Large-scale distributed computing systems

Lecture 7:

Data Management

February 2014

Johan Montagnat
CNRS, I3S, MODALIS
http://www.i3s.unice.fr/~johan/

Course overview

- 1. Distributed computing and models
- 2. Remote services
- 3. Infrastructures and deployment
- 4. Workload and performance modeling
- 5. Workflows
- 6. Authentication, authorization, security
- **7. Data management**
- 8. Evaluation

Course content

- 6. Data Management
  - Distributed data management
  - Distribution, replication
  - P2P

Distributed data

- Scientific data records increase permanently
  - Astronomy / astrophysics observations
  - Satellite data, climate, atmosphere, geophysics data
  - Epidemiology data, medical records
  - Biological data, gene annotations and structure, genomes
  - Scientific instruments, *e.g.* high energy physics records
- Target PB repositories
- Usually distributed
- Potentially sensitive
- File systems limitations
  - 2^{31} bytes per file / inodes (towards 64 bytes file systems)
  - ~10000 files/directory

Requirements

- Very large scale distribution, transparent access
  - Heterogeneous formats
  - Virtualization of distributed resources
  - Coherency of remote data updates and replicated data
- Performance, scalability
  - Data transfers and data access strategies: often depend on access patterns
  - Parallel access, multiple users
- Fault tolerance
  - Resources failures and network service interruption
- Reliability
  - Long term availability of data, non repudiation
- Access control, data protection
  - Flexible access control, on-storage and on-network protection

Distributed data management

- Centralized approach: file catalogs, indexes
  - Handles heterogeneity, legacy storage
  - Direct access to data, bottlenecks (limited scalability), central point(s) of failure
  - Depend on external storage data management policies
  - Ease coherency and data protection
- Decentralized approach: peer-to-peer
  - Very scalable, no critical point, distribute dat search load
  - Strategies performance dependent on data access patterns
  - Robust, unreliable environment with many peer failures
  - Low data protection
- Hybrid centralized / decentralized
  - Replicated catalogs or P2P networks with redundancy, QoS...

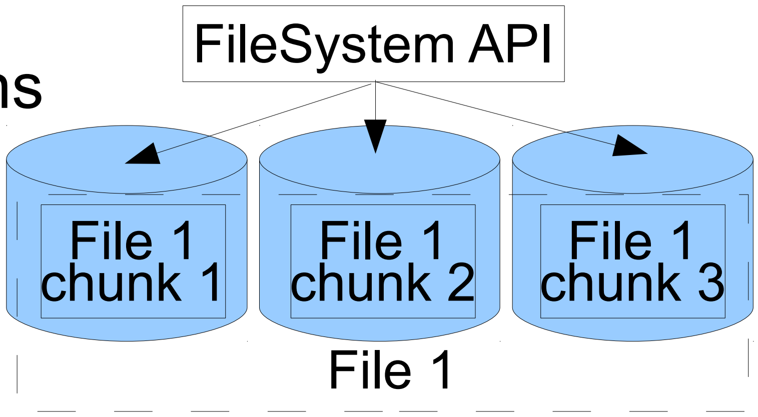
Distributed file systems

Distributed file systems

- First idea: extend existing approach (local file system) to distributed resources
- Parallel I/O
  - Performance
- Network File System
  - Network extension of file systems
- Andrew File System
  - Secured, large-scale extension with collaborative caching
- Grid File System
  - Emphasis on heterogeneity management
  - Ambitious objectives for information life-cycle management

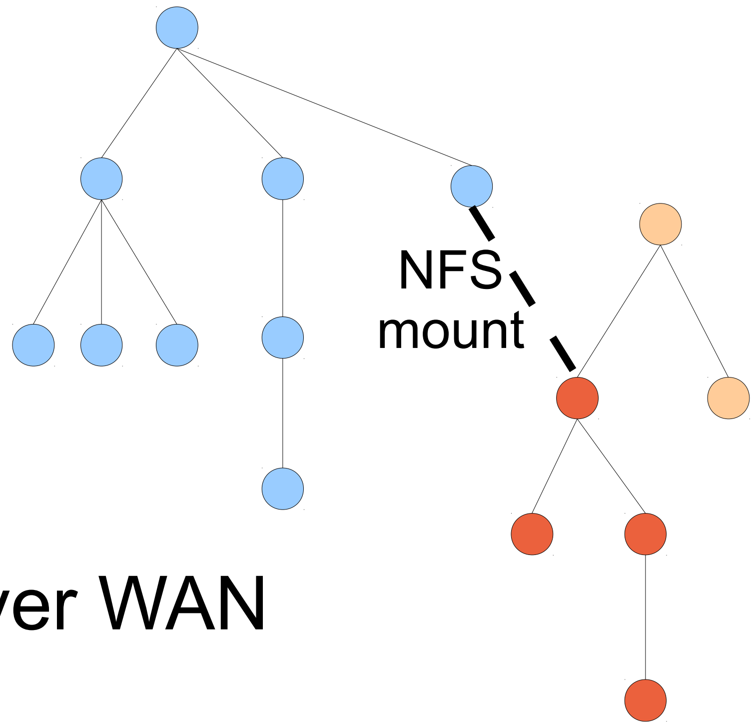
Parallel file systems

- Focus on high performance
  - Local resources with high connectivity
  - Independent industrial implementations (IBM, SGI...)
- Performance
  - Parallel I/O
  - Can be exploited by parallel programs...
  - ...or sequential programs in case of disk bus saturation
  - Dedicated architectures available
- Storage Area Network (SAN)
  - Network interfaced storage resources
  - High performance network as disk bus (fiber channel)



NFS: Network File System

- Multiple (partial) file systems viewed as one
- C/S model
  - Scalability limitation
  - Usually across LAN
- Security limitations
  - User IDs mapping?
  - Special control for UID 0 (root)
  - Transfers over WAN?
- Obvious performance limitations over WAN
  - Caches
  - Automount
  - ...

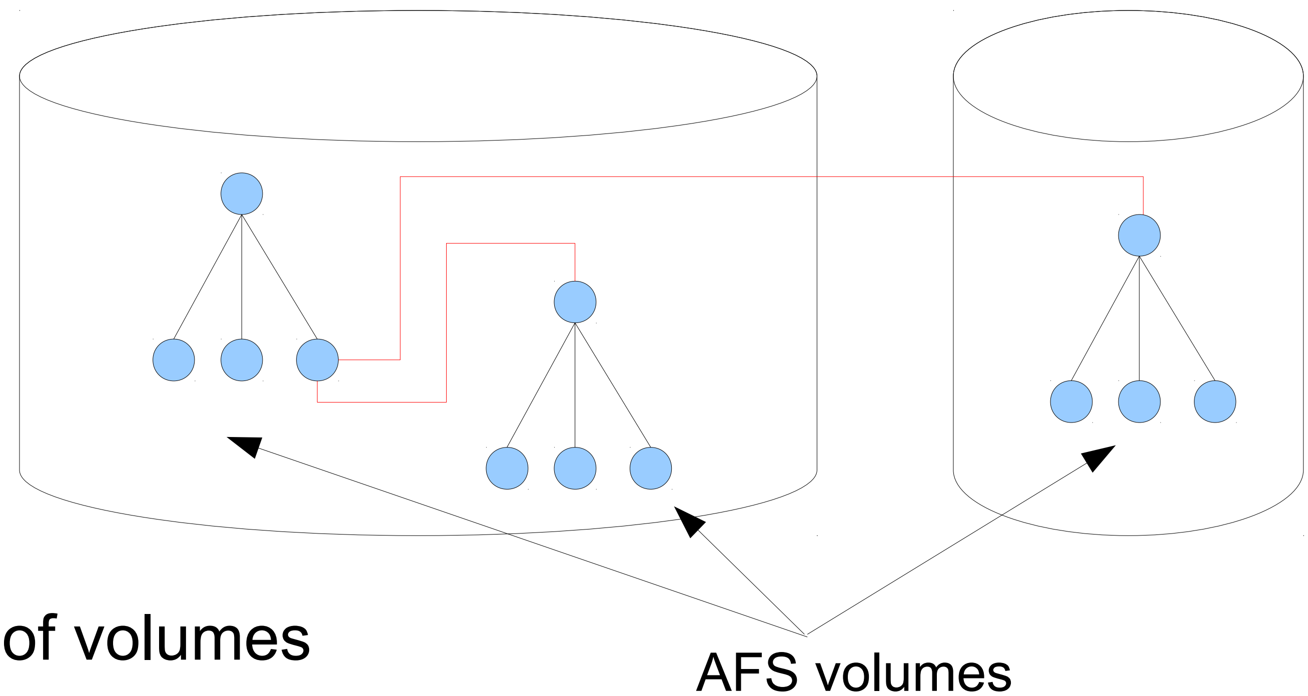


AFS: Andrew File System

- File system on the NFS model with focus on
  - Security (Kerberos authentication, ACL control)
  - Distribution (caches)
  - Scalability (tens of thousand client per cells)
- Collaborative caching
  - File locking strategy for ensuring coherency during updates (avoid too large, shared records)
  - Modification on local caches
  - Cached files listed on AFS server
  - Notification mechanisms in case of file modification to all caches (with recovery on network failure)

AFS: Andrew File System

- Space partitioning by AFS volumes
  - Files hierarchy hosted on a single storage device
  - Logical view (mountpoints, migration of volumes possible)



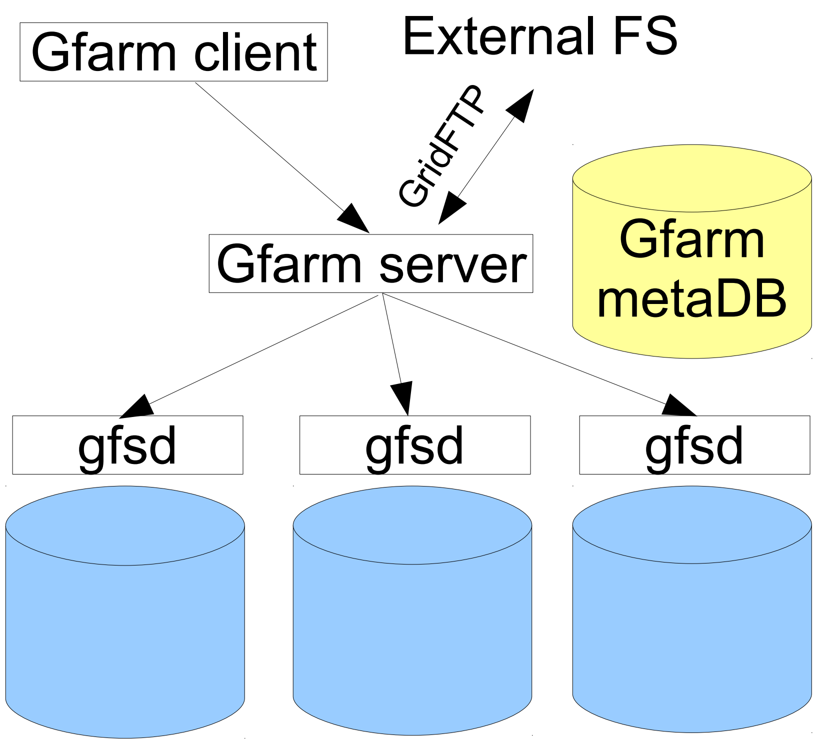
- Clones
  - Read-only copies of volumes
- Influenced NFSv4 features

GFS: Grid File System

- Open Grid Forum working group standard
- Targets
  - Standard interface for multiple resources
  - Plug-n-play resources
  - Federation of logical resource name space
  - Information lifecycle management (data placement and retention policies)
  - Object based storage
  - Context management
  - Bulk and asynchronous operations
- Storage resources virtualization
  - Abstraction layer to manage heterogeneity

GFS implementation: Gfarm

- Architecture
  - Local resource virtualization through gfarm daemon
  - Specific C/S protocol + external protocols supported (e.g. GridFTP)
  - System metadata store

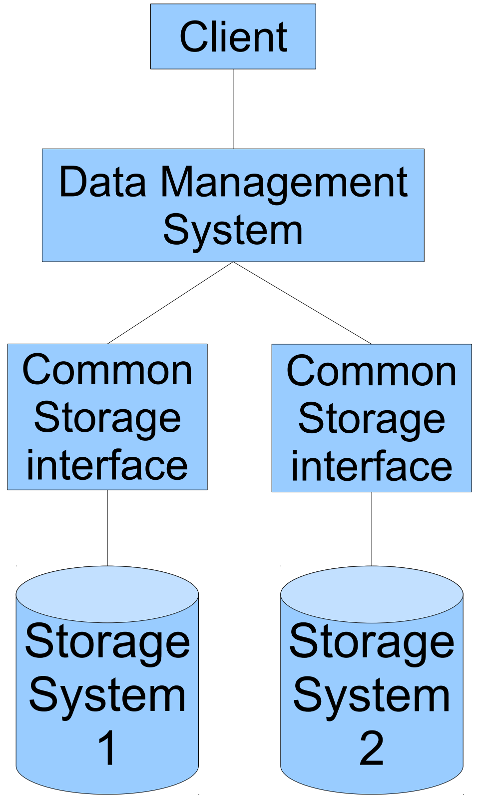


- Reliability
  - Replication
- Performance
  - Parallel IO
- Interoperability
  - Grid credentials recognized
  - FUSE component to mount on UNIX file systems

File catalogs and replication

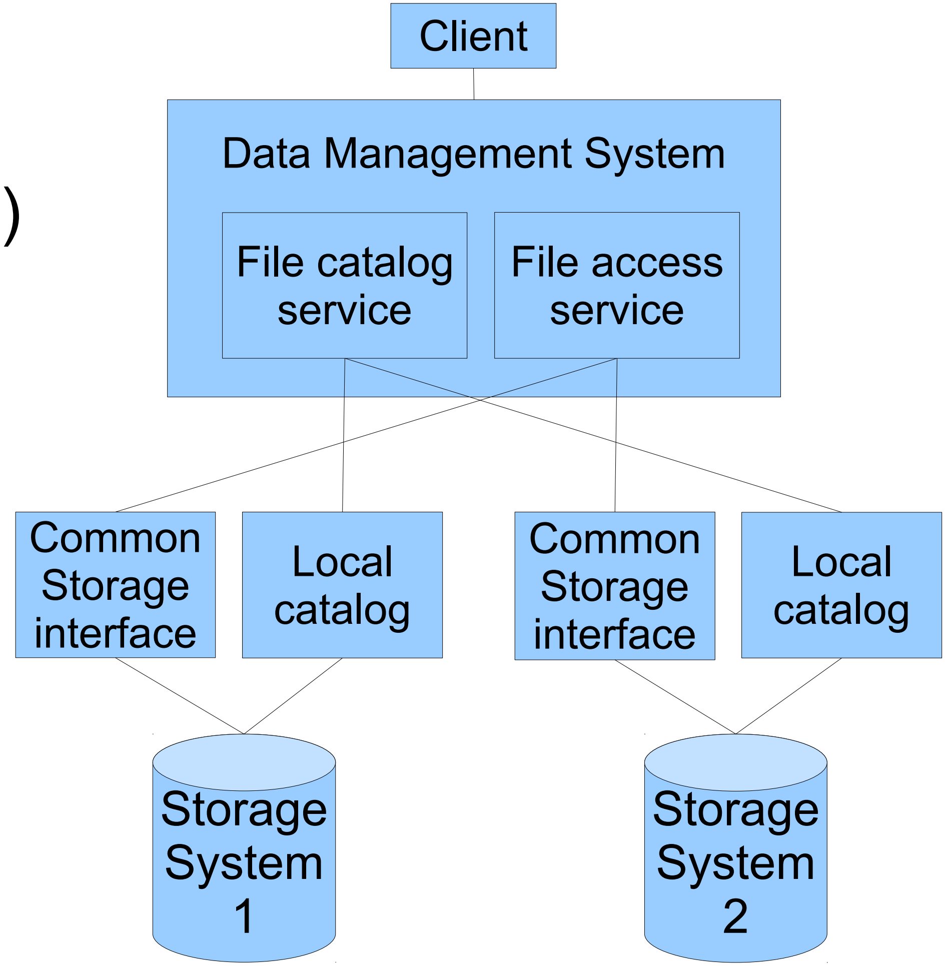
Splitting storage and data management

- Handle heterogeneity
  - Standard interface to storage resources
  - Storage service
- Manage data at a higher level: Additional services to handle:
  - Distribution, load management
  - Availability, replication
  - Performance, caching, transfers scheduling
  - ...
- Require adequate support at storage-level
  - Security (data access control, data protection)
  - Data locks...



File catalog

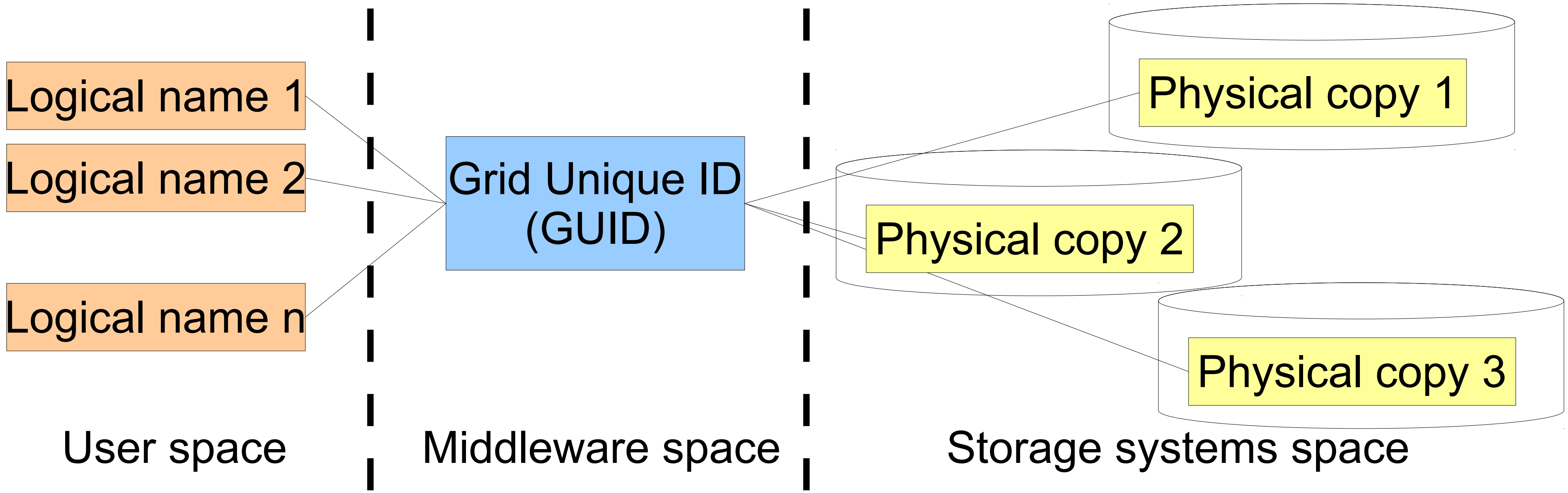
- Unique view of file hierarchy
  - (Local) sub-catalogs mapping to a single file tree view
  - Centralized entry point
- Additional services
  - Access control
  - System metadata (checksum...)
  - User-defined metadata



File replication

- Files replication over different sites enable
  - Improved performance (use closest replica)
  - Improved reliability (if a server is out of reach, replica may still be available)
  - Easy to set up (in read-only mode at least)
- Drawbacks
  - Multiple copies coherency problem
  - Define replication policies: by hand or automatic (mirror, partial mirror)
  - Does not solve the storage size granularity problem
- Distribution
  - Synchronize access controls on different storage
  - Give a logical view of several physical replica

File Replication



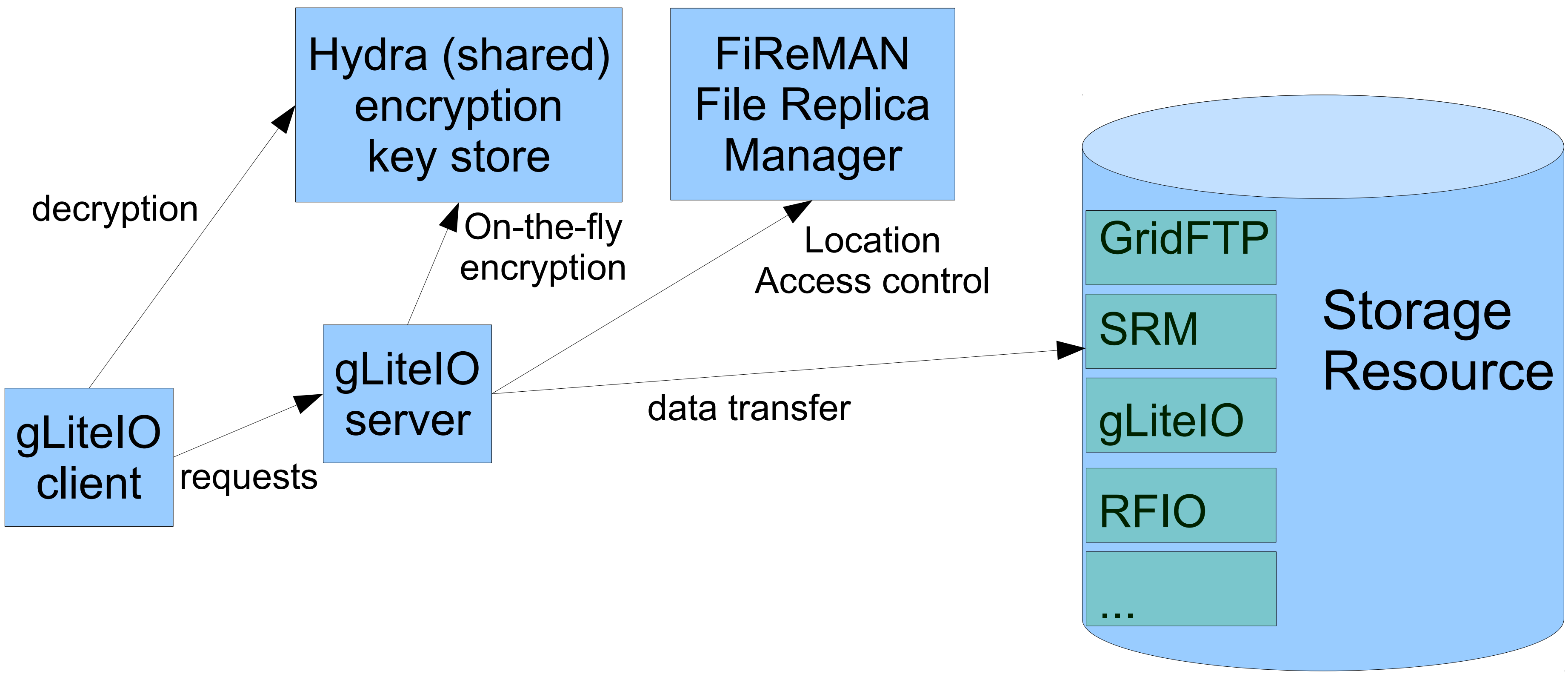
- GUID: Grid-wide Unique IDentifier (system use)
- Logical names: user names, many-to-1 association
- Physical names: URI kind (location), 1-to-many association
- File catalogs map logical, system and physical spaces

SRM: Storage Resource Manager

- SRM is an OGF standard
  - Early version 1 wide spread, but basic functionality lacking (access control...)
  - Version 2 well supported with corrections
  - Current version 3 hardly supported (complexity)
- Common interface to all storage resources
  - File access and transfer
  - Directories and space management
  - Files life time management
  - Targets large and hybrid (tape/disk) storage: space reservation, file prefetching and pinning
- Only individual storage resources management
  - Limited to local resource management
  - No high level data management policy

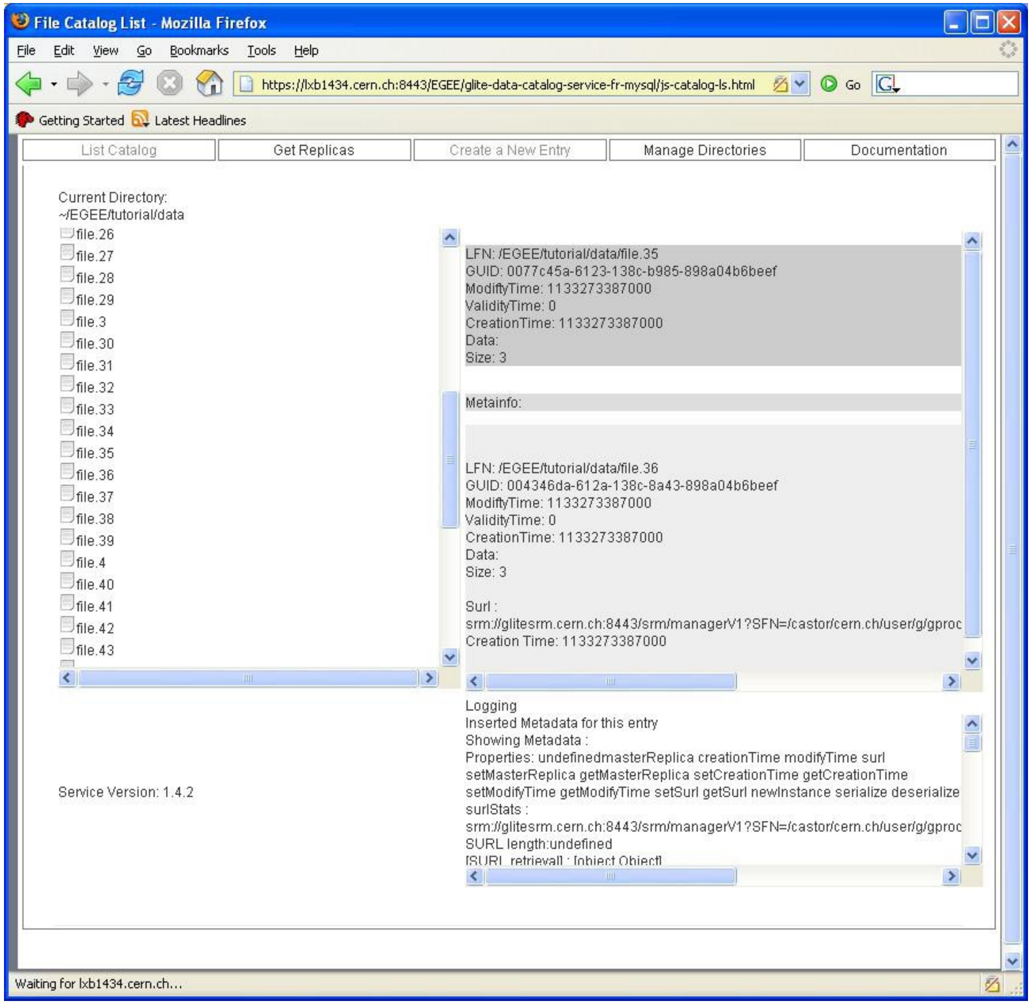
gLite Data Management System

- Collaboration of services
  - File catalog: data location and replication
  - Encryption key store: on the fly data encryption
  - gLiteIO server: IO interface and access control



The FiReMan file catalog

- FiReMan: File Replica Manager
- Resolves logical file names to GUIDS to physical location (URL) of files
- Secured access: VOMS groups, ACL support
- File attributes support (metadata indexed on files GUIDs)
- CLI and simple APIs
- Web-based interface
- Exposing interfaces suitable for matchmaking (synchronized with workload management system)



Hydra distributed key store

- Unique key generated for each file
  - GUID – file key association
- AES encryption algorithm
- Key splitting for improved security
  - Shamir shared secret algorithm
- Access control on keys based on ACLs
- Different access levels
  - Full access to file and encryption key for authorized users
  - Access to (encrypted) file but not keys for file administrators

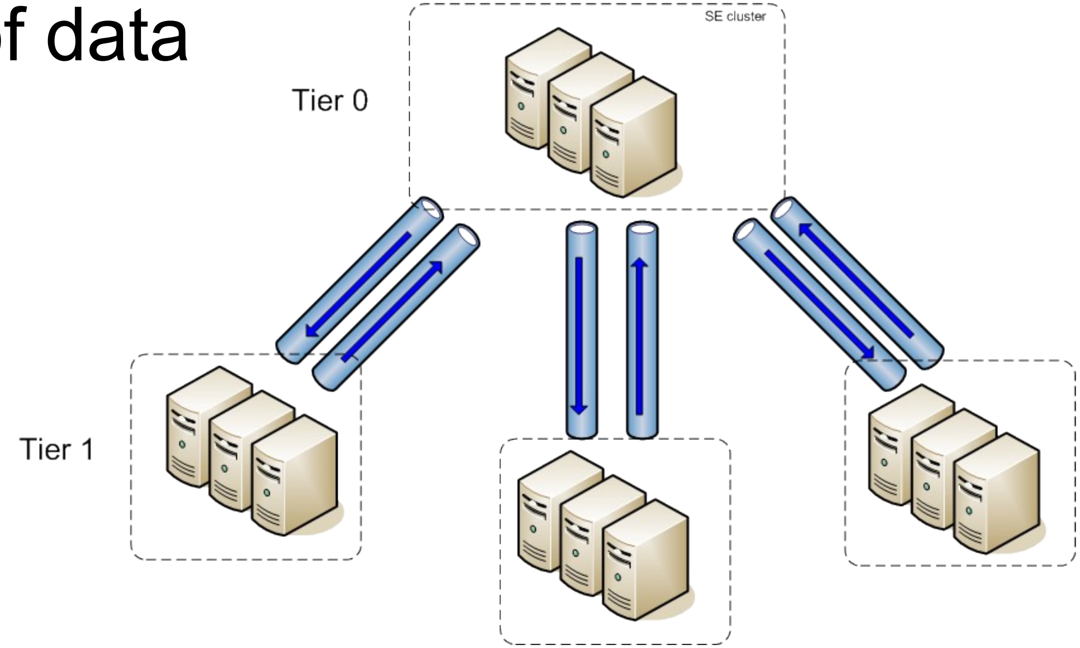
File transfer

GridFTP

- Security
  - Grid credentials-based authentication and authorization (single sign-on)
- Third-party transfer
  - Server-to-server file transfers for administration needs
- Performance
  - Multiple parallel TCP streams
- Striped
  - Data interleave
- Partial transfer
- Restart on failure
- QoS negotiation (buffer, window size)

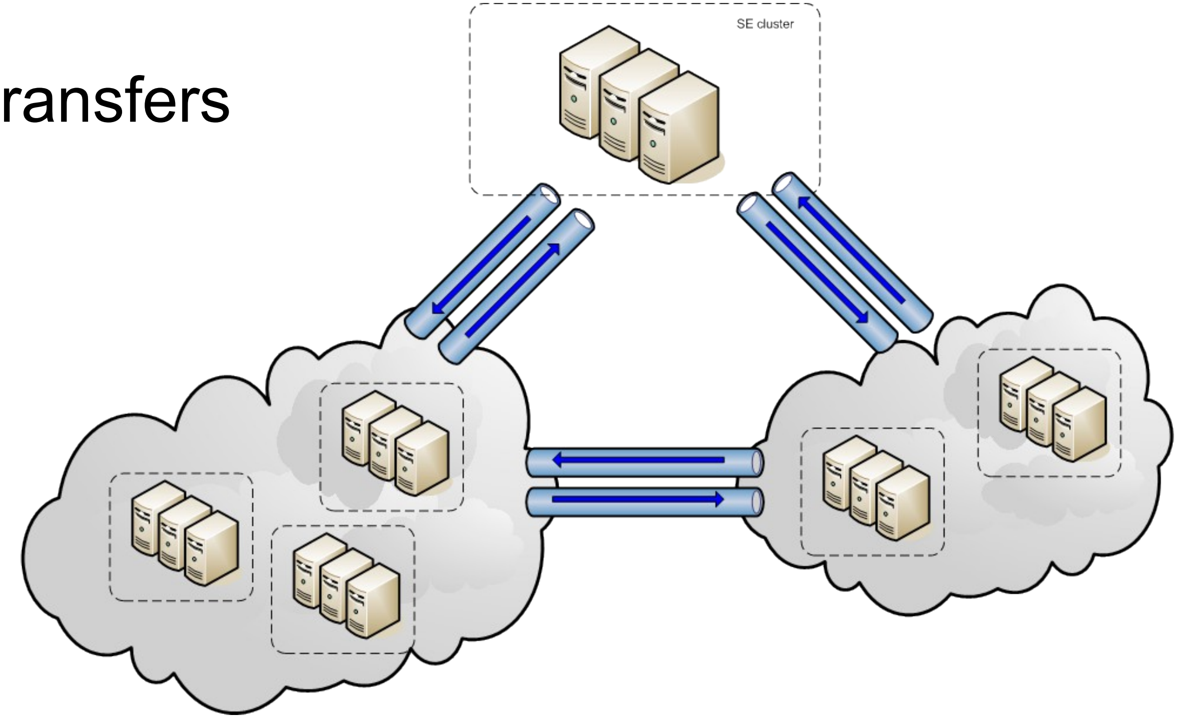
File Transfer Service

- Sequential file transfer (e.g. FTP)
  - Unfair for smaller files
- File Transfer Service
  - Supports grid credential (single sign-on) and SRM protocol
  - Scheduled transfers
  - Error recovery
  - Simplified management of data sets
- For very large amounts of data



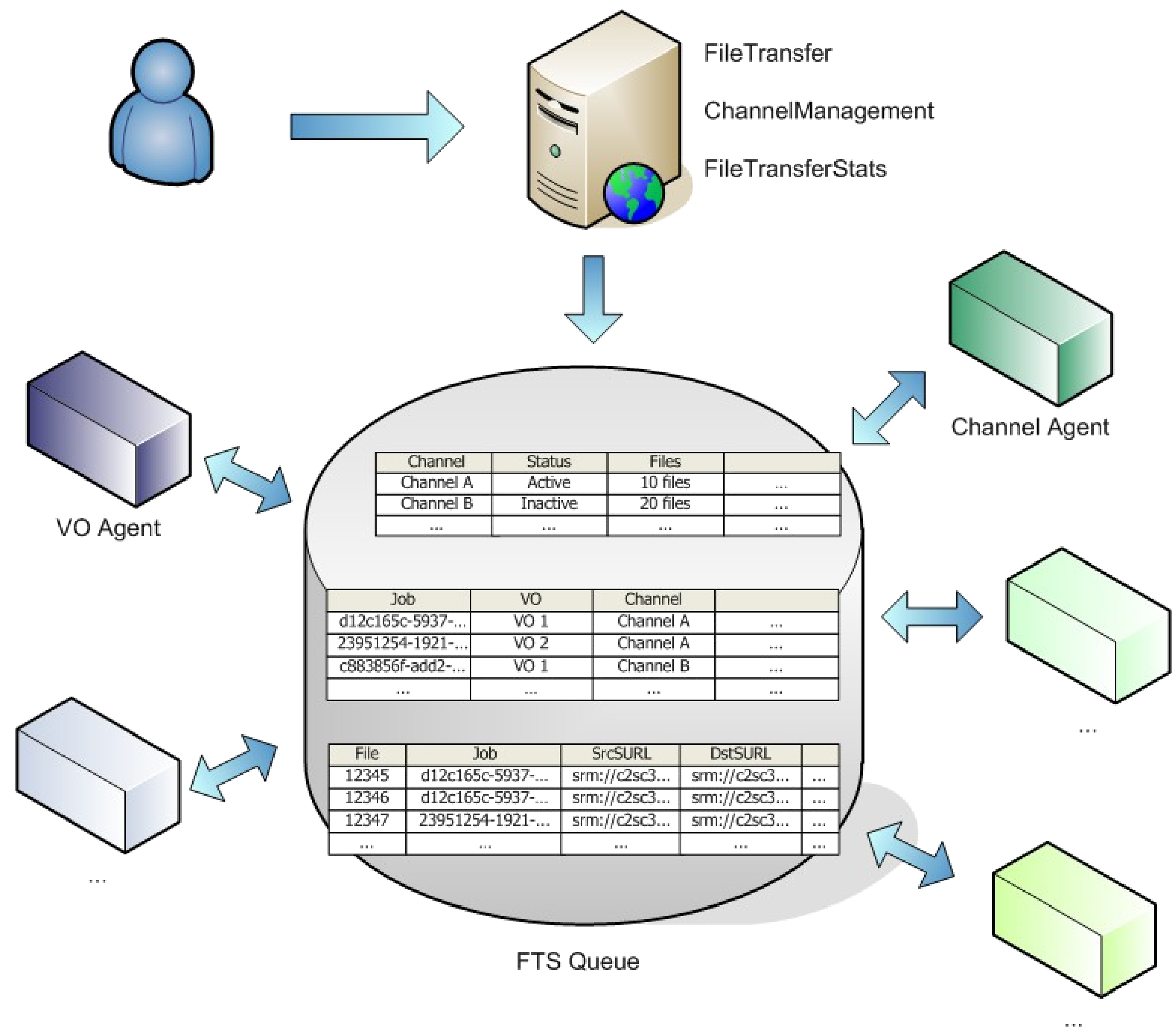
Key Concept: transfer channel

- Logical unit of management
  - Represent a directed network pipe between two sites
- Mono-directional
- Independently manageable
  - State
  - Number of streams
  - Number of concurrent transfers
- Inter-VO scheduling
  - VO share
- Site Grouping
  - Define multiple targets



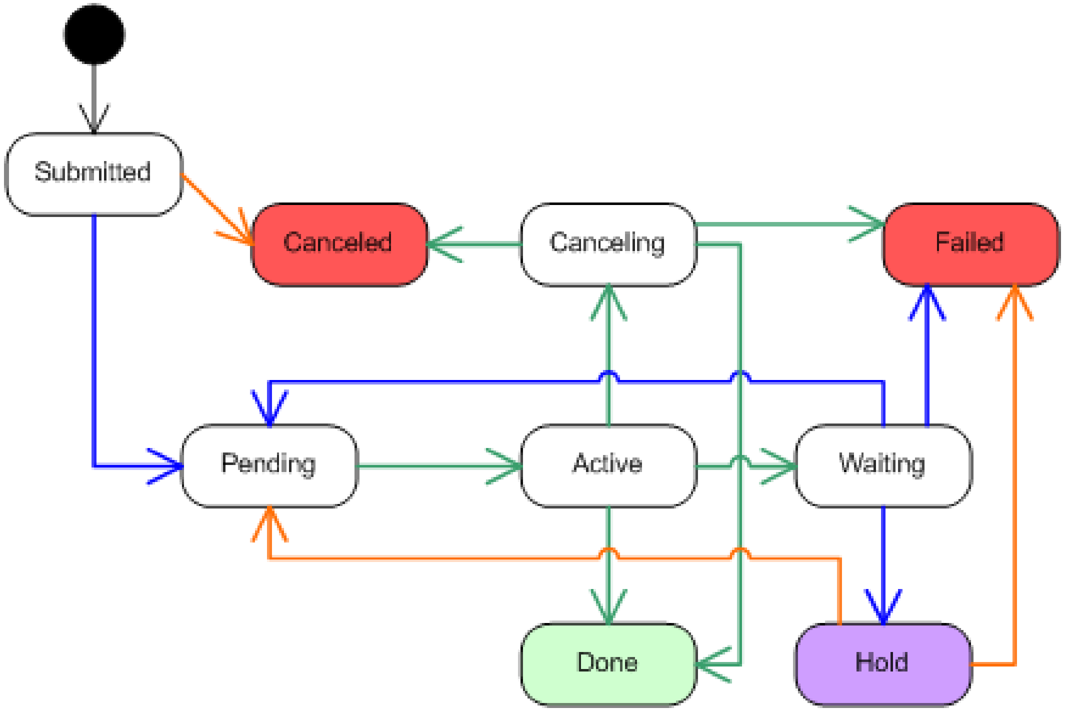
FTS Architecture

- FTS Web Service
  - User, Administration and Monitoring Interfaces
- File Transfer Queue
- File Transfer Agents
  - VO Agents
  - Channels Agents
- SOAP API and CLI



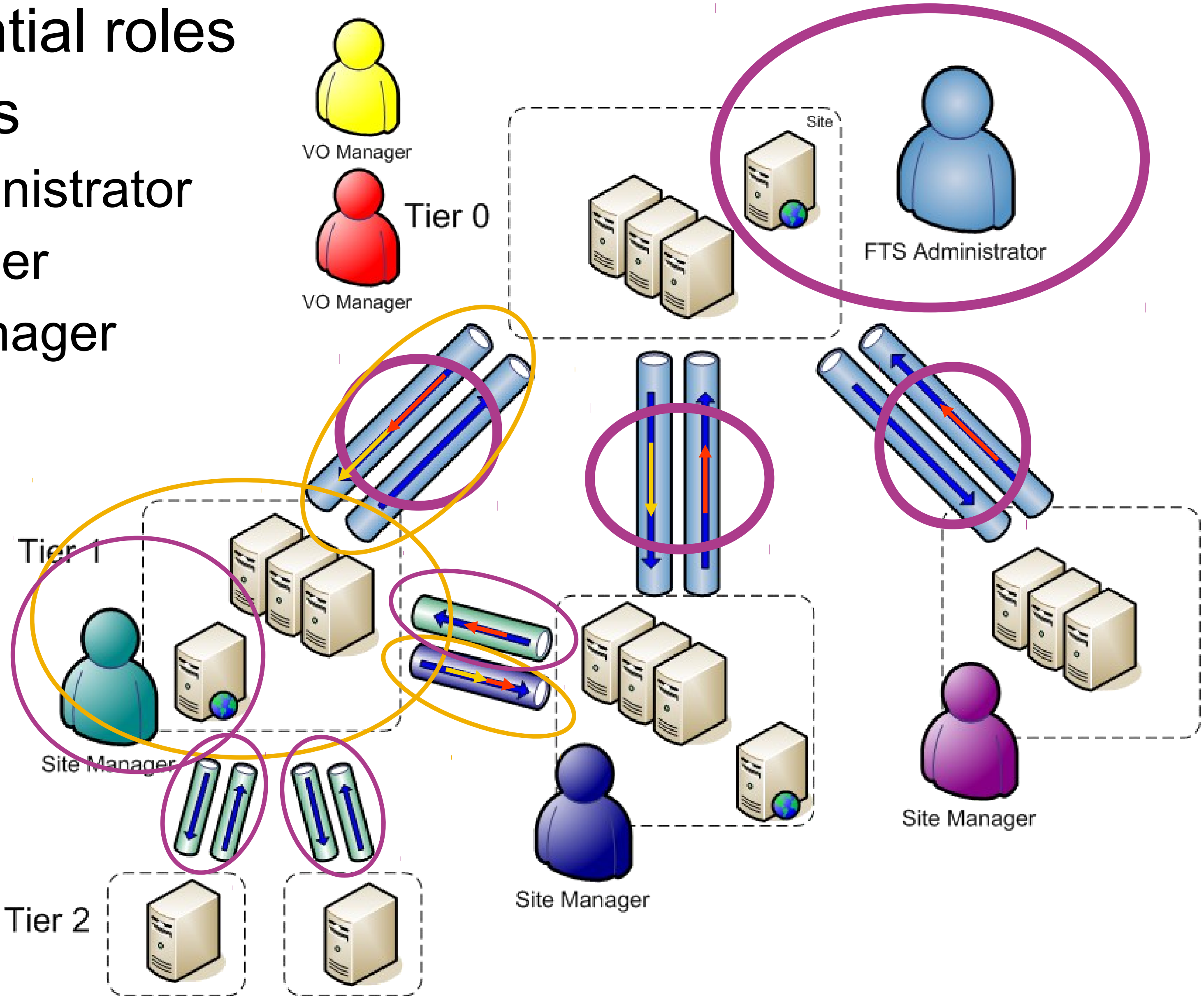
Transfer jobs

- Channel has a number of properties
  - State (Active / Inactive / Drain / Stopped / Halted)
  - Number of concurrent files transfers
- Scheduler
  - Queue of request
  - State machine



Security: Roles

- VOMS credential roles
- Different users
  - Service administrator
  - Submitter User
  - Channel Manager
  - VO Manager
  - Vetoed User



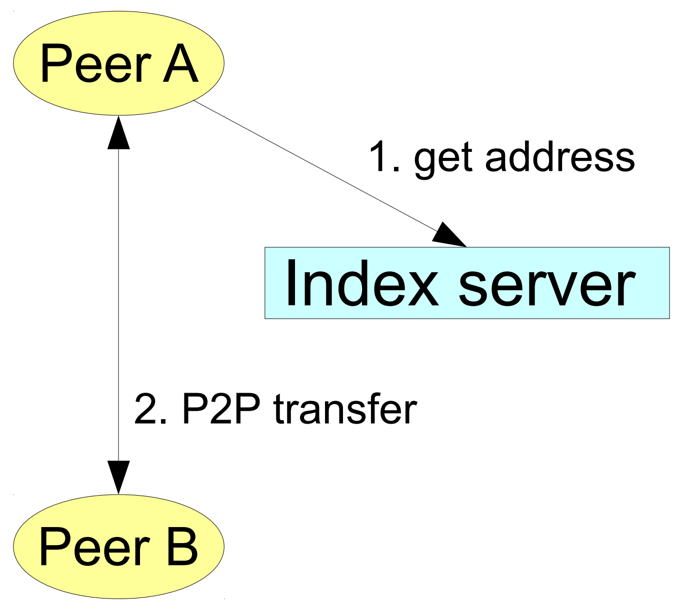
P2P networks

P2P: Peer-to-Peer

- Away from the Client-Server model
  - All peers contribute, no centralized / critical server
- Decentralization
  - Avoid single point of failure
  - Aggregate multiple resources
  - (Dynamically) extend network of participants
- Expected properties (not all compatible!)
  - Minimum data search time
  - Minimum network overhead
  - Scalability
  - Fault tolerance
  - Reliability, completeness

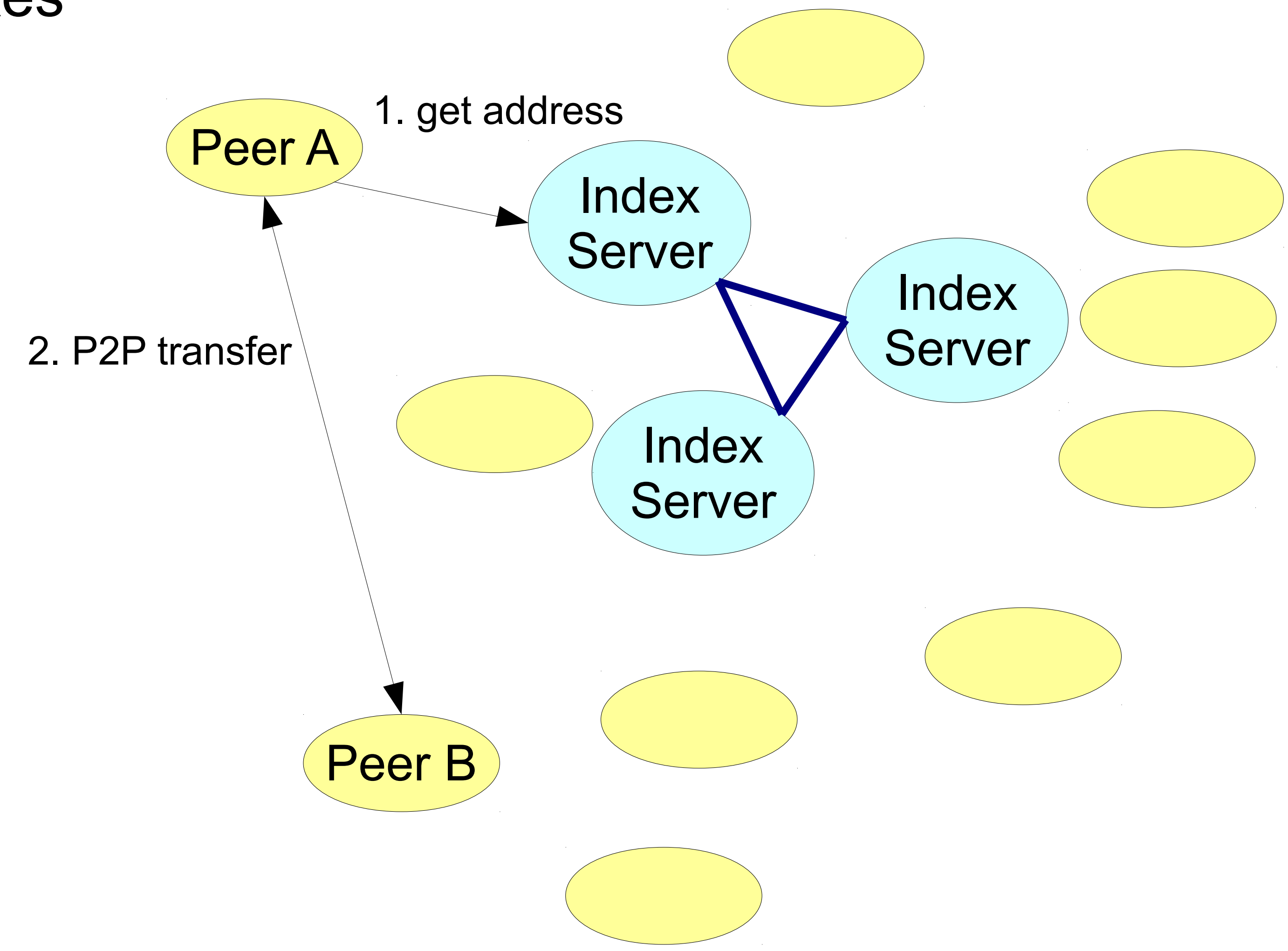
Napster: centralized index

- Centralized index
  - Central point of failure
  - Index size limitation
  - Very efficient lookup
- Peer-to-Peer data tansfers
  - Shared data delivery (no high load on servers)
  - Shared bandwidth (no bottleneck)



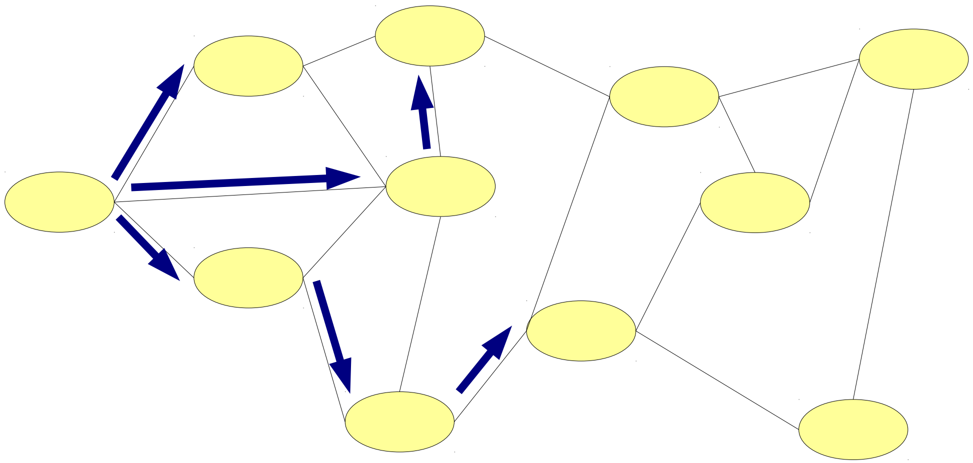
Towards centralized/decentralized

- Extend scalability
  - Replicate indexes



Gnutella: decentralized, flooding

- Each peers is connected to a couple of neighbors
  - Typically 3-4 neighbors
  - Need a bootstrap mechanism (IRC, Gnucache, ping...)
- Searches by flooding the peers network
  - Message flooding in the peer network
  - Unique IDs to detect loops
  - Maximum Time-To-Live (TTL) to limit expansion (typically 7)
  - In the order of 10 000 peers
- File transfers
  - Direct P2P
  - HTTP protocol, GET requests



Gnutella early protocol

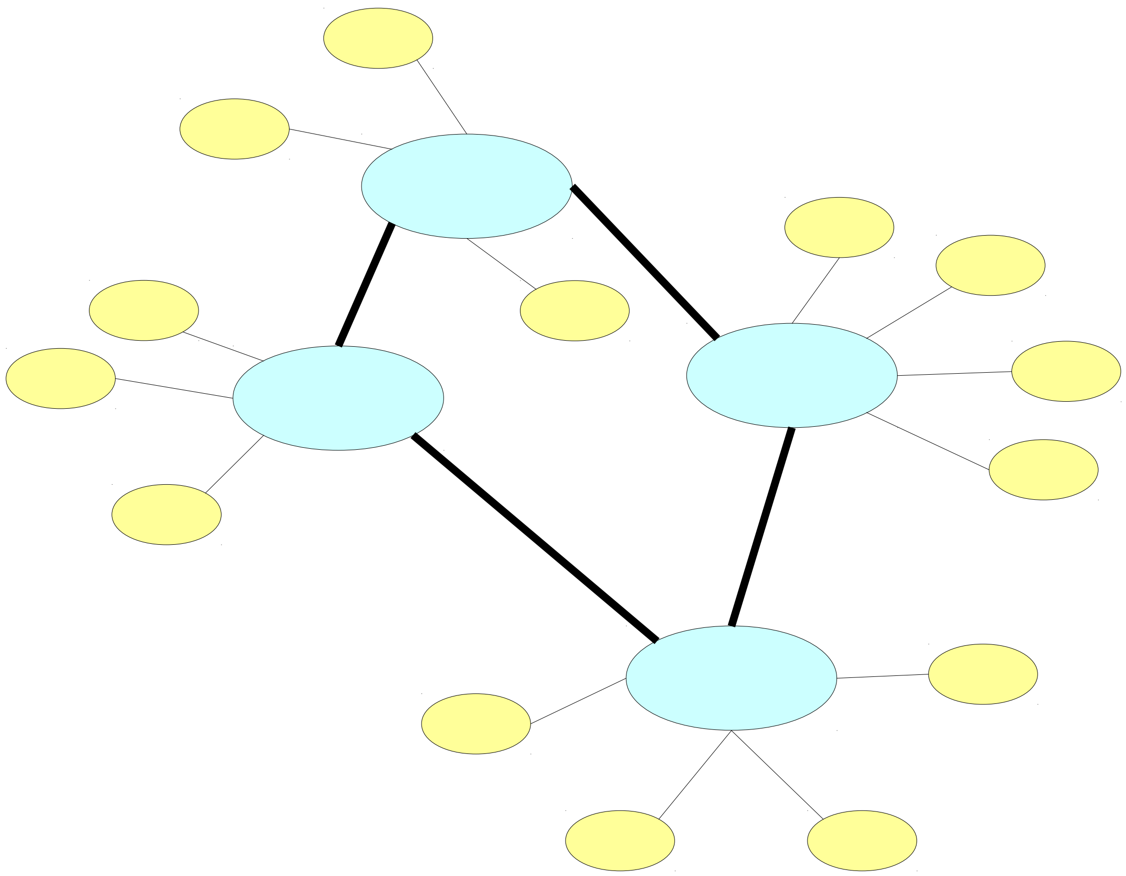
- Messages
  - No sender IP: response track back the route of requests
  - Ping: discover new peers
  - Pong: reply to ping (include responder IP/port)
  - Query: search for data
  - QueryHit: return found data (include responder IP/port)
  - Push: bypass firewalls by requesting outbound connection (include sender IP)
- Decentralization
  - Limited horizon (7 hops, ~10 000 peers), no guaranteed data retrieval
  - No routing, large communication overhead for flooding

Gnutella weaknesses

- Ad-hoc topology of the overlay network
  - No differences between physical network leafs
  - Critical bridges appear
- Network heterogeneity
  - Limited connection peers create bottlenecks
  - 56 kbits connection limitation: 560 bytes query x 10 queries / seconds x 3 peers makes more than 25% of the traffic
- Query length
  - Multiply connection latencies, affected by TCP/IP timeouts
- No load balancing
  - In practice, it is found that two third of users do not serve files
  - 1% of host serve 37% of files (20% serve 98%)

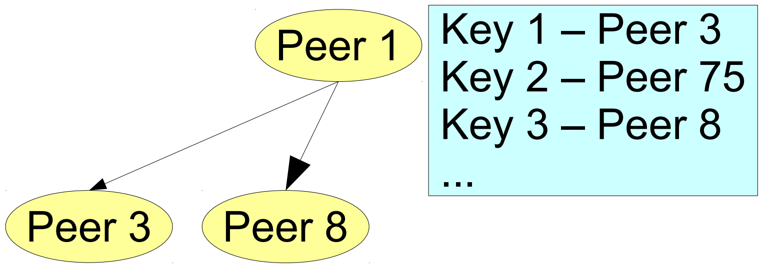
Towards centralized/decentralized

- Gnutella super-peers
  - Reflector nodes with query/response caches
  - High bandwidth connection



FreeNet: decentralized, routing

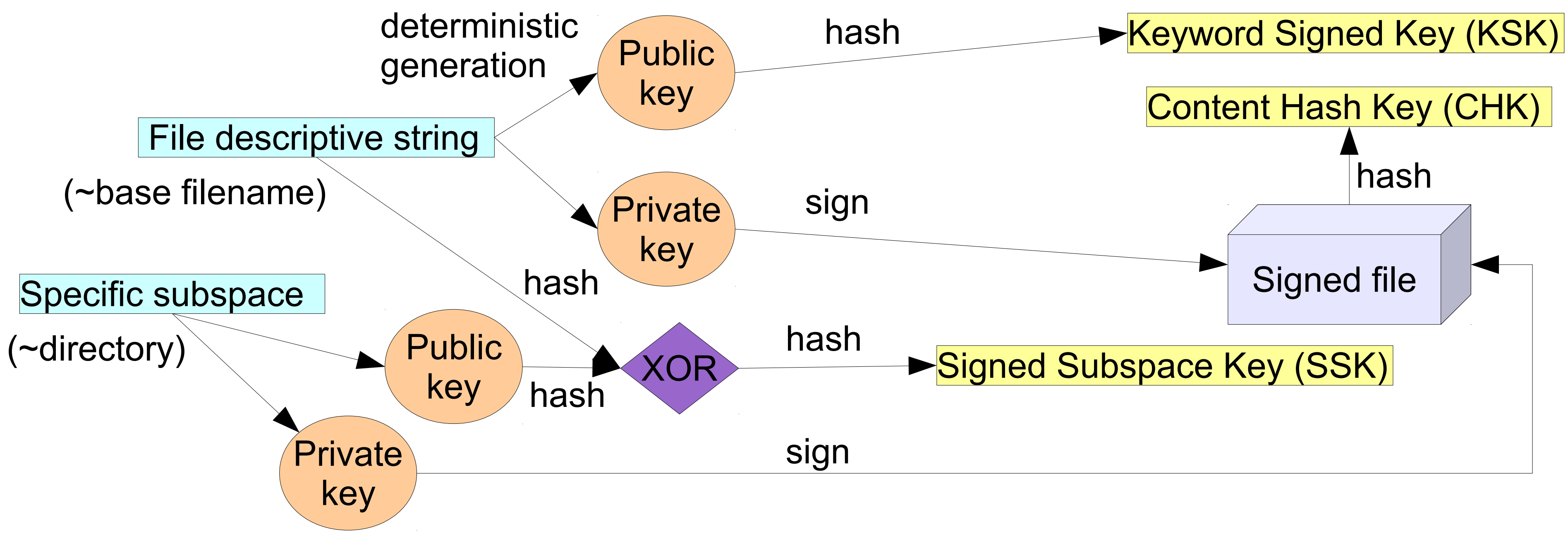
- Different focus
  - File storage instead of data search
  - R/W access to data



- Adaptive routing to overcome gnutella limitations
  - Routing tables: keys associated to files, (key, peer) pairs table
  - Most visited files replication
  - Least visited files expiration
  - Best effort quality of service: no guaranteed result
- Routing algorithm
  - Keys clusters (similar keys are close in the overlay network)
  - TTL + mix-net strategy (restart failed queries far away)
  - Routing table updates on query hit results

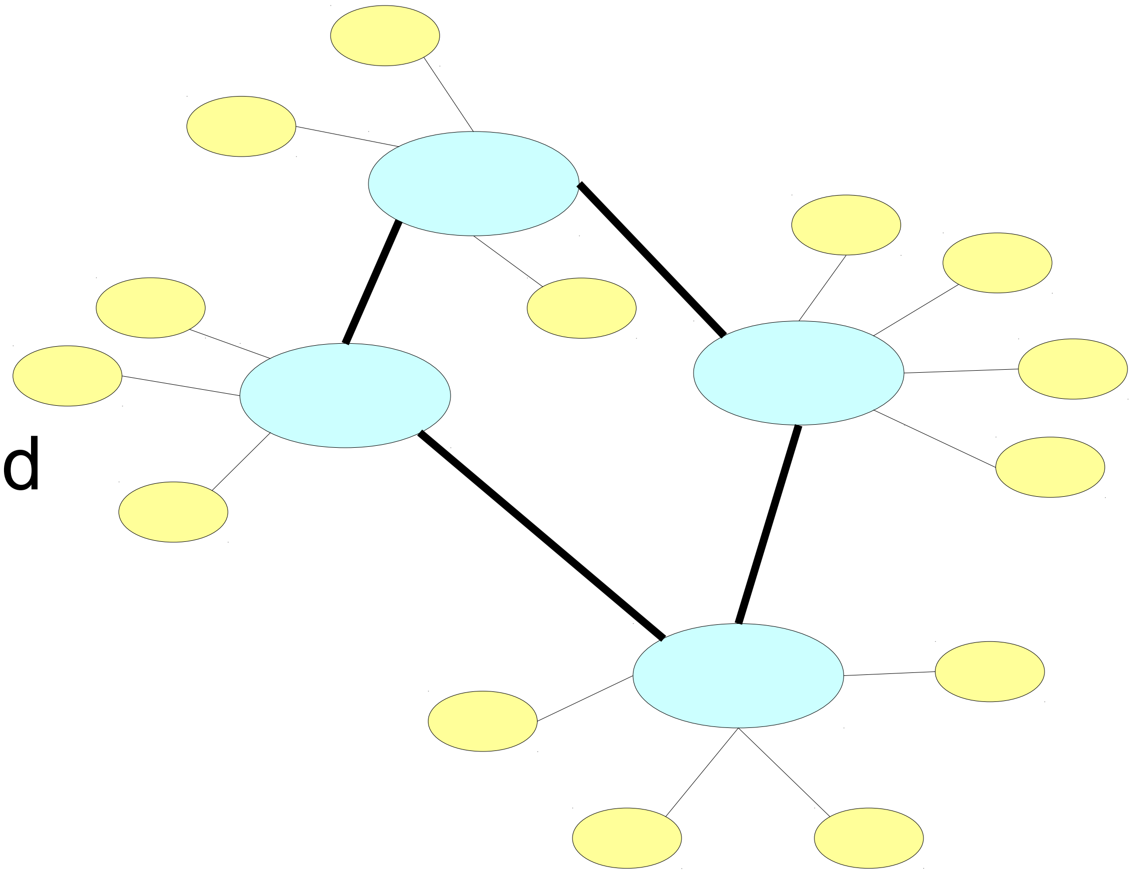
Data keys, unique identifiers

- SHA-1 hash function
  - Non-reversible, Sensitive to input changes, Collision resistant
- 3 keys: data integrity, authentication and privacy
  - CHK ensures integrity
  - SSK signature: pseudonymous identity of the inserter
  - KSK signature: document-specific identifier
  - Symmetric encryption ensures data protection



Towards centralized/decentralized

- Nodes identification
  - Keys are associated to nodes as well
- Scalability
  - Up to 100 000s nodes
- Convergence observed towards a centralized/decentralized model



DHT, Overlay networks

- DHT: Distributed Hash Tables
  - Extension of hash tables to distributed systems
  - Scalable, fault tolerant
  - Combine decentralization (Gnutella), efficiency (Freenet) and guaranteed result (Napster)
- Applications
  - Distributed file systems, P2P file sharing
  - Web caching
  - Multicast / anycast
  - DNS (Domain Name Service)
  - Instant messaging

DHT, Overlay networks

- Overlay network
  - Structured, logical network
  - Key space partitioning scheme among participating nodes
  - Routing between nodes overlayed on top of the Internet network
- Example
  - (key, value) = (SHA1(data), data)
- Expected properties
  - Decentralization, scalability, fault tolerance
  - Security, anonymous in some cases
  - Load balancing
  - Data integrity
  - Performance

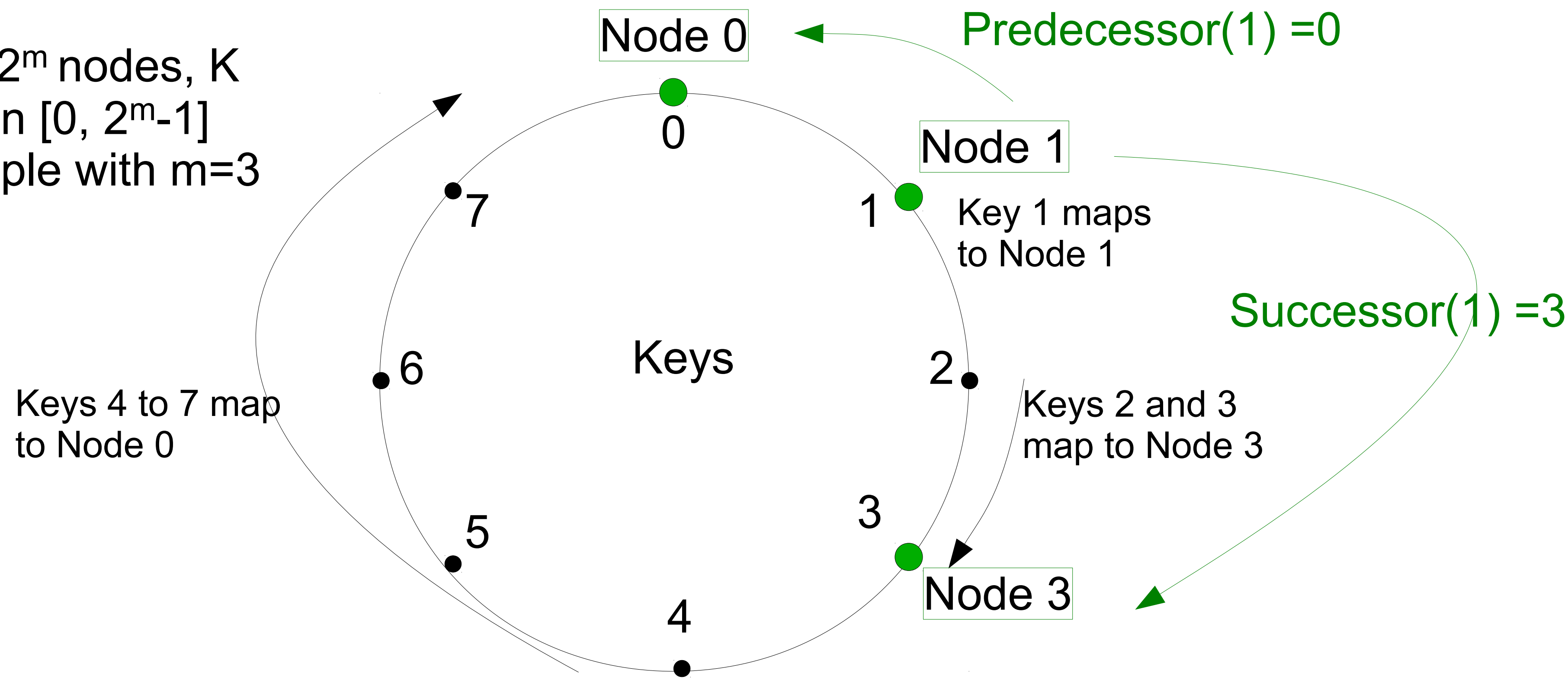
Key space partitioning

- Distance function in key space $\delta(k_1, k_2)$
  - Each node is assigned an ID key n
  - A node contains files which key k are closest to n ($\delta(k, n)$ min)
- Consistent hashing functions
  - Removing/adding a node only change the set of keys of nodes with adjacent IDs
  - Minimize data reorganization due to nodes leave / arrival
- Key-based routing
  - For any key k, a node either owns k or has a link to a node closer to k in its routing table
  - Greedy algorithm for data discovery
  - Antagonist goals: minimize both route lengths and neighborhood sizes. Typically O(log(n))/O(log(n))

DHT: Chord

- Chord
  - Keys are points on a circle
  - δ is the number of hops on the circle traveling clockwise

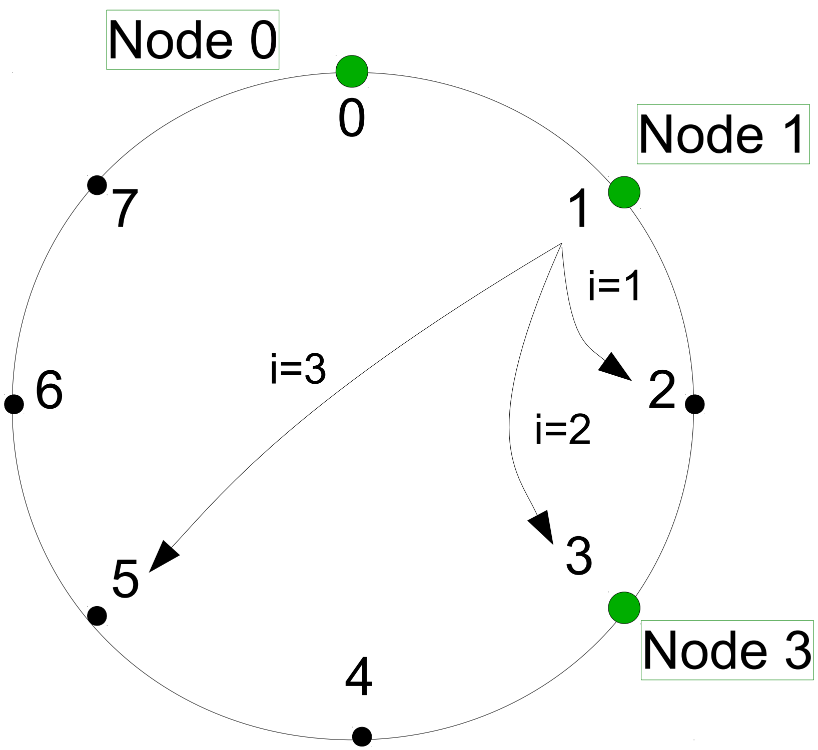
N <= 2^m nodes, K keys in [0, 2^m-1]
Example with m=3



- Each node is responsible for (1+ε)K/N keys with high probability with ε = O(log(N))

Chord routes

- Routing following the predecessor/successor pointers: O(N)
  - Any node becomes a critical point of failure
- m-entries finger table for each node n
  - i^{th} entry: node s = successor($(n + 2^{i-1}) \bmod 2^m$)



Finger table for Node 1 (n = 1)
F(1) = successor($1 + 2^0 \bmod 8$)
= successor(2)
= Node 3
F(2) = successor(3) = Node 3
F(3) = successor(5) = Node 0

Finger table for Node 3
F(1) = F(2) = F(3) = Node 0

Chord routes

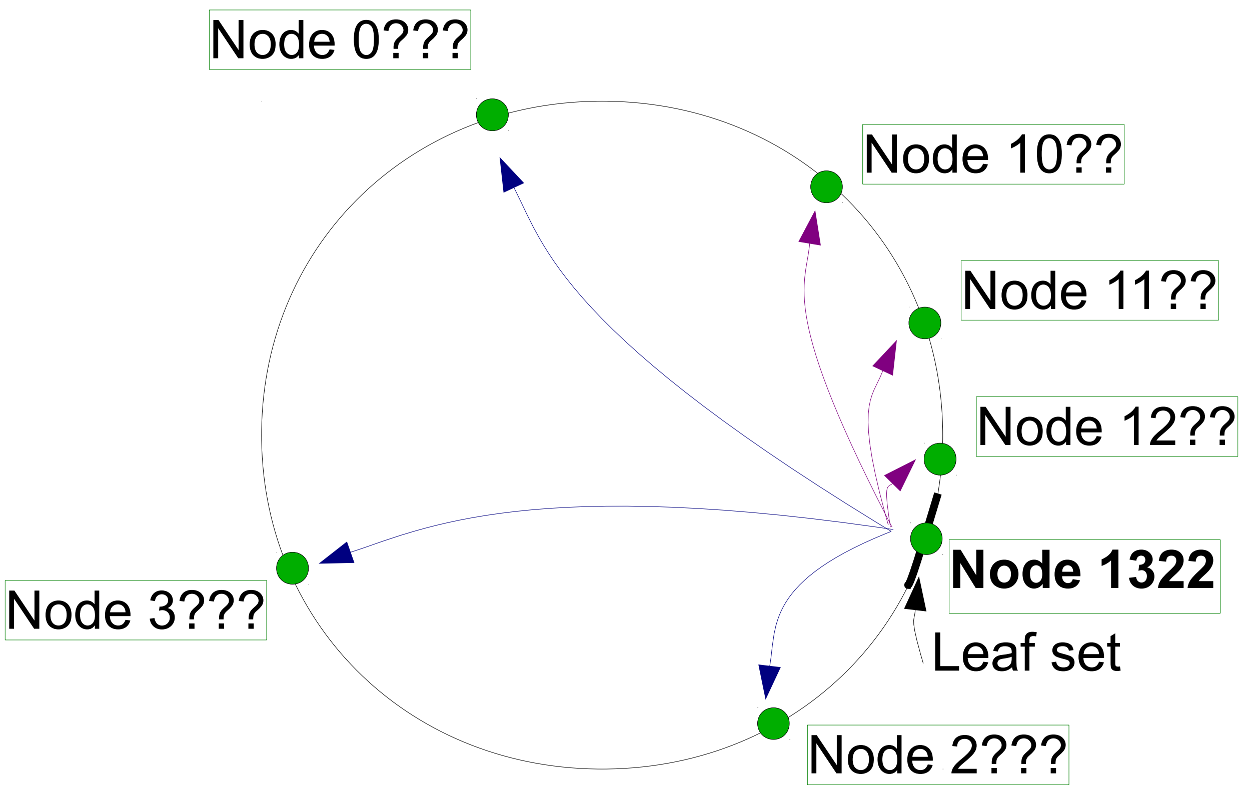
- Routing table properties
  - m = log(N) entries
  - No immediate routing (e.g. Node 3 finger table does not contain the successor of key 1)
- Routing for key k
  - If it is known, send request to the immediate successor of k
  - Otherwise, send the request closer to k, to the closest known predecessor of k
  - With high probability, the route length is O(log(N))
- Insertion / deletion of nodes
  - Require a predecessor pointer to each node for reverse tracks
  - Create new node routing table and update other routing tables
  - With high probability, this require $O(\log^2(N))$ messages

DHT: Pastry

- Minimize message travel distance
  - Proximity metric (e.g. number of IP hops)
- Circular hash table's key space
  - m=128, random node IDs assignment
  - Node IDs are thought as numbers in base 2^b (typically, b=4)
  - Robust to L/2 adjacent simultaneous node failures (L=2^b or 2^{b+1})
- For any node:
  - Routing table: by ID prefix (base 2^b)
    - Size $\log_{2^b}(N) \times (2^b - 1)$, maximum $\log_{2^b}(N)$ hops
  - Neighborhood set: M (=2^b or 2^{b+1}) closest peers in term of proximity metric
  - Leaf set: L numerically closest peers (divided in 2 groups with smaller and larger IDs)

Pastry routing

- Example with b=2 (base 4), m=8 (4 digits node Ids)
- Route to one node in the leaf set if within range or use the routing table: $O(\log_{2^b}(N))$



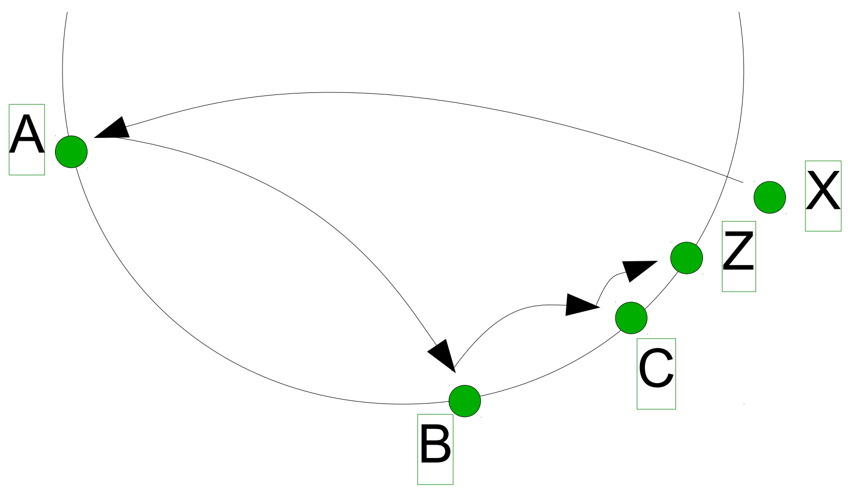
2^b (columns) × $\log_{2^b}(N)$ (rows)

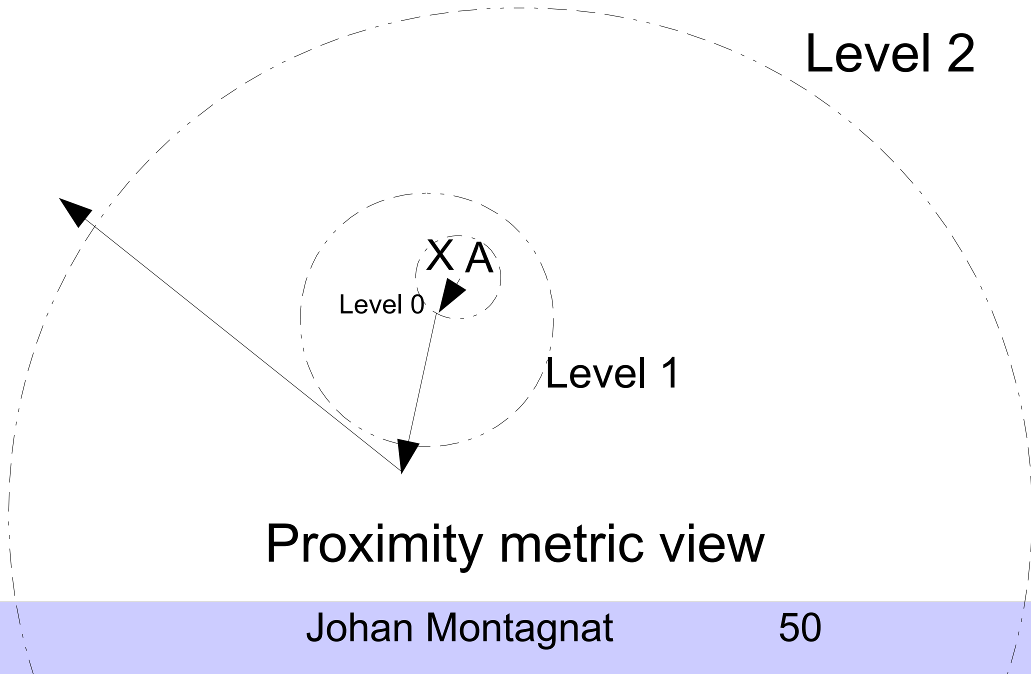
	0	1	2	3
0 digit	0???		2???	3???
1 digit	10??	11??	12??	
2 digit	130?	131?		133?
3 digit	1320	1321		1323

Some cells may be left empty when no corresponding node is known

Locality property

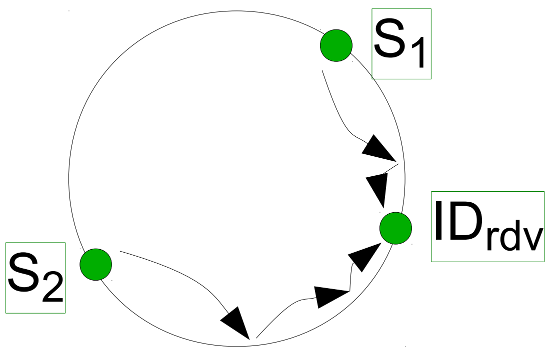
- Adding nodes
  - X joins, know A close geographically, search for Z closest ID
  - Z leaf set is close to X leaf set
  - X build routes from reasonably close A, B, C... Z nodes
- All table entries of any node refer to a node that is near (proximity metric) among nodes with appropriate prefix





Pastry applications

- PAST distributed file system
  - Locality property highly desirable to minimize file transfers
- SCRIBE publish/subscribe system
  - A set of subscribers $\{S_i\}$ are interested in a topic with ID_t
  - A rendez-vous node with ID_{rdv} close to ID_t is selected
  - Subscribers send a registration message ID_t with which is registered all along the path to ID_{rdv}
  - The publisher send messages to ID_{rdv}
  - Messages are multicasted to the reverse tree of all subscribers paths

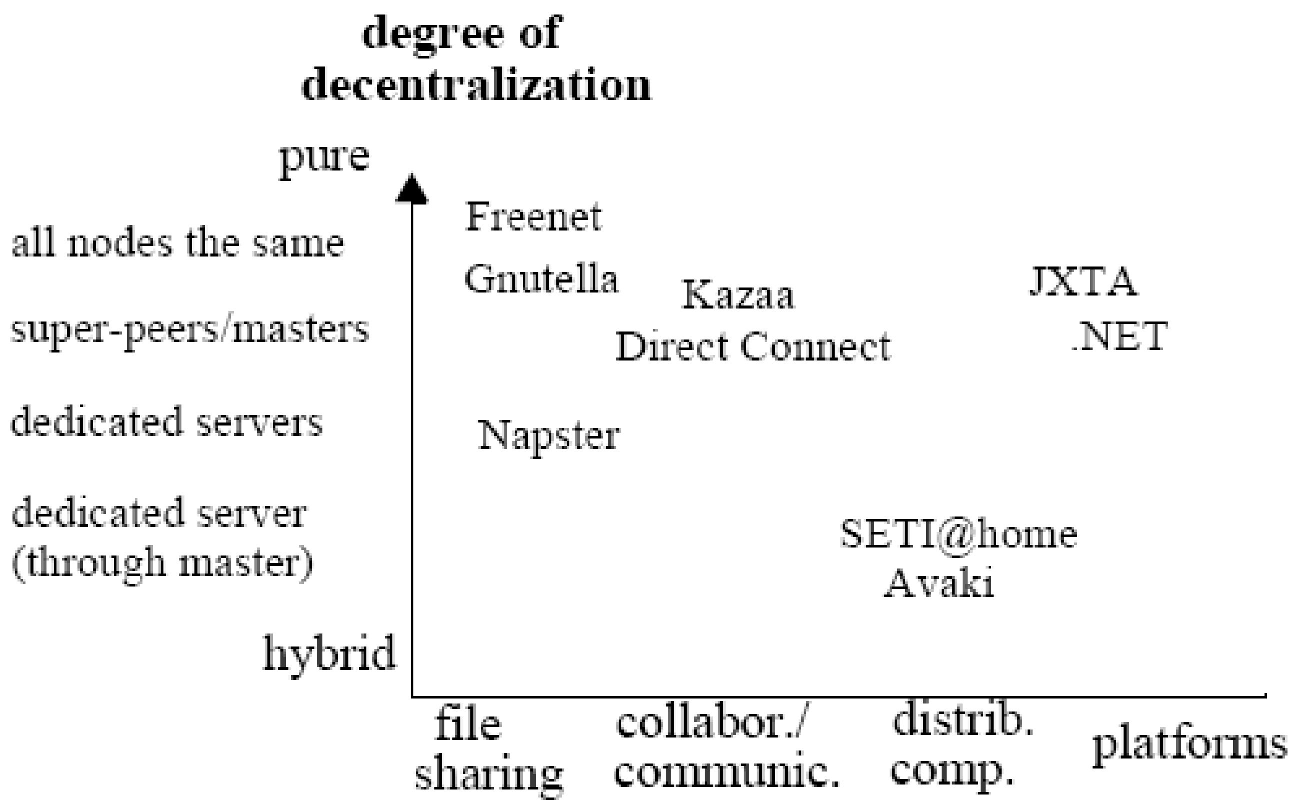


Other DHTs

- Tapestry
  - Optimize routing tables (dynamically maintained, efficiency by minimizing messages latency)
  - Implements multicasting
  - Applications: OceanStore distributed storage, Spamwatch decentralized spam filter, Bayeux multicasting application...
- CAN
  - d-dimensional Cartesian coordinate key space
  - O(d) route tables, $O(dN^{1/d})$ lookup cost
  - Independent of N: matches Chord/Pastry for d = log(N) but N is meant to evolve while d is constant
- And many others...

The success of P2P networks

- Mostly based on read-only multimedia content retrieval
- Extension to load management based on a degree of centralization



P2P challenges

- Data access control
  - Most P2P networks ignore access control
- Availability of data
  - Table updates on node deletion but data inaccessible if the service interruption was not scheduled
  - Some effort for providing replication
- Data updates
  - Most data read-only

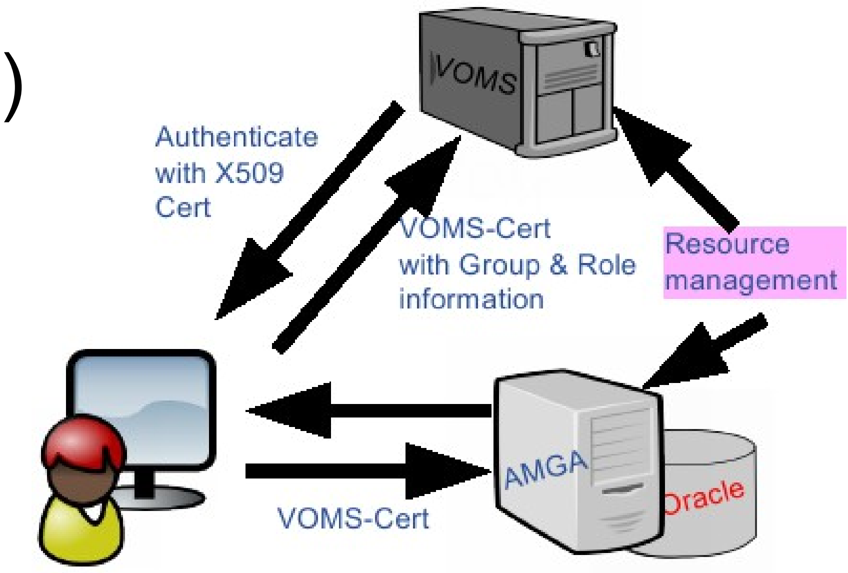
Metadata management

Metadata

- Metadata
  - Any (secondary) data related to the (main) data
  - Usually stored in databases (relational, XML) by opposition to files
  - Especially important to handle heterogeneity
- Simple metadata indexed on files
  - System metadata: file size, checksum, etc
  - User metadata: file format, file descritption, etc
- General metadata, complex relational schema
  - Not necessarily directly indexed on data file
  - E.g. patient information attached to many medical data files
  - Require flexible and extensible metadata schema coupled with metadata search engine

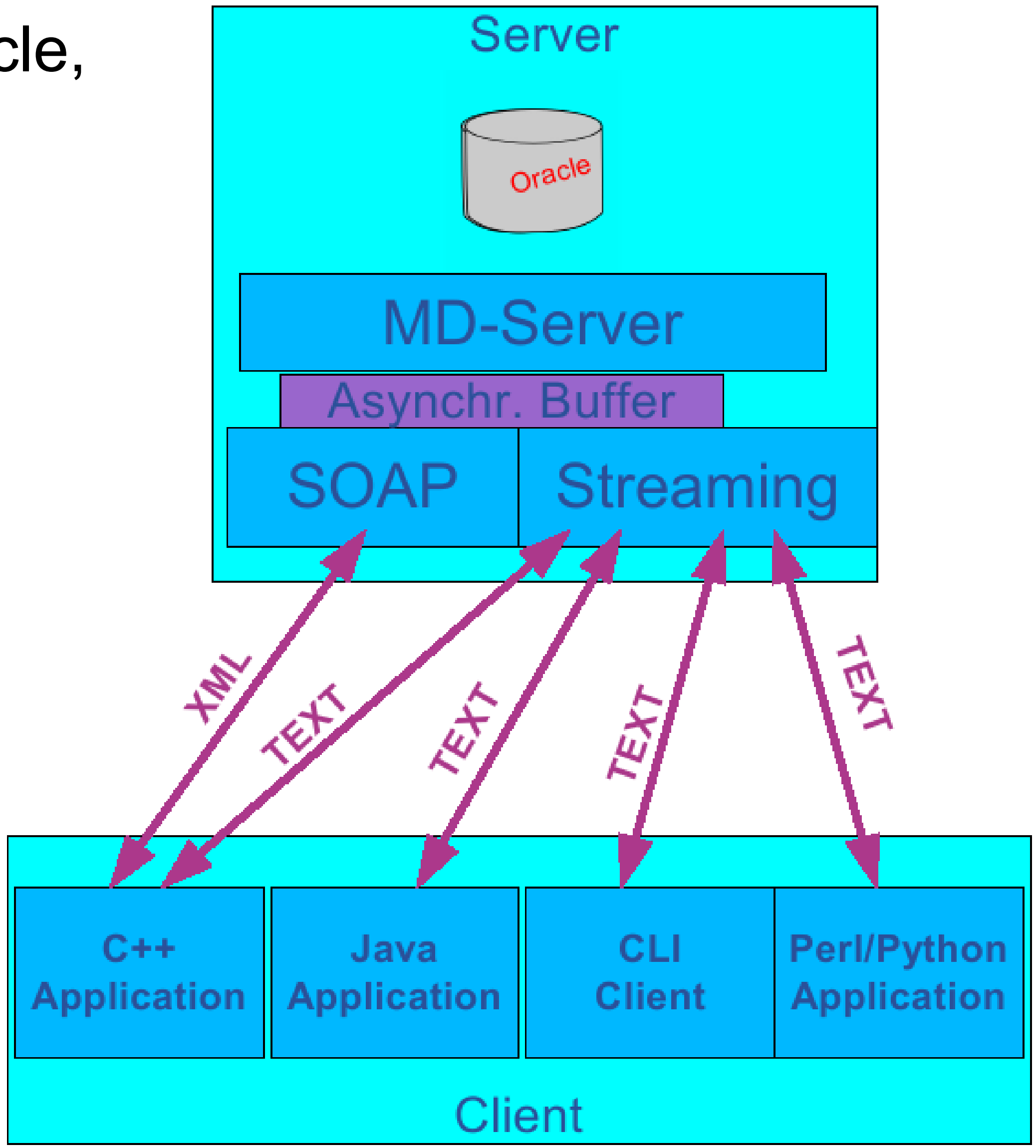
AMGA database front-end

- Grid credential-based authentication
  - Single sign-on
- Secured communications (TLS)
- ACL-based access control
  - Per table, per entry
- Different back-end
  - Heterogeneity, legacy DBs
- Performance
  - Streamed bulk operation
  - Scales to hundred concurrent client (back-end limit)
- Support for replication
- Proprietary interface / query language



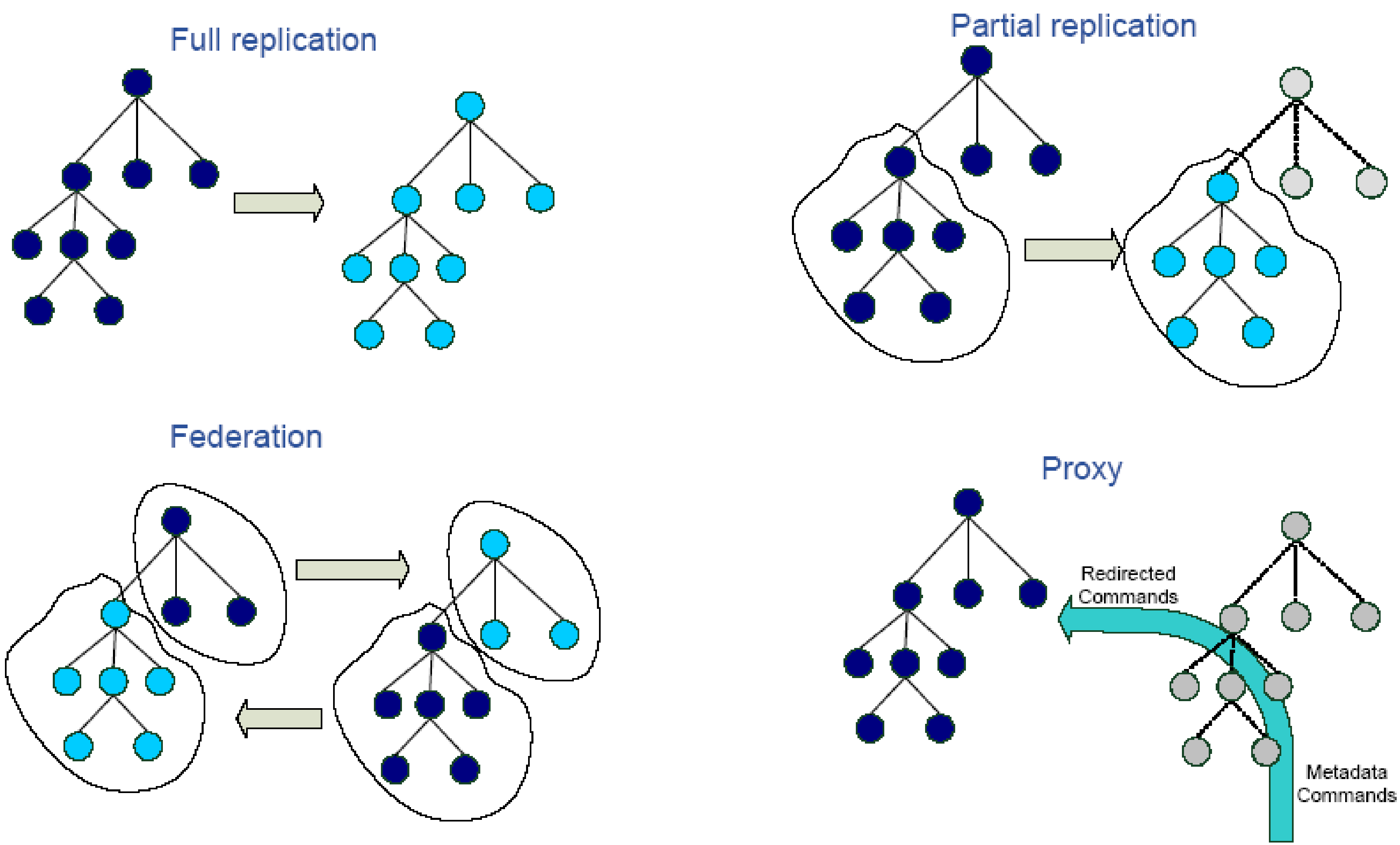
AMGA infrastructure

- Back-end
  - PostgreSQL, MySQL, Oracle, SQLite
- SOAP and text interfaces
- Streaming capability
  - Especially for WAN communications
- Secured communications
  - Optionally
- Client APIs
  - C++, Java, Perl, Python



AMGA replication

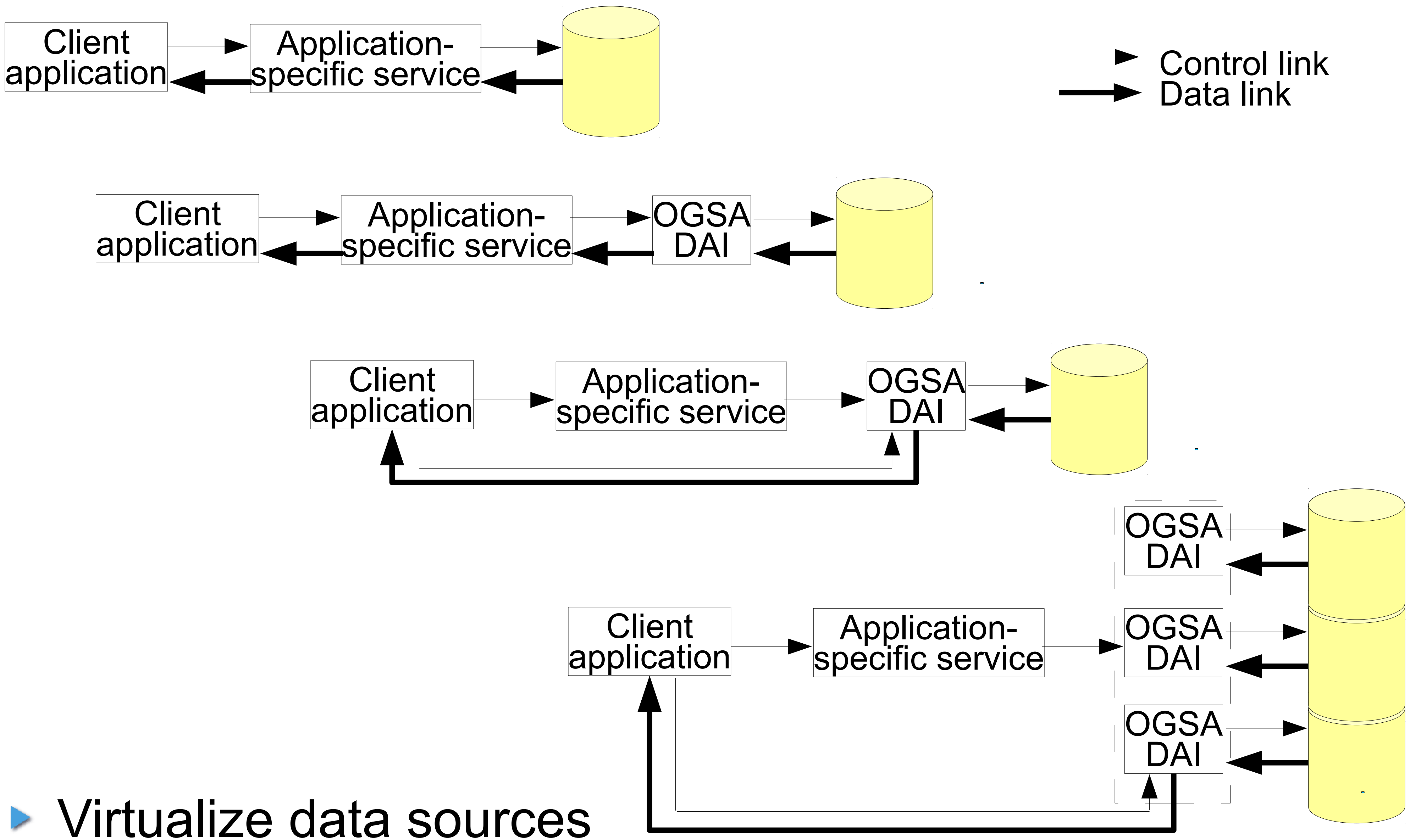
- Replication of databases, hierarchical approach



OGDA DAI: Data Access Integration

- OGSA DAI
  - UK eScience project, http://www.ogsadai.org.uk
  - Part of the OGF DAIS-WS working group
- Middleware to assist with access and integration of data from different sources
  - Relational or XML databases
  - Files
  - Data query, transformation and delivery components
  - Service-based distributed query processor

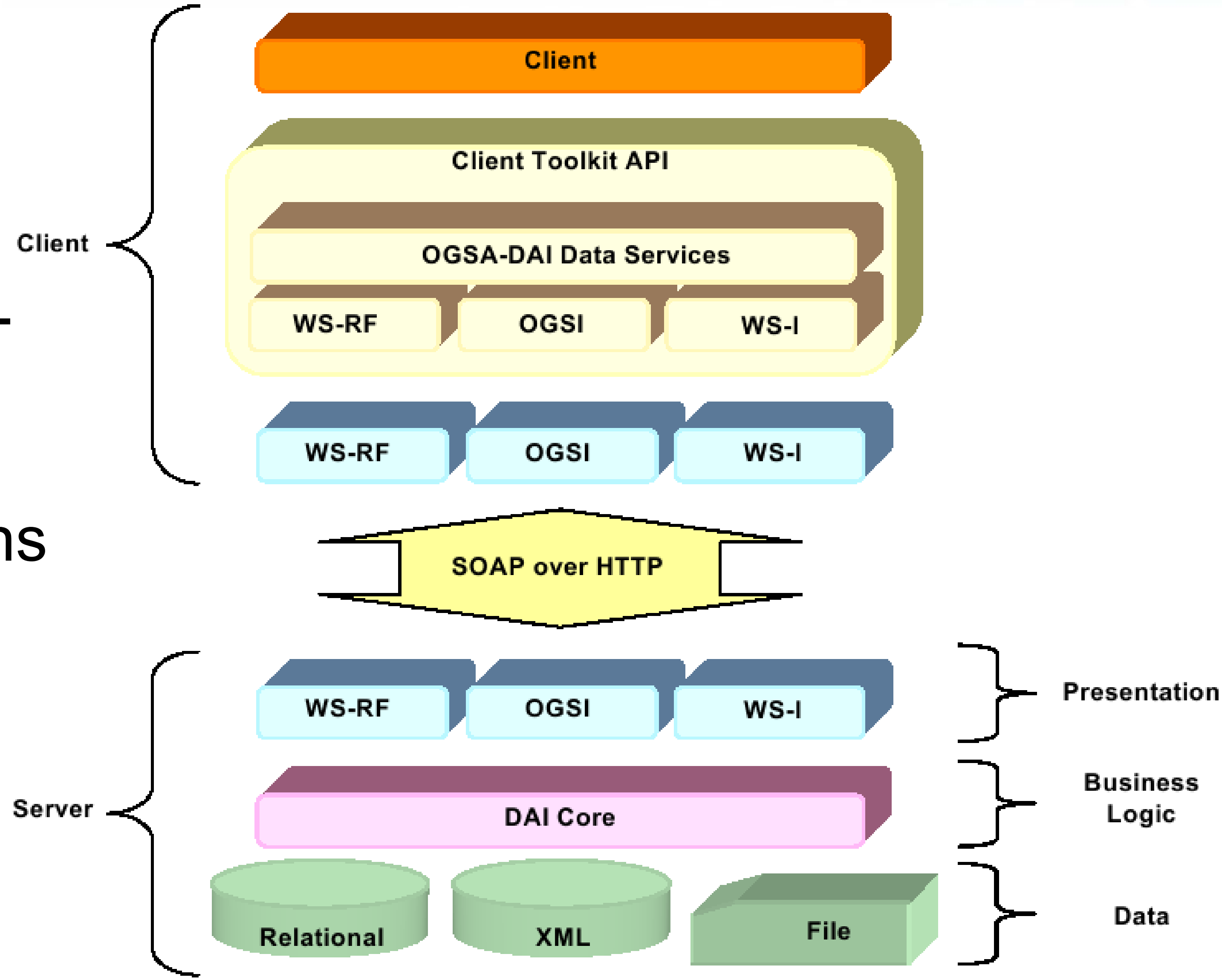
OGSA DAI integration



- Virtualize data sources

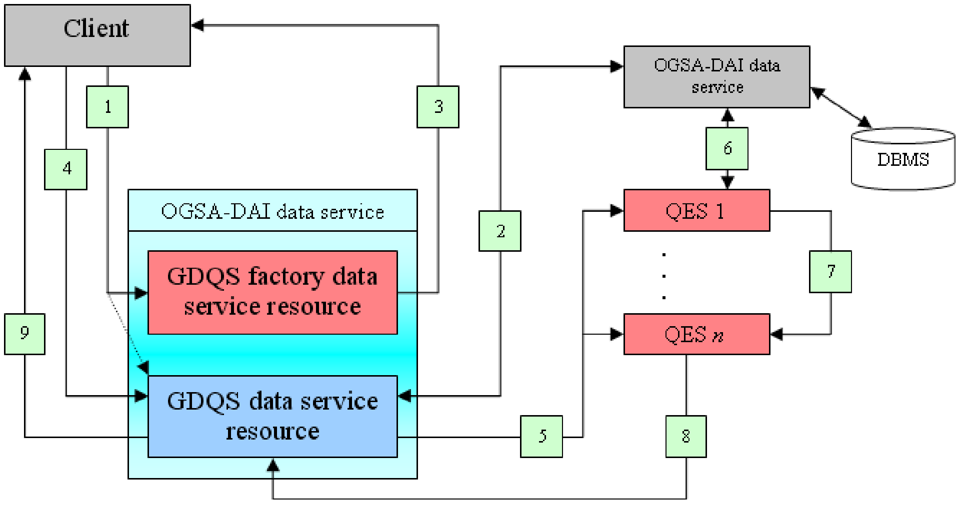
OGSA DAI architecture

- Service-based
- Client / Server-side services
- Standard communications



OGSA DQP: Distributed Queries Processing

- Query service, interfaced to OGSA DAI services and other Web Services
- Parallel database technologies
  - Exploit queries implicit parallelism, distributed data sources
- Query Evaluator Service
  - Evaluates query partition
- Distributed Query Service
  - Coordinates QES partitions



References

- I. J. Taylor, “From P2P to Web Services and Grids: peers in a client/server world”. Springer.
- I. Stoica, R. Morris, D. Karger, M. Frans Kashoek and H. Balakrishnan, “Chord: a scalable P2P lookup service for internet applications”, ACM SIGCOMM 2001, San Diego, USA, Aug. 2001.
- A. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems", IFIP/ACM Intl Conf. on Distributed Systems Platforms (Middleware), Heidelberg, Germany, Nov. 2003.