Secure Distributed Storage in Peer-to-peer networks

Øyvind Hanssen

07.02.2007

Motivation

- Mobile and ubiquitous computing
  - Persistent information in untrusted networks
- Sharing of storage and information
  - But privacy and integrity
- Digital archiving
  - Very durable storage
  - Very robust storage
  - But high availability
- Scalability
  - Global network...

Outline

- Peer to peer computing
  - Infrastructure, overlays ...
  - Structured vs. unstructured
- Structured overlays (distributed hash tables)
  - Example: Pastry
  - (Other: Chord, CAN, Tapestry, etc.)
  - Some security issues
- Secure Storage
  - Challenges
  - Techniques: Cryptographic, byzantine agreement
  - Examples: Past, Oceanstore, Pesto, Pacisso,

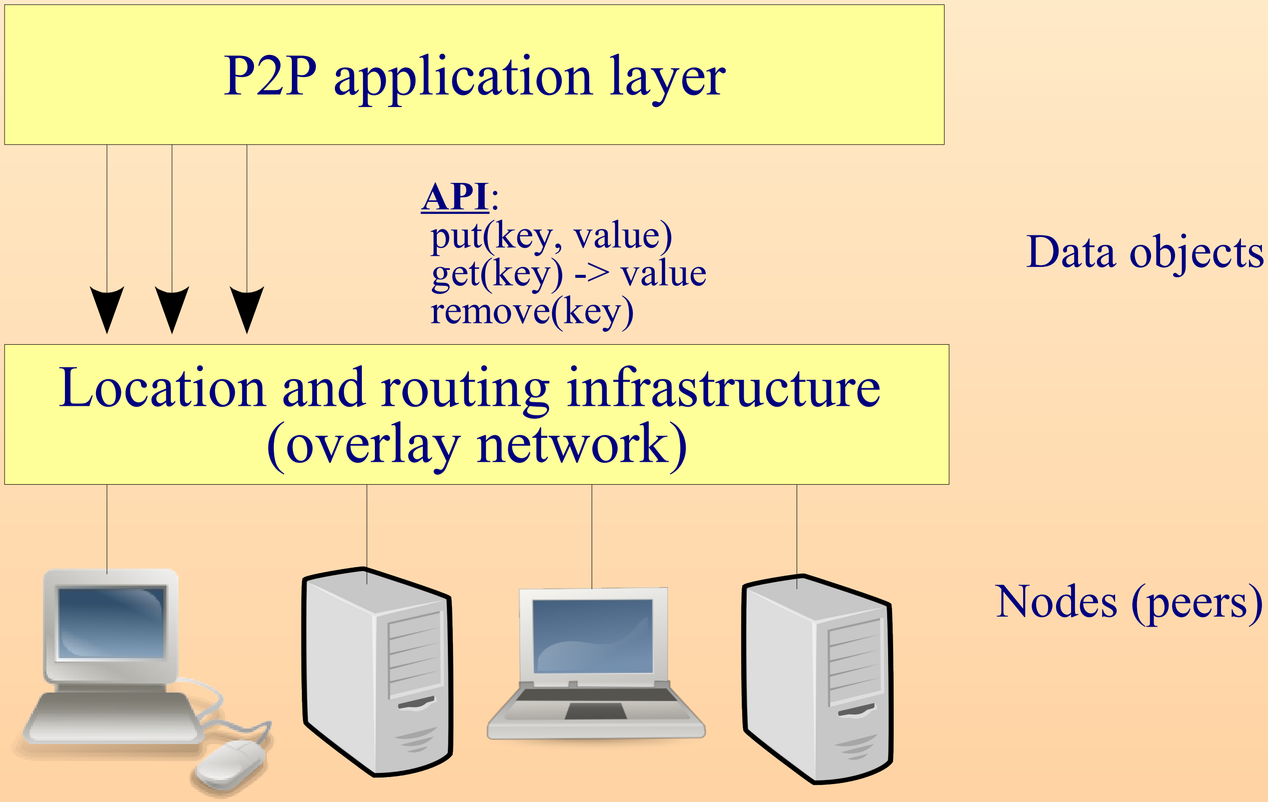
What is P2P computing?

- Different definitions in litterature
  - Strictest: Totally distributed system in which all nodes are completely equivalent
  - “...class of applications that take advantage of resources ... available at the edges of the internet” (Shirky, 2000)
  - “...the sharing of computer resources and services by direct exchange between systems” (Milojicic et.al 2002)
  - “... interconnected nodes able to self-organize into network topologies with the purpose of sharing resources ... capable of adapting to failures... without requiring the intermediation or support of a global centralized server or authority” (Androutsellis & Spinnellis, 2004)

P2P applications

- Communication and collaboration
  - E.g. ICQ, Jabber, Skype
- Distributed computation
  - E.g. SetiAtHome
- Internet service support
  - E.g. Multicast systems
- Database systems
  - Queries, semantic web etc..
- Content distribution
  - File sharing
  - Storage systems (focus: persistence, security)

Infrastrucure



Overlay networks

- Centralization
  - Purely decentralized
    - *All nodes are equal*
  - Partially centralized
    - *Some nodes are “more equal than others”*
    - *But there should be no single points of failure*
  - Hybrid decentralized
    - *Central servers*
- Network structure
  - Unstructured
    - *Loose rules, ad hoc*
  - Structured
    - *Content placed deterministically at locations*

Network structure

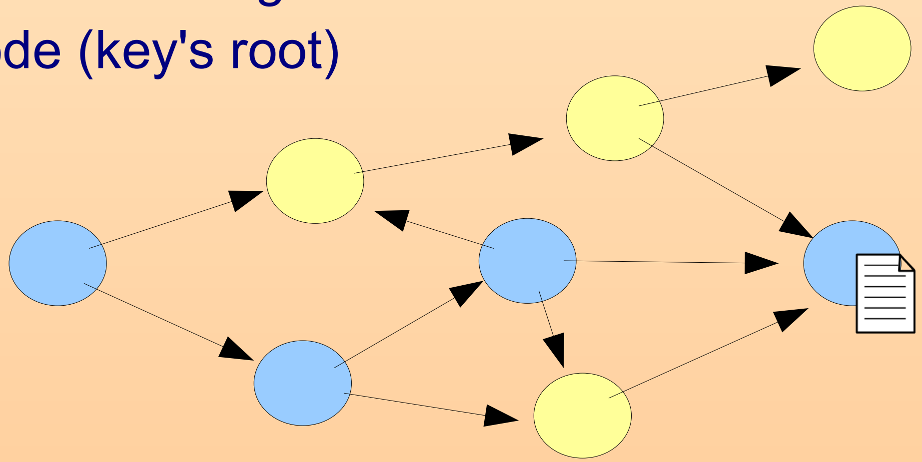
- Unstructured P2P
  - Typically: Flooding to send queries
  - Good for popular items, bad for rare items
  - Cannot guarantee that item is found
- Structured P2P
  - Distributed Hash Tables
  - Efficient location of rare items, some overhead for popular items
  - Can guarantee that item is found
  - Scalable

Outline

- Peer to peer computing
  - Infrastructure, overlays ...
  - Structured vs. unstructured
- Structured overlays (distributed hash tables)
  - Example: Pastry
  - (<u>Other</u>: Chord, CAN, Tapestry, etc.)
  - Some security issues
- Secure Storage
  - Challenges
  - Techniques: Cryptographic, byzantine agreement
  - Examples: Past, Oceanstore, Pesto, Pacisso,

Distributed hash tables

- Goal: Locate data objects identities to nodes
- Uniform "random" identifiers
  - Assigned to nodes (nodeId)
  - Assigned to application objects (keys)
- Routing
  - Each node has a routing table and neighbour set
  - Collectively maps key to node (key's root)
- Replica function



Pastry

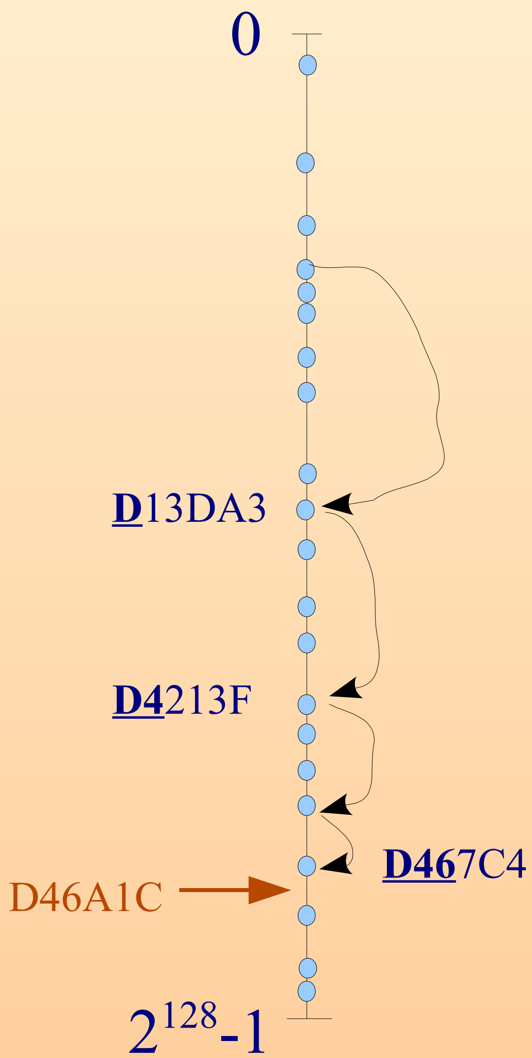
- Nodeids/data keys
  - 128 bit
  - Sequence of digits with base 2^b
- Routing table
  - 2^b columns, $128/2^b$ rows (typically 16x8)
  - Each entry contains IP address of node.
    - *Try to select one which is "nearby"*
  - In addition: A neighbour set (+- l/2 nodeId's. l depends on N)

Prefix routing (Pastry)

Routing table for nodeId 65A1xxxx

0	1	2	3	4	5		7	8	9	A	B	C	D	E	F
60	61	62	63	64		66	67	68	69	6A	6B	6C	6D	6E	6F
650	651	652	653	654	655	656	657	658	659		65B	65C	65D	65E	65F
65A0		65A2	65A3	65A4	65A5	65A6	65A7	65A8	65A9	65AA	65AB	65AC	65AD	65DE	65AF
..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..
..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..
..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..
..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..

Routing



- Each step: At least one more digit
- If no entry found, try a node which is numerically closer (neighbour list).
  - Random, with some preference for "nearby" nodes.
- If not found, we have reached the destination.
- $O(\log_{16} N)$ hops

Security issues in DHT

- Routing attacks
  - Incorrect lookup
  - Incorrect routing updates
  - Partition
- Storage and retrieval attacks
  - Deny existence of data, refuse to serve
  - <u>Censorship</u>: Take control of all replica roots
  - <u>Solution</u>: secure/verifiable nodeId assignment
  - Sybil attack. Attacker gets multiple nodeId's
- Misc. attacks
  - Inconsistent behaviour
  - Overload targeted nodes
  - Trick system into unnecessary rebalancing
  - Unsolicited response messages

Outline

- Peer to peer computing
  - Infrastructure, overlays ...
  - Structured vs. unstructured
- Structured overlays (distributed hash tables)
  - Example: Pastry
  - (Other: Chord, CAN, Tapestry, etc.)
  - Some security issues
- Secure Storage
  - Challenges
  - Techniques: Cryptographic, byzantine agreement
  - Examples: Past, Oceanstore, Pesto, Pacisso,

Challenges

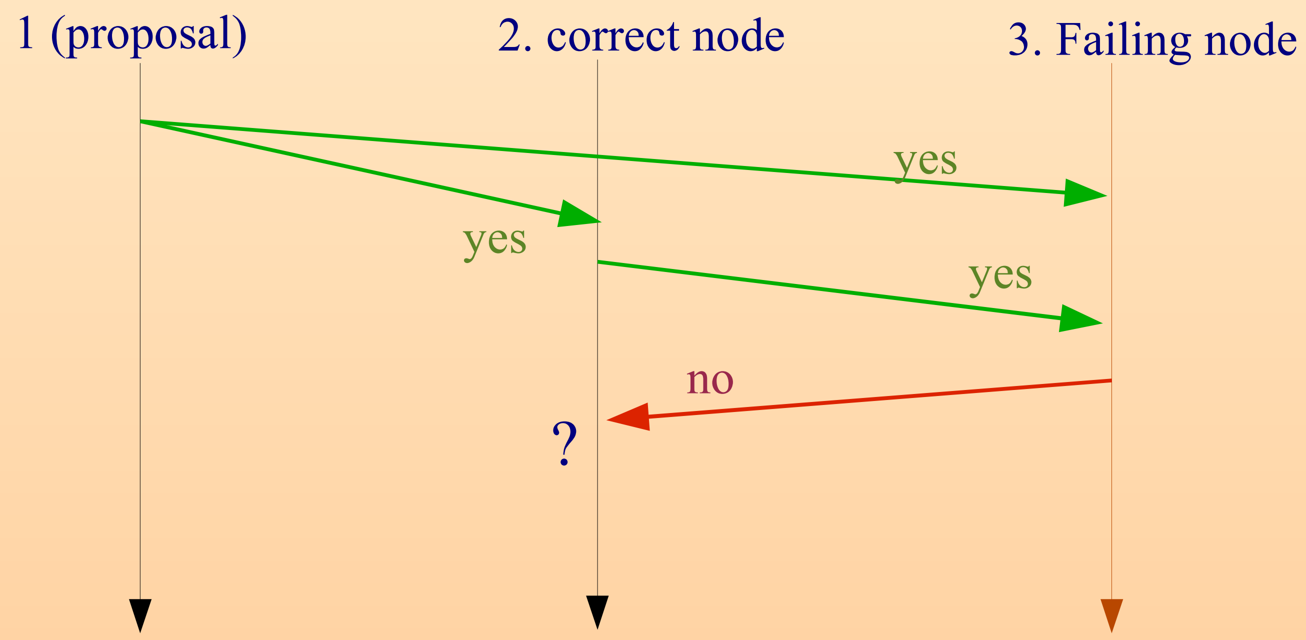
- Availability and durability
- Consistency among updates and replicas
- Security on top of <u>untrusted</u> P2P network
  - Secure storage: Privacy and integrity
  - Authorisation without central authority
  - Authentication without central authority

Basic mechanisms

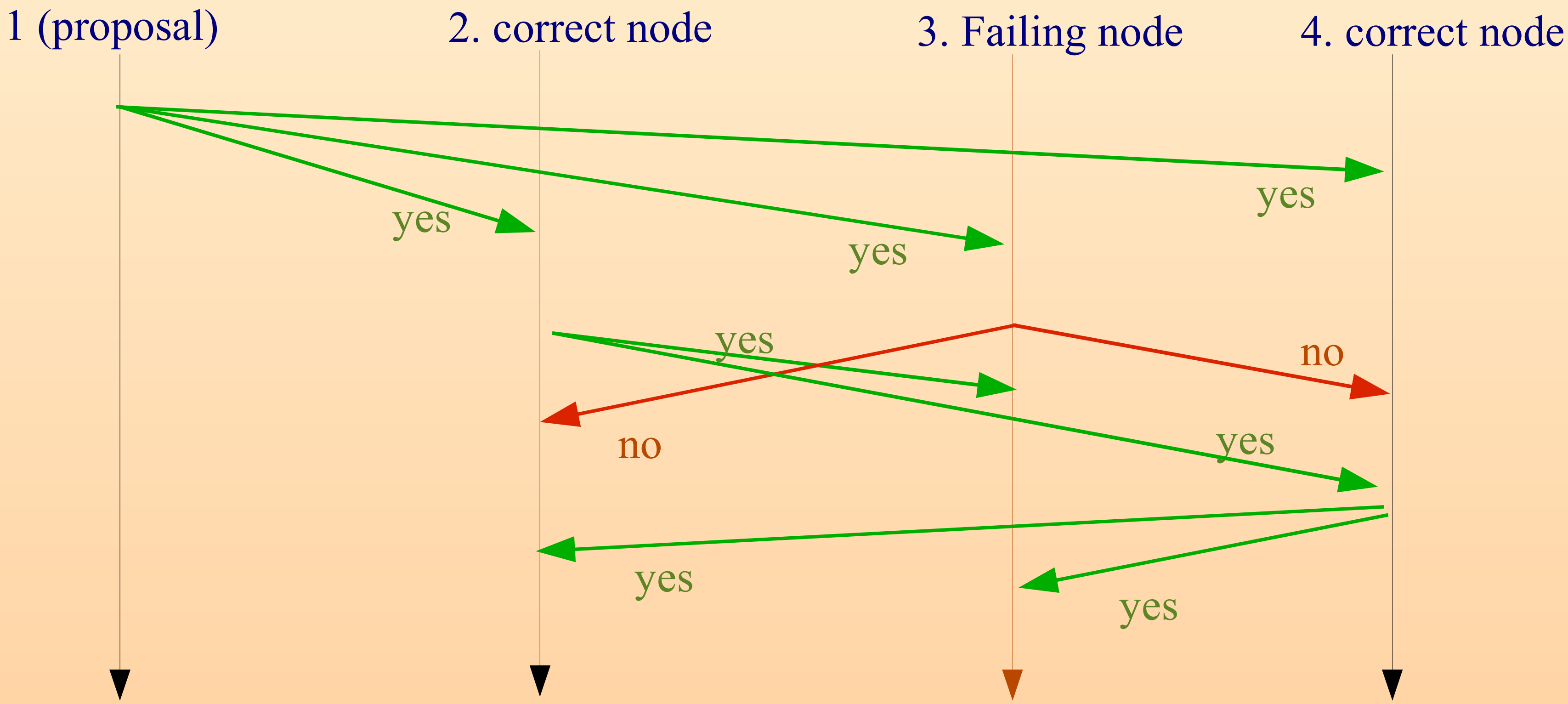
- Cryptography
  - Symmetric crypto
    - *Same key for encrypting and decrypting*
  - Asymmetric crypto (or public-key crypto)
    - *<u>Two keys</u>: One for encrypting and one for decrypting*
    - *One key is <u>public</u> and one is <u>private</u> (kept secret)*
    - *<u>Encrypt</u>: Encrypt with public key.*
    - *<u>Sign</u>: Encrypt with private key.*
- Certificate
  - A signed statement
- Secure hash
  - Difficult to reproduce a given hash value by modifying content content
  - (one way function)

Byzantine agreement

- Consensus, despite failing participants...
- Solvable if no more than m of n = 3m+1 are faulty



Byzantine agreement



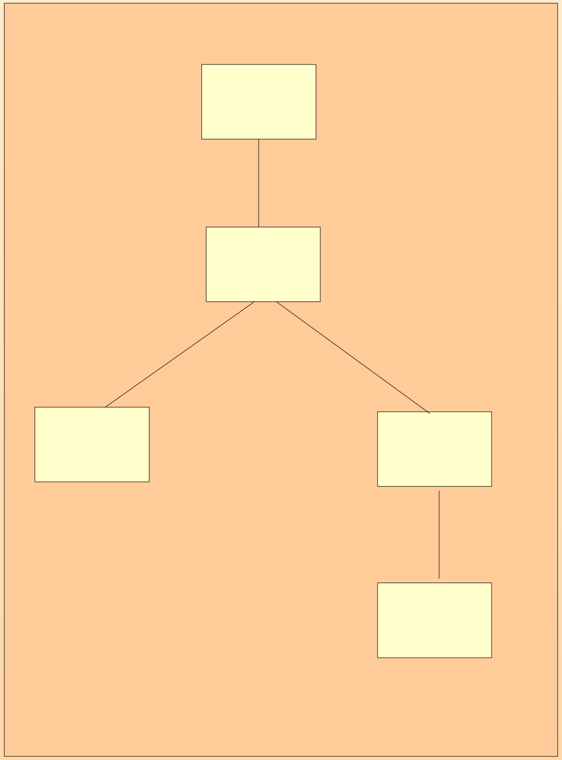
Some techniques

- Encrypted data
  - Predicates: compare-version, compare size, compare-block, search
  - Operations: replace-block, insert-block, delete-block, append
- Self certifying data
  - Secure hash and possibly a signature
- Information dispersal / erasure coding
  - Encode files into m blocks where any n < m blocks are sufficient to reproduce them. More efficient than simple replication.
- Shamir's Secret sharing
  - A secret key K can be split into a number of shares. Any subset of size k can reproduce K. k-1 shares can not reproduce K.
  - Can be combined with mutual signing protocols
- Smartcards

Past w/smartcards

- Based on Pastry
- Smartcards
  - Each node, each user
  - private/public key
  - Certificate - signed by issuer (broker)
  - Maintain storage quotas (enforce contract)
- Files
  - Immutable ...
  - FileID (160 bit)– secure hash of filename, owners public key.
    - *128 most significant bits used to locate node*
  - File certificate:
    - *FileID, replication factor, date, secure hash of content*
    - *Signed by owner (owner's smartcard!)*
  - Reclaim certificate:
    - *Storage of FileID can be reclaimed*

Immutable Objects



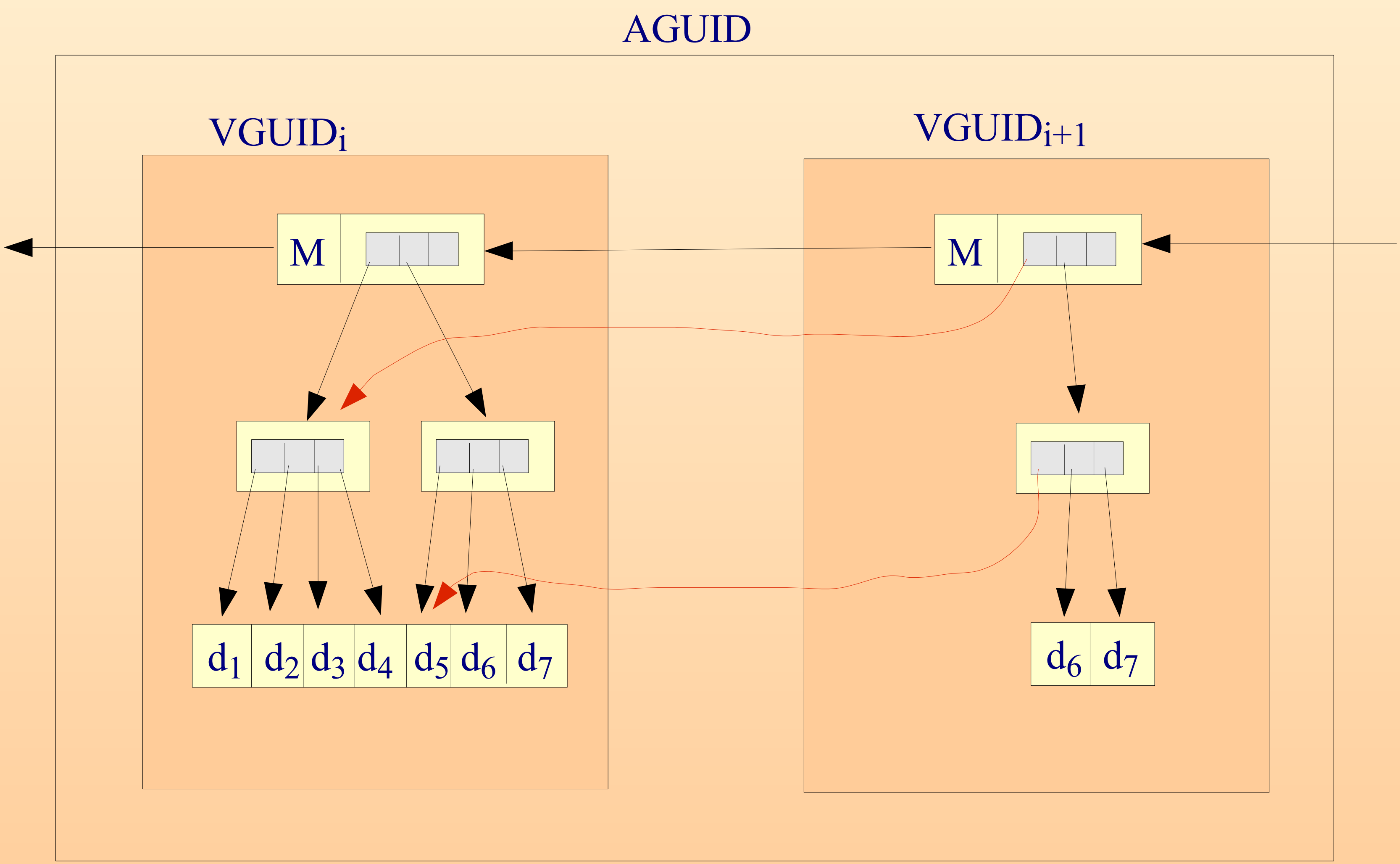
- Mutable files by having multiple versions.
- Simplifies some issues related tocaching and replication.
- Update – write a new version

- What is the latest valid version?
- Consistency, serialisability requirements?

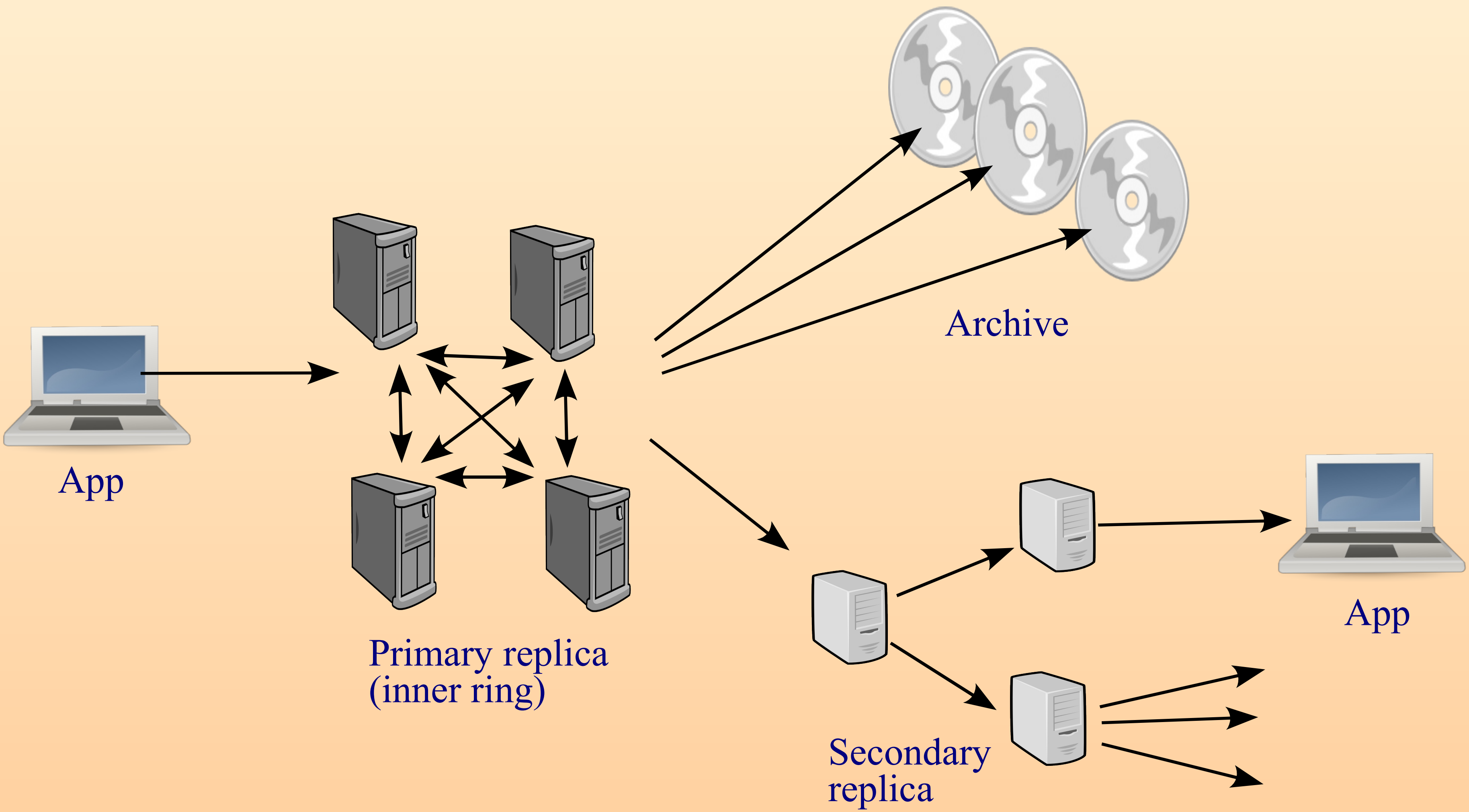
OceanStore/Pond

- Durability, availability, flexible update-semantics..
- Some highlights
  - Built on top of Tapestry (similar to Pastry)
  - Versioning
  - Erasure coding for storage + secondary replicas and caching
  - Uses cryptography and digital certificates
  - Updates: List of predicate/action pairs
  - Each data object assigned an “<u>inner ring</u>” of nodes
    - *Primary replica and update semantics*
    - *Byzantine agreement protocol*
    - *Private key sharing*
    - *Proactive threshold signature scheme (replace private key shares)*

OceanStore/Pond



OceanStore Update



Other approaches

- Pesto
  - User-User contracts (outside Pesto)
  - User decides whom to “trust” for specific tasks
  - Symmetric crypto
- Pacisso
  - Access control by “gatekeeper” nodes
  - Key-sharing, byzantine agreement ...
- Plutus
  - Lazy revocation, key-rotation...

... and more

Conclusions

- Second generation P2P overlays
  - Analogy: Distributed hash table
  - Provides deterministic routing and randomized placement
  - Can support replication, locality, etc..
  - Security issues mostly denial of service...
- Secure storage systems on top of overlays
  - Hard to achieve without some central/trusted components or trusted authorities
    - *Smartcards, PKI's*
    - *Trusted groups of nodes instead of single nodes*
  - Cryptographic methods
    - *Key management*
  - Replication, redundant encoding
  - Versioning, file block level replication
  - Another layer?

Litterature

- S. Androutsellis-Theotokis, D. Spinellis, A Survey of Peer-to-Peer Content Distribution Technologies, *ACM Computing Surveys, Vol. 36, No. 4, December 2004, pp. 335-371*
- E.K. Lua, J. Crowcroft, M. Pias, R. Sharma, S. Lim, A Survey and Comparison of Peer-to-Peer Overlay Network Schemes, I*EEE Communications Survey and Tutorial, March 2004*.
- D.S. Wallach, A Survey of Peer-to-Peer Security Issues, in *Proc. Intl. Symposium on Software Security, November 2002*.
- W. Sit, R. Morris, Security Considerations for peer-to-peer distributed hash tables, in *Proc. 2nd Intl. Workshop on Peer-to-Peer Systems*.
- P. Druschel, A. Rowstron, PAST: A large-scale, persistent peer-to-peer storage utility, In *Proc. 8th Workshop on Hot Topic in Operating Systems, 2001*
- J. Kubiatowicz, Extracting Guarantees from Chaos, *CACM, February 2003*.
- J. Kubiatowicz, et. al. OceanStore: An Architecture for Global-Scale Persistent Storage, In *Proc. ACM ASPLOS, 2000*.
- S.Rhea, et. al. Pond:the OceanStore Prototype, In *Proc. 2nd Usenix Conference on File and Storage Technologies 2003*.
- F.W. Dillema, T.Stabell-Kulø, Pesto Flavoured Security, *SRDS 2003*.
- E. Coç, M. Baur, G. Caronni, PACISSO: P2P Access Control Incorporating Scalability and Self-Organization for Storage Systems, *Sun Microsystems SMLI TR-2007-167, June 2007*