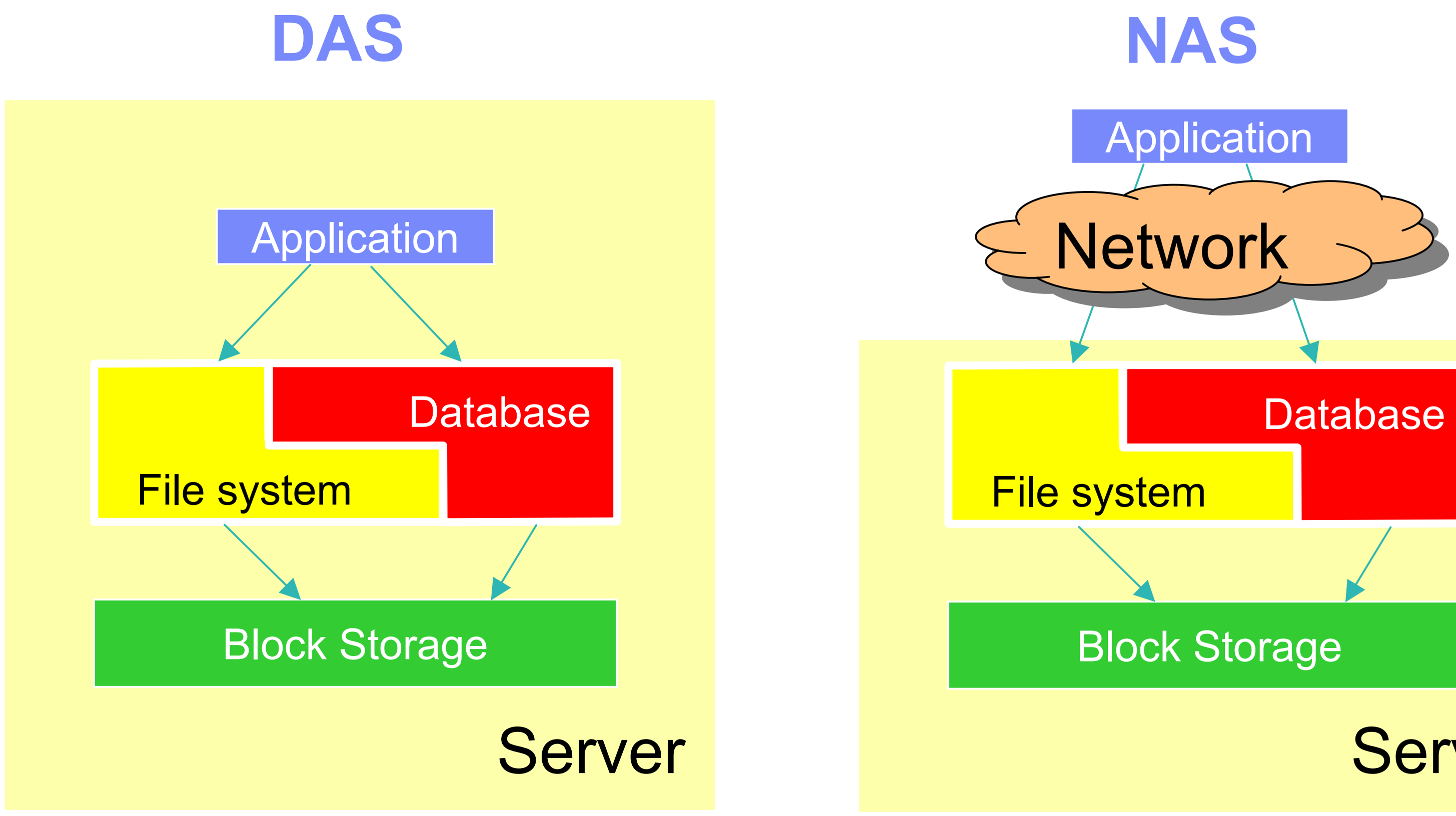
Towards an Object Store

Alain Azagury, Vladimir Dreizin, Michael Factor, Ealan Henis, Dalit Naor, Noam Rinetzky, Ohad Rodeh, Julian Satran, Ami Tavory, Lena Yerushalmi

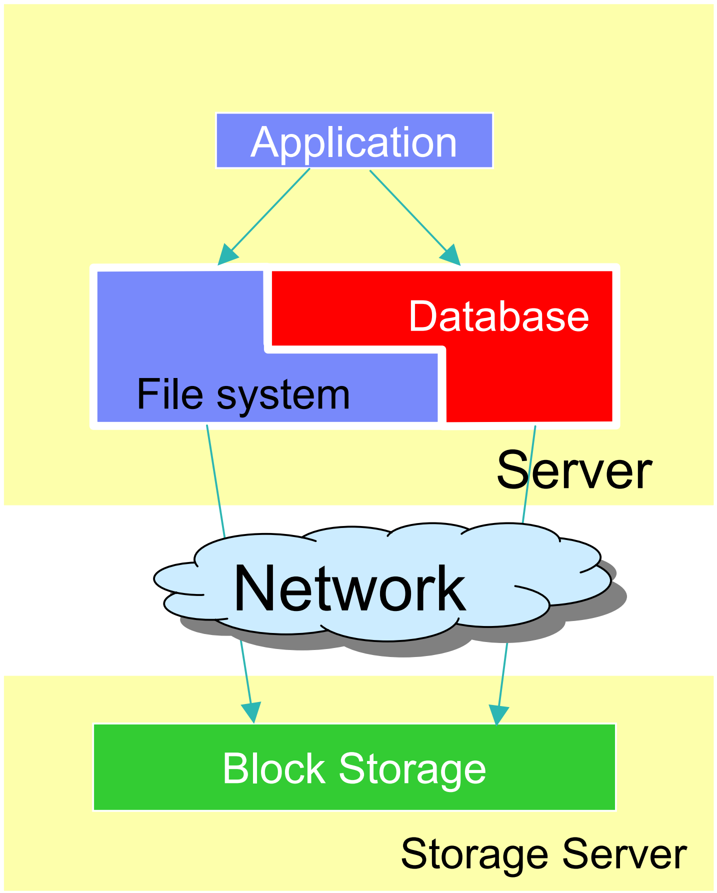
IBM Research Lab at Haifa

From Direct Attached Storage to Network Attached Storage…

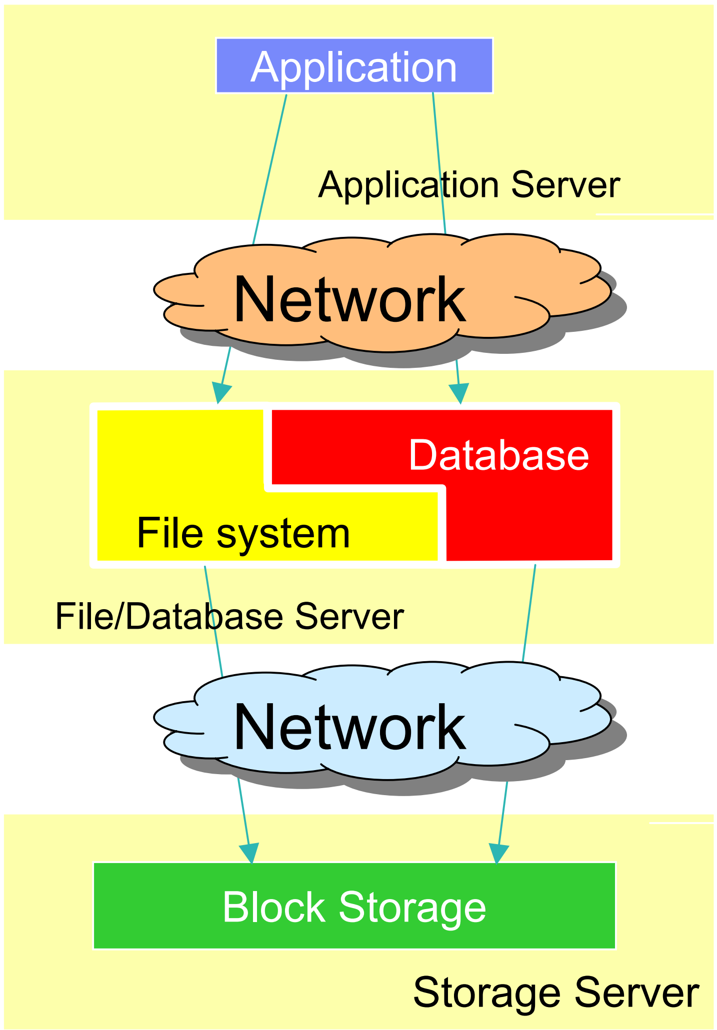


…to Storage Area Networks

SAN



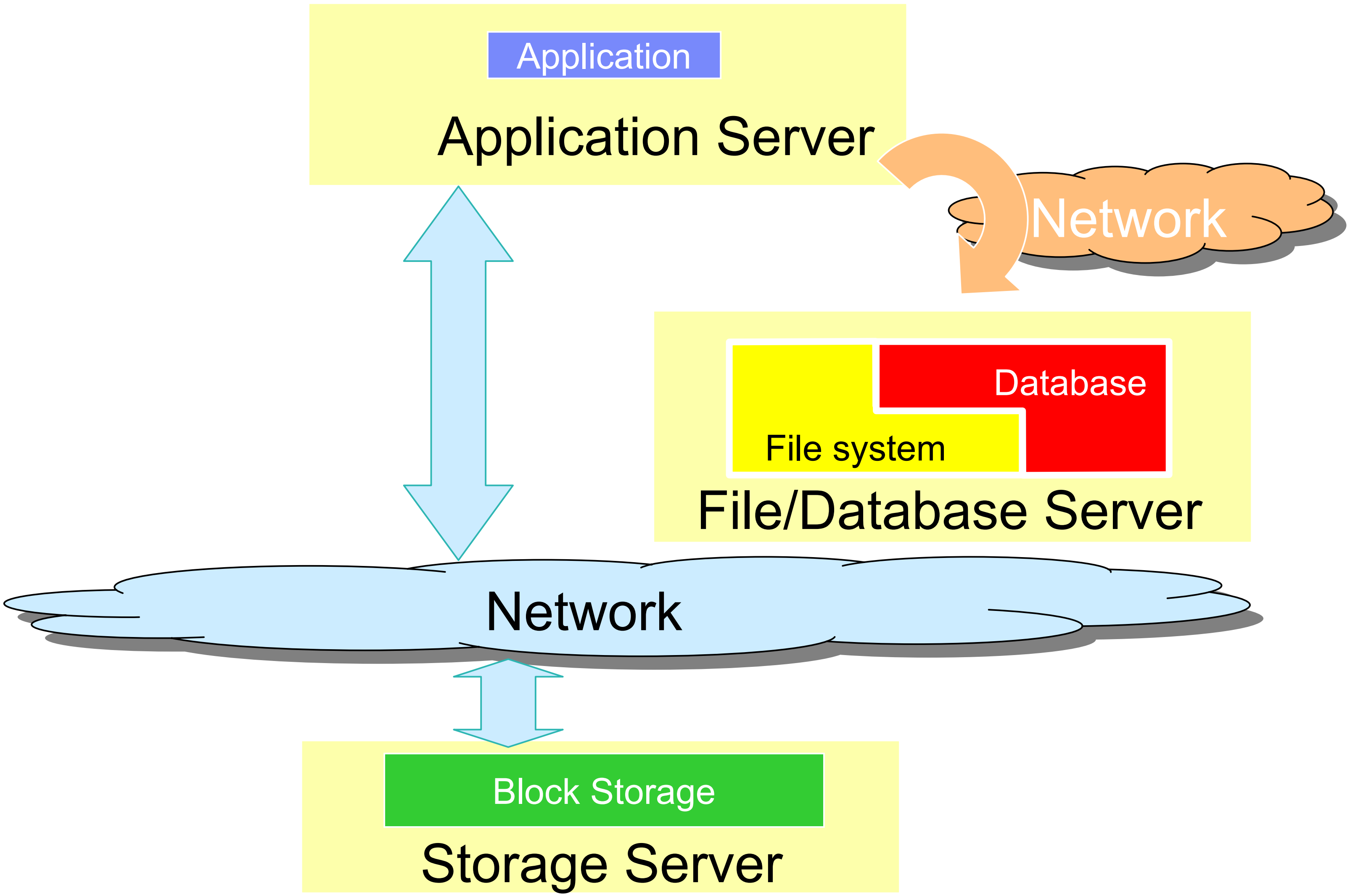
Combined Technologies



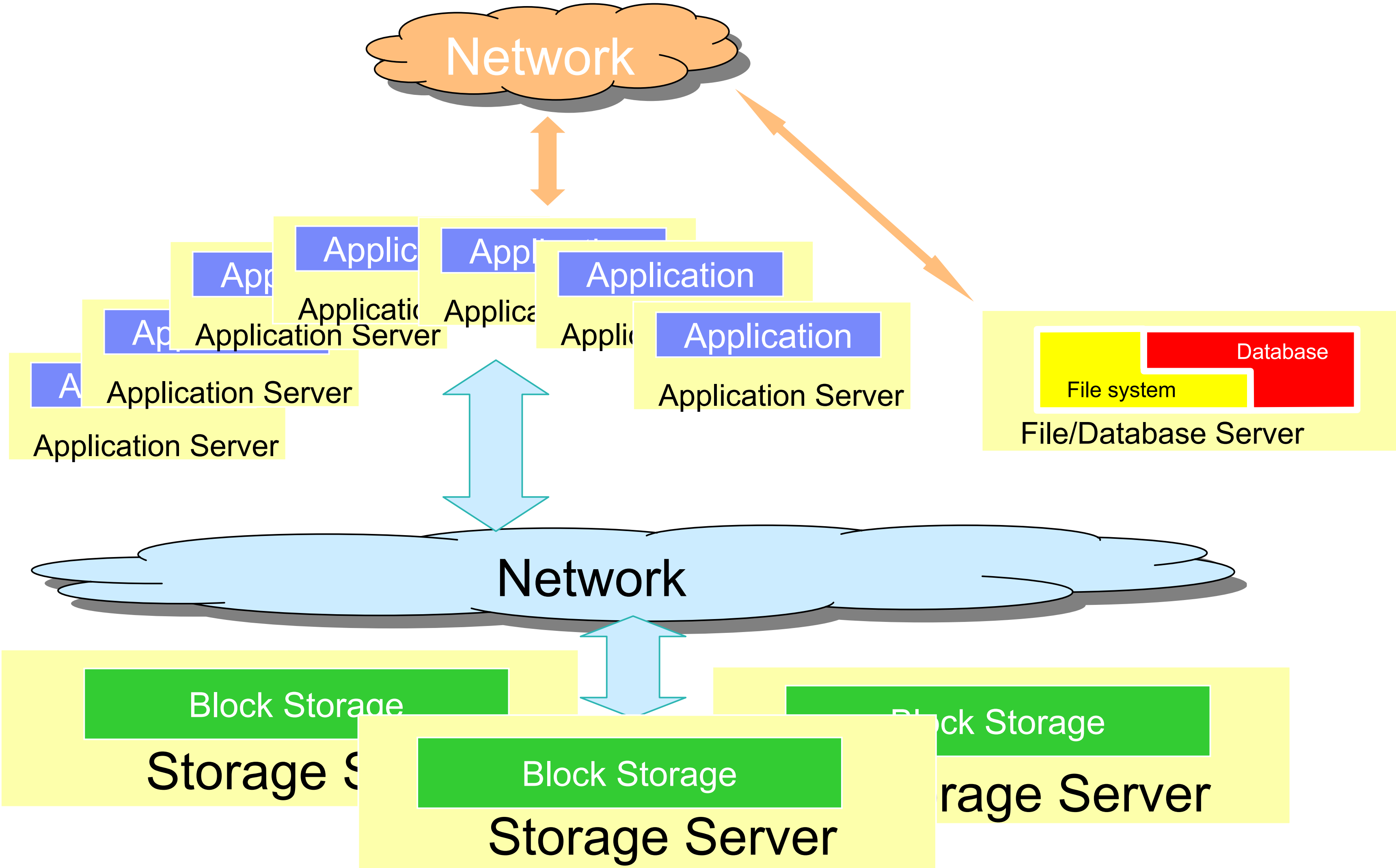
Network: IP
Protocol: CIFS NFS

Network – FibreChannel/IP
Protocol – FCP/iSCSI (SCSI)

SAN promise – Unmediated Access to Data

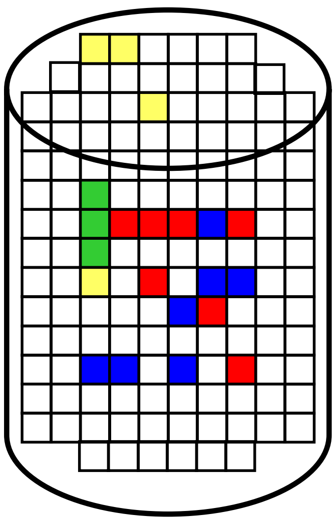


SAN promise – Disk Sharing



Object Storage

Today's Block Device

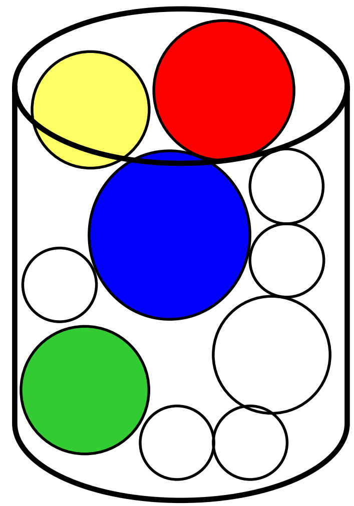


Operations
- read block
- write block

Security
- Weak
- Full disk

Allocation
- External

Object Store



Operations
- read object offset
- write object offset
- create object
- delete object

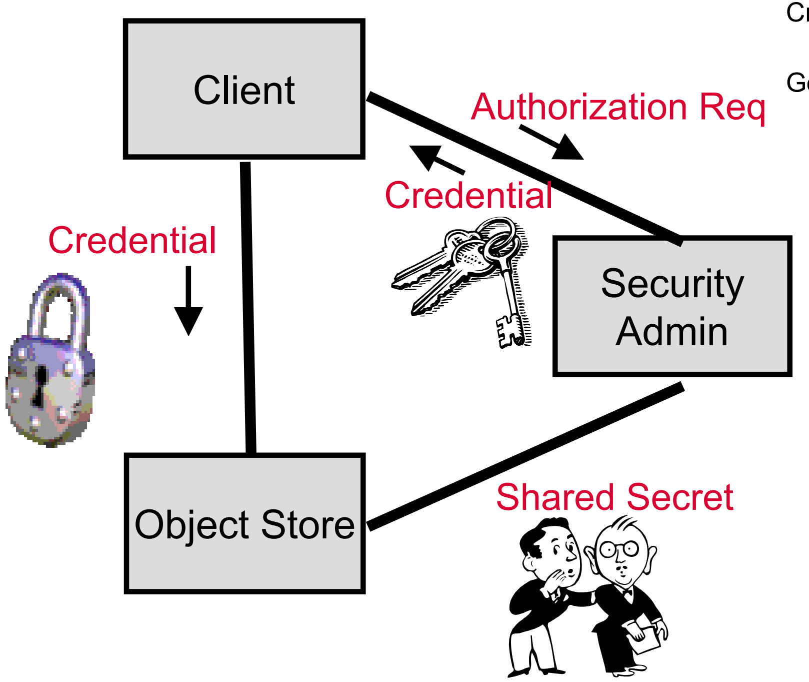
Security
- Strong
- Per Object

Allocation
- Local

Object Store Operations

- Basic Operations
  - Create Object
  - Delete Object
  - Write Offset in Object
  - Read Offset in Object
- Administrative Operations
- Basic abstract flow
  - Create an object, getting back an object ID
    - Clients responsibility to remember the ID
  - Send requests to read and write the object given the ID
  - Delete the object when done using

Object Store Security



All operations are secured by a credential

Security achieved by cooperation of:

- Admin - authenticates, authorizes and generates credentials.
- ObS - validates credential that a host presents.

Credential is cryptographically hardened

- ObS and admin share a secret

Goals of Object Store security are:

- Increased protection/security
  - At level of objects rather than whole LUs
  - Hosts do not access metadata directly
- Allow non-trusted clients to sit in the SAN
- Allow shared access to storage without giving clients access to all data on volume

Object Store Operations Read

- Read
  - Parameters: Object Store ID, Object ID, Offset, Length, Credentials
  - Basic steps an object store must provide
    - Receive request
    - Validate credentials
    - Find allocation data for indicated object
      - Map offset and length to a collection of LBAs in an underlying block storage
    - Stage the data if necessary
    - Gather the data and return to the host
  - Issues and Variants
    - Block alignment
    - Read of non-allocated data (sparse vs. past "end" of object)

Object Store Operations Write

Write

- Parameters: Object Store ID, Object ID, Offset, Length, Credentials, Data
- Basic steps an object store must provide
  - Receive request
  - <u>Validate credentials</u>
  - <u>Find allocation data</u> for indicated object
    - Determine if the indicated range is already bound to a collection of underlying LBAs
  - <u>If not already bound</u>
    - <u>Determine the mapping</u>
    - <u>Update the metadata</u>
  - Destage the data to the indicated LBAs
- Issues/Variants
  - Use of a non-volatile write cache
    - Late binding
  - Hardening the metadata updates
  - Ensuring metadata updates are only modified if data is hardened
  - Block alignment

Capability Structure

				Object Rights						ObS Rights			
Type	ObS ID	Object ID	R	W	A	T	C	D	I	F	C*	I*	Ver

Type

Does the credential apply to a specific object or entire object store

Object Rights

Read, **W**rite, **A**ppend, **T**runcate, **C**reate (given an ID), **D**elete, **I**nfo

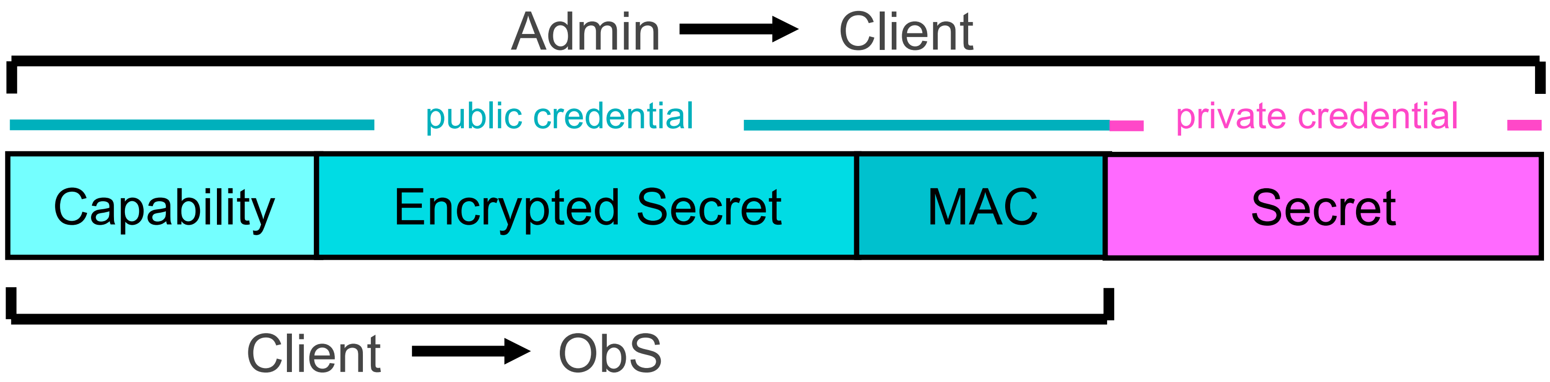
Object Store (ObS) Rights

Format, **C**reate (ObS generates ID), **I**nfo on Object Store

Ver(sion)

Used to allow the credential to time out

Credential Structure



Capability

Operations that the credential entitles

Encrypted Secret

A Secret generated by the Admin

A different secret for every credential

Encrypted with a key Admin shares with the ObS

MAC -- Message Authentication Code

Standard cryptographic hash on the capability and the encrypted secret

Ensures host cannot alter/forge a credential

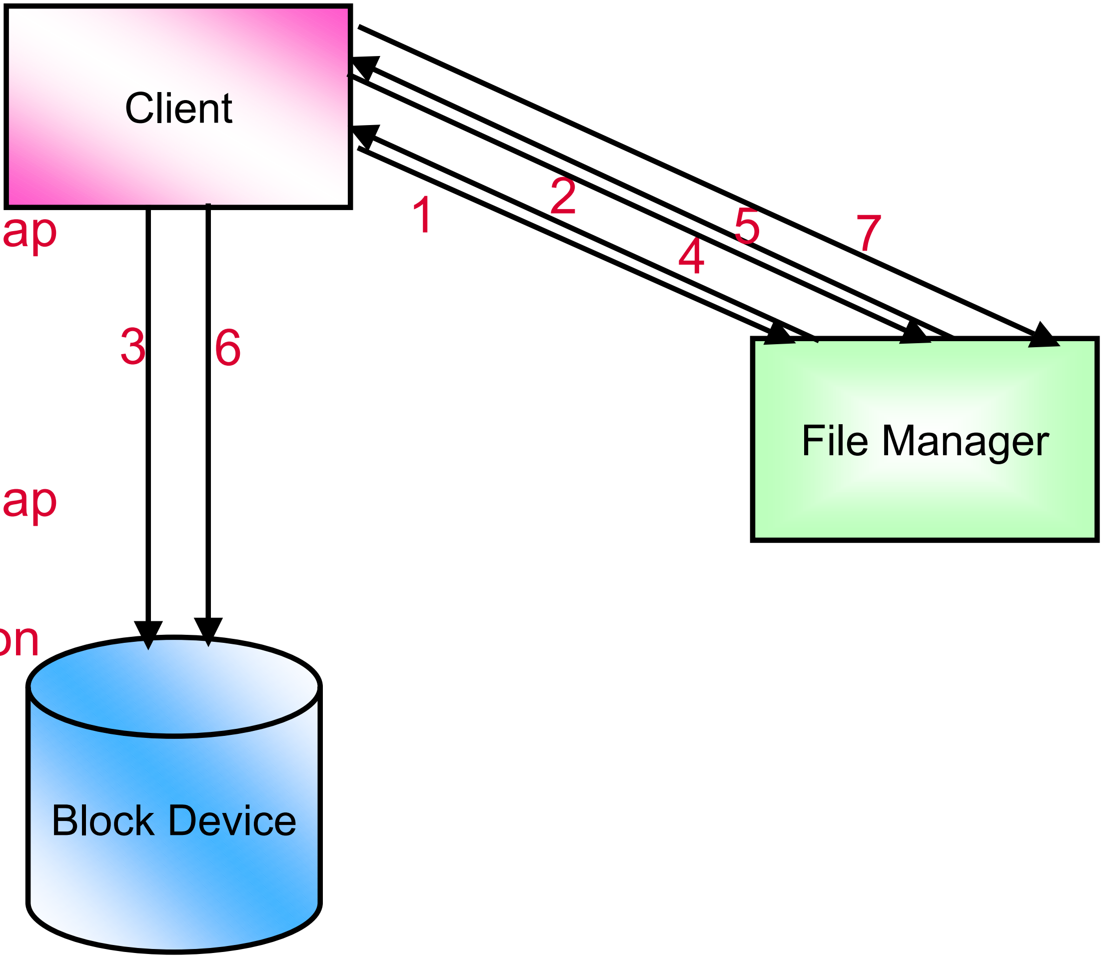
Secret

The (un-encrypted) secret

Used by the client to verify that it got the credential form the Admin

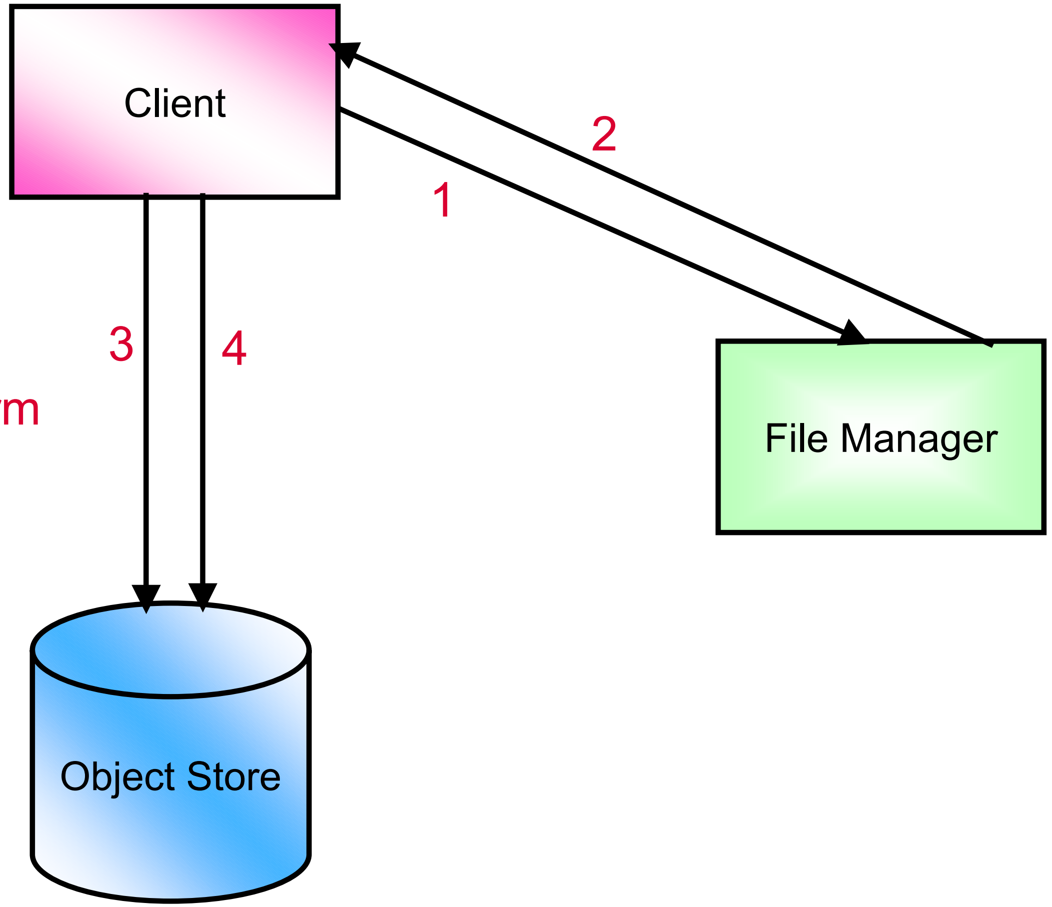
SAN File System without an Object Store

1. Create File
2. Return allocation bitmap
3. I/O (unsecured)
4. Request additional allocation bitmap
5. Return allocation bitmap
6. I/O (unsecured)
7. Return actual allocation data to File Manager



SAN File System with an Object Store

1. Create File
2. Return Locks and Credentials
3. Create Object and perform I/O with Credentials
4. I/O with Credentials



Blocks vs. Objects vs. Files

Block Storage

- Fastest
- Small set of operation
- No connection to user's abstraction
- Dumb device
- No access security

Object Storage

- Fast
- Small set of operations
- Object usually maps to user abstraction
- Local space management
- Enable end-to-end management
- Access is Secure

File Storage

- Slow
- Rich set of operations (sometimes more than what the application requires)
    - Locking
    - Hierarchical name service
    - …
- File usually maps to user abstraction
- Access is Secure (strength depending on the protocol)

How Real Are Object Stores?

Object Stores

- First big push
  - Garth Gibson, et al., NASD -- CMU, Panasas
- DSF Storage Manager
- Antara – a prototype Object Store
- Lustre and Object Store Target (LLNL)
- CAS (EMC Centera – A Write-Once ObS)
- . . .

Drivers

- iSCSI
  - IP access to storage exacerbates SAN security problems
- Data Sharing Facility (DSF)
  - A highly scalable research file system which incorporated an object-store like component to ensure local space allocation
- Storage Tank and other SAN file systems
  - Shared access requires SAN security (or trusted clients!)

What is Antara*?

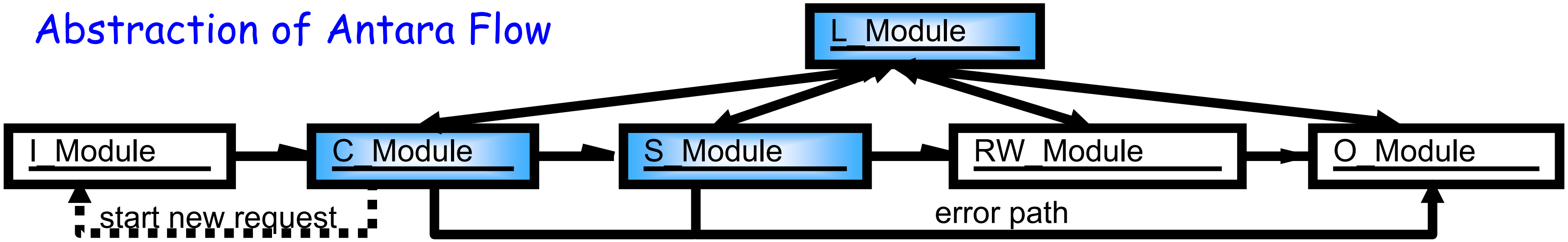
- **A prototype implementation of an object store as a standalone control unit**
  - First generation ObS prototype
    - Generation zero: DSF Storage Manager
    - Generation two…
  - Initial focus on integrity and recoverability of metadata and scalable design
    - Current focus on demonstrating "reasonable" performance
  - Multiple versions have been developed
- **Features**
  - IP connectivity (but design is fairly transport independent)
  - Supports the Antara ObS protocol
    - Similar in functionality to T10 draft but not SCSI
    - Commands currently supported:
      - Open/Close Session, Create/Delete Object, Read/Write/Append, Truncate, . . .
  - Export a single object store
  - Current version assumes a non-volatile store
    - Prior versions did not have this assumption

* "interior" in Sanskrit

General Design

- **A conceptual pipeline**
  - Uses completion ports to avoid context switches
  - In most cases same operating system thread handles multiple stages
- **The modules are:**
  - I-Module (communication **i**nput, connection management)
  - S_Module (**s**ecurity)
  - C_Module (**c**ontrol and dispatching)
  - L_Module (**l**ookup, metadata tables and locking mechanisms)
  - RW_Module (**r**ead-**w**rite of data and log of metadata)
  - O_Module (communication **o**utput)
- **The C, L, and S modules are mostly object store unique**
  - I, RW, and O provide functions that are in most control units

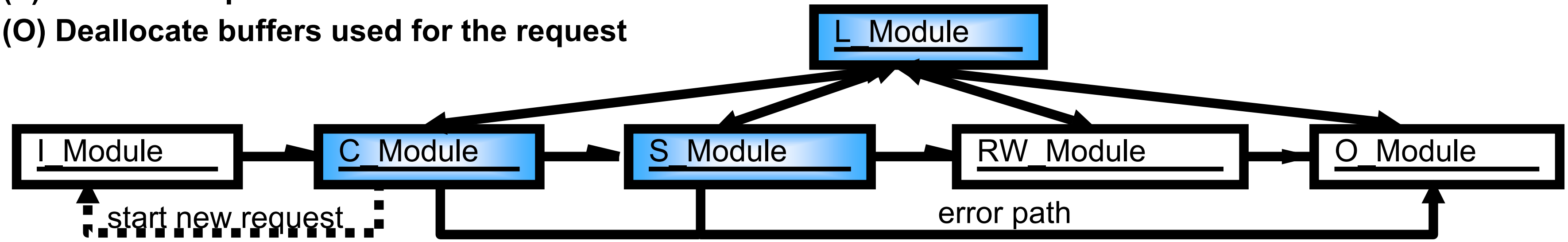
Abstraction of Antara Flow



Flow for Read and Write in Antara

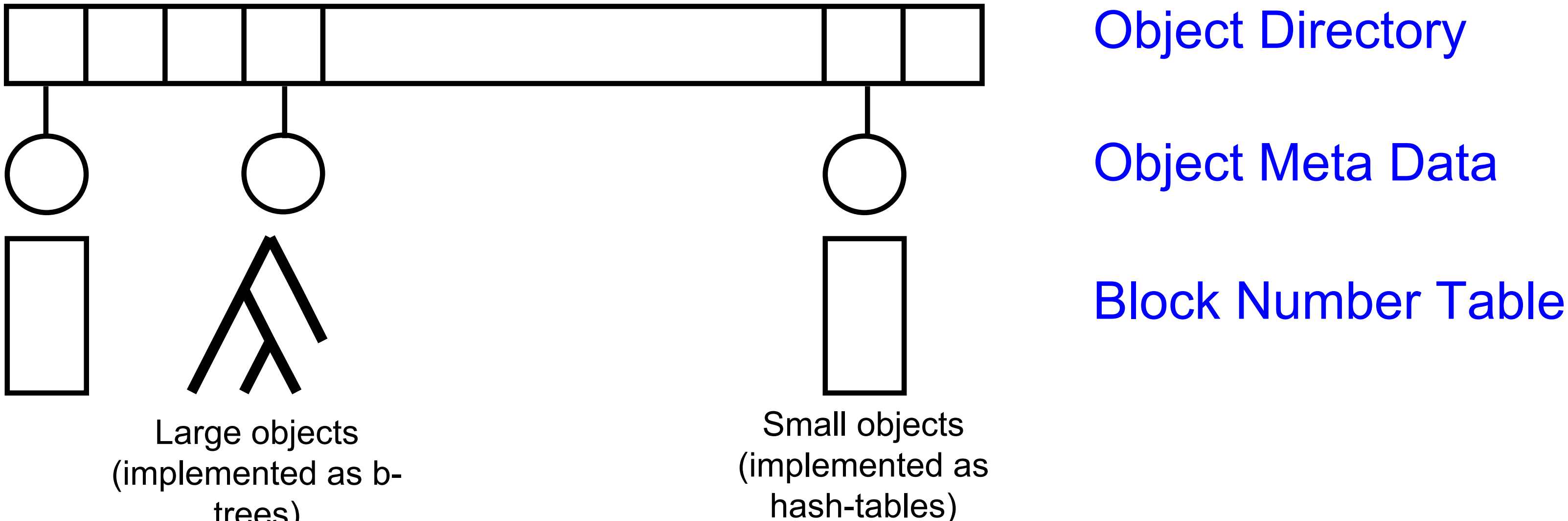
1 **(I) Receive CB, determine needed buffers, allocate and receive data into buffers**
2 **(C) Mark the request as running; Delay this request if clashes with other operation**
3 **(C,I) Notify OS ready to receive the next request from the client**
   Additional requests handled in parallel
4 **(S) Check Security (e.g., proper session, credentials, etc.)**
5 **(L) Perform necessary lookups (and/or allocations in case of writes)**
   Put allocation information in the request
6 **(RW) Read/write the data from the cache**
7 **(L) Mark the request as done**
   1Allows delayed requests to start, e.g., read-write conflict although read response may need to wait for log complete
8 **(L) Log the request (this is a no-op in the case of read)**
   Ensures hardening of the allocation information
9 **(L) Mark the request as logged (this is a no-op in the case of read)**
   Allows delayed requests to complete

- **(O) Send the reply**
- **(L) Mark the request as sent**
- **(O) Deallocate buffers used for the request**



Metadata

- **Significant effort invested in designing efficient, scalable data structures**
  - Minimize contention for concurrent clients and multiple processors
- **Object Directory maps from OID to object's metadata via a sparse hashtable**
- **Object metadata includes object size and a block number table**
- **Block number table maps from offset in block to extents**
  - Small objects use a dynamic, linear-probe, hash table.
  - Large objects use a b-tree.
- **Freespace managed via a buddy list bitmap**



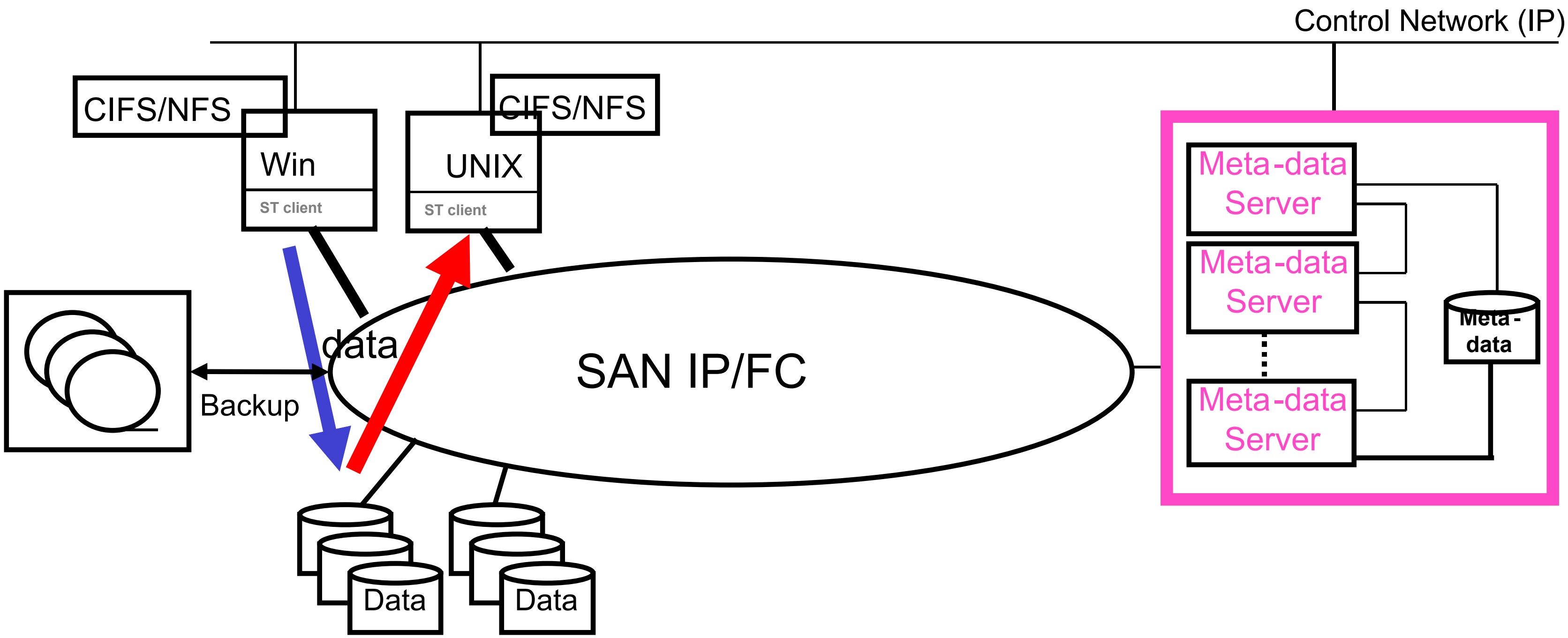
Metadata *(cont)*

- **The linear-probe hash table used for mapping OIDs to OMDs was chosen for parallelism**
  - Not all of the metadata can fit into main memory.
  - Some metadata accesses will require disk accesses
  - Shared/exclusive locks, created on-the-fly, allow locking only specific entries of the hash table.
- **The block-number table implementations were chosen for minimizing page-faults**
  - For large objects, b-trees were chosen based on similar choices in databases.
  - For small objects, an entire b-tree page is inefficient in terms of space and time.
- **Entries in block number table represent extents**
  - Extents range from one block up to the maximum amount of data transferred in a write request

Antara Space Allocation

- **Goals**
  - Logically consecutive blocks of an object will map to consecutive LBAs
  - Avoid fragmentation
  - Quick allocation decisions
- **Approach**
  - Maintain a MRU cache of objects receiving allocating writes
  - Non-persistently associate with each object a large extent of consecutive LBAs
  - For an allocating write, take space from large extent at appropriate offset
    - Persistently update free space bitmap

Storage Tank Background (IBM Research Almaden)



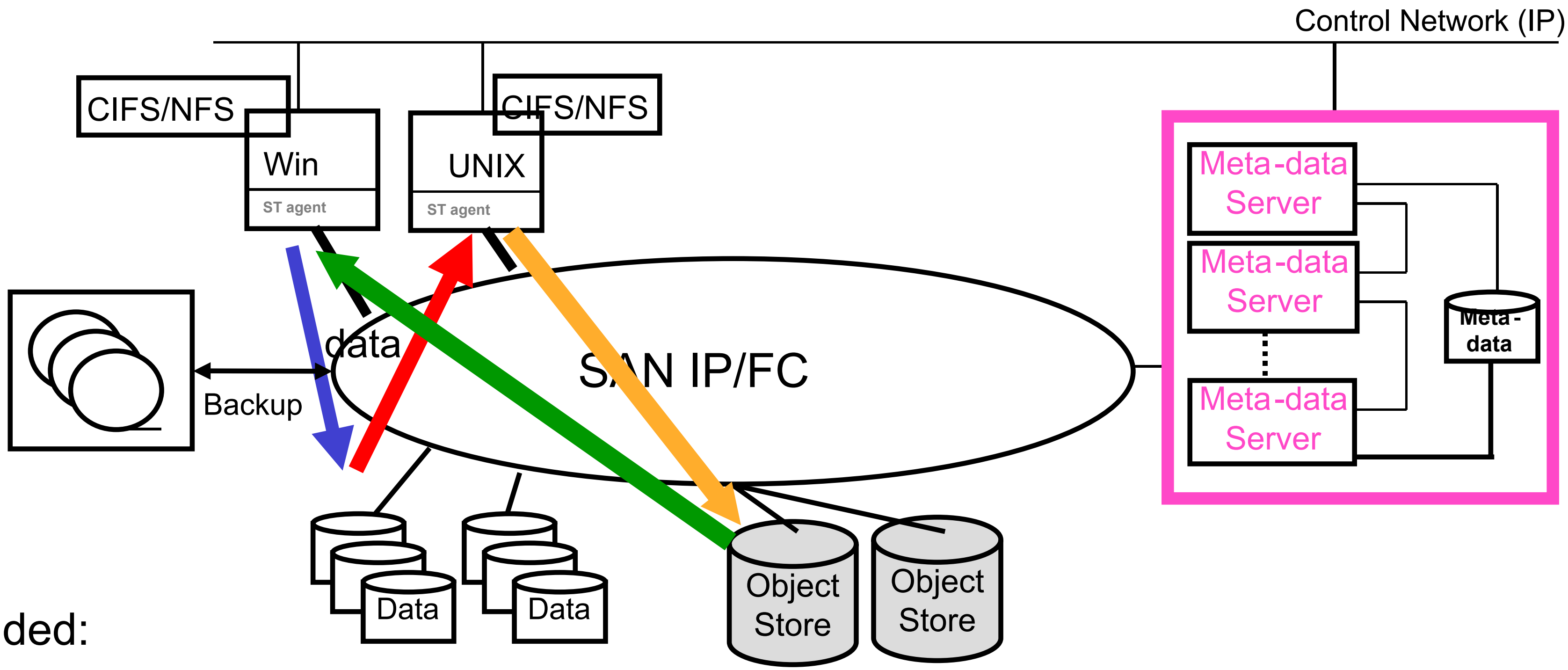
Capabilities:

- Performance and semantics similar to local file system
- Sharing like NAS
- Policy-based, centralized storage management

Object Store and Storage Tank

- **Storage Tank was designed with an Object Store in mind**
- **Object Store will be at same level as control units**
- **To integrate an Object Store into Storage Tank requires changes to**
  - Meta Data Server
  - Storage Tank Client
- **Main customer benefit will be security and protection**
  - Other benefits will follow
- **ObS initiator is the callable module that interacts with the ObS Target**
  - Initiator integrated into both Storage Tank client and Meta Data Server
  - Current Initiator comes in several flavors
    - Synchronous and Asynchronous
    - User-mode and Kernel-mode

Object Store added to Storage Tank



added:

- SAN security
- Scalability
- Manageability