content addressing

Can locate file either by content-based or id-based querying

The current state of a file is determined by collecting all the necessary waynodes

Example file waynode

< File name="t3.txt" type="File" version="1.0:initial" size="72" location="URL" version_history="1.0:initial" fileID="id1" source="node1" cotentHash="123"/>

Directories

Directories are also represented by waynodes

- Each user's directory waynodes represent only locally hoarded files
- Directories are identified by their name alone

Directories are constructed by collecting these waynodes

- All waynodes for a directory are collected
- The sets of files are then merged into a single view for the user

Example Directory Waynode

```
<dir name="foobar" type="hierarchical">
    <file name="t1.txt" ID="123">
    <file name="t2.txt" ID="413">
<dir/>
```

Consistency Model

File consistency

- May not be able to locate latest version
  - May or may not know about the existence of the latest version
- May modify an old version
- Same conflict resolution mechanism works
  - Determinism of automatic resolution allows pair-wise reconciliation

Directory consistency

- Nothing new except … deletes