MONTANA STATE UNIVERSITY

Chapter 2

Peer-to-peer (P2P) applications

P2P Systems

- Characteristics
  - No always-on server
  - Arbitrary end systems directly communicate
  - Peers are intermittently connected and change IP addresses

- Unstructured P2P systems
  - Bit-Torrent, TOR, Blockchains
- Structured P2P systems
  - Distributed Hash Tables (DHTs)

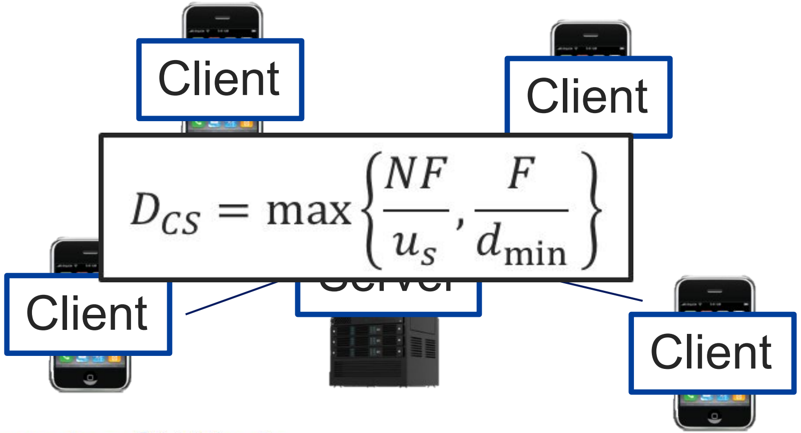
What are some examples of P2P systems in use today?

Scalability of P2P Architectures

How much time does it take to distribute a file of size F to N clients?

Client-Server architecture

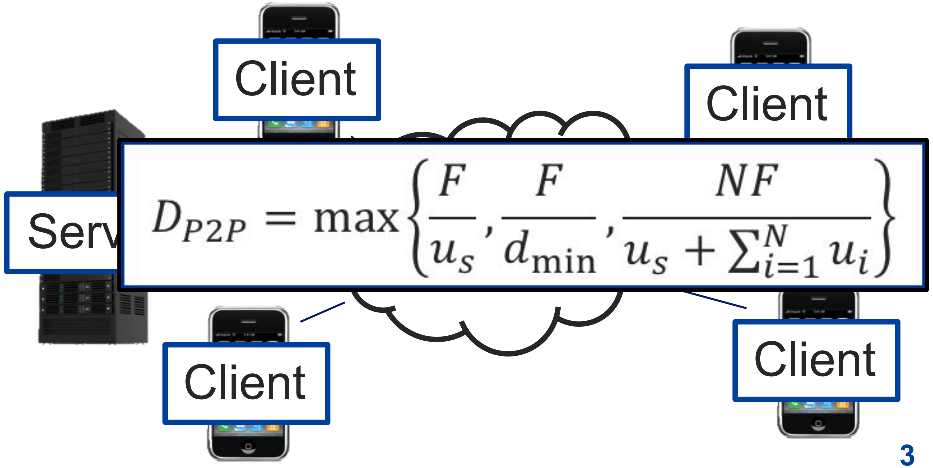
- Server can upload data at rate u_s
- Clients download data at rates $d_1, d_2, \dots, d_N$



$$D_{CS} = \max\left\{\frac{NF}{u_s}, \frac{F}{d_{\min}}\right\}$$

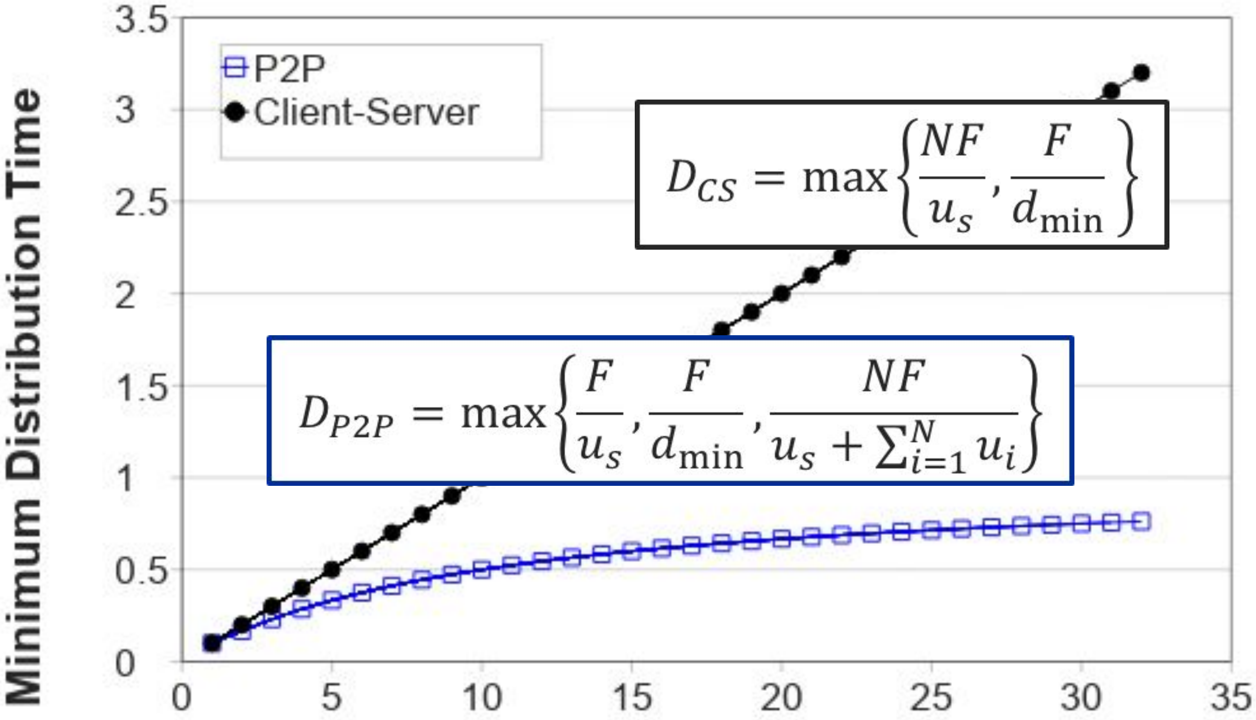
P2P Architecture

- Clients can also upload data at rates $u_1, u_2, \dots, u_N$



$$D_{P2P} = \max\left\{\frac{F}{u_s}, \frac{F}{d_{\min}}, \frac{NF}{u_s + \sum_{i=1}^{N} u_i}\right\}$$

Distribution time



$$D_{CS} = \max\left\{\frac{NF}{u_s}, \frac{F}{d_{\min}}\right\}$$

$$D_{P2P} = \max\left\{\frac{F}{u_s}, \frac{F}{d_{\min}}, \frac{NF}{u_s + \sum_{i=1}^{N} u_i}\right\}$$

P2P architectures are *self-scaling*

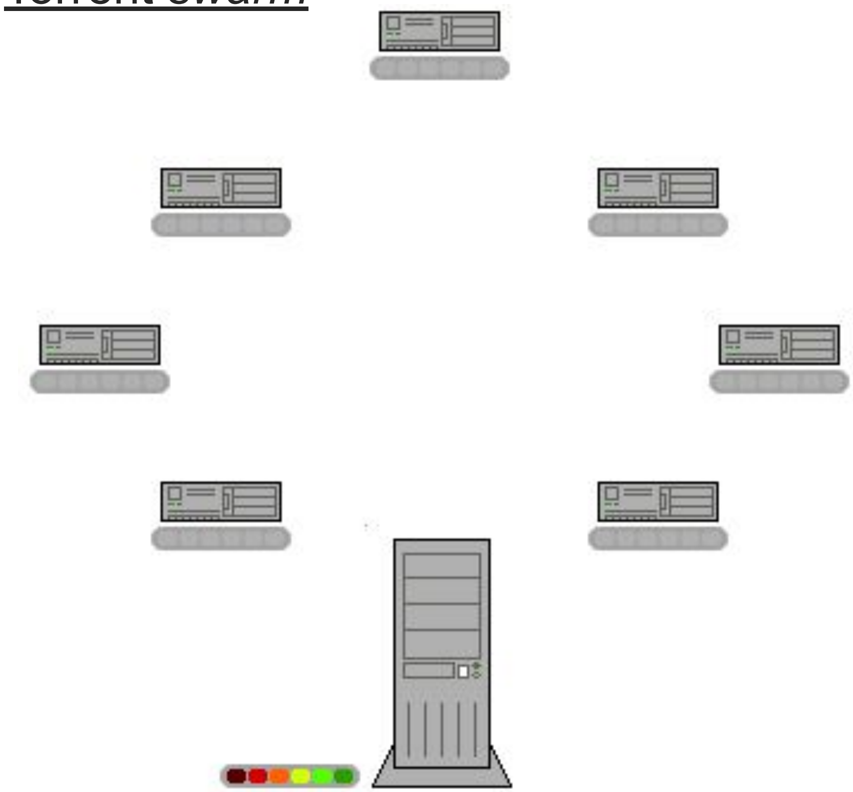
Assume:

$$\frac{F}{u} = 1 \text{ hour}$$
$$u_s = 10u$$
$$d_{\min} \geq u_s$$

- **Client server** distribution time grows with the number of clients
- **P2P** distribution time approaches 1 hour as number of clients grows

- Key Motivation:
  - File popularity exhibits temporal locality
  - Flash crowds, Slashdot effect, etc.
- Functionality
  - File chunks distributed to peers
  - Collaborative download
  - Has some “real” publishers
- Mechanism - Swarming
  - Publish: Run a tracker server
  - Search: Out-of-band
  - Join: get list of peers from tracker
  - Fetch: Direct shard exchange with peers

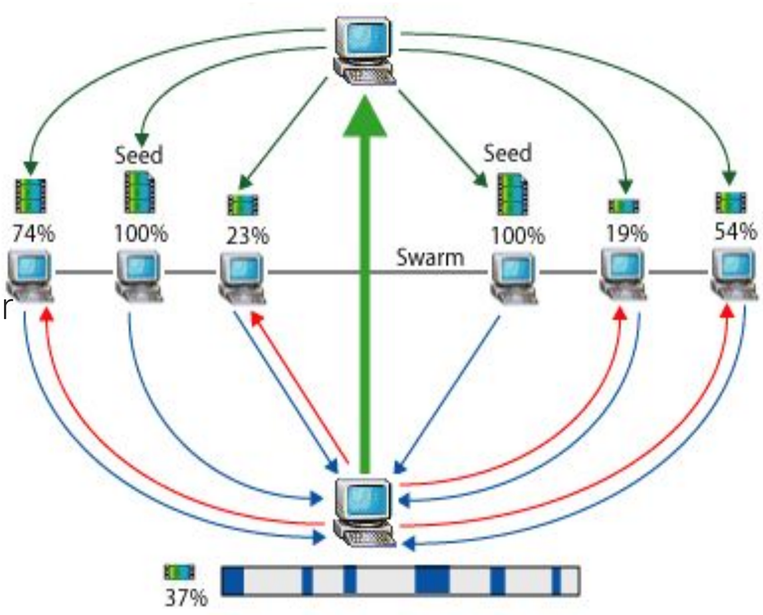
Torrent *swarm*



BitTorrent continued

- Which chunks to download?
  - Rarest first mechanism
  - Equalizes the number of copies of each chunk in the system
- From which peers to download?
  - “Tit-for-tat” sharing strategy
  - Allow download to N peers with highest upload rates
  - Allows peers with similar upload rates to find each other
- How to bootstrap peers?
  - Opportunistic unchocking mechanism
  - Allows download from random peer
  - Allows new peers to start
  - Allows shift to download from faster peers

Torrent *swarm*

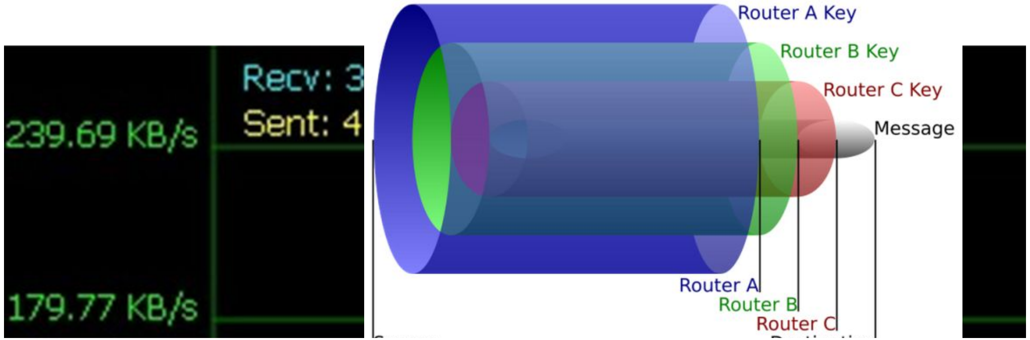


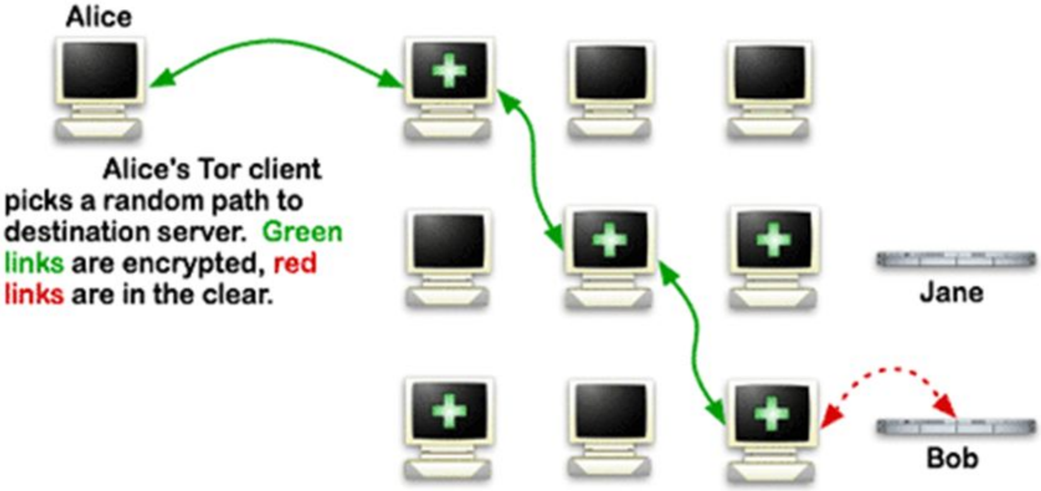
What are some limitations to BitTorrent scalability?

Unstructured P2P Middleware

The onion router (TOR)

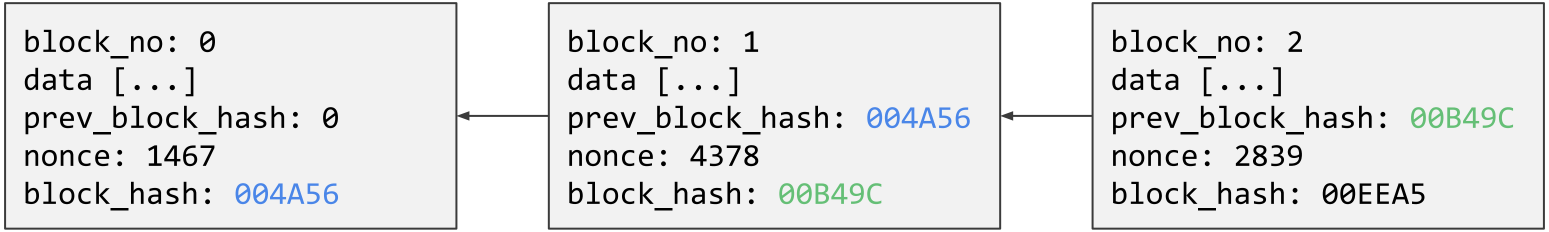
- History
  - Deployed in 2002
  - Conceal user’s identity and network activity
- Functionality
  - Sender obtains a set of router keys
  - Each router only knows next hop
  - Intermediate routers cannot read message
- Vuze includes built-in Tor support
- Can be used to access the darknet
- Invisible Internet Project (I2P)
  - Support pseudoanonymous services
  - Fully distributed node database





Alice's Tor client picks a random path to destination server. Green links are encrypted, red links are in the clear.

Blockchains



```
block_no: 0
data [...]
prev_block_hash: 0
nonce: 1467
block_hash: 004A56
```

```
block_no: 1
data [...]
prev_block_hash: 004A56
nonce: 4378
block_hash: 00B49C
```

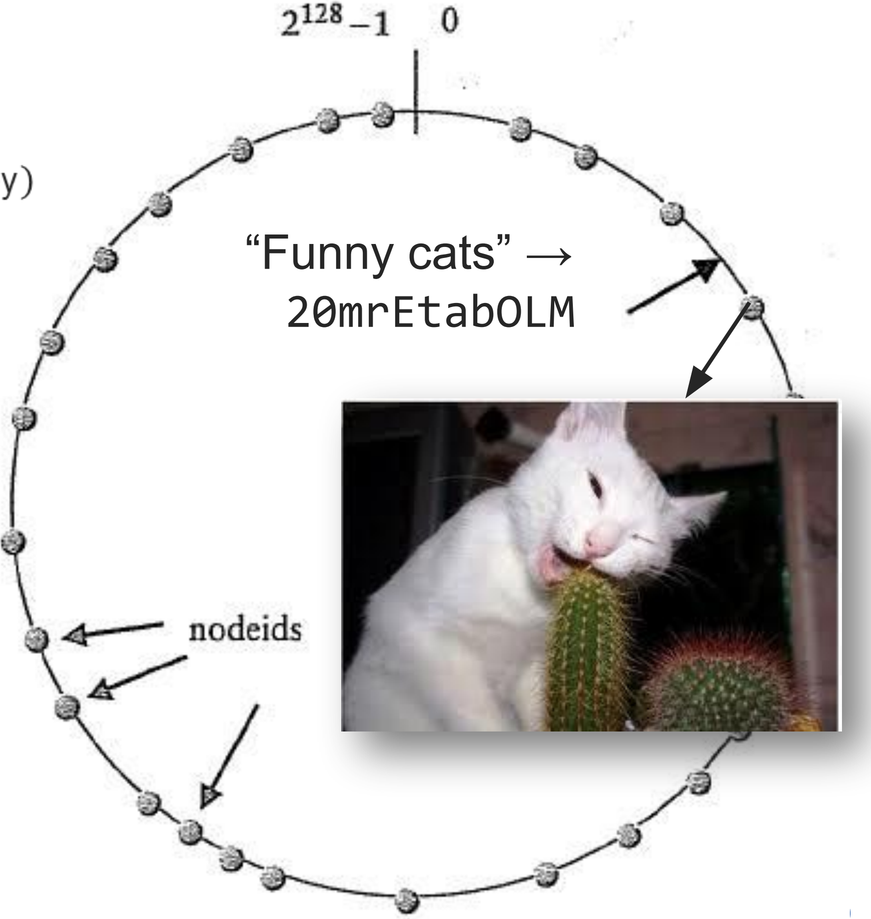
```
block_no: 2
data [...]
prev_block_hash: 00B49C
nonce: 2839
block_hash: 00EEA5
```

- A miner sends a new block to other miners, who verify it and start working on the following block
- A series of cryptographically linked blocks
- Cannot modify older blocks without modifying newer ones

- Consensus over data, and so over a series of distributed system actions
  - Monetary transfers
  - Smart contract executions

Structured P2P

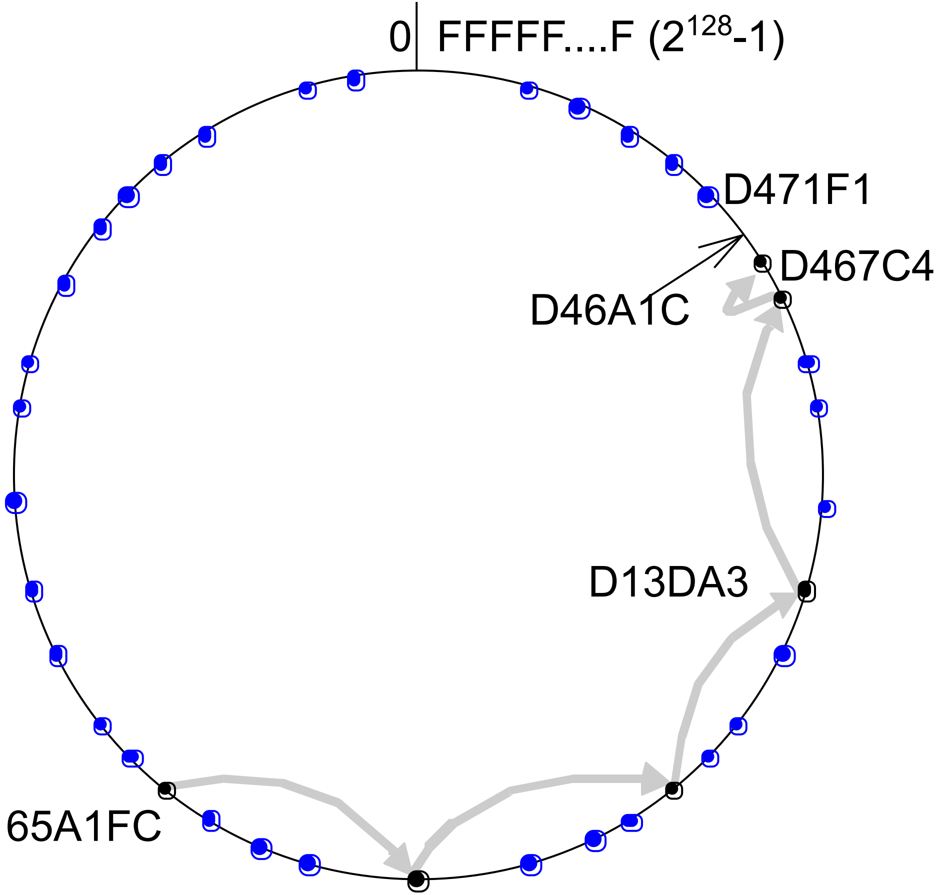
- History
  - 1990's: collision resistant hash functions
    - For $a \neq b$, $\mathrm{P}(H(a) = H(b))$ is very low
    - Globally Unique Identifier (GUID) from H(key)
  - 2001: Chord, Pastry, CAN, …
- Functionality
  - Distributed <key, value> search
    - Eg. <"Funny cats", value> → <20mrEtabOLM, value>
  - Fully distributed and self-organizing
- Distributed Hash Table (DHT)
  - Join: join ring with GUID from public key
  - Publish: put(GUID, value)
  - Search/Fetch: value = get(GUID)



"Funny cats" → 20mrEtabOLM

Search in a structured P2P

- Each node maintains links to 4 closest **`nodeids`** in each direction
- Queries for **`objid`** forwarded to closest **`nodeid`** in the routing table
- Eventually **`nodeid`** responsible for **`objid`** is found

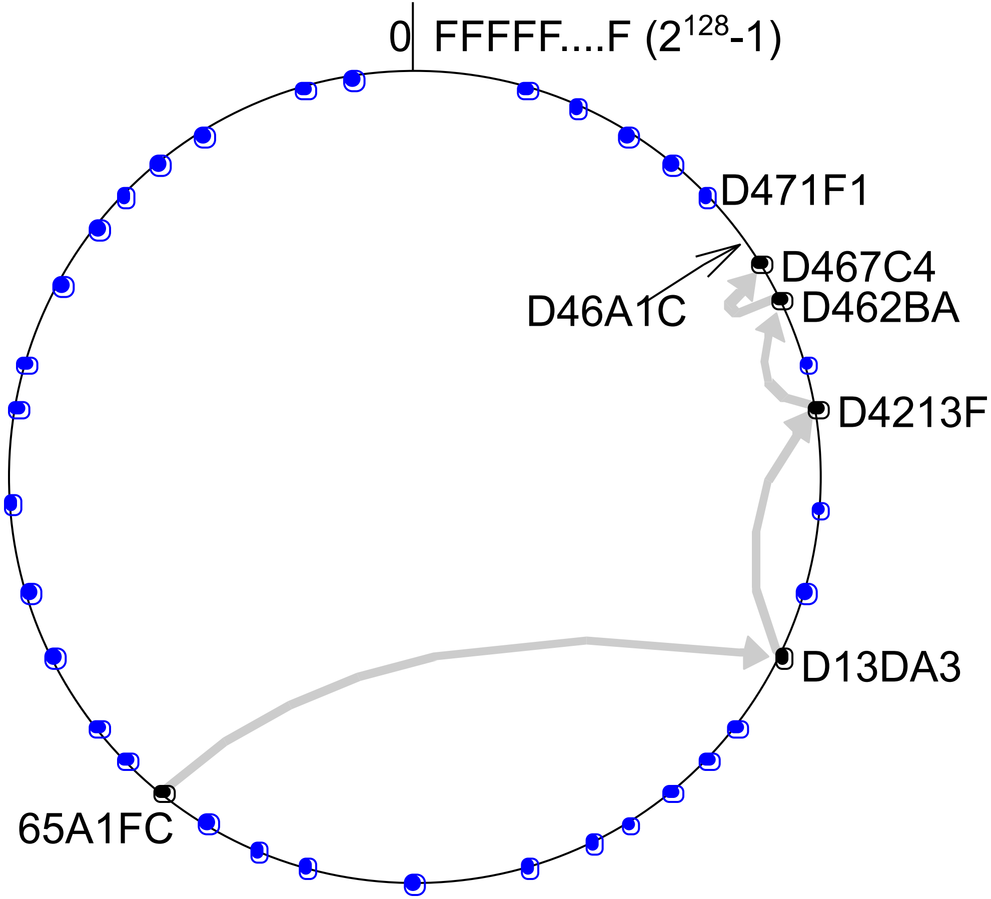


$O(\log N)$ DHT search

- Each node maintains links to **nodeids half** way across the ring, **quarter** way across the ring, **eight** of the way, etc.

- Queries for **objid** forwarded to **nodeid** closest in the routing table to **objid**

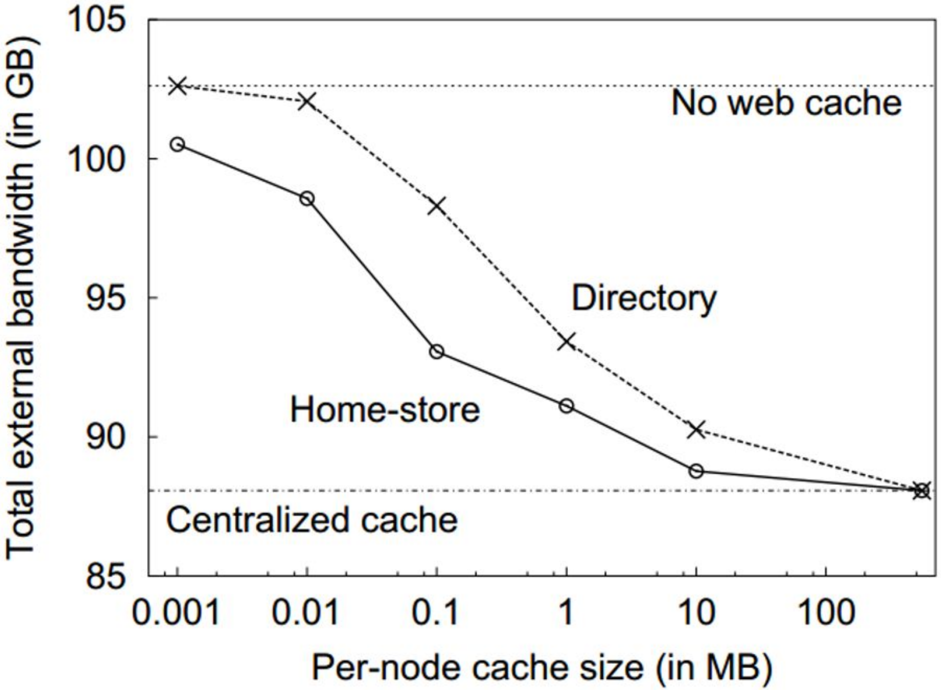
- Queries satisfied in $O(\log N)$



Structured P2P Middleware

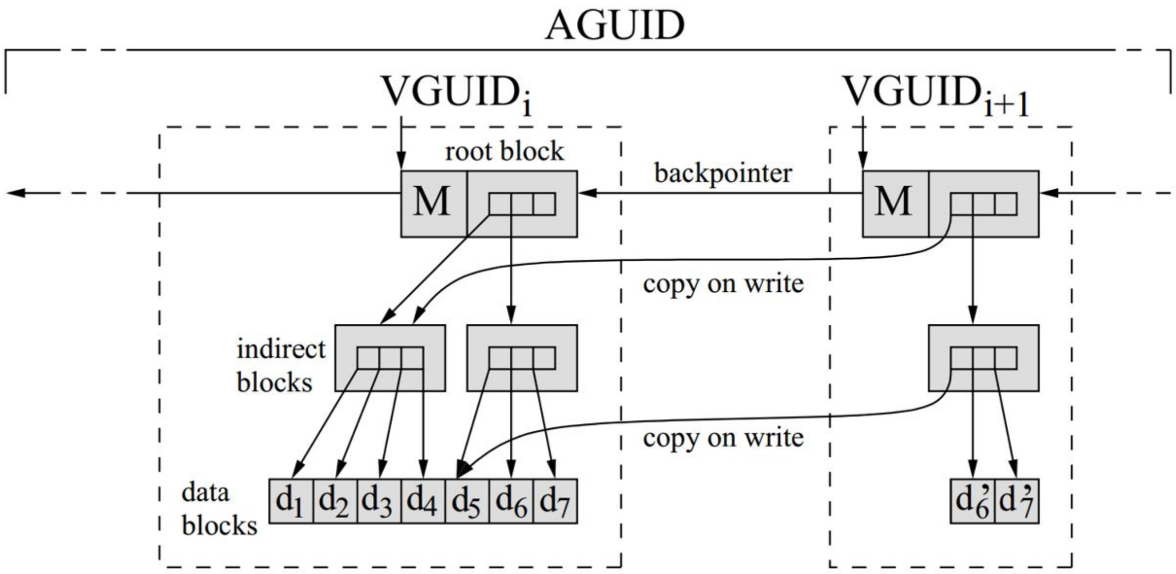
- Squirrel
  - Distributed Web cache
  - MS Research 2002
  - Stores web objects http://wiki.org/../network.png on nodes within LAN

- Performance
  - Lookups require several LAN hops (~1ms each) vs. WAN latency of (~10-100ms)
  - Same hit rate as centralized cache with modest per node resources

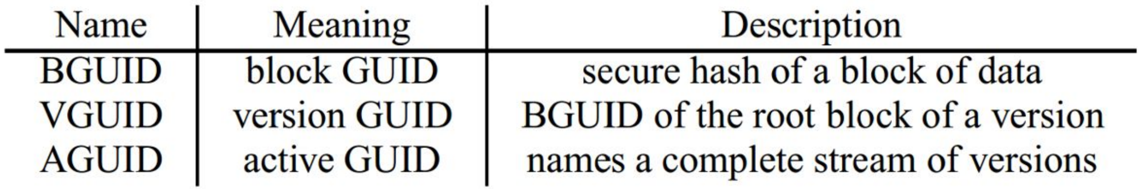


P2P-based Middleware

- OceanStore
  - Distributed file system
  - Supports mutable objects
  - Pond prototype 2003 based on Tapestry DHT



- Slower than NFS on LAN
- Both really slow on WAN



Name	Meaning	Description
BGUID	block GUID	secure hash of a block of data
VGUID	version GUID	BGUID of the root block of a version
AGUID	active GUID	names a complete stream of versions

Unstructured vs. Structured P2P

	Unstructured P2P	Structured P2P
Advantages	• Self-organizing • Naturally resilient to node failure	• Guaranteed to locate objects (if exist) • Time and complexity bounds • Low message overhead
Disadvantages	• Cannot offer guarantees on locating objects • Prone to excessive messaging that limits scalability	• Need to maintain complex overlays • High control traffic overhead in dynamic environments

MONTANA
STATE UNIVERSITY