Experiences in Building and Operating ePOST, a Reliable Peer-to-Peer Application

Alan Mislove†‡
Ansley Post†‡
Andreas Haeberlen†‡
Peter Druschel†

†Max Planck Institute for Software Systems
‡Rice University
Reliable P2P Systems: Myth or Reality?

• For the past few years, much research interest in p2p
  • **Highly scalable** in nodes and data
  • Utilization of **underused resources**
  • **Robust** to large range of workloads and failures

• Most deployed systems are not reliable [Kazaa, Skype, etc]
  • None attempt to **store data reliably, durably, or securely**
  • Lead some to conclude p2p can’t support reliable applications

• Question: **Can peer-to-peer systems provide reliable service?**
Demonstration Application: ePOST

- ePOST is an email service built using decentralized components
  - Completely decentralized, no ‘email servers’

- Email one of the most important Internet applications
  - Privacy
  - Integrity
  - Durability
  - Availability

- Wanted to develop system to a point where people rely on it
ePOST: Deployment

- Built and deployed ePOST within our group
  - Running for over 2 years
  - Processed well over 500,000 email messages

- Built ePOST to be more reliable than existing email systems
  - 16 users used ePOST as primary email
  - Even my advisor!

- Many challenges found by building the system
  - After challenges solved, provides reliable service
  - Robust; numerous times ePOST was only mail service working
Rest of Talk

• ePOST in detail

• Challenges faced in building and deploying ePOST

• Conclusion
**ePOST: Architecture**

- Each participating node runs **mail servers for the local user**
  - Email service **looks the same to users**

- Data stored cooperatively on participating machines
  - Machines form overlay
  - Replicated for redundancy

- All data **encrypted and signed**
  - Prevents others from reading your email
ePOST: Architecture

- Each participating node runs **mail servers for the local user**
  - Email service **looks the same** to users

- Data stored cooperatively on participating machines
  - Machines form overlay
  - Replicated for redundancy

- All data **encrypted and signed**
  - Prevents others from reading your email
ePOST: Metadata Storage

- Folders represented using logs
  - Entries represent changes
  - All entries self-authenticating

- Log head points to most recent entry
  - Signed by owner due to mutability
  - Only local node has key material

- All writes performed by owner
  - Map multi-access problem to single-writer
ePOST: Metadata Storage

- Folders *represented using logs*
  - Entries represent changes
  - All entries self-authenticating

- Log head points to most recent entry
  - *Signed by owner* due to mutability
  - Only local node has key material

- All writes performed by owner
  - Map multi-access problem to single-writer
ePOST: Metadata Storage

• Folders represented using logs
  • Entries represent changes
  • All entries self-authenticating

• Log head points to most recent entry
  • Signed by owner due to mutability
  • Only local node has key material

• All writes performed by owner
  • Map multi-access problem to single-writer
Challenges Faced
Challenges Faced

- Network partitions
- NATs and firewalls
- Routing anomalies
- Node churn
- Correlated failures
- Resource consumption
- Data storage
- Slow nodes
- Hidden single points of failure
- Data corruption
- Comatose nodes

- Complex failure modes
- Very unsynchronized clocks
- Lost key material
- Disconnected nodes
- Power failures
- Resource exhaustion
- Spam attacks on relays
- Java eccentricities
- Congested links
- PlanetLab slice deletion
- ...
Challenges Faced

- Network partitions
- NATs and firewalls
- Routing anomalies
- Node churn
- Correlated failures
- Resource consumption
- Data storage
- Slow nodes
- Hidden single points of failure
- Data corruption
- Comatose nodes
- Complex failure modes
- Very unsynchronized clocks
- Lost key material
-Disconnected nodes
- Power failures
- Resource exhaustion
- Spam attacks on relays
- Java eccentricities
- Congested links
- PlanetLab slice deletion
- ...

19.04.2006 EuroSys'06 Conference, Leuven, Belgium
Challenge: Network Partitions

- Overlay originally had **no special provisions** for network partitions
  - Did not envision partitions as a significant problem

- When a network failure occurs, nodes detect others to be dead
  - **Multiple overlays reform**

- Network usually fails at access links
  - Generally one large overlay and one small overlay
Challenge: Network Partitions

- Overlay originally had **no special provisions** for network partitions
  - Did not envision partitions as a significant problem

- When a network failure occurs, nodes detect others to be dead
  - **Multiple overlays reform**

- Network usually fails at access links
  - Generally one large overlay and one small overlay
Challenge: Network Partitions

• Overlay originally had **no special provisions** for network partitions
  • Did not envision partitions as a significant problem

• When a network failure occurs, nodes detect others to be dead
  • **Multiple overlays reform**

• Network usually fails at access links
  • Generally one large overlay and one small overlay
How frequent are partitions?

- Partitions occur often in PlanetLab
- Usually a single subnet (PlanetLab site) becomes partitioned
Impact of Network Partitions

- Tradeoff between consistency and availability under partitions
  - Well-known tradeoff
  - ePOST resolves this in favor of availability

- Partitions cause consistency problems
  - Small partitions have data inaccessibility
  - Mutable data can diverge
  - Partitions persist unless action is taken
Impact of Network Partitions

- Tradeoff between consistency and availability under partitions
  - Well-known tradeoff
  - ePOST resolves this in favor of availability

- Partitions cause consistency problems
  - Small partitions have data inaccessibility
  - Mutable data can diverge
  - Partitions persist unless action is taken
Partitions: Overlay Reintegration

- To reintegrate overlay
  - Nodes remember recently deceased nodes
  - Periodically query these nodes, and integrate missing nodes into overlay

- Protocol is periodic, and therefore stable
  - Tested on simulated failures as well as Planetlab
  - Overlay heals as expected
Partitions: Overlay Reintegration

- To reintegrate overlay
  - Nodes remember recently deceased nodes
  - Periodically query these nodes, and integrate missing nodes into overlay

- Protocol is periodic, and therefore stable
  - Tested on simulated failures as well as Planetlab
  - Overlay heals as expected
Partitions: Data Divergence

- In ePOST, log-based data structure
  - Forked logs must be merged
  - Data divergence unlikely due to single-writer behavior

- To repair logs, merge entries, cancel destructive operations
  - Ensures no data loss
Partitions: Data Divergence

- In ePOST, log-based data structure
  - Forked logs must be merged
  - Data divergence unlikely due to single-writer behavior

- To repair logs, merge entries, cancel destructive operations
  - Ensures no data loss
Partitions: Data Divergence

- In ePOST, log-based data structure
  - Forked **logs must be merged**
  - Data divergence unlikely due to single-writer behavior

- To repair logs, merge entries, cancel destructive operations
  - Ensures no data loss
Challenge: Routing Anomalies

- Overlay assumed that *any two participating nodes could communicate*

- Internet routing anomalies (routing intransitivity) a problem
  - Nodes *disagree about the liveness of other nodes*
Challenge: Routing Anomalies

- Overlay assumed that any two participating nodes could communicate.

- Internet routing anomalies (routing intransitivity) a problem.
  - Nodes disagree about the liveness of other nodes.
Challenge: Routing Anomalies

- Overlay assumed that **any two participating nodes could communicate**

- Internet routing anomalies (routing intransitivity) a problem
  - Nodes **disagree about the liveness of other nodes**
Effect of Routing Anomalies

- Routing anomalies cause nodes to disagree on membership
  - Objects on disputed nodes **may be inaccessible**

- Example: DHT lookup inconsistency
  - Overlay route locates object
  - Direct return path fails
  - Failure is permanent until node churn creates a new owner
Effect of Routing Anomalies

- Routing anomalies cause nodes to disagree on membership
  - Objects on disputed nodes may be inaccessible

- Example: DHT lookup inconsistency
  - Overlay route locates object
  - Direct return path fails
  - Failure is permanent until node churn creates a new owner
Effect of Routing Anomalies

- Routing anomalies cause nodes to disagree on membership
  - Objects on disputed nodes may be inaccessible

- Example: DHT lookup inconsistency
  - Overlay route locates object
  - Direct return path fails
  - Failure is permanent until node churn creates a new owner
Routing Anomalies: Solution

• Liveness messages forwarded using source routing [DSR, IP]

• Nodes advertise best routes to other nodes
  • If direct path fails, route through another node

• With source routing, we see about 8% indirect links in PlanetLab ring
Routing Anomalies: Solution

- Liveness messages forwarded using source routing [DSR, IP]
- Nodes advertise best routes to other nodes
  - If direct path fails, route through another node
- With source routing, we see about 8% indirect links in PlanetLab ring
Challenge: Correlated Failures

- Initially assumed **diverse node population**
  - Independent failure probability

- But many sources of correlated failures
  - DNS entries
  - Possible worm attack

- Can cause data loss

- Solution: **Glacier [NSDI’05]**
  - Erasure codes and redundancy to mask failure
  - Survive 60% failure with 10x storage overhead
**Challenge: Correlated Failures**

- Initially assumed *diverse node population*
  - Independent failure probability
- But many sources of correlated failures
  - DNS entries
  - Possible worm attack
- Can cause data loss
- **Solution: Glacier [NSDI’05]**
  - Erasure codes and redundancy to mask failure
  - Survive 60% failure with 10x storage overhead
Initially assumed diverse node population
  • Independent failure probability

But many sources of correlated failures
  • DNS entries
  • Possible worm attack

Can cause data loss

Solution: **Glacier** [NSDI’05]
  • Erasure codes and redundancy to mask failure
  • Survive 60% failure with 10x storage overhead
Challenge: Resource Consumption

- Studied hard drive growth vs. data creation rate
  - Determined sufficient space

- But did not anticipate spam explosion

- After 6 months, 75% garbage
  - Sufficient space, but high bandwidth
  - Maintaining replicas of garbage

- Solution: Lease-based storage
  - Renew useful objects
  - Avoids insecure delete operation
Challenge: Unsynchronized Clocks

• Assumed *loosely synchronized clocks*
  • Error of a few hours

• Did not hold
  • One user was *2 years behind*

• Caused user’s lease requests to fail
  • Never deleted any stored data

• Solution: *Counter-based leases*
  • Do not use absolute time
Challenge: Unsynchronized Clocks

- Assumed loosely synchronized clocks
  - Error of a few hours

- Did not hold
  - One user was 2 years behind

- Caused user’s lease requests to fail
  - Never deleted any stored data

- Solution: Counter-based leases
  - Do not use absolute time
Challenge: Unsynchronized Clocks

- Assumed *loosely synchronized clocks*
  - Error of a few hours

- Did not hold
  - One user was *2 years behind*

- Caused user’s lease requests to fail
  - Never deleted any stored data

- Solution: *Counter-based leases*
  - Do not use absolute time
Challenge: Unsynchronized Clocks

• Assumed *loosely synchronized clocks*
  • Error of a few hours

• Did not hold
  • One user was *2 years* behind

• Caused user’s lease requests to fail
  • Never deleted any stored data

• Solution: *Counter-based leases*
  • Do not use absolute time
Conclusion

• Question: Can peer-to-peer systems build reliable applications?
  • Yes!

• Built ePOST, a reliable decentralized mail system
  • Many users relied on ePOST for primary mail

• Many challenges to providing reliable service
  • Network partitions, routing anomalies, ...

• Challenges and techniques applicable to other systems
  • Human time-scale events, eventual consistency
  • Instant messaging, whiteboards, newsgroups, blogs, ...
Questions?

http://www.epostmail.org

Thanks to all of the ePOST users!