Experiences Leveraging DHTs for a Security Application

Amit Levy with:
Roxana Geambasu (Columbia)
Yoshi Kohno (UW)
Arvind Krishnamurthy (UW)
Hank Levy (UW)
Paul Gardner (Vuze)
Vinnie Moscaritolo (PGP)

Outline

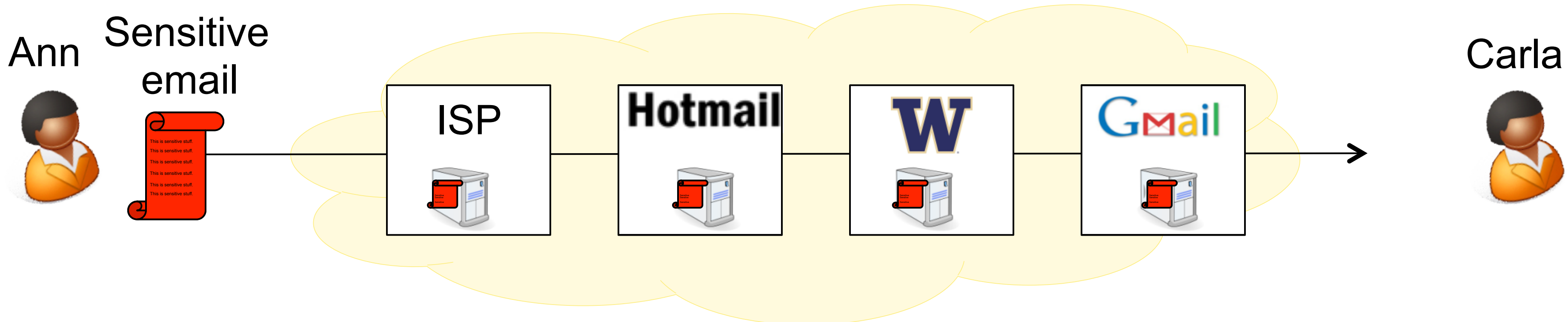
- Vanish – a self-destructing data system
- Challenges building Vanish on a global-scale P2P DHT
- Comet – the DHT we wish we had

Vanish:
Increasing Data Privacy with Self-Destructing Data

The Problem: Two Huge Challenges for Privacy

1. Data lives forever
   - On the web: emails, Facebook photos, Google Docs, blogs, …
   - In the home: disks are cheap, so no need to ever delete data
   - In your pocket: phones and USB sticks have GBs of storage
2. Retroactive disclosure of both data and user keys has become commonplace
   - Hackers
   - Legal actions
   - Border seizing
   - Theft

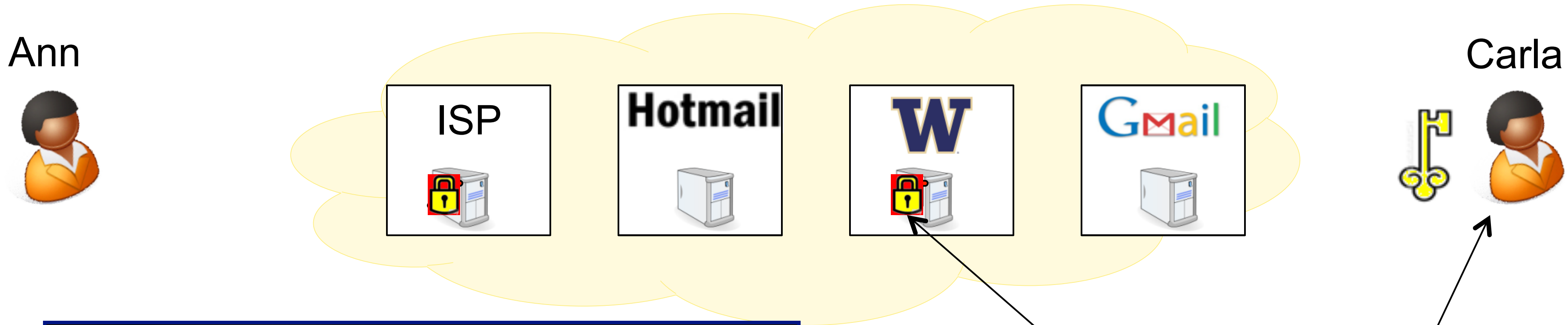
Motivating Problem: Data Lives Forever



How can Ann delete her sensitive email?

- She doesn't know where all the copies are
- Services may retain data for long after user tries to delete

Why Not Use Encryption (e.g., PGP)?



cnet news

February 26, 2009 1:30 PM PST

Judge orders defendant to decrypt PGP-protected laptop

A federal judge has ordered a criminal defendant to decrypt his hard drive by typing i… a ruling…

V3.co.uk formerly vnunet.com

UK police can now demand encryption keys

vnunet.com, 03 Oct 2007

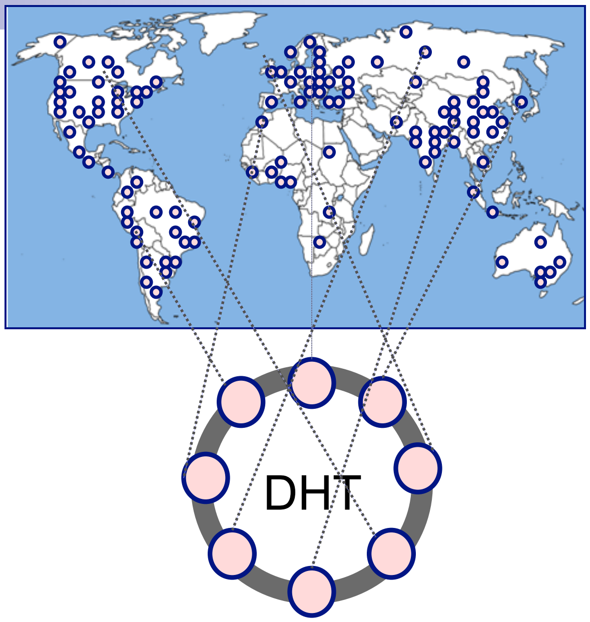
People in the UK who encrypt their data are now obliged by law to give up the encryption keys to law enforcement officials if requested under the Regulation of Investigatory Powers Act 2000 (RIP Act).

Vanish: Self-Destructing Data System

- Traditional solutions are not sufficient for self-destructing data goals:
  - PGP
  - Centralized data management services
  - Forward-secure encryption
  - …
- Let's try something completely new!

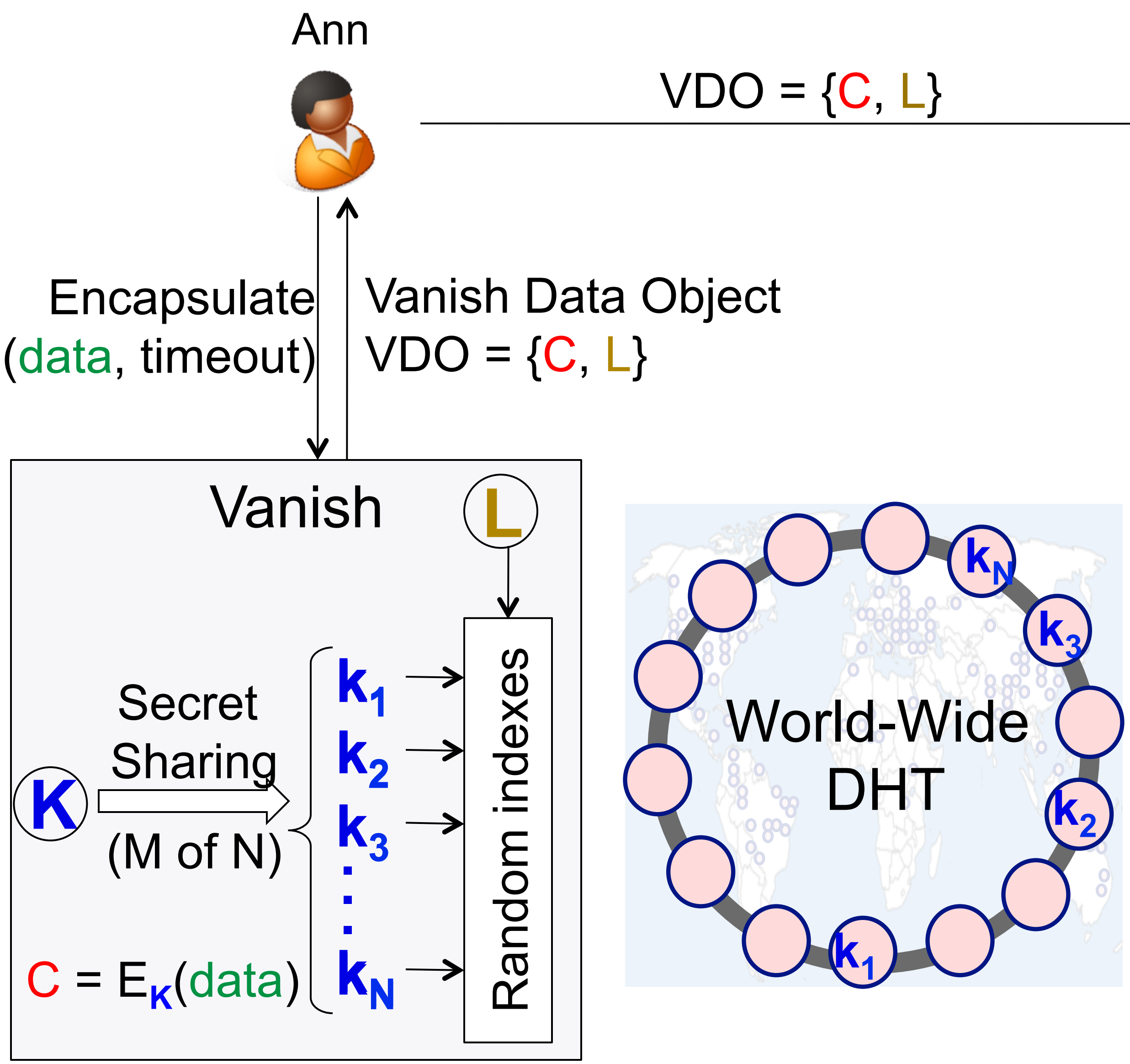
Idea:
Leverage P2P systems

Distributed Hashtables (DHTs)

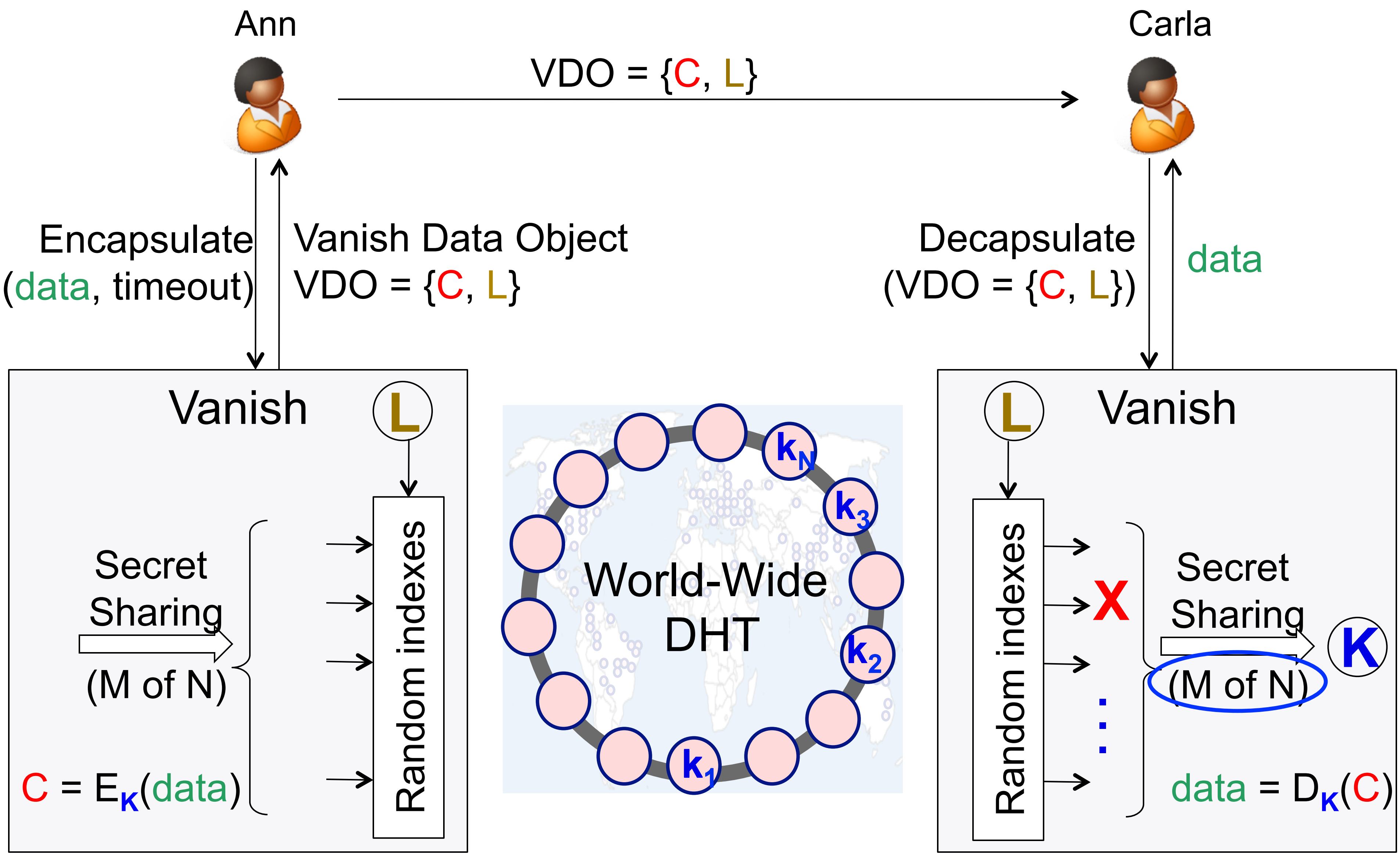


- Hashtable data structure implemented on a P2P network
  - Get and put (index, value) pairs
  - Each node stores part of the index space
- DHTs are part of many file sharing systems:
  - Vuze, Mainline, KAD
  - Vuze has ~1.5M simultaneous nodes in ~190 countries
- Vanish leverages DHTs to provide self-destructing data
  - One of few applications of DHTs outside of file sharing

How Vanish Works: Data Encapsulation

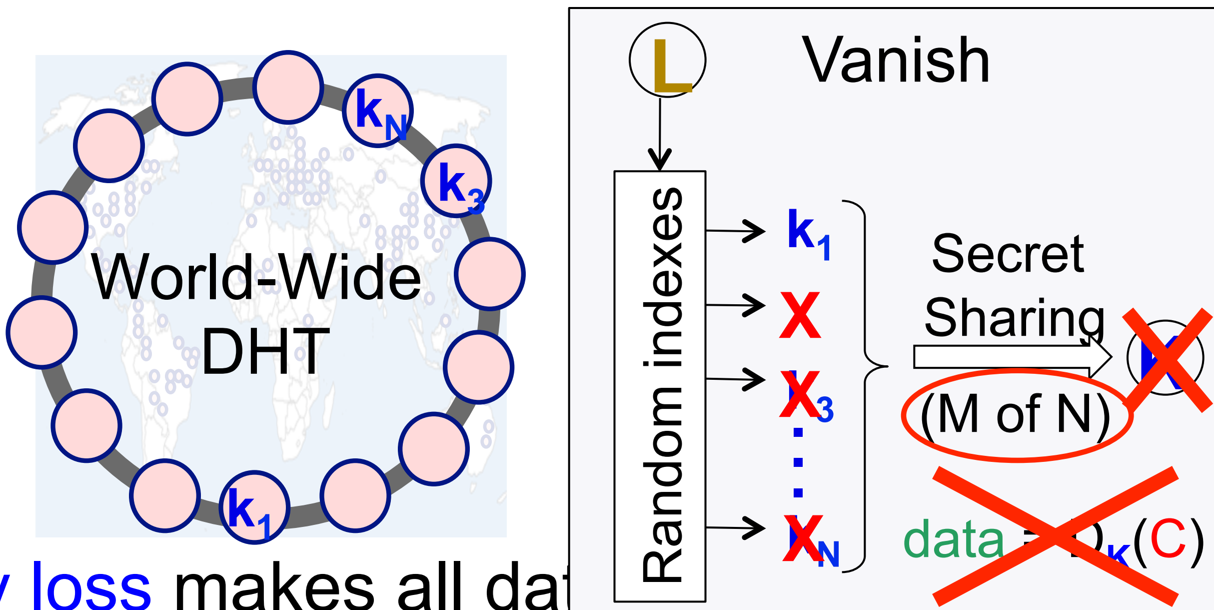


How Vanish Works: Data Decapsulation



How Vanish Works: Data Timeout

- The DHT loses key pieces over time
  - Built-in timeout: DHT nodes purge data periodically
  - *Natural churn: nodes crash or leave the DHT (note for later)*



- Key loss makes all dat … y unreadable

Evaluation

- Experiments to understand and improve (won't cover):
  1. data availability before timeout
  2. data unavailability after timeout
  3. performance
  4. security

- Highest-level results:
  - Tradeoffs are necessary between availability, performance and security.
  - Secret sharing parameters (N and M) affect tradeoffs

Conclusions

- Two formidable challenges to privacy:
  - Data lives forever
  - Disclosures of data and keys have become commonplace
- Vanish combines global-scale DHTs with secret sharing
- ***Vanish ≠ Vuze-based Vanish***
  - Customized DHTs, hybrid approach, other P2P systems
  - Further extensions for security in the paper

Vuze DHT Weaknesses

- Static data timeouts
- Over-replicates
  - Maintains 20 replicas of each key-value pair
  - Three replicas is sufficient for availability
- Over-eager replication
  - *push-on-join*
  - Many nodes join the system for very short periods
- Weak Sybil protections
  - Single IP can take on up to 64K identities
  - A laughable number of machines can defeat Vanish in a preemptive data harvesting attack

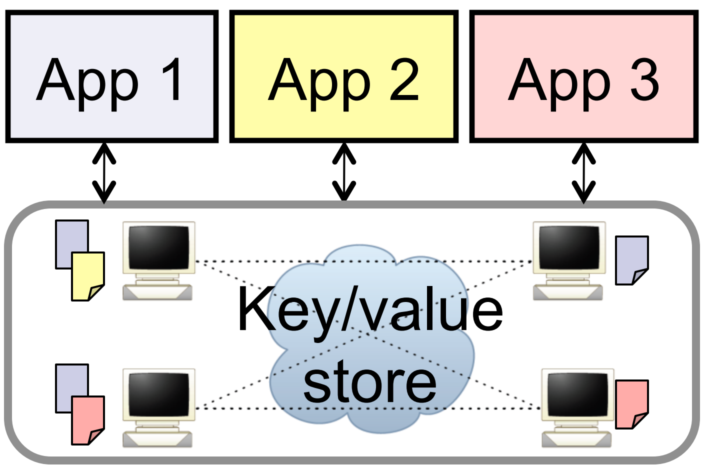
Vuze DHT Weaknesses

- Fixes
  - Variable data timeout (specified by flags)
  - No *push-on-join*
  - Variable (and smart) replication factor
  - Limit replicas per IP prefix
  - …
- Changes were simple, but deploying them was ***difficult***:
  - Need Vuze engineer
  - Long deployment cycle
  - Hard to evaluate before deployment

Comet: An Active Distributed Key-Value Store

Challenge: Inflexible Key/Value Stores

- Applications have different (even conflicting) needs:
  - Availability, security, performance, functionality
- But today's key/value stores are one-size-fits-all
- Motivating example: our Vanish experience

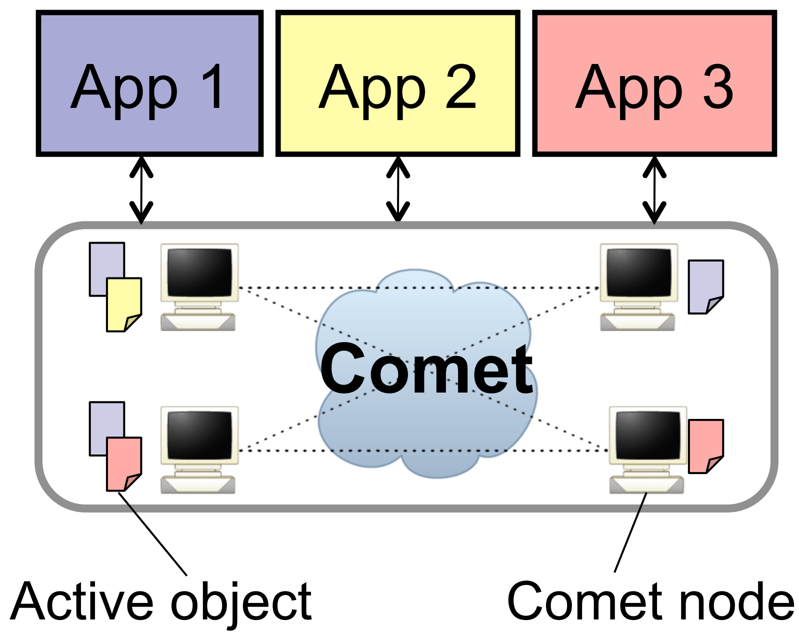


Extensible Key/Value Stores

- Allow apps to customize store's functions
  - Different data lifetimes
  - Different numbers of replicas
  - Different replication intervals
- Allow apps to define new functions
  - Tracking popularity: data item counts the number of reads
  - Access logging: data item logs readers' IPs
  - Adapting to context: data item returns different values to different requestors

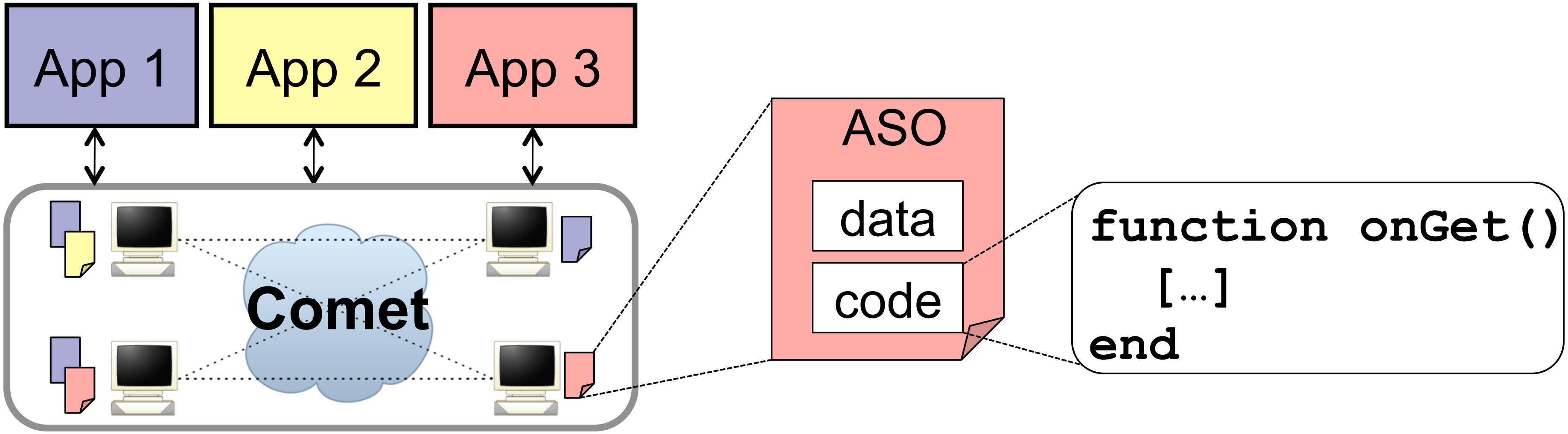
Comet

- DHT that supports application-specific customizations
- Applications store active objects instead of passive values
  - Active objects contain small code snippets that control their behavior in the DHT



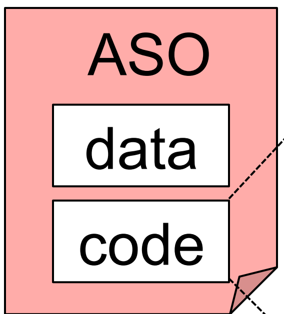
Active Storage Objects (ASOs)

- The ASO consists of data and code
  - The data is the value
  - The code is a set of handlers that are called on **`put/get`**



Simple ASO Example

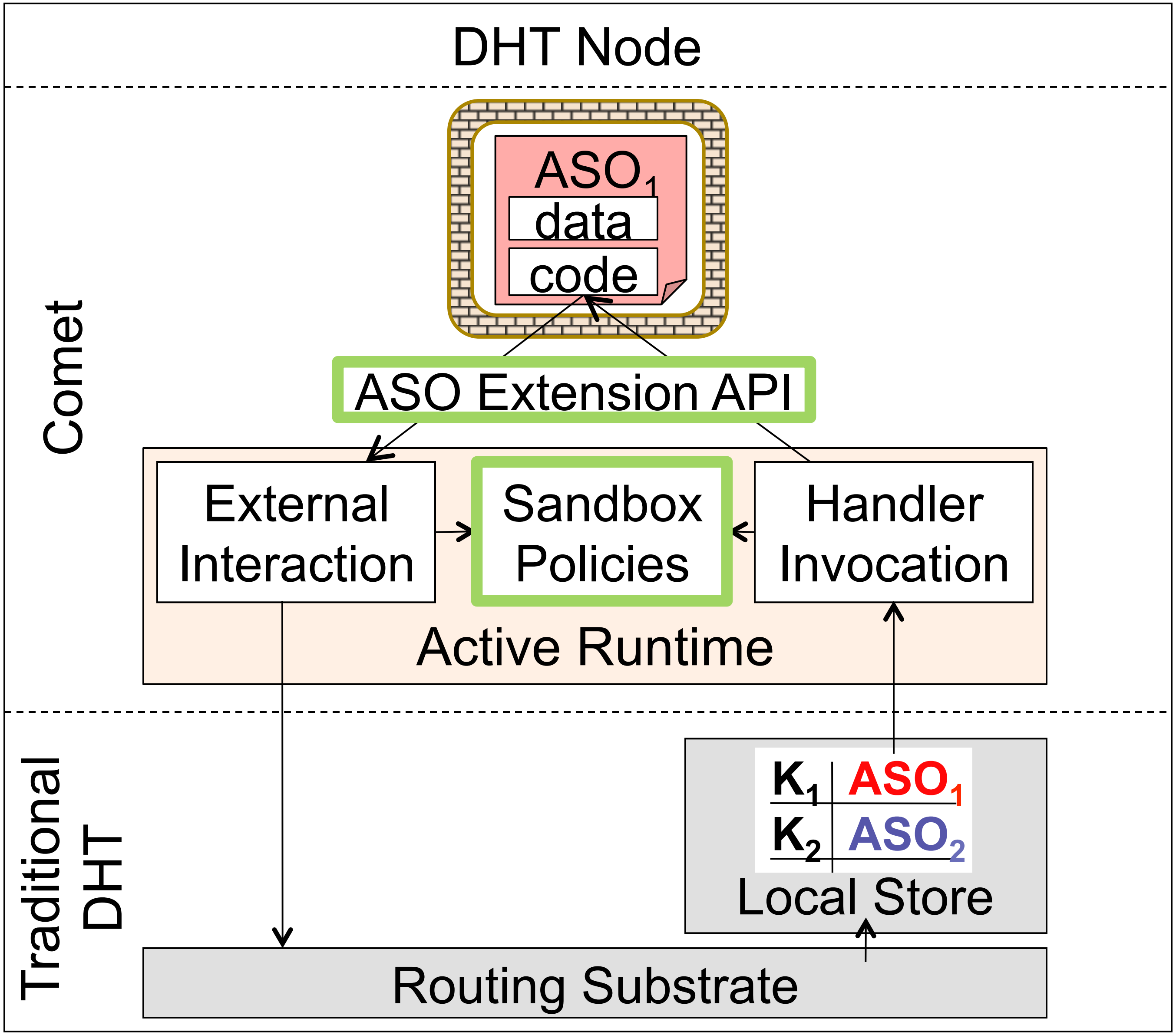
- Each replica keeps track of number of **`gets`** on an object




```
aso.value = "Hello world!"
aso.getCount = 0
function onGet()
   self.getCount = self.getCount + 1
   return {self.value,
   self.getCount}
end
```

- The effect is powerful:
  - Difficult to track object popularity in today's DHTs
  - Trivial to do so in Comet without DHT modifications

Comet Architecture



Comet Prototype

- We built Comet on top of Vuze and Lua
  - We deployed experimental nodes on PlanetLab
- In the future, we hope to deploy at a large scale
  - Vuze engineer is particularly interested in Comet for debugging and experimentation purposes

Comet Applications

Applications	Customization	Lines of Code
Vanish	Security-enhanced replication	41
	Flexible timeout	15
	One-time values	15
Adeona	Password-based access	11
	Access logging	22
P2P File Sharing	Smart Bittorrent tracker	43
	Recursive gets*	9
P2P Twitter	Publish/subscribe	14
	Hierarchical pub/sub*	20
Measurement	DHT-internal node lifetimes	41
	Replica monitoring	21

* Require signed ASOs (see paper)

Three Examples

1. Application-specific DHT customization
2. Context-aware storage object
3. Self-monitoring DHT

1. Application-Specific DHT Customization

- Example: customize the replication scheme

```
function aso:selectReplicas(neighbors)
  [...]
end
function aso:onTimer()
  neighbors = comet.lookup()
  replicas = self.selectReplicas(neighbors)
  comet.put(self, replicas)
end
```

- We have implemented the Vanish-specific replication
  - Code is 41 lines in Lua

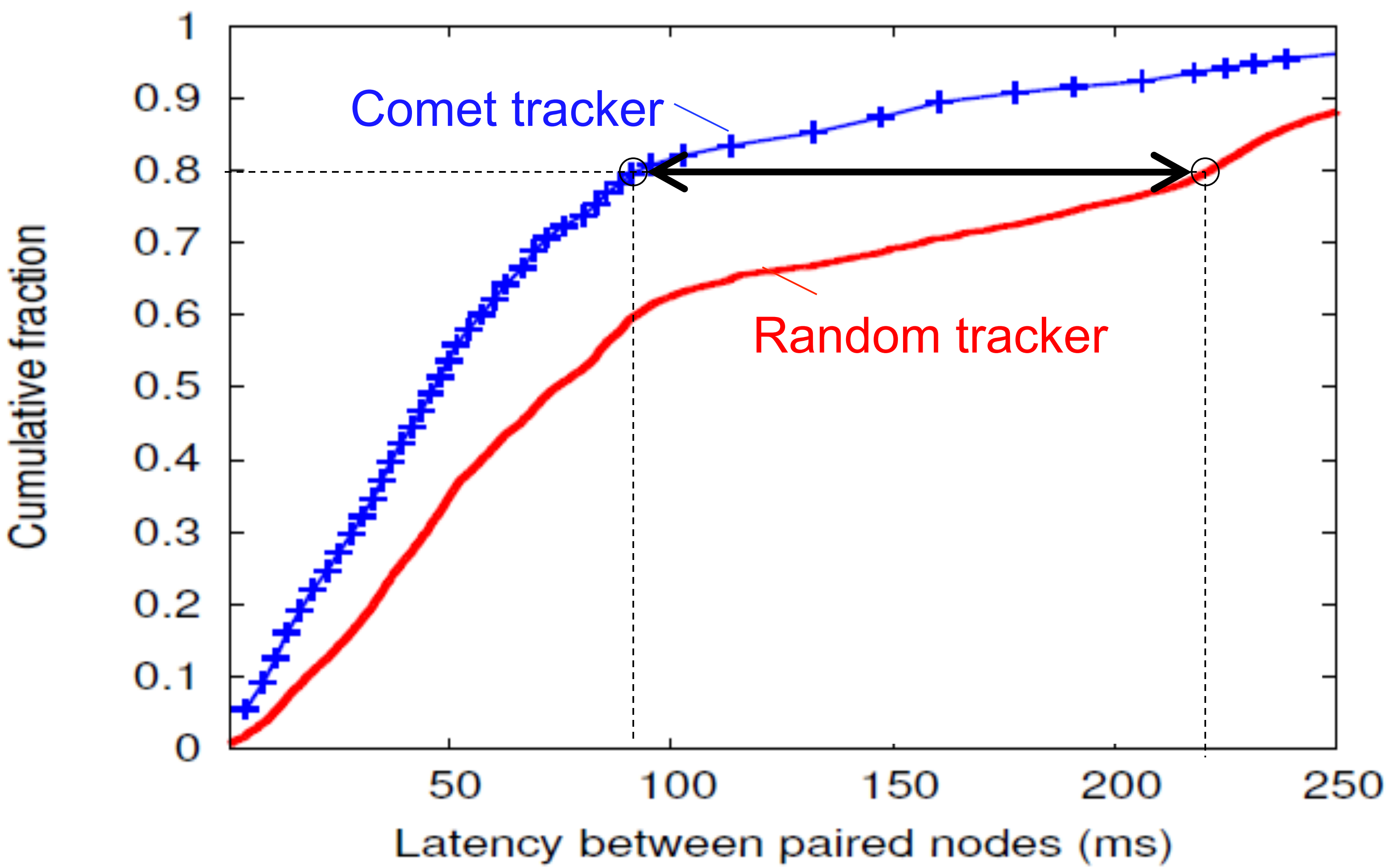
2. Context-Aware Storage Object

- Traditional distributed trackers return a randomized subset of the nodes

- Comet: a proximity-based distributed tracker
  - Peers `put` their IPs and Vivaldi coordinates at `torrentID`
  - On `get`, the ASO computes and returns the set of closest peers to the requestor

- ASO has 37 lines of Lua code

Proximity-Based Distributed Tracker



3. Self-Monitoring DHT

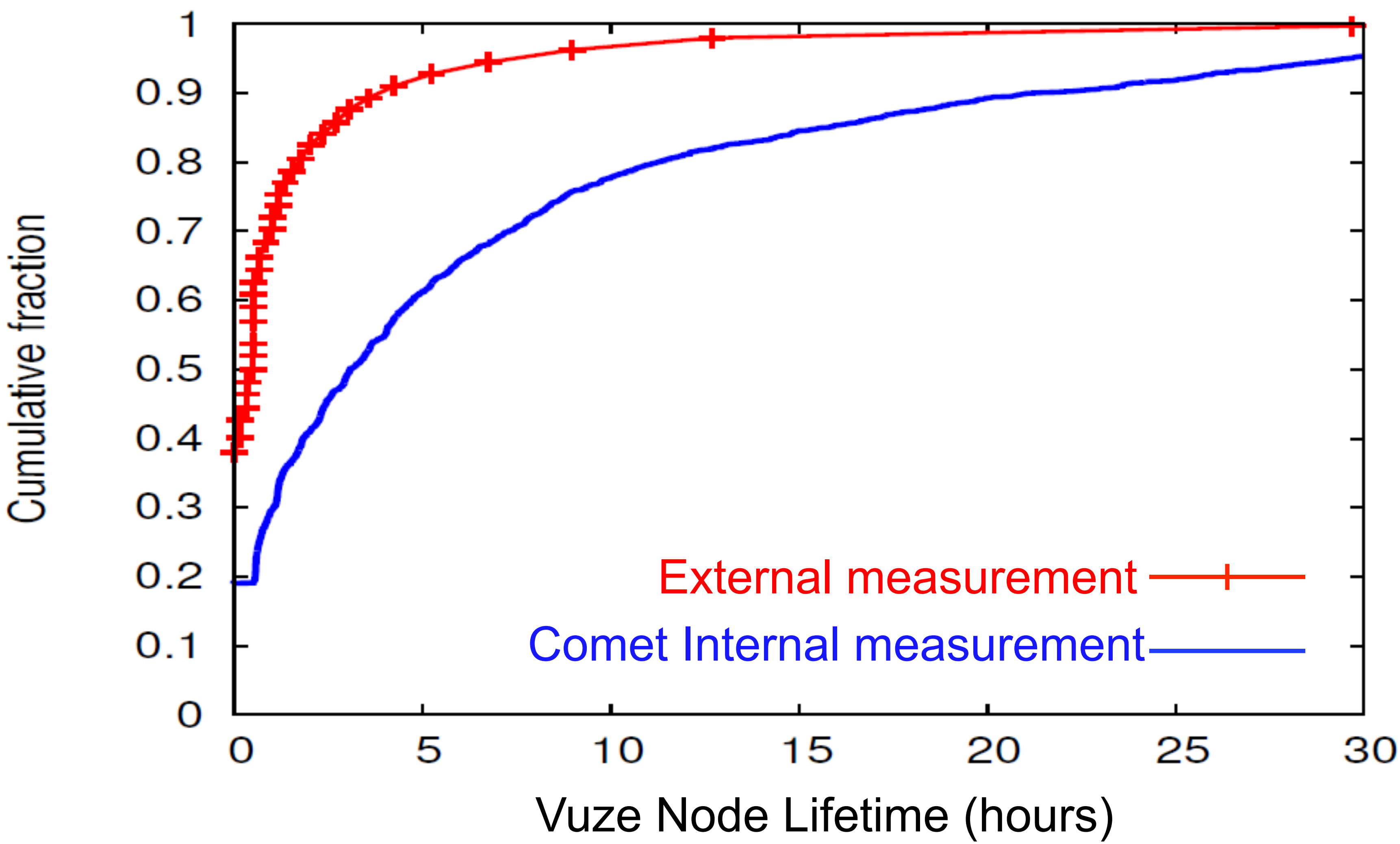
- Example: monitor a remote node's neighbors
  - **`Put`** a monitoring ASO that "pings" its neighbors periodically

```
aso.neighbors = {}

function aso:onTimer()
  neighbors = comet.lookup()
  self.neighbors[comet.systemTime()] = neighbors
end
```

- Useful for internal measurements of DHTs
  - Provides additional visibility over external measurement (e.g., NAT/firewall traversal)

Example Measurement: Vuze Node Lifetimes



Remember the bit about churn?

- We tried using churn to control data lifetime in Vanish
  - The numbers were all wrong
  - Data stayed around for way too long
- Very difficult to accurately measure churn (or size) in current global-scale DHTs
  - Many firewalled nodes - only speak to their neighbors
  - Contribute to data resilience but are unreachable by clients (show up as dead in external measurements)
- Measuring internally
  - Results that better matched our observations in Vanish
  - May be the only option - don’t control nodes in the system
  - Depends on what you want to measure

Conclusions

- Global scale DHTs are a useful abstraction for security
  - But it turns out not to be that simple
  - Totally non-idealized environment
  - Hard to simulate with small deployments
  - Hard to get changes deployed
- Is there hope with extensibility?
  - Able to modify DHT behavior per appication
  - Able to test easily
- Where are we now?
  - Some interest from Vuze for their own purposes but still no deployment
  - Could deploy our own cluster but not very useful even at the scale of planet lab