CS 524 A

Introduction to Cloud Computing

Module 10:

1) REST API
2) Content Delivery
3) More on Storage, including Access Security

Outline

- Representational State Transfer (REST) and REST programming style vs. Remote Procedure Call (RPC)
- Content Delivery

  How Akamai works
- More on Storage
  - The evolution of storage in the Cloud
  - Access security in block-level NAS
  - File-level storage virtualization

Make sure you read the textbook as Homework #5 requires more than what we can cover in class

REpresentational State Transfer (REST)
The Cloud API Style

Reading

Fielding, Roy Thomas.

Architectural Styles and the Design of Network-based Software Architectures.

Doctoral dissertation, University of California, Irvine, 2000.

http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm

Some history: Remote Procedure Call (RPC)

- Involves specifying the precise interface (binding)
- Defines procedures and their parameters in detail
- Is carried by
  - *marchalling* the procedure parameters in the request (through *stubs*);
  - carrying the procedure call on a remote machine; and
  - returning the result to the client
- Has been implemented on the Web in ***Simple Object Access Protocol*** (*SOAP*), typically carried by HTTP

Web Scale Demands that

- the browser code not be *bound* to the service interface

  In other words, when you introduce a new service at www.example.*FreeBeer.com*, the browser code must not change

- the client code be bound to the interface defined by
  - URIs
  - HTTP *methods* invoked on each URI
  - *Representations* (documents that describe resources) obtained as the result of that reference (often defined by *metadata*)
- the service state not be stored at the server

An Example

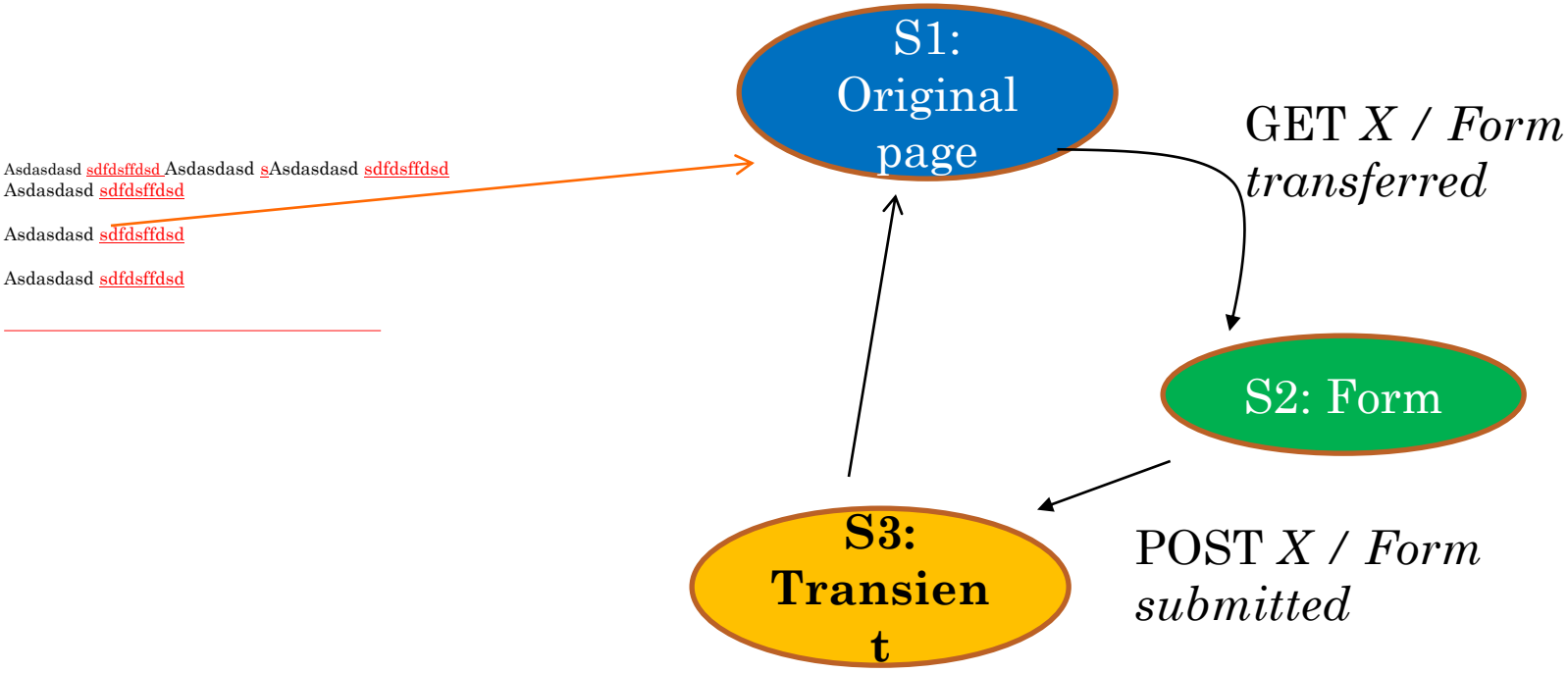
Invoking the *HTTP GET* method *on* *http://www.example.com/freebeer/Lagers* returns a *representation* of a certain resource as an HTML document

- The resource might refer to *something* stored in a database and converted into HTML format (as we saw before)
- The page may contain image URIs, which will be retrieved, too by *GET*, but these are almost certainly the resources that are stored as JPEG or JIG files on some server
- The same resource may be referenced by another URI (e.g., *http://www.example.com/freebers?style=lager*)
- The same resource may have multiple representations, depending on a specific format established in *content negotiations*

Constraining the uniform remote interfaces

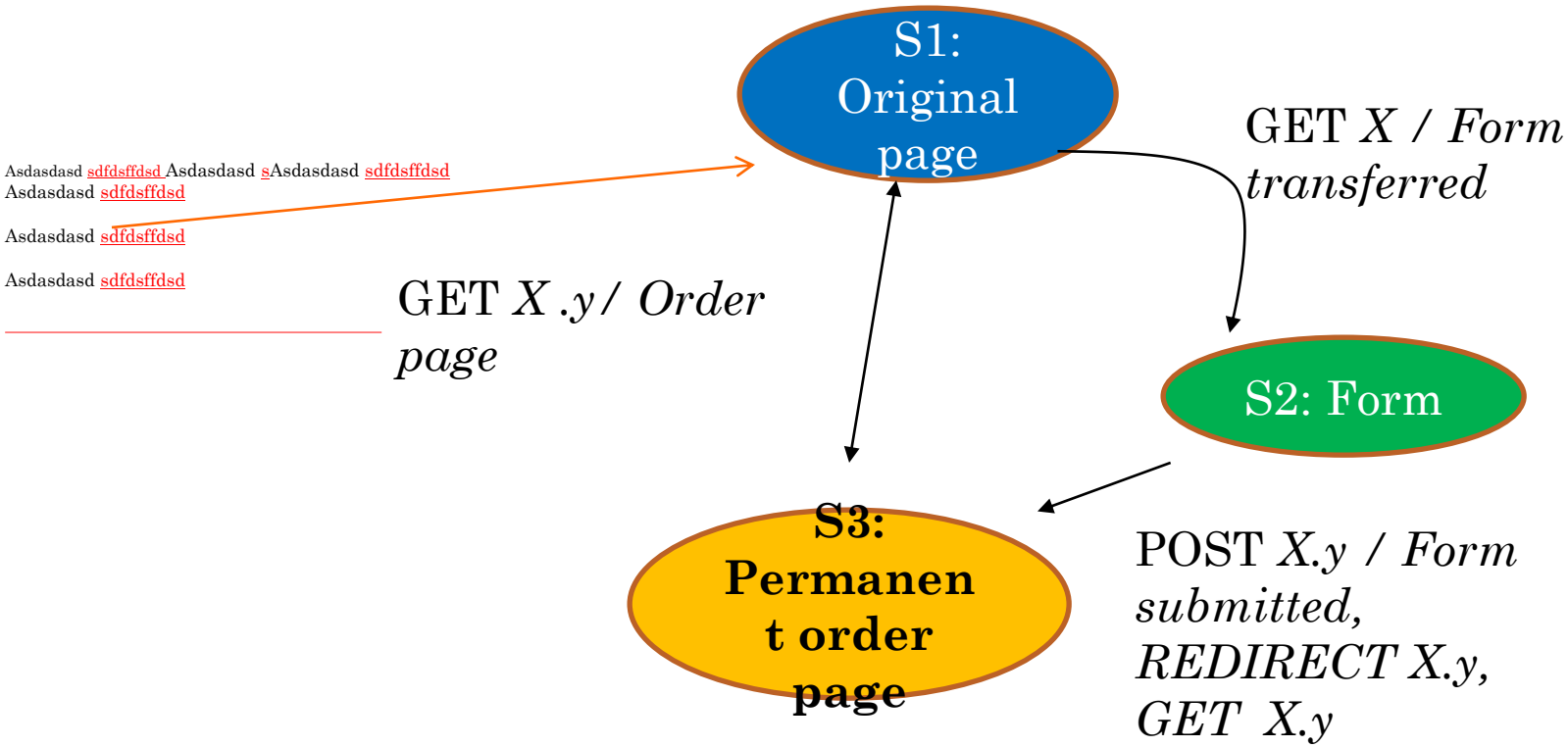
- Each URI on a site has a set of methods it responds to
- Each <URI, Method> pair typically defines a specific format (e.g., *form-url* for POST), and it may define additional constraints
- The constraints are further defined by the type of hypermedia (generalization of text hyperlinks extending to graphics, video, voice, etc.)

The Application State Machine: Transient States



A transient state is *not* associated with a URI, and so there is no way to return to it without resubmitting a form

The Application State Machine: Eliminating a Transient State through *PRG*



POST/REDIRECT/GET (PRG) is a mechanism to eliminate transient states *REDIRECT* is a versatile method, whose various applications to dynamic resource discovery, *authentication,* and *authorization,* we will encounter in the rest of the course

Roy Fielding's Rules for RESTful API

A REST API

- should concentrate on the definition of the media types used for representing resources and driving application state
- should be entered with no prior knowledge beyond the initial URI (bookmark) and set of standardized hypermedia types
- should not depend on any single communication protocol (instead, it should adapt to an existing protocol through metadata)
- should not rely on any changes to the communication protocols
- must not define fixed resource names or hierarchies (instead, it should allow servers to instruct clients on how to construct appropriate URIs, as is done in HTML forms and URI templates, by defining those instructions within hypermedia types and link relations)
- should never have "typed" resources that are significant to the client

Fielding's design principles

- Separate the client (user interface) concerns from the server (data storage) concerns *(no client-to-server interface binding)*
- Ensure that each request from client to server contains all of the information necessary to understand the request (*server is stateless*)
- Ensure that data within a response to a request be implicitly or explicitly labeled as cacheable or non-cacheable *(elimination of extra interactions*)
- Use *layering* to encapsulate legacy services and to protect new services from legacy clients
- Use *code-on-demand*
- Apply four constraints:
  1. *identification* of resources;
  2. *manipulation* of resources *through representations*;
  3. *self-descriptive messages*; and
  4. *hypermedia* as the engine of application state

The Definition Sequence for designing RESTful services

1. Define the data and operations
   - Data types (e.g., types of beer available)
   - Operations (e.g., *searching* for a given type of beer, *ordering*, *asking for suggestions*)
2. Define the application state machine
   - Effectively a flowchart, or a precedence chart (e.g., *finding* beer must precede *ordering*)
   - Consider the links on a page as a mechanism to drive the state
3. Define resources and their respective methods
   - Start with re-formulating data types into the resource names
   - Map the respective operations into HTTP methods
4. Define the resource URIs
   - Assign each resource one or more URIs
   - Ensure that hierarchical information is represented in the path portion of the URI (e.g., www.example.freebeer.com/lager/pilsener/Pilsner_Urquell)
5. Define input representations

   Select a subset of input representation form based on the specifics of the information to be passed
6. Define the resource representations
   - Capture the state of the resource
   - Express the methods to be used
7. Define response codes and metadata
   - <meta name="keywords" content="beer, lager, ale" >
   - <meta http-equiv="expires" content="Tue, 8 May 2012 20:00:00 GMT">

Complexity warning: An example of an **EVIL** linked representation

What you see:

```
<a href="./evil_code.exe">
 <img src="./beer.jpg"/>
 </a>
<p>
Click on the picture to see a larger mug!
</p>
```

What you get after clicking:

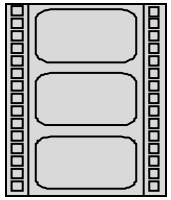
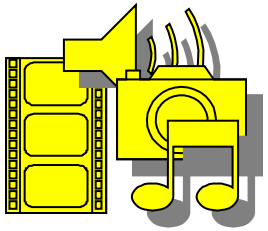
+

Browsers must check media compatibility in representations, but…

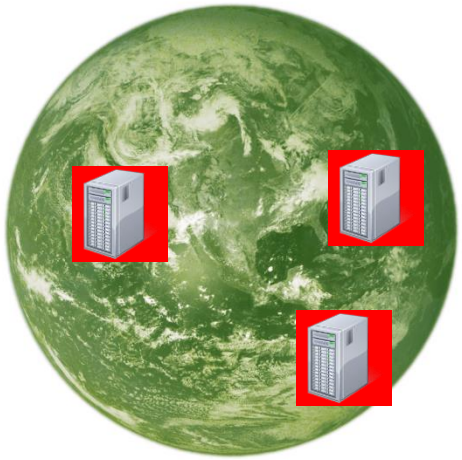
Content Delivery Networks (CDNs)

The CDN Business Model: Make money on the *World-Wide Wait* (A. Tanenbaum)

1. A CDN charges *content providers* for delivering their content efficiently


2. A CDN pays ISPs for allowing to place the CDN servers in their specific locations



3. A CDN pays the Cloud provider for VMs at specific geographic locations

Redirection Requirements

- The client's request must be redirected to the server nearest to the client
- The redirection must be performed without DNS modification

How can this be done?

The process

1. CDN gets the web site from the content provider
2. Each page is run through a preprocessor, which modifies all URLs related to large (say, streaming media) objects
3. The original page *stays* on the provider's site, but the media URLs now point to a CDN's own central server (which has no media but knows how to do **hard math!)**
4. Once that CDN server gets a user's request, it
   - Determines user's location (**not that easy)**
   - Finds the nearest *actual* server with the cached media **(very hard)**
   - *REDIRECTs* the user to the server so found

An experiment: view-source:http://www.nytimes.com/

- </script><script type="text/javascript" src="http://cdn.krxd.net/kruxcontent/releases/krux-4.9.10.2.js"></script><!-- End KRUX Digital Control Tag for New York Times -->

nslookup cdn.krxd.net -->

Name: e5180.g.akamaiedge.net

Address: 23.15.242.251

Aliases: wildcard.krxd.net.edgekey.net

- <script type="text/javascript"> if (/iPad|iPod|iPhone/.test(navigator.userAgent)){ document.write('<img id="mastheadLogo" width="379" height="64" alt="The New York Times" src="http://i1.nyt.com/svg/nytlogo_379x64.svg">'); } else { document.write('<img id="mastheadLogo" width="379" height="64" alt="The New York Times" src="http://i1.nyt.com/images/misc/nytlogo379x64.gif">'); } </script>

Nslookup i1.nyt.com -->

Name: a1859.g.akamai.net

Address: 64.211.162.81, 64.211.162.97

Aliases: i1.nyt.com.edgesuite.net

Akamai Technologies, inc.

- Was founded by 1998 by the MIT graduate student Daniel Lewin (May 14, 1970 – September 11, 2001) and his Applied Mathematics professor Tom Leighton
- Is now a lead CDN and Cloud platform. (Please study the materials at http://www.akamai.com and get *facts* about its customer base)
- Employs its own DNS servers with name servers for *acamai.net* performing *dynamic redirection*, which it pioneered

Akamai means "intelligent" in

Locating the "nearest" server

- is *not* based on geography alone
- is based (at least) on two factors:
  1. The *shortest high-capacity* path
  2. Load already carried by a candidate node
- Starts with simple ideas
  - The HTML re-write for *akamaising* is relatively simple
  - DNS programming is fairly simple (see http://research.microsoft.com/en-us/um/people/ratul/akamai.html)
- Builds on complex mathematics for caching algorithms that (see **E. Bommaiah, K. Guo, M. Hofmann, and S. Paul**: *Design and Implementation of a Caching System for Streaming Media over the Internet*. IEEE Real Time Technology and Applications Symposium 2000 and also http://www.siam.org/pdf/news/790.pdf)

Study math!

Bibliography

- M. Hofmann and L. Beaumont, *Content Networking: Architecture, Protocols, and Practice*, The Morgan Kaufmann Series in Networking, 2005.

- D.R. Karger, E. Lehman, T. Leighton, M. Levine, D. Lewin, and R. Panigrahy, *Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the World Wide Web*, Proceedings of the 29th ACM Symposium on Theory of Computing, ACM, ACM Press, May 1997, 654–663.

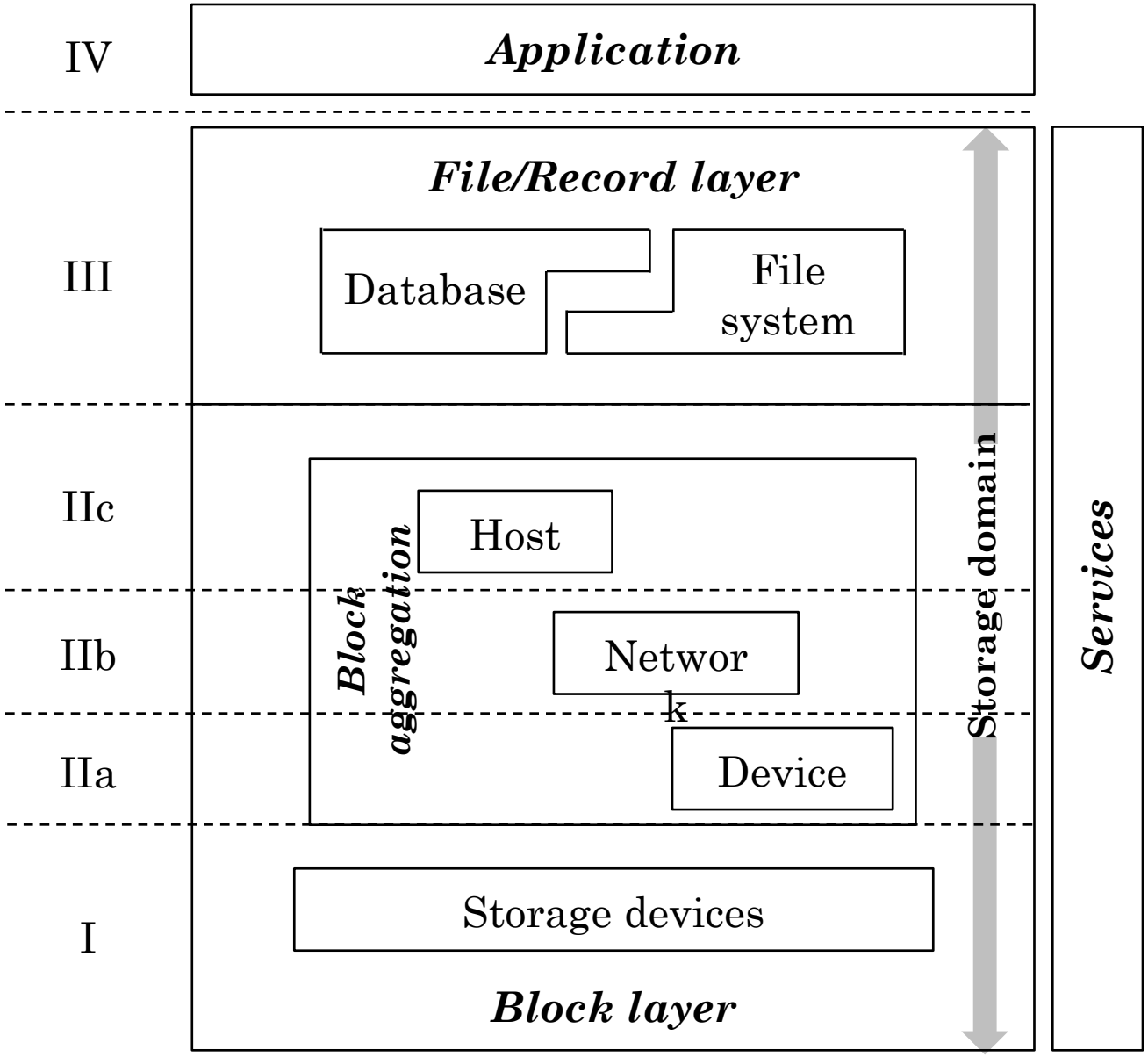
Back to the Three Data center Components

- *Compute*
- *Storage* 
- *Networking*

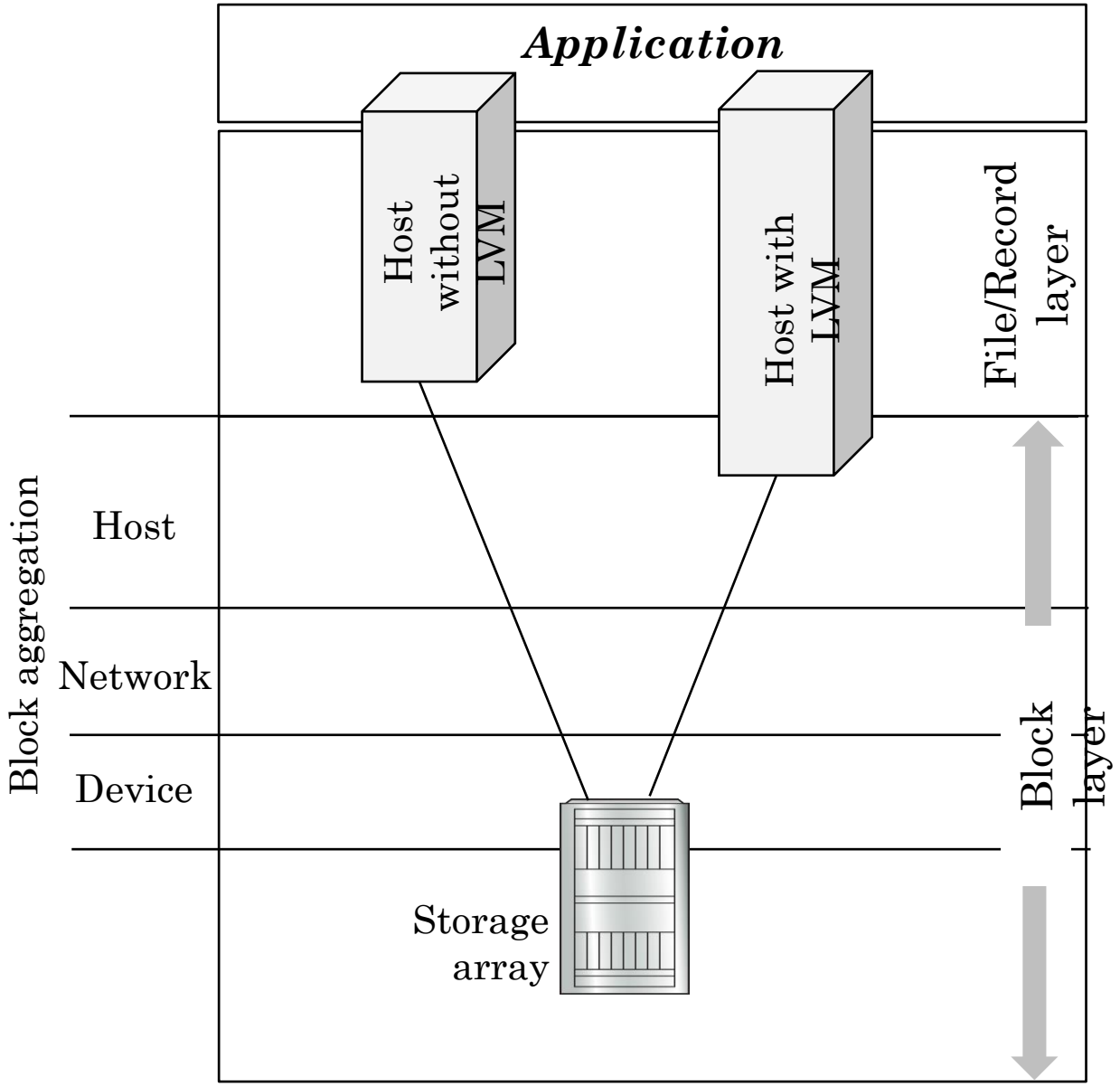
Coping with the data deluge

1. The storage capacity needs to keep increasing
2. Stored data need to be secured
3. Access to the data needs to be more efficient

The Storage Networking Industry Association (SNIA) Shared Storage Model



Direct-attached Storage

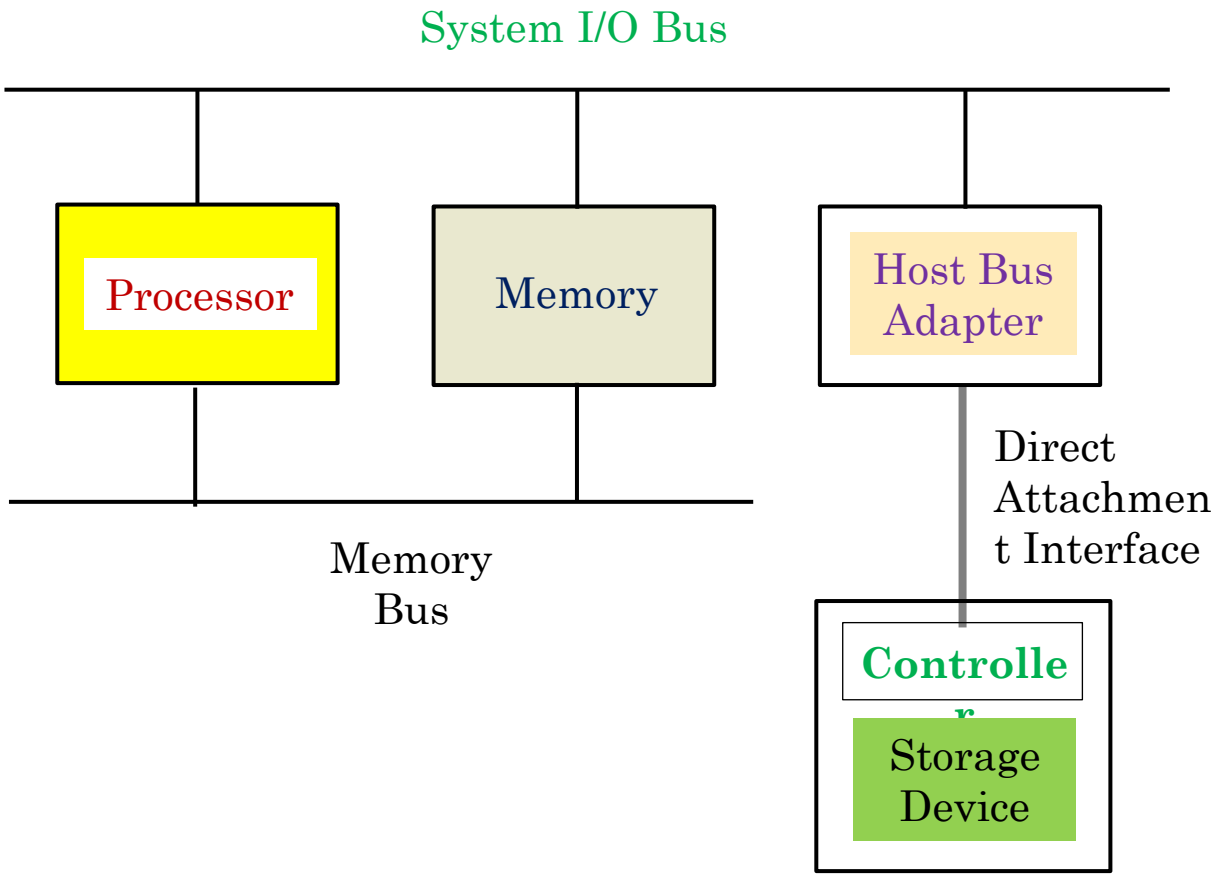


LVM: Logical Volume Manager

About DAS

- It is dedicated to a single host
- It is connected by a point-to-point link
- It is controlled by the host
- It keeps host-critical local data (boot image, swap space, etc.)
- It can be
  - *Internal* (e.g., hard disk inside the server) or
  - *External*, attached through a standard interface (e.g., the *Small Computer System Interface [SCSI]*)

A schematic view of the direct attachment interface



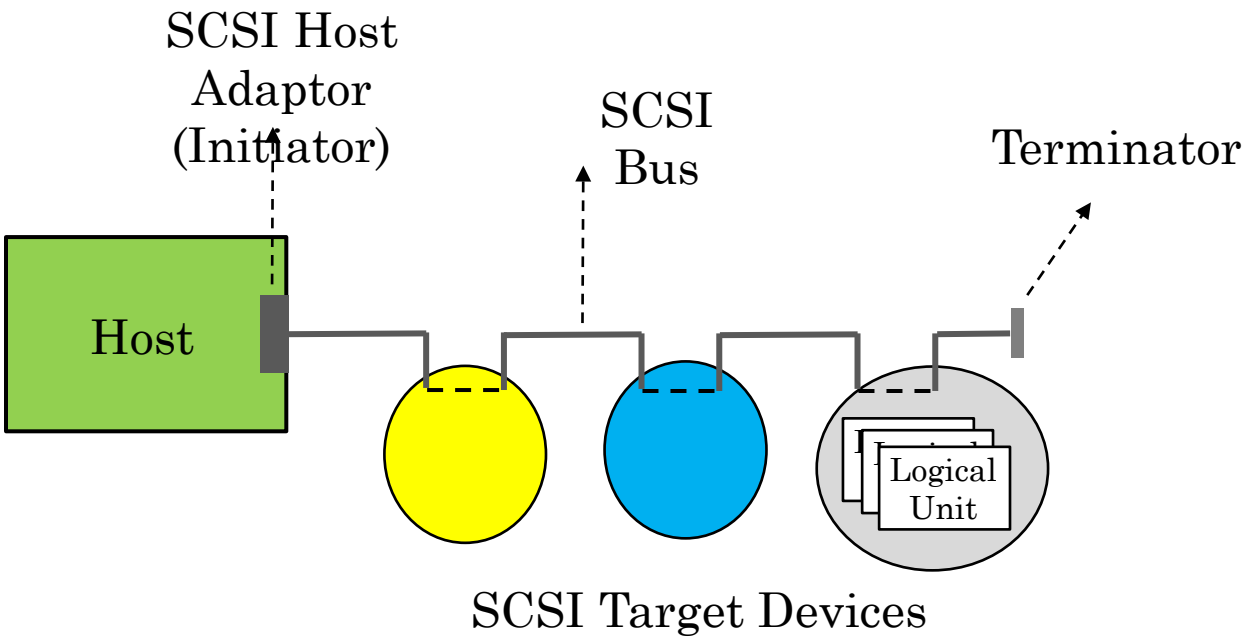
SCSI

- Is standardized by the *American National Standards Institute (ANSI)* in the **X.3.131-1986** specification
- Is based on the *Shugart Associates System Interface (SASI)* circa 1979
- Defines
  - a parallel bus for attaching various types of devices (disks, tape drives, CD ROM, … drones)
  - a daisy-chaining scheme (next slide)
  - priority levels within the chain
  - Logical Unit Numbers (LUNs) that represent devices to an OS

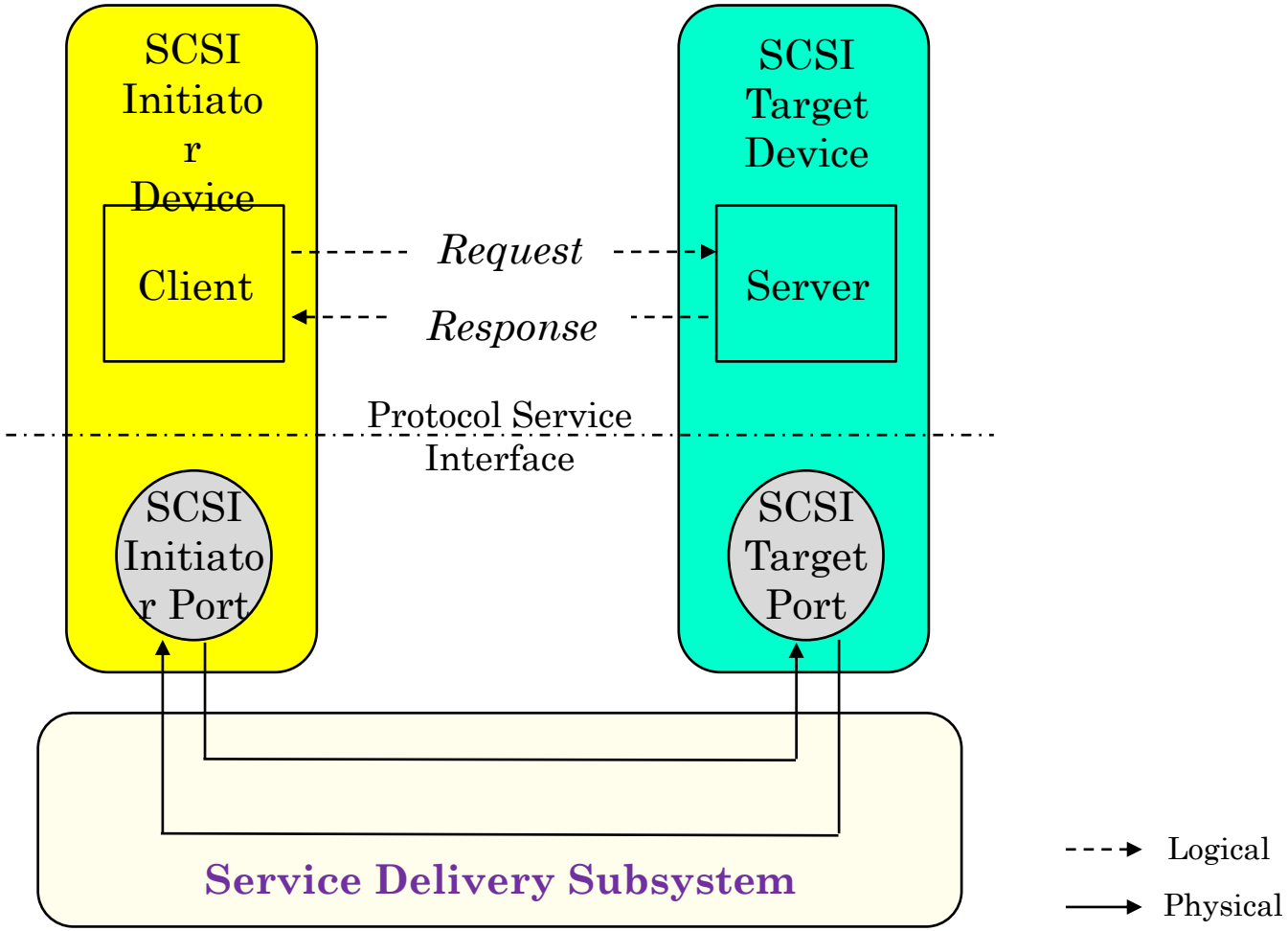
CD Juke Box as an example of a multi-LUN device

- Each CD is a separate unit, which has its own Logical Unit Number

An Example SCSI Configuration (Daisy-Chaining)



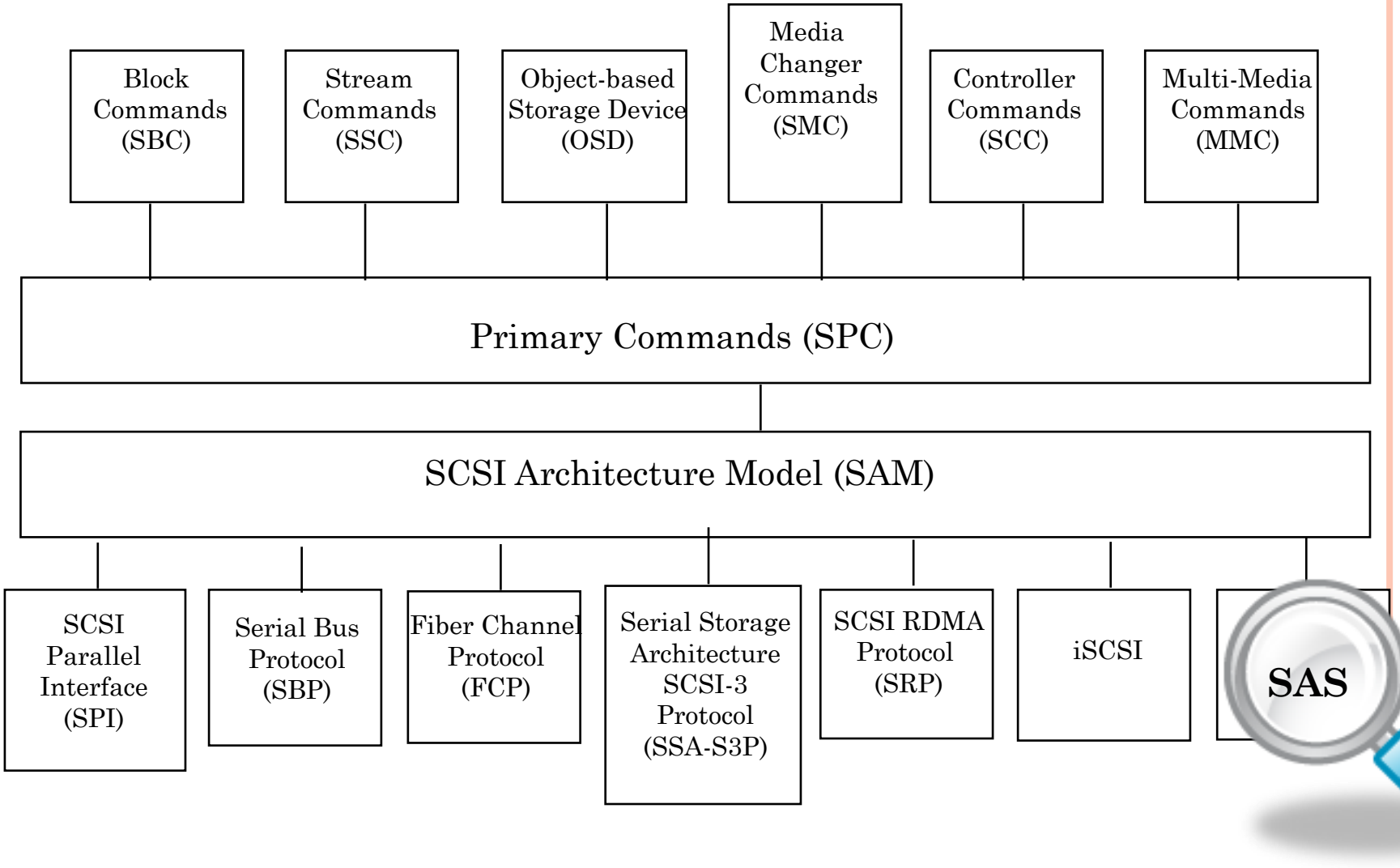
The SCSI Client-Server Model



A comparison of different SCSI versions

Version	Data Bus Width (bit)	Clock Rate (MHz)	Throughput (MBps)	No. of Devices
SCSI-1	8	5	5	8
Fast SCSI (SCSI-2)	8	10	10	8
Fast Wide SCSI (SCSI-2)	16	10	20	16
Ultra SCSI (SCSI-3)	8	20	20	8
Ultra Wide SCSI (SCSI-3)	16	20	40	16
Ultra2 SCSI	8	40	40	8
Ultra2 wide SCSI	16	40	80	16
Ultra3 SCSI	16	40	160	16
Ultra-320 SCSI	16	80	320	16
Ultra-640 SCSI	16	160	640	16

The world of SCSI Standards



Problems with earlier SCSI technology

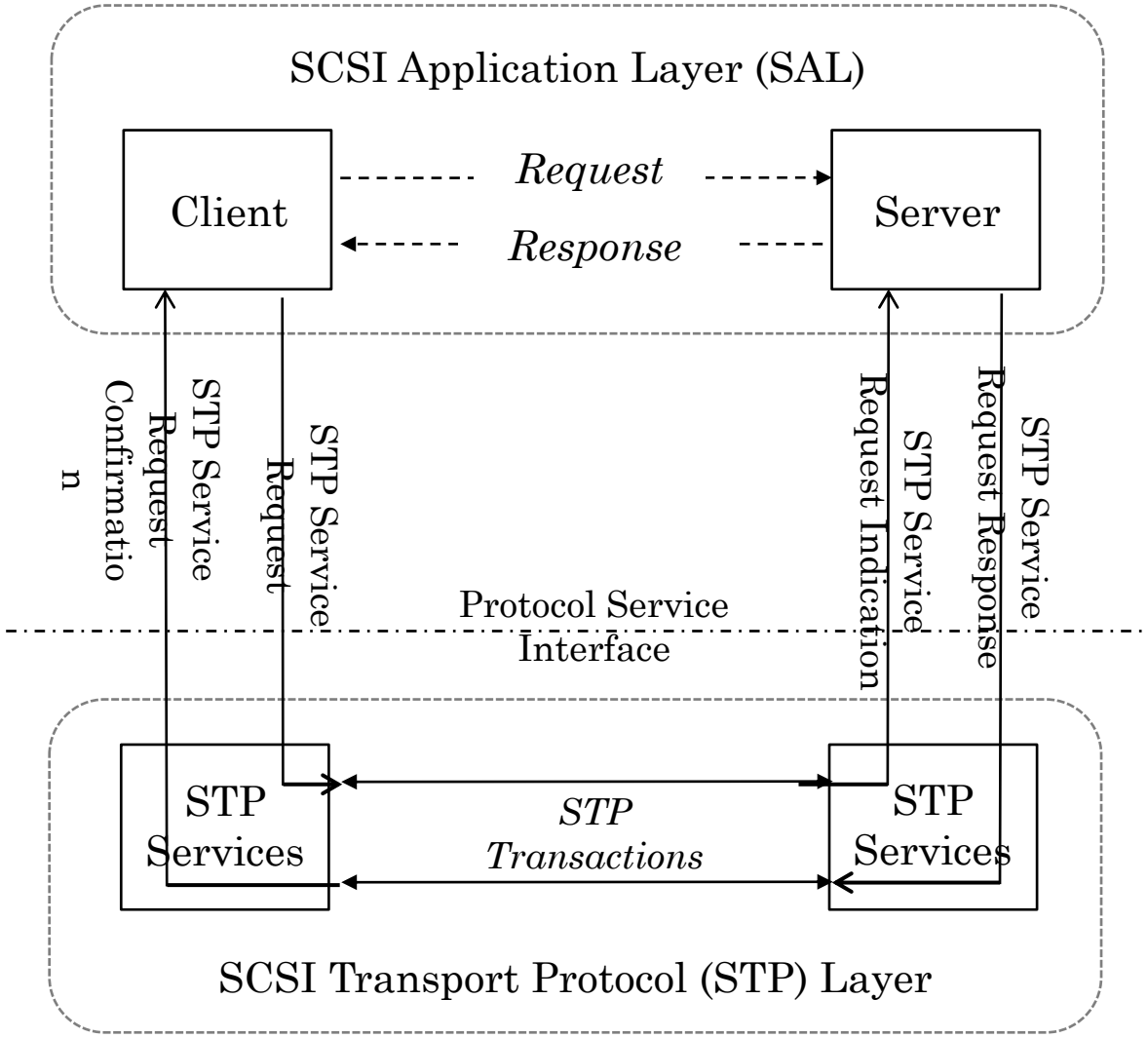
- Cross-talk
- Inconsistent signal arrival times (timing skew)

Those are caused by the existence of multiple lines used to transfer data in parallel.

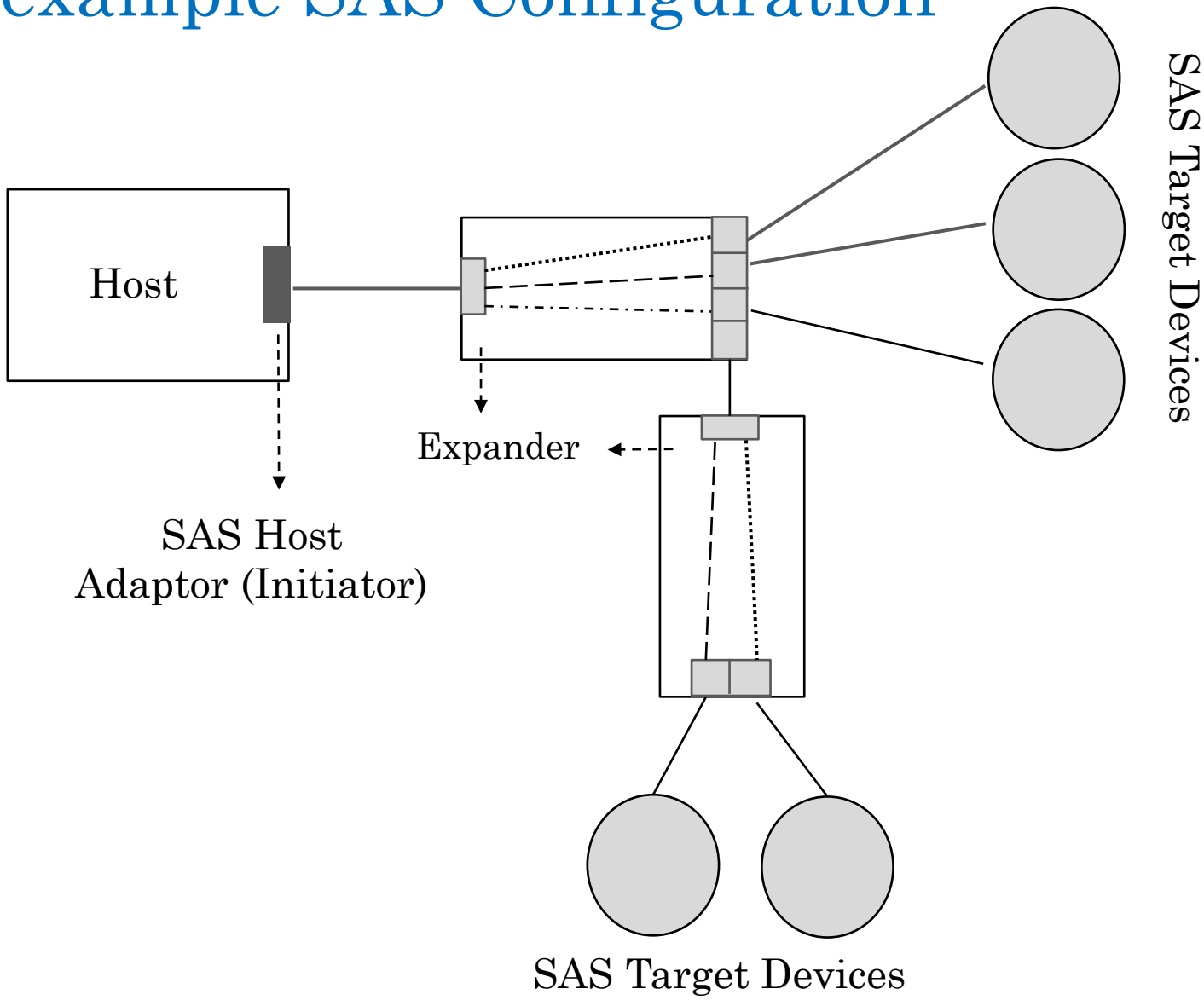
The departure from daisy-chaining…

- The *Serial Attached SCSI (SAS)* standard, ANSI INCITS 376-2003, published in 2003, represented a shift to *serial attachment*
- Major influence: the *Advanced Technology Attachment (ATA)* interface already implemented in PCs
- ATA has evolved into *Serial ATA (SATA)*, which has influenced SAS.

The SCSI inter-layer relationship



An example SAS Configuration

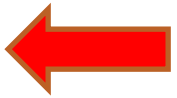


The serial attached SCSI architecture

ATA: Advanced Technology Attachment
SSP: Serial SCSI Protocol
STP: Serial ATA Tunneled Protocol
SMP: Serial Management Protocol

Application Layer SCSI operations, ATA operations, SAS management
Transport Layer Frame definitions of SSP, STP, and SMP
Port Layer *Port management*
Link Layer *Connection management and flow control for SSP, STP, and SMP*
Phy Layer *Line coding, out-of-band signaling*
Physical Layer *Physical and electrical characteristics of cables and connectors*

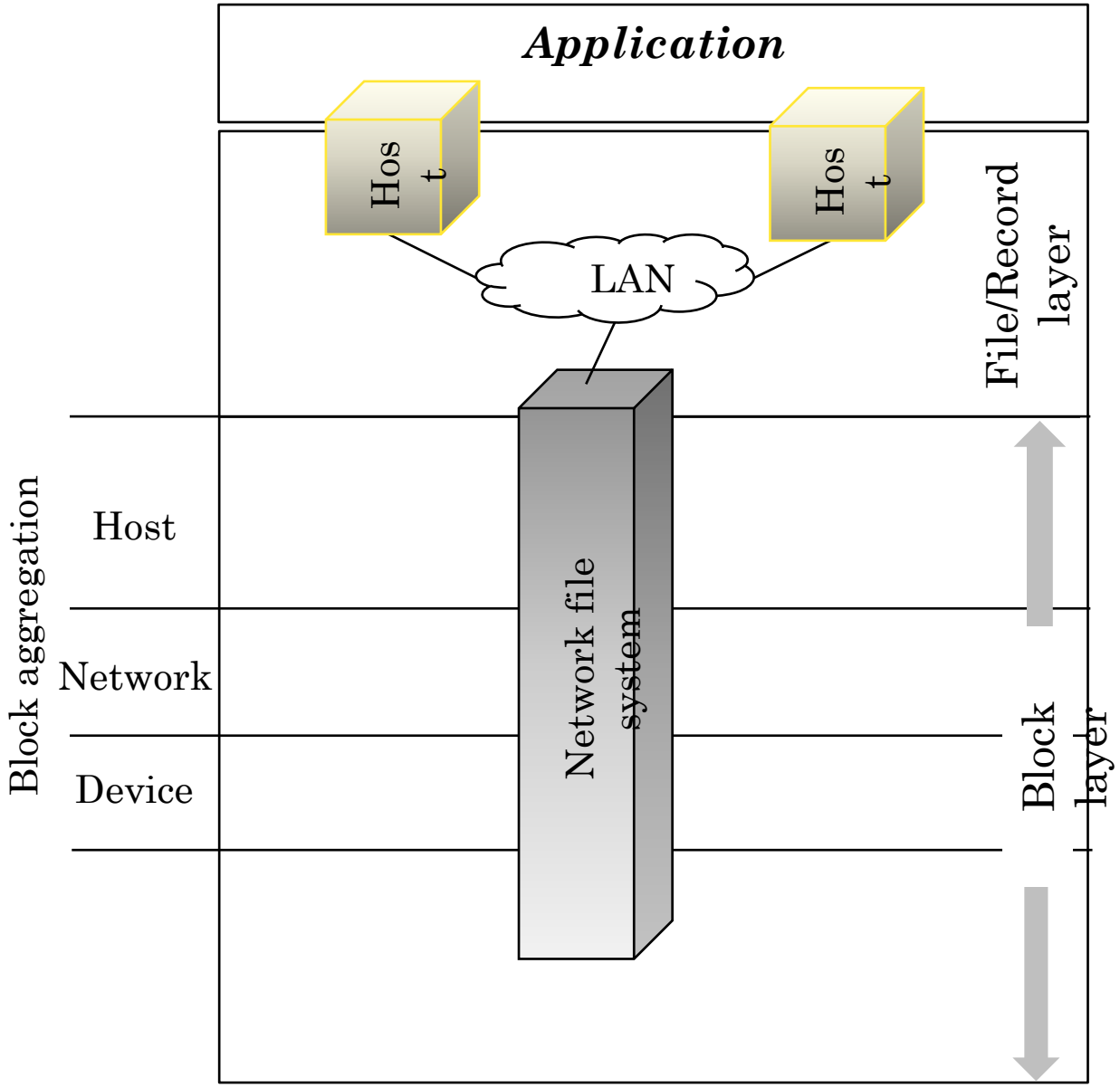
Storage (checkpoint)

- ***Direct-attached storage (DAS)*** Done
- ***Network-Attached Storage (NAS)*** 
- ***Storage Area Network (SAN)***

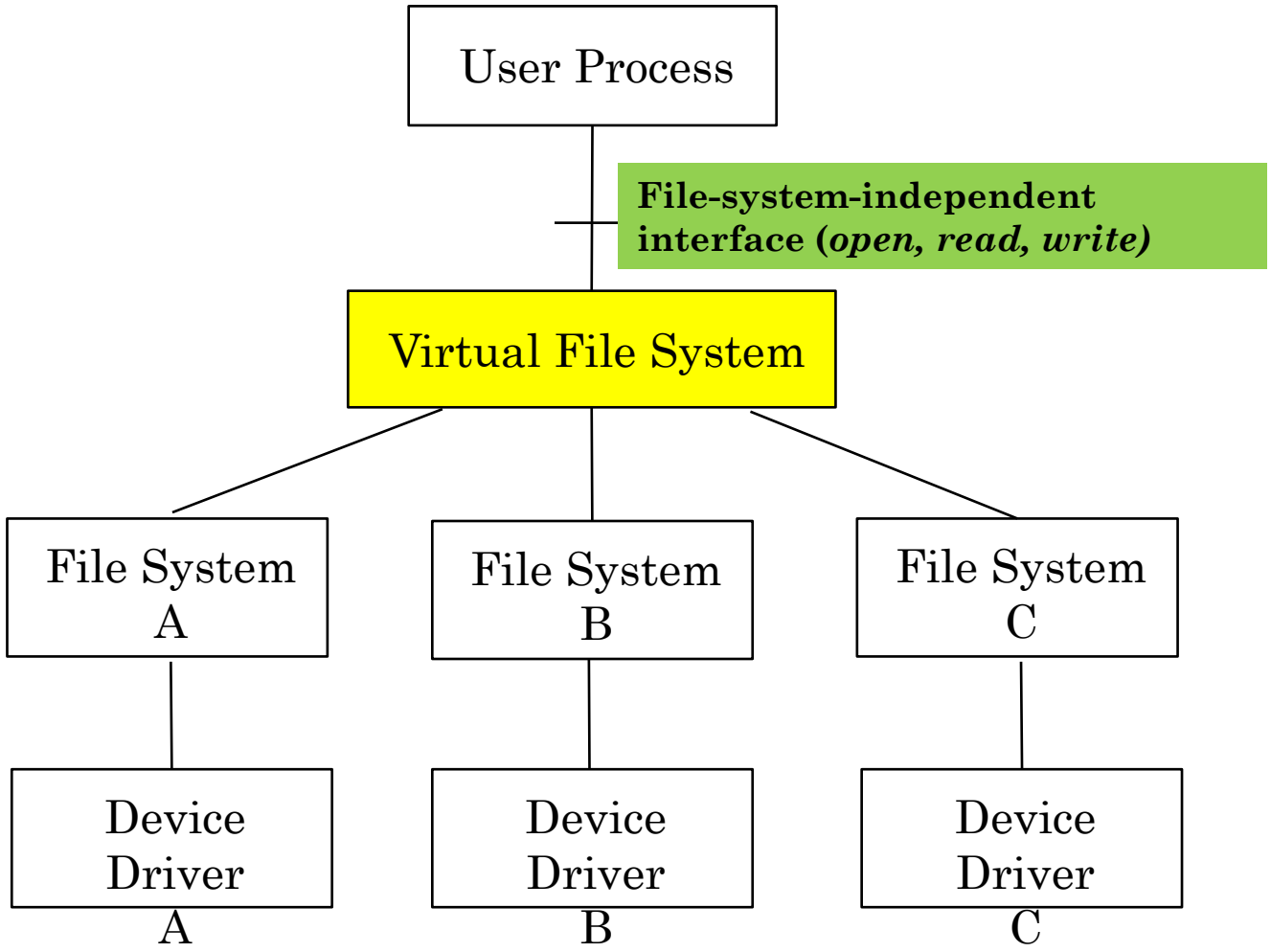
Network-Attached Storage

- Provides the *file-* or *object-level* access over LAN
- First appeared in the form of the *Network File System (NFS)* designed in 1980 by *Sun Microsystems*
- Uses an abstract file interface, *Virtual File System (VFS)* to shield from actual file implementation—both locally and over-the-network. (VFS primitives: *open, close, read, write*)

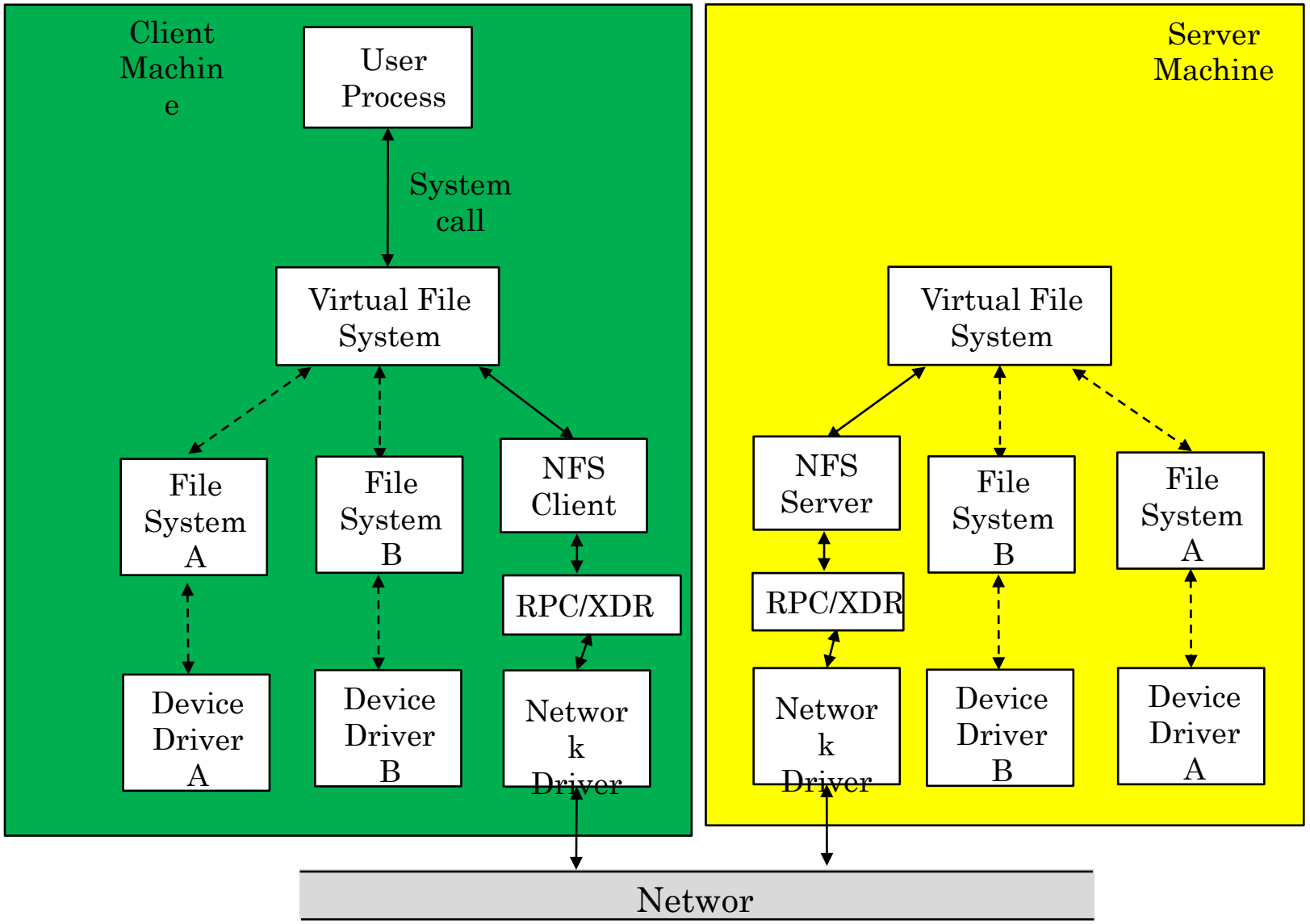
Network File System



The file system abstraction



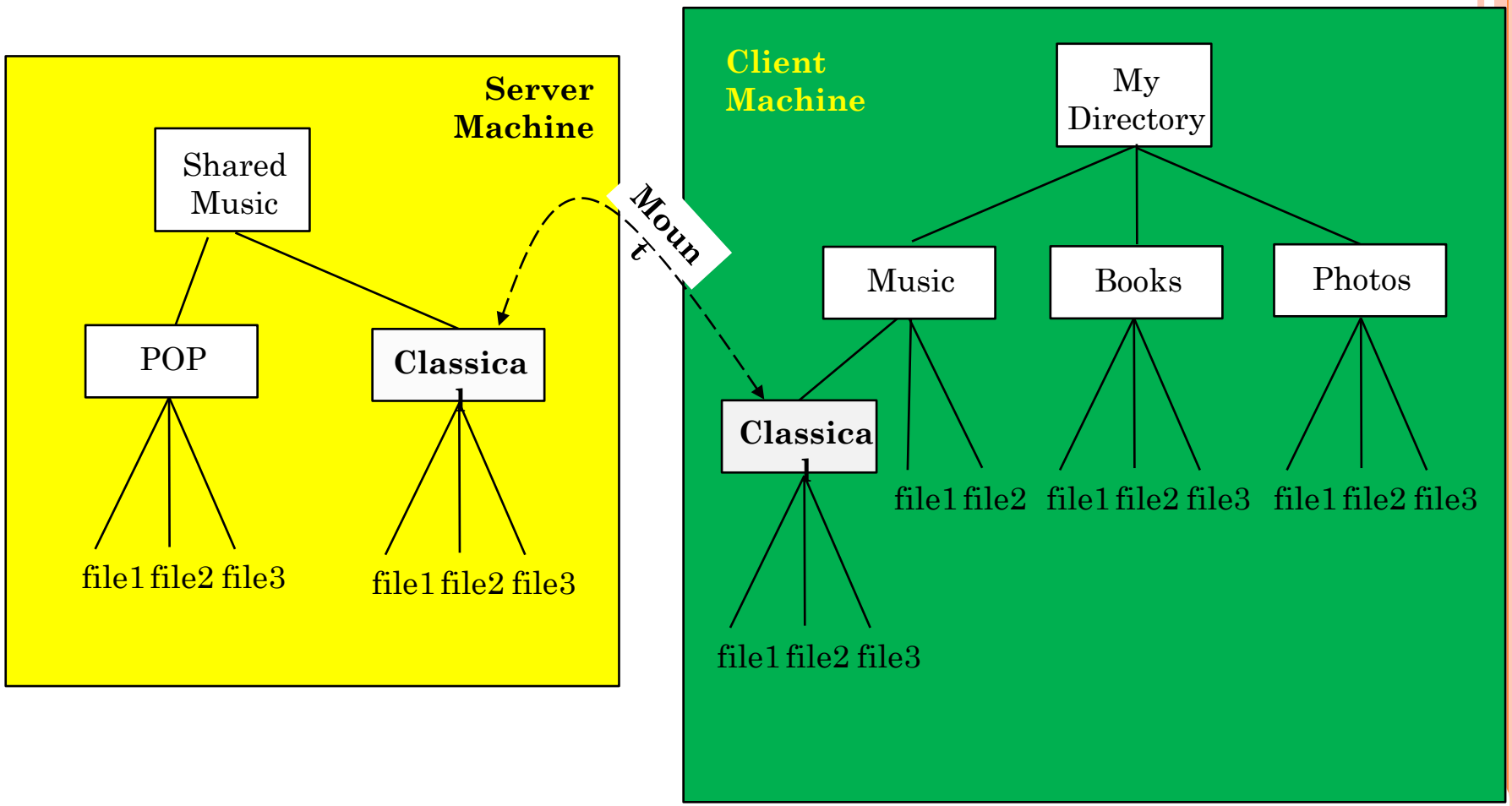
The SUN MS implementation of the Network File System



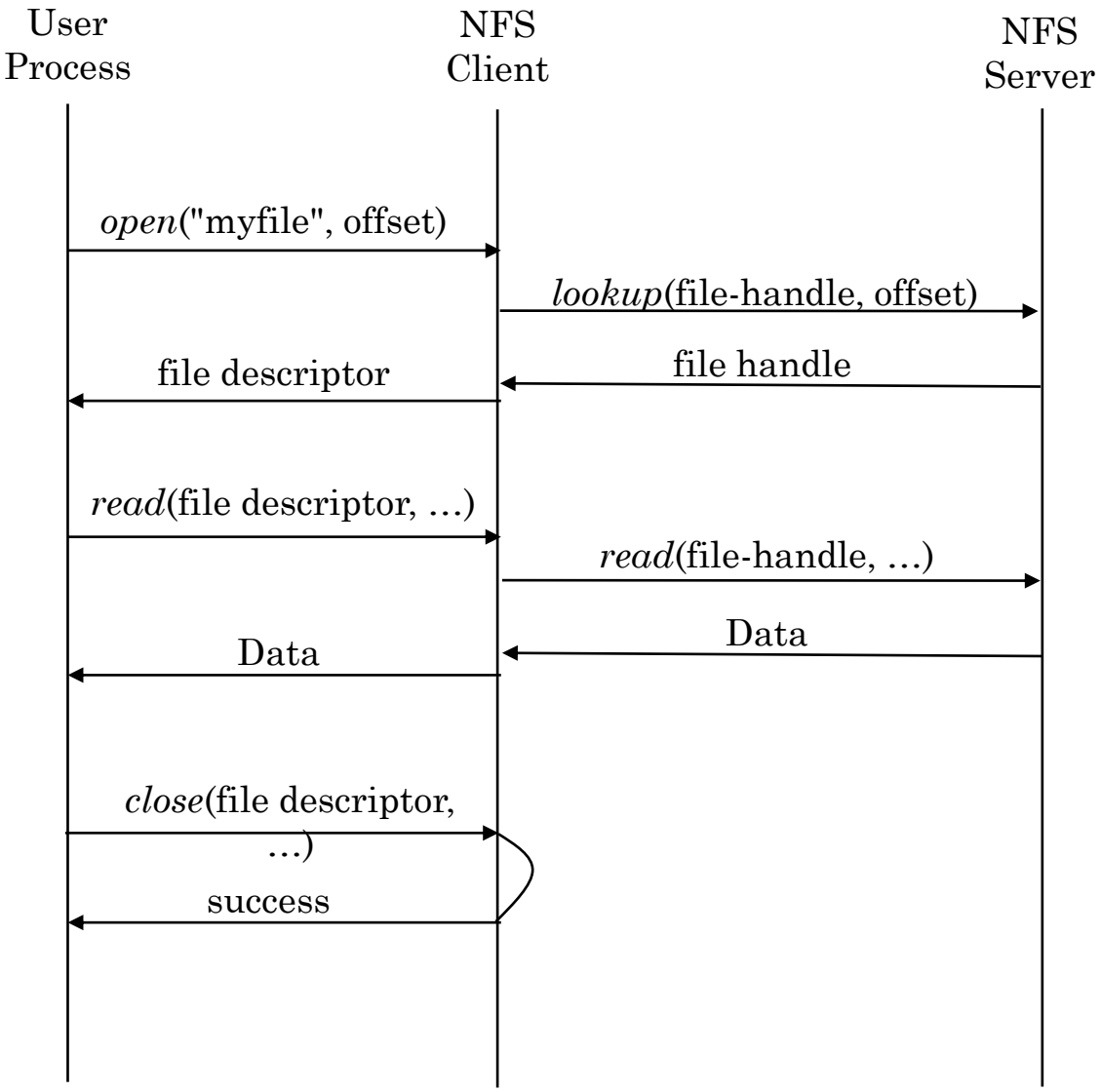
RPC: Remote Procedure Call
XDR: eXternal Data Representation

The server is *stateless*!

Mounting a remote directory through NFS

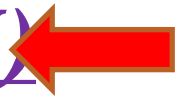


Remote file operations



Storage (checkpoint)

- ***Direct-attached storage (DAS)*** Done
- ***Network-Attached Storage (NAS)*** Done
- ***Storage Area Network (SAN)***

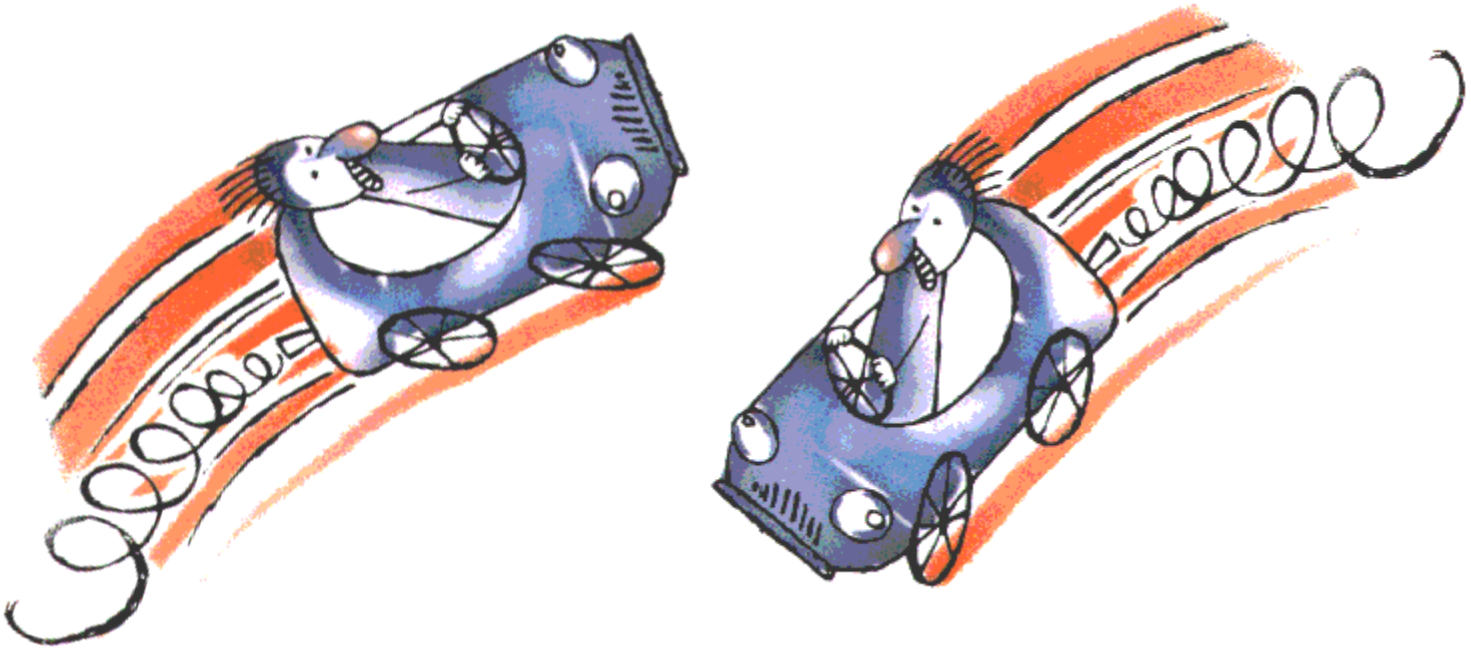


Motivation for Developing SAN: Resource sharing

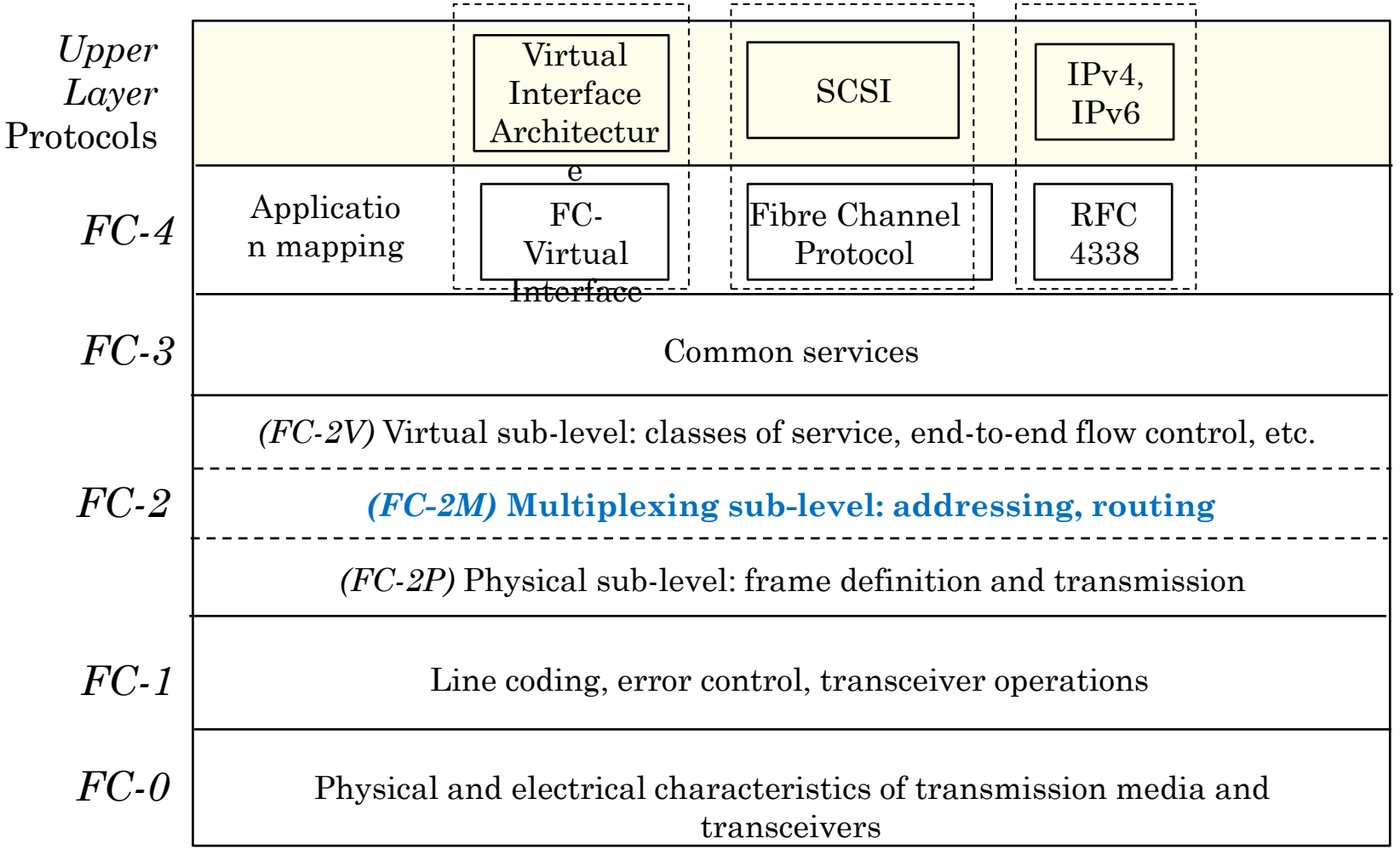
- DAS ***does not*** support resource sharing
- NAS is an improvement, but has throughput limitations as its many implementations ***do* not** support the block-level I/O access granularity
- **SAN solves this problem** by employing the ***fibre channel (FC)*** technology
  - Specified in ANSI X.3.230-1994
  - Combining the qualities of a serial I/O bus **and** a switched network
  - Tailored to interconnecting storage systems to allow resource pooling and block-level access to pooled resources

IT'S ALL ABOUT SPEED!

- The fastest FC supports **3.2 GBps** throughput in each direction



The Fibre Channel (FC) structure



FC-2M Connection types

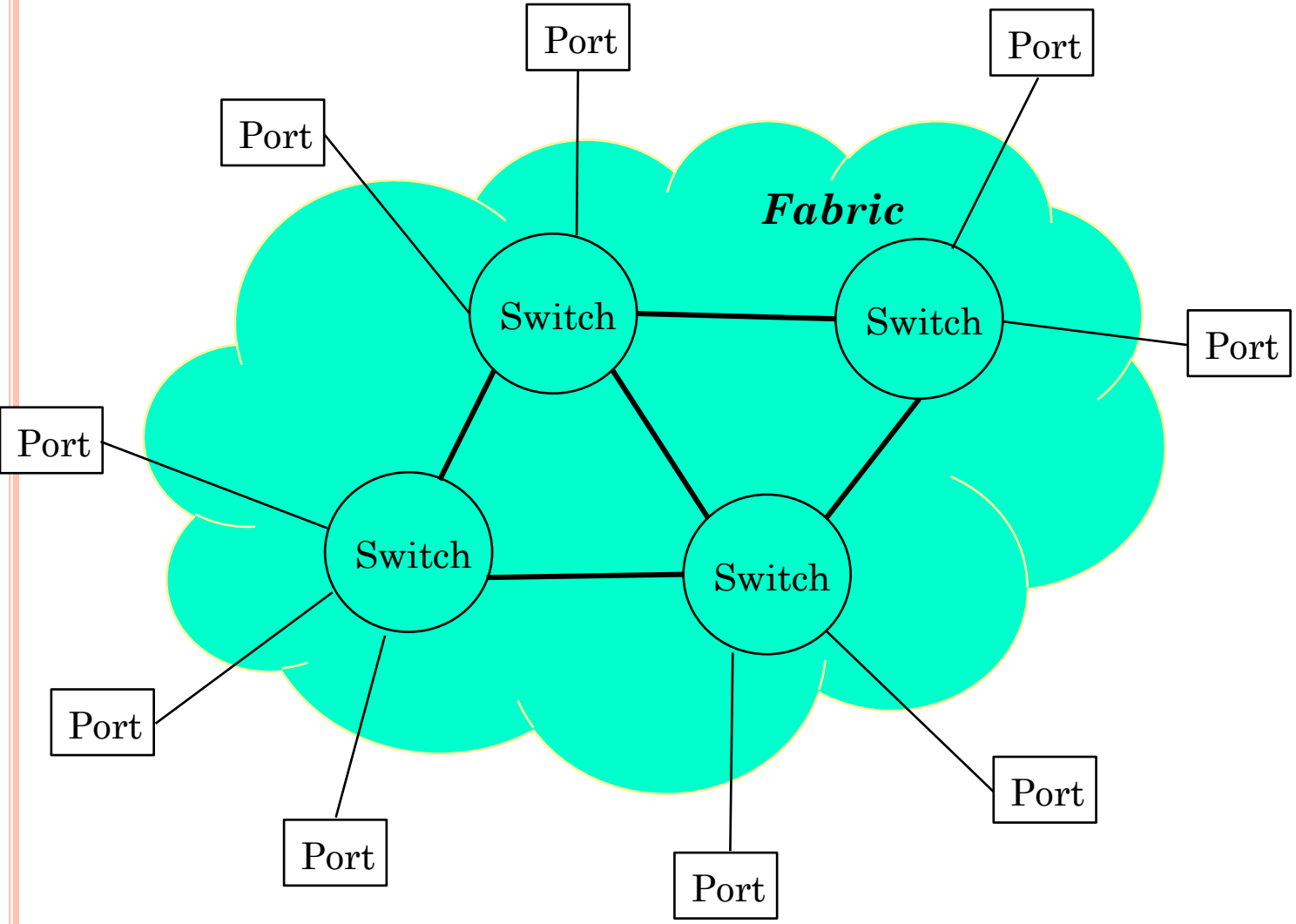
1. *Point-to-Point*
2. *Fabric* (most flexible)
3. *Arbitrated Loop*

A Node
(storage device, host bus adaptor [HBA])

A Port

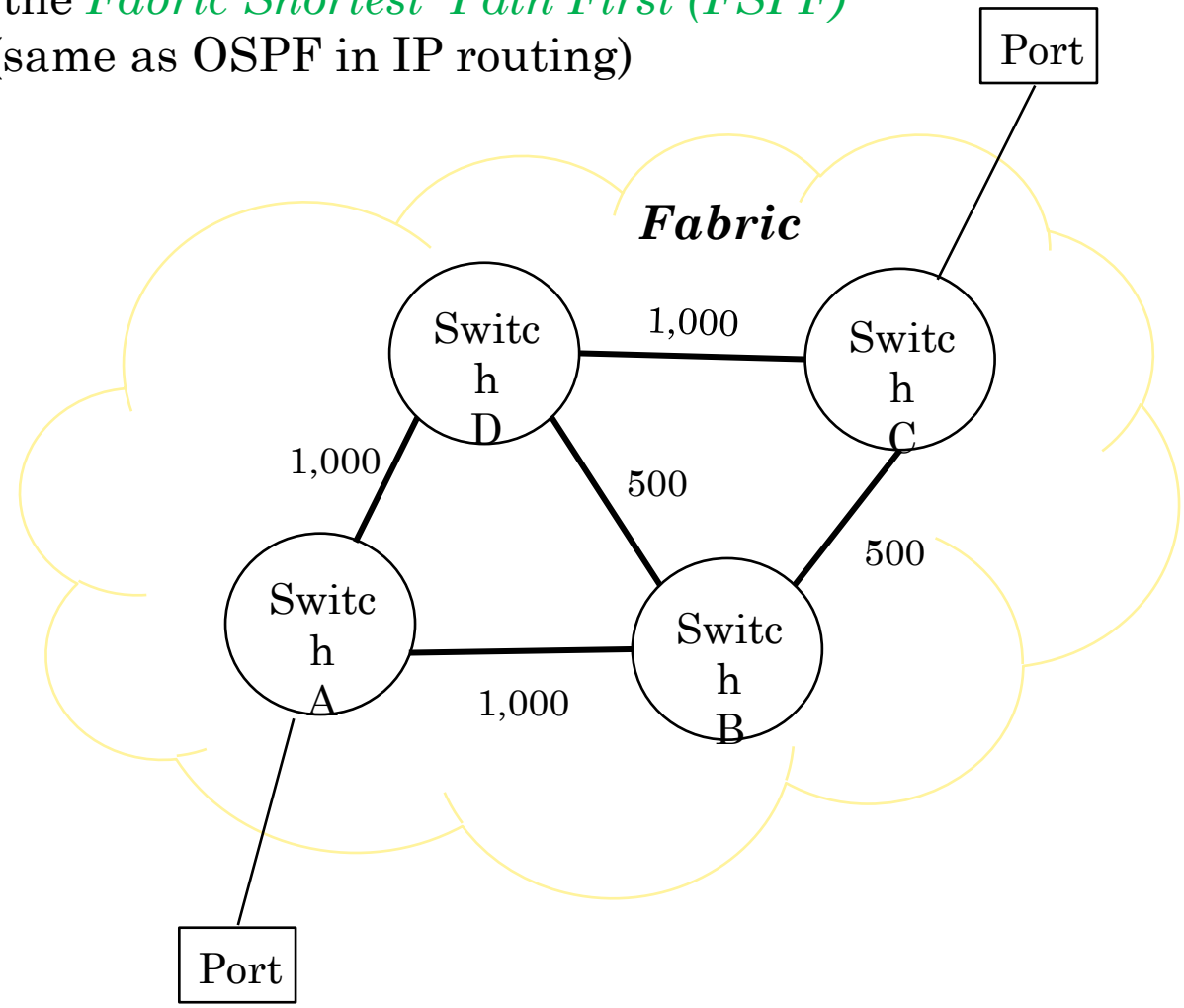
- 24-bit **address space** (for routing) and
- 64-bit **name space**

An Example of Fabric Topology

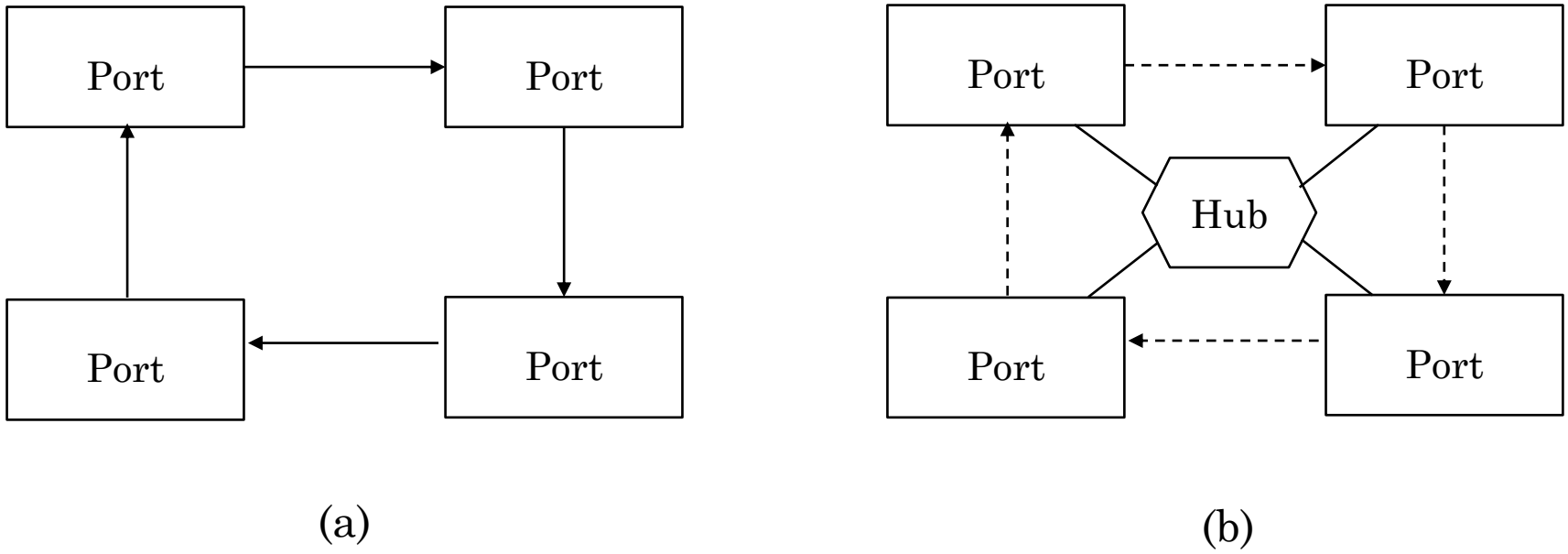


A Weighted-Path *Fabric* Network

- Implements the Layer 3 OSI Routing Function
- Employs the *Fabric Shortest Path First (FSPF)* protocol (same as OSPF in IP routing)



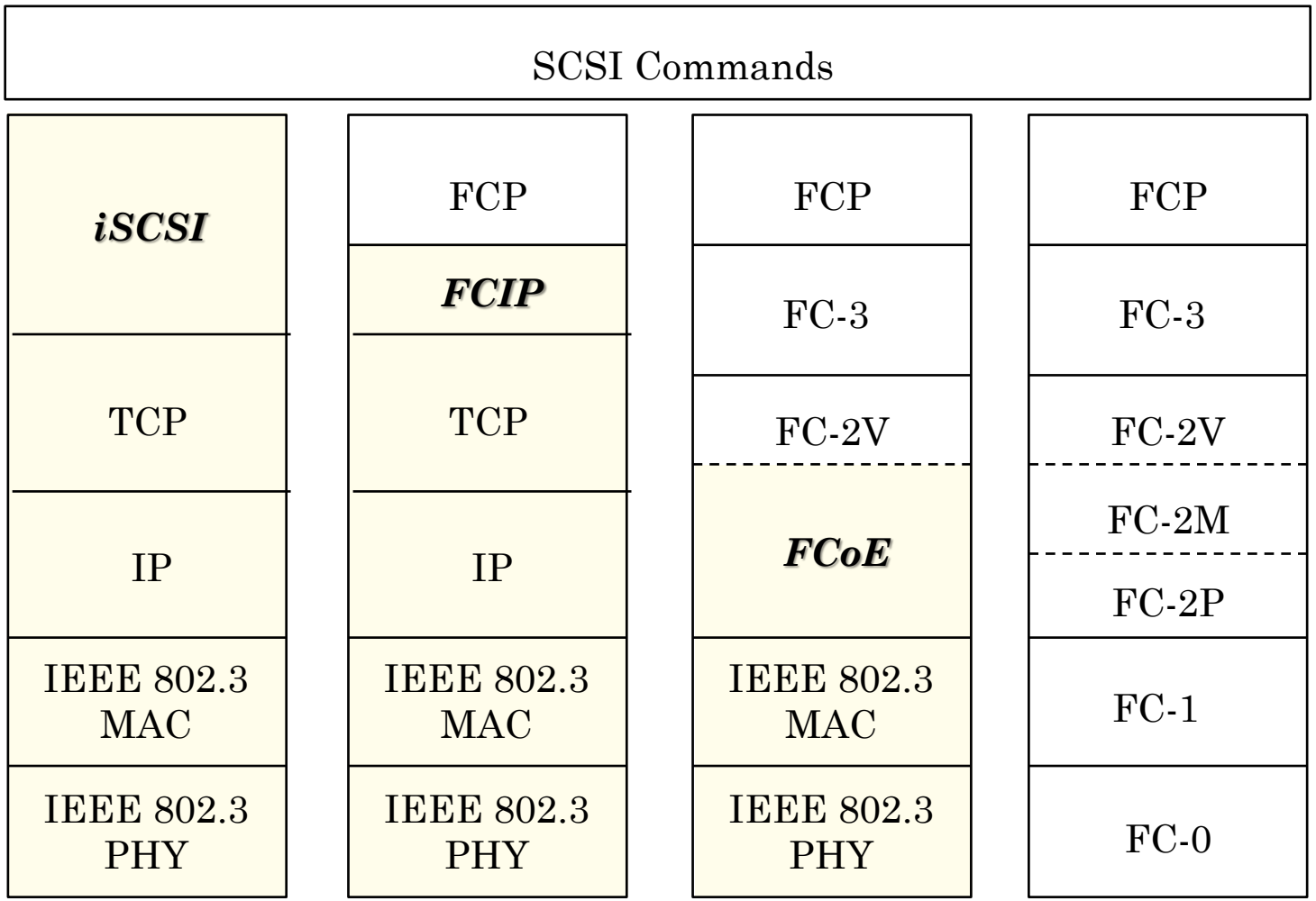
An example of arbitrated loop topology



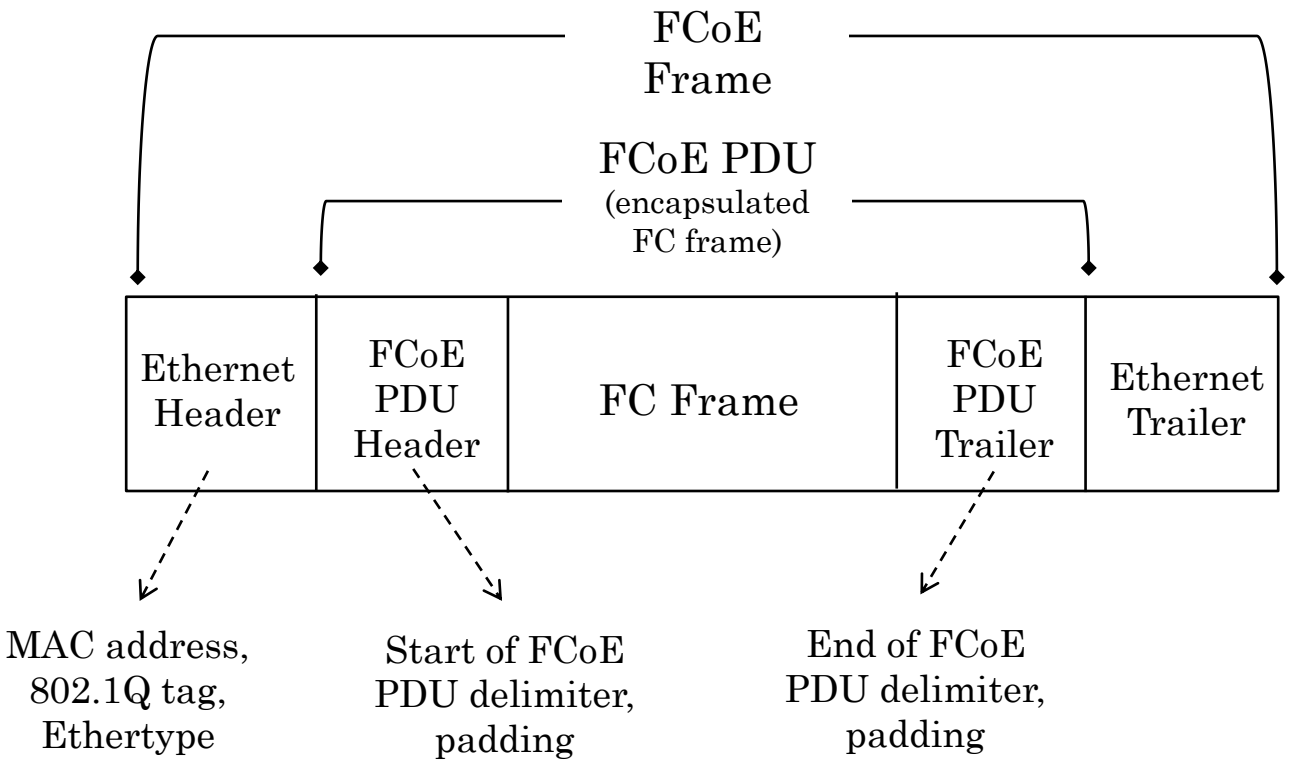
Convergence of SAN and Ethernet

- FC SAN has performed very well, but maintaining a bespoke network is expensive
- Reducing capital- and operational expenses (CAPex and OPex) is the major motivation for "convergence"—that is moving away from SAN to the IP-over-the-Ethernet networking
- SAN's layered structure helps to achieve that!

Example converged storage protocol stacks

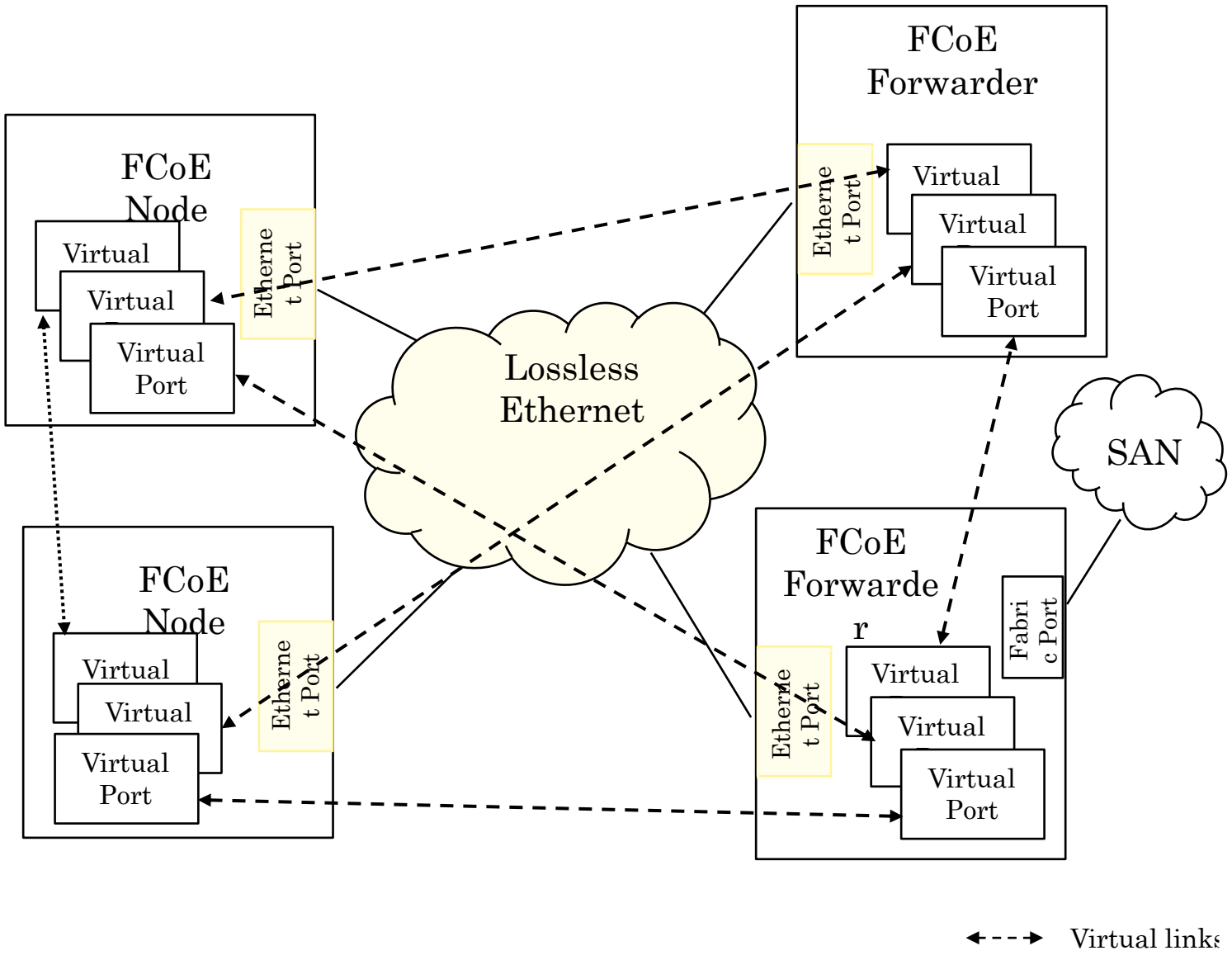


The FCoE frame structure

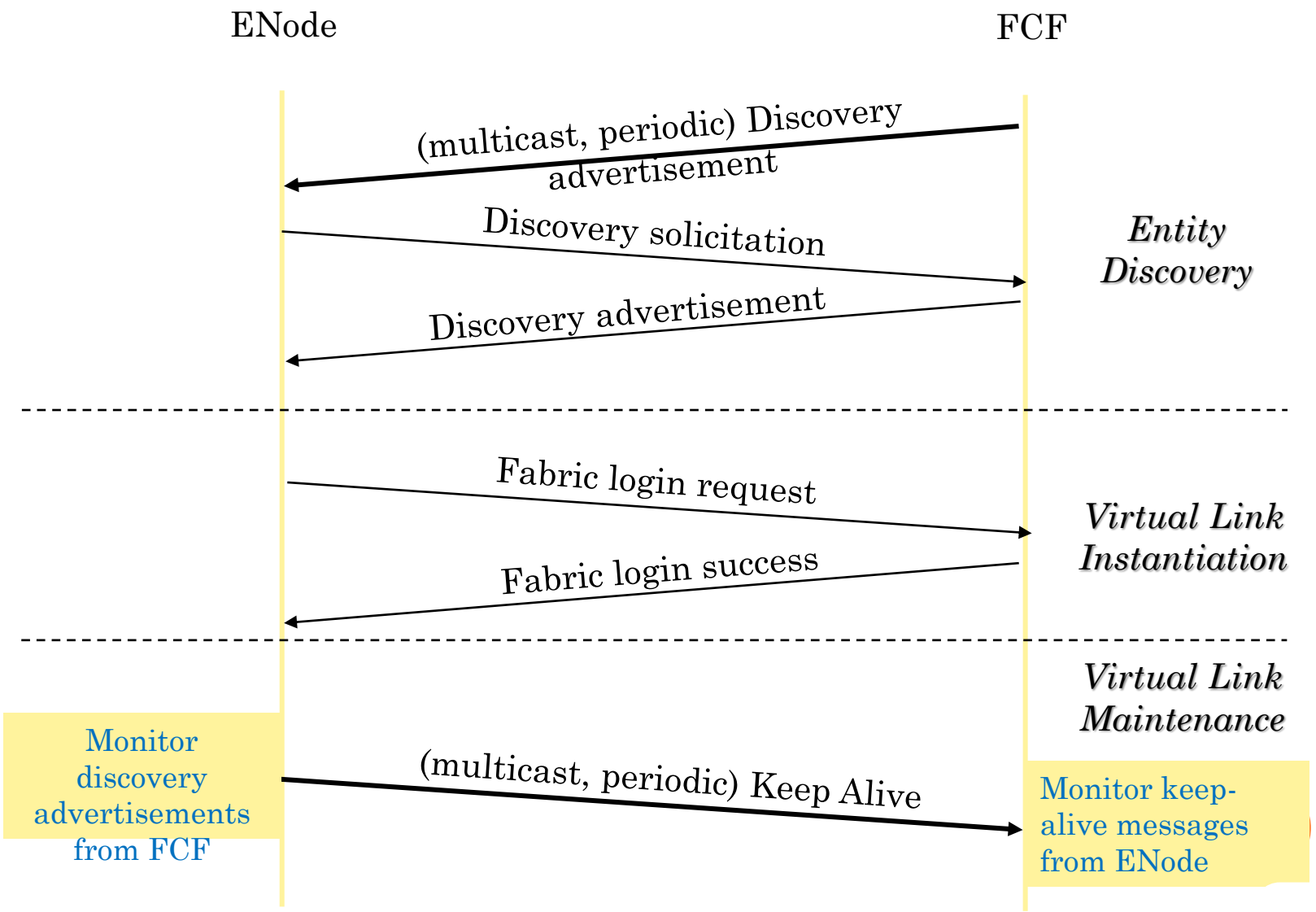


PDU: Protocol Data Unit

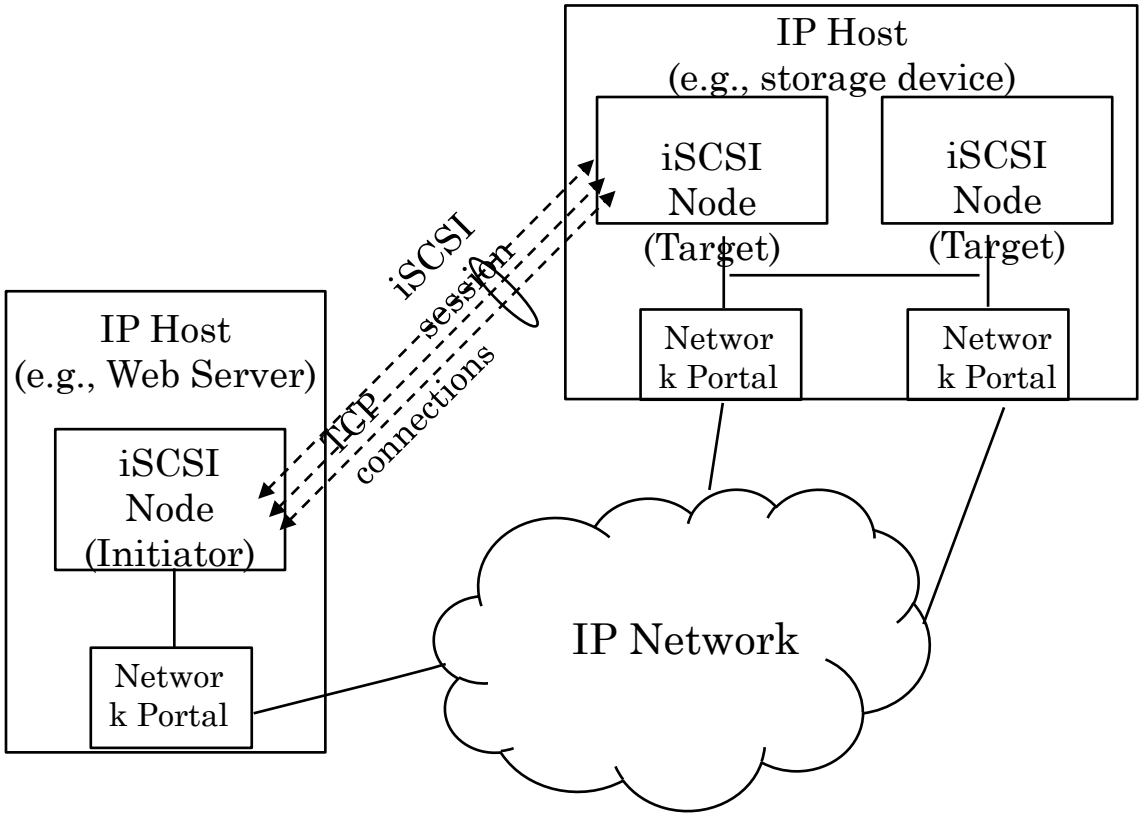
Conceptual FCoE architecture



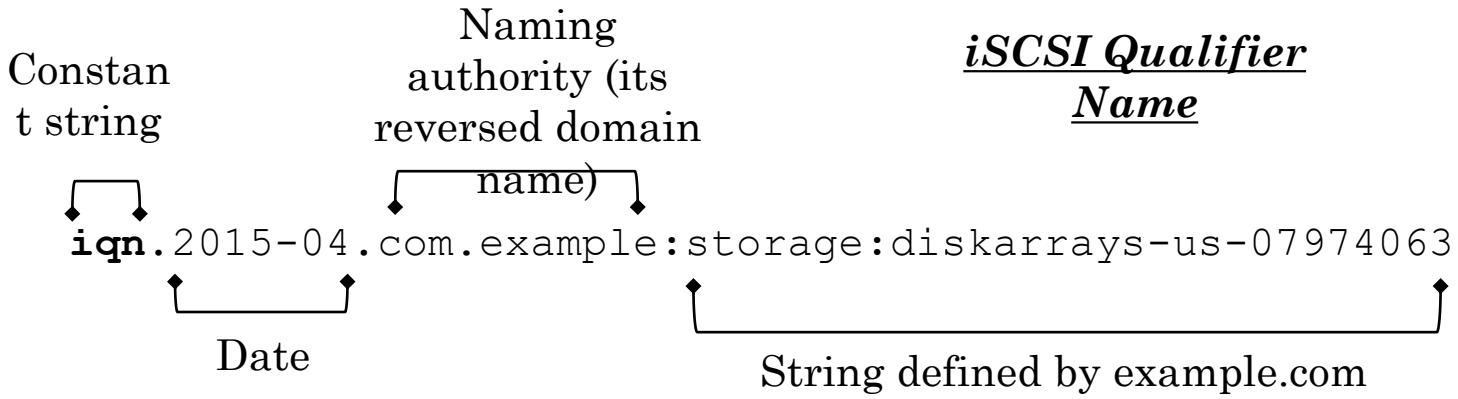
High-Level FIP Operations

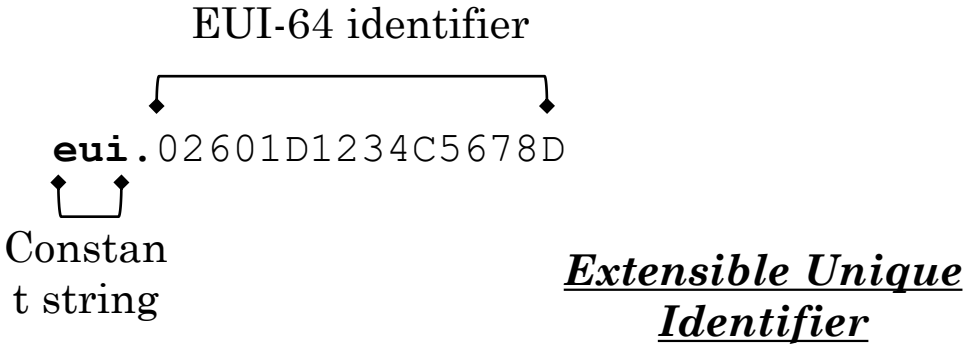


iSCSI conceptual model

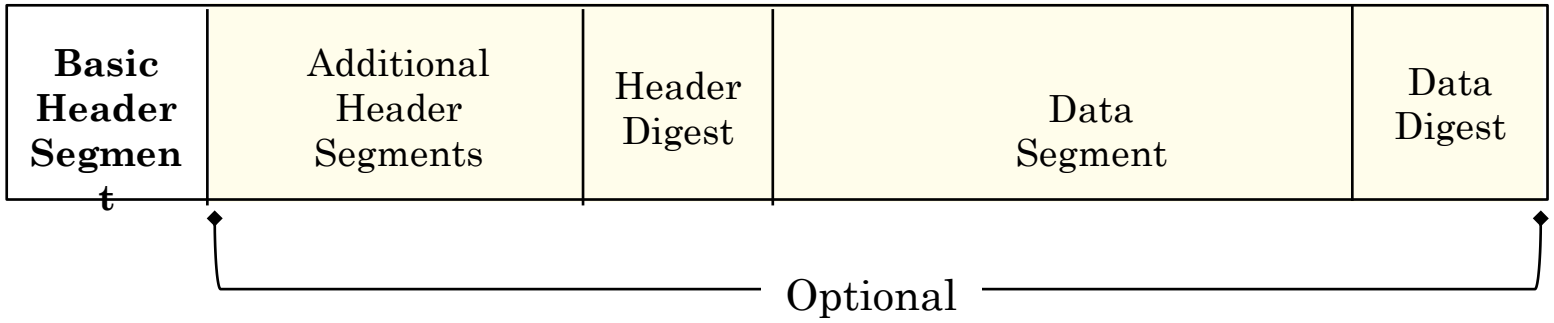


iSCSI Naming



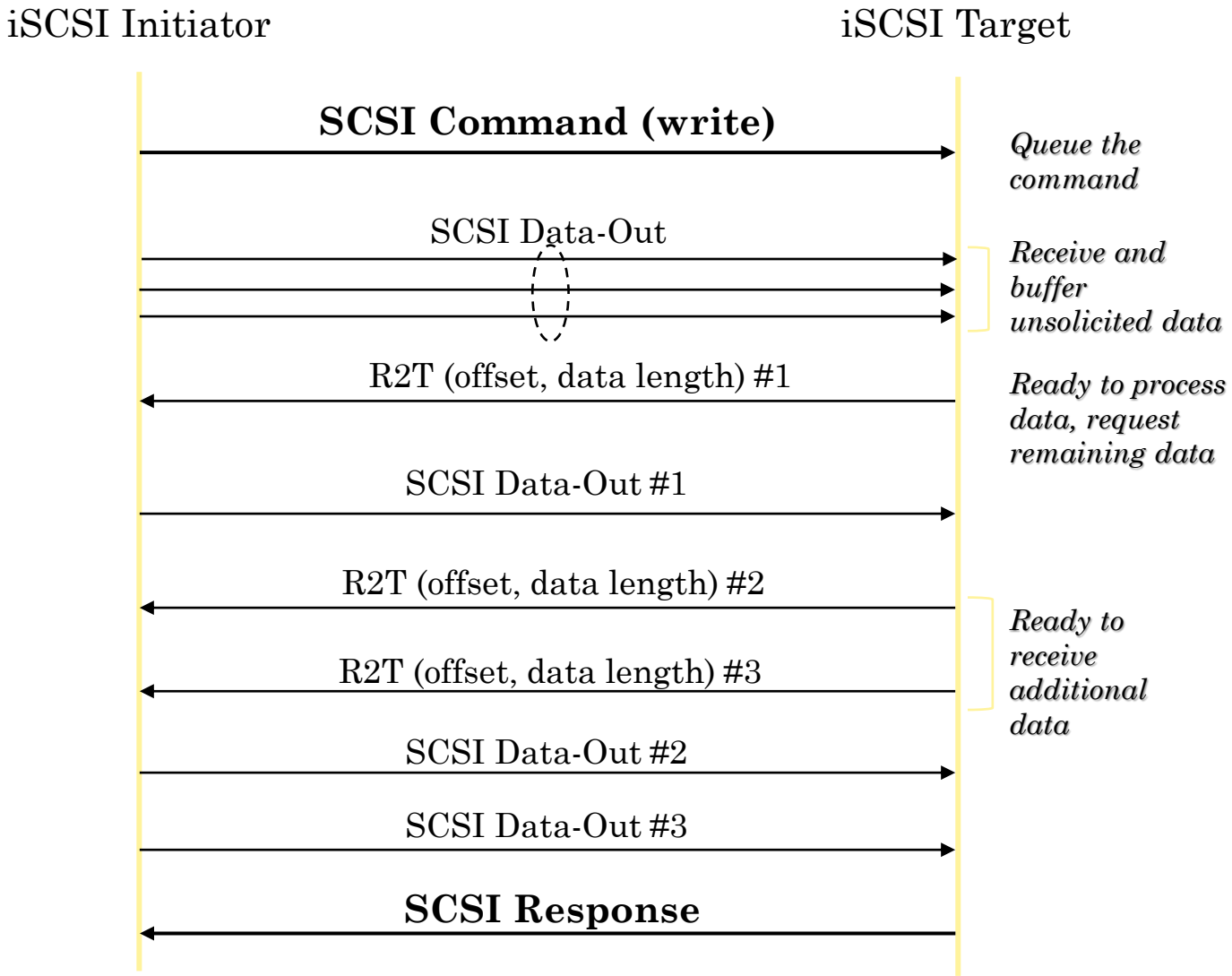


The format of an iSCSI Protocol Data Unit

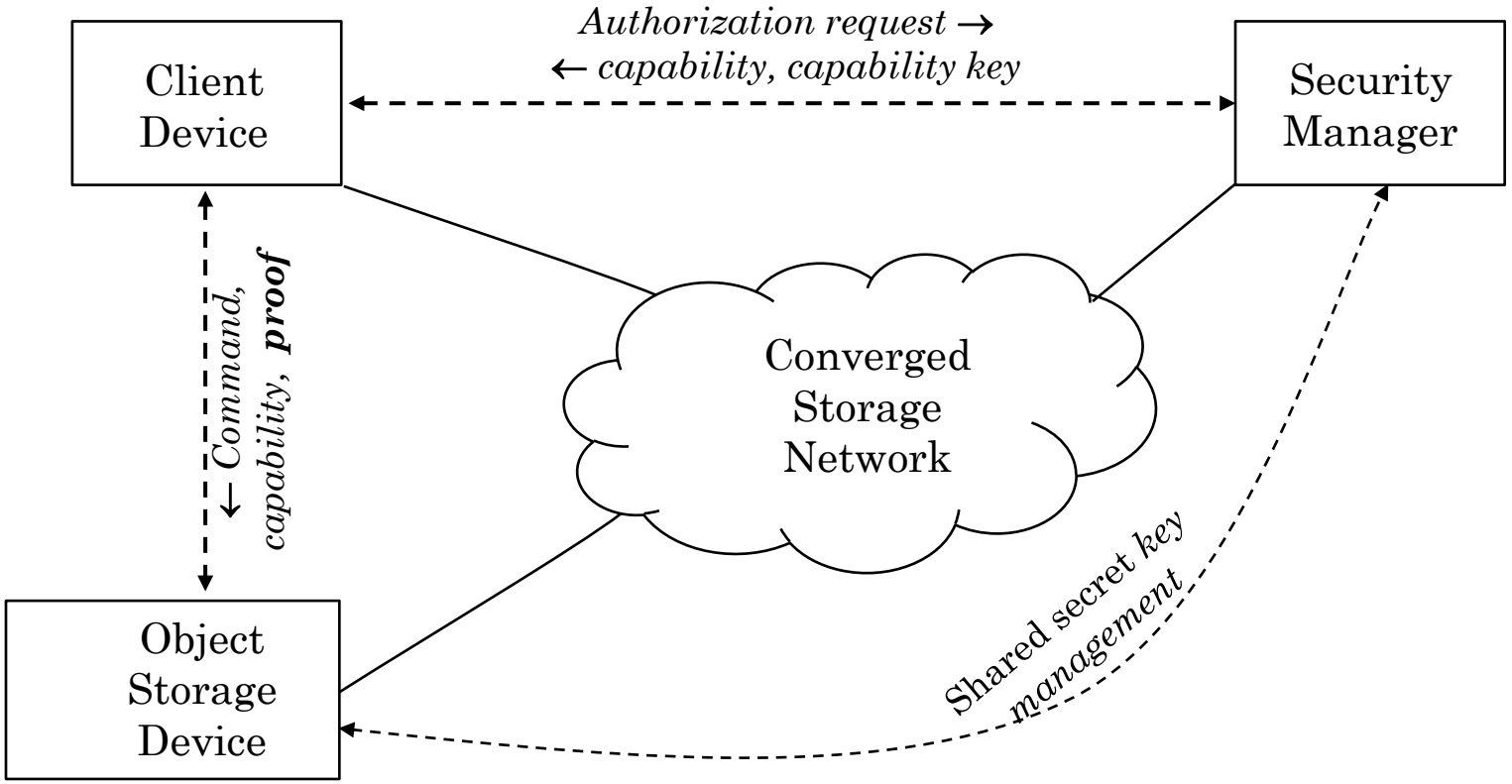


PDU: Protocol Data Unit

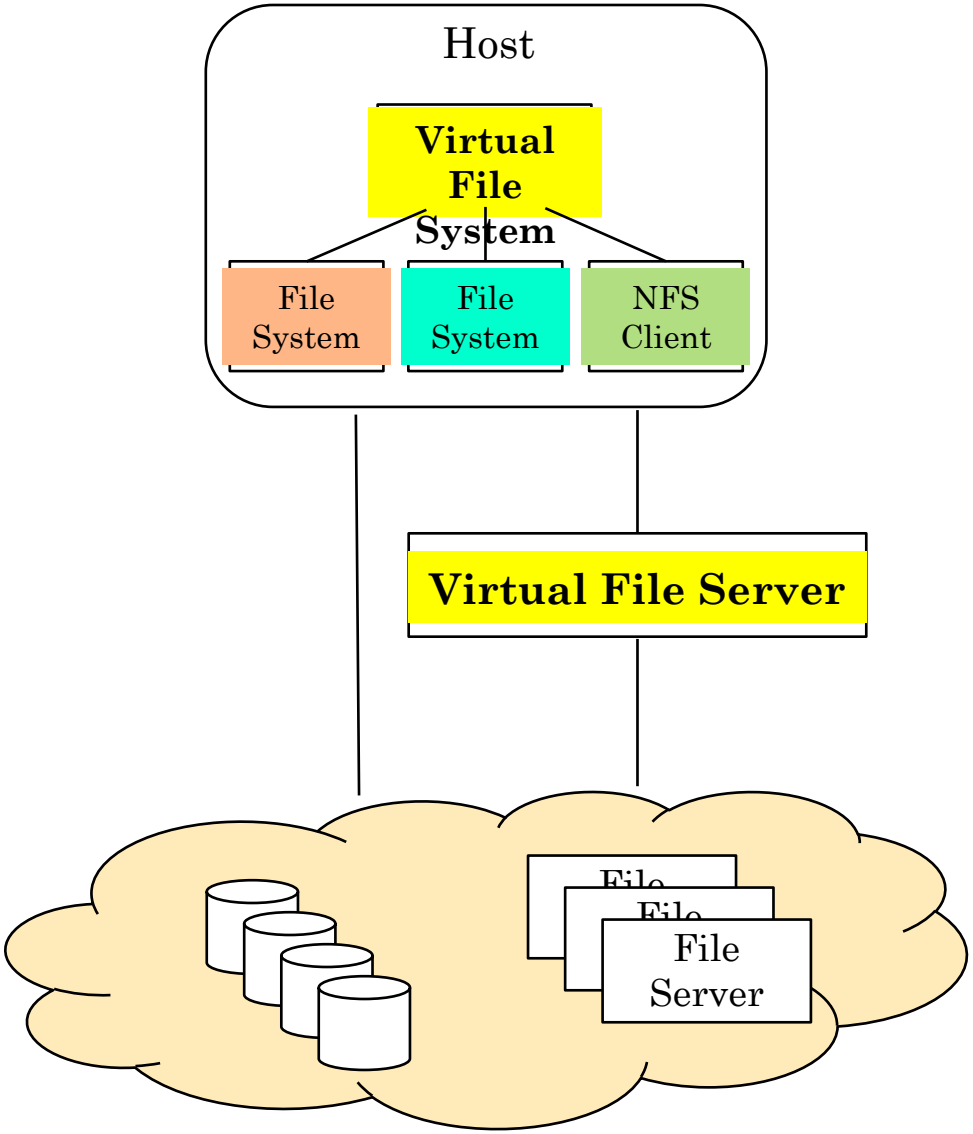
The information flow for the *write* operation



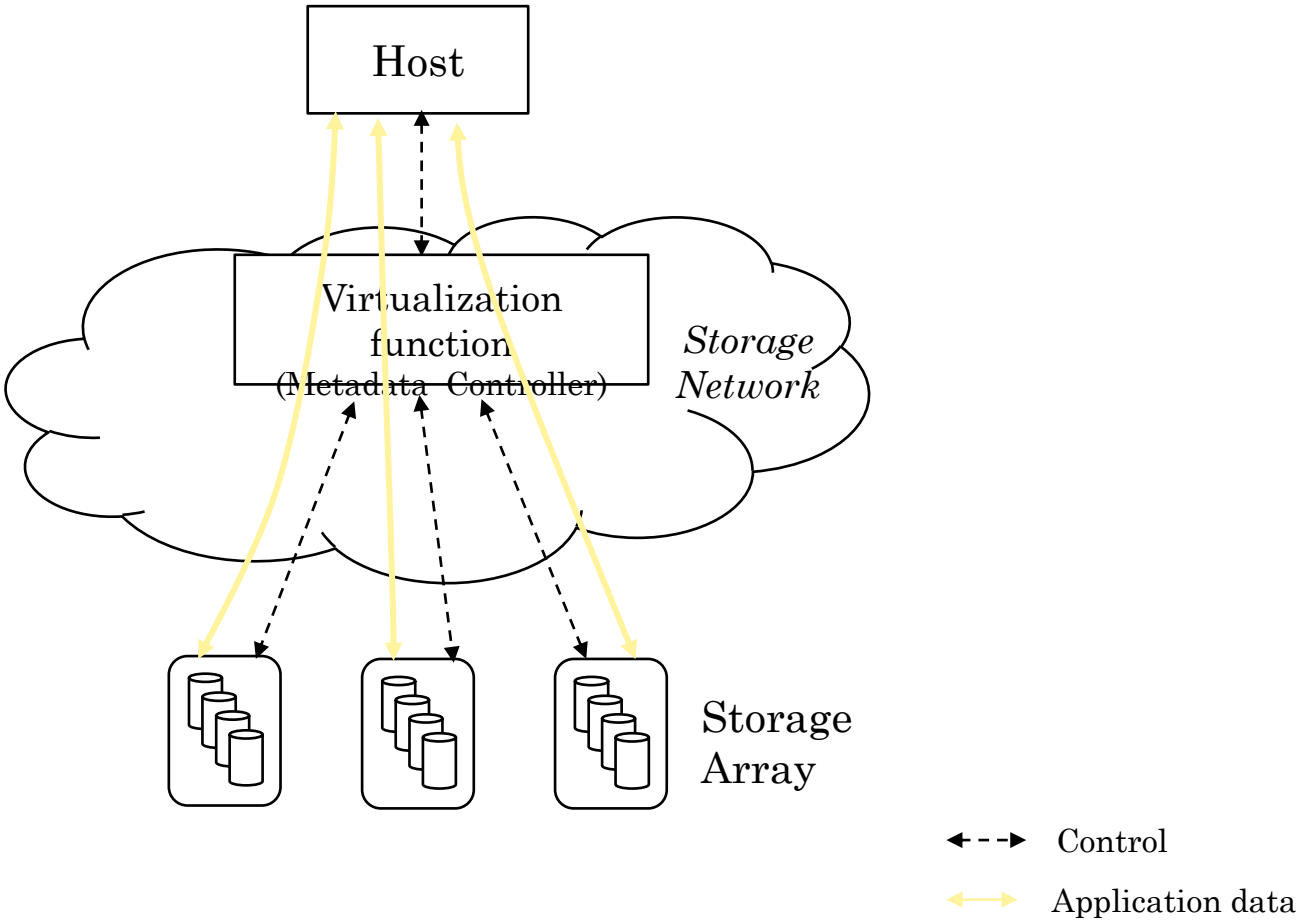
The object storage access control model



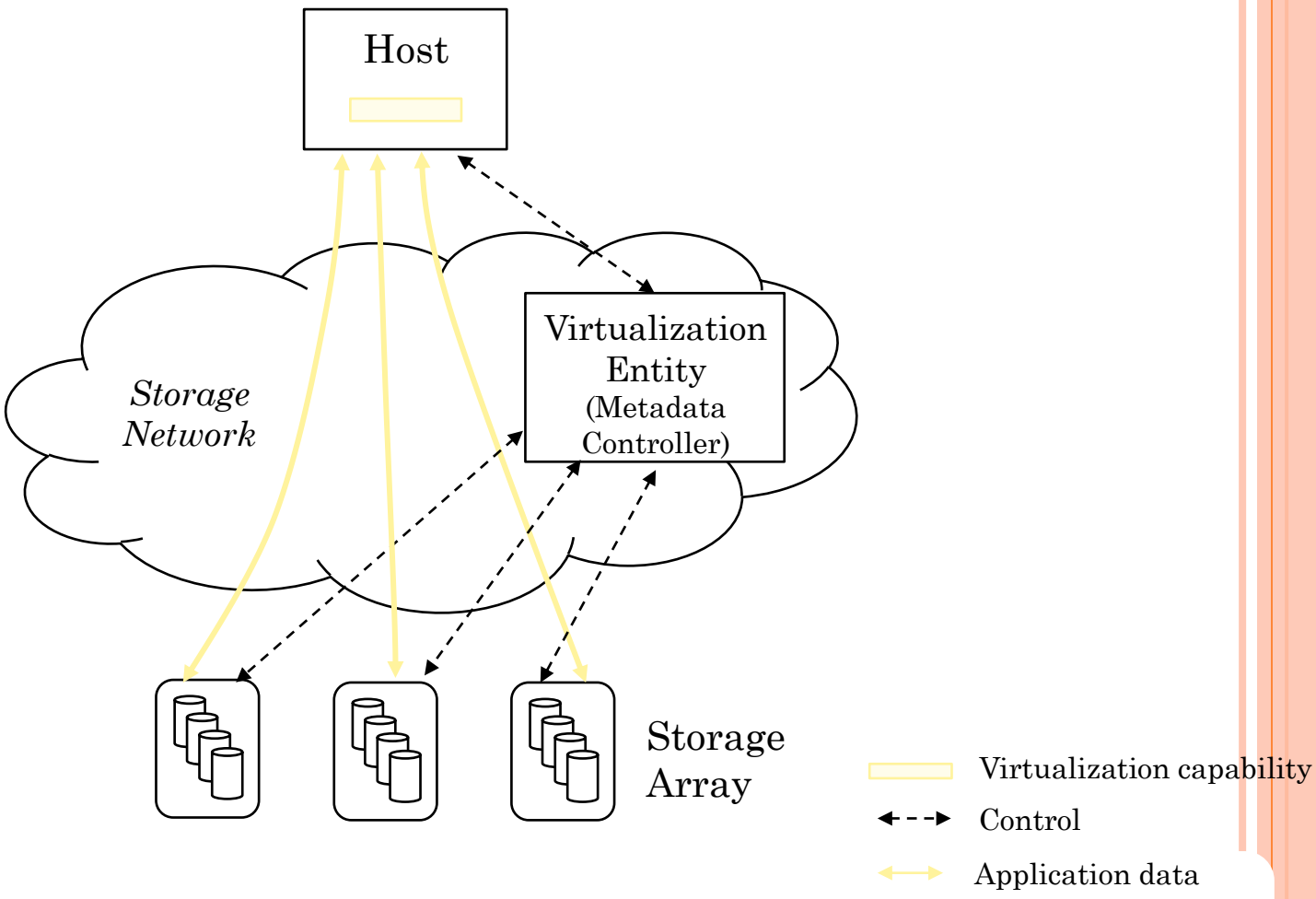
File-level storage virtualization



In-band storage virtualization



Out-of-band storage virtualization

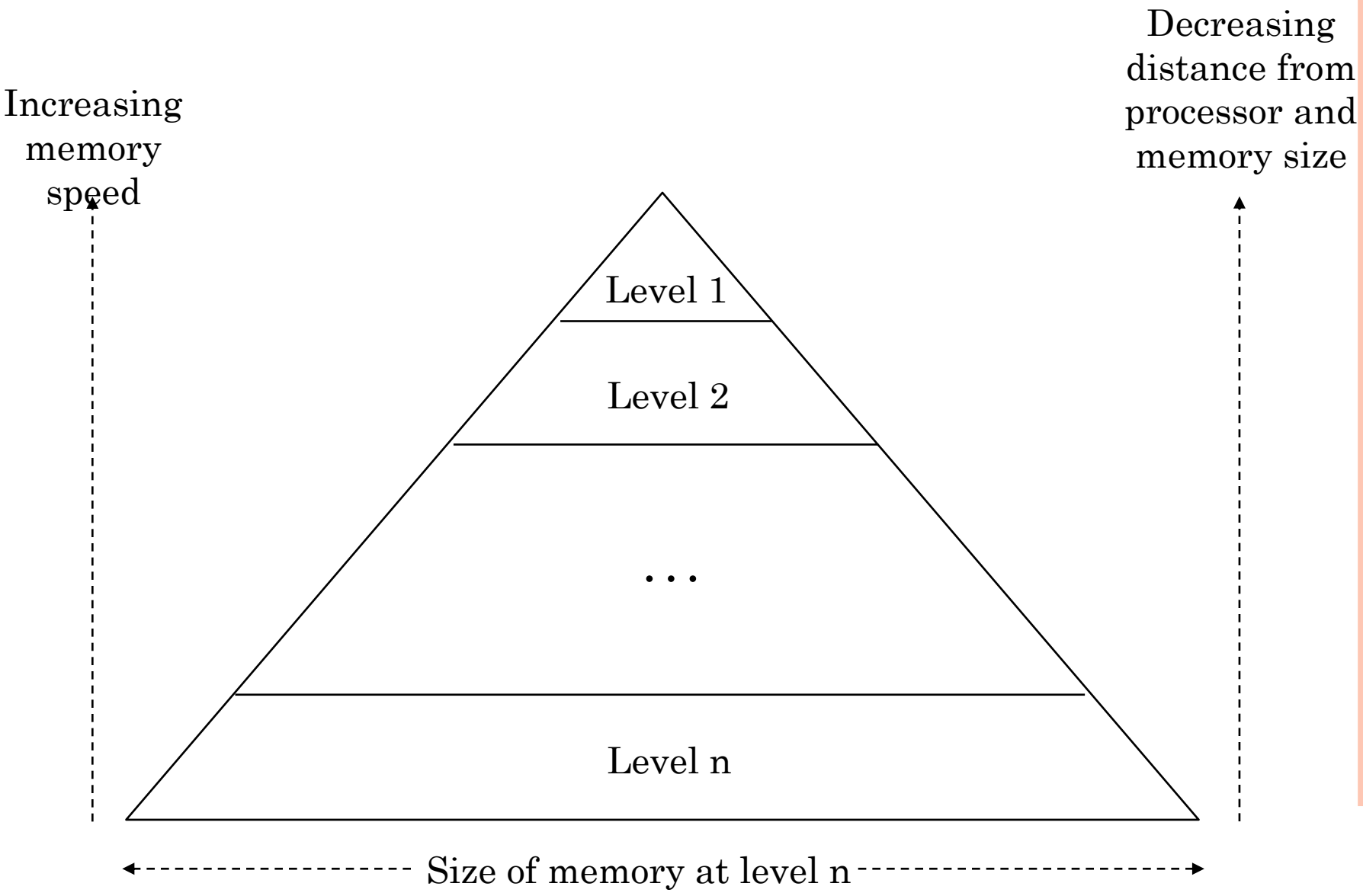


Performance comparison of storage technologies

	Relative Access Time	Relative Cost	Retention Time
SRAM	$\sim 10^{-7}$	$\sim 10^{4}$	the duration when it is powered
DRAM	$\sim 10^{-5}$	$\sim 10^{2}$	<< 1 second
Flash	$\sim 10^{-2}$	~10	years
Magnetic Disk	1	1	years
Magnetic Tape	$\sim 10^{-4}$	~0.5	years

SRAM: Static Random Access Memory
DRAM: Dynamic Random Access Memory

General memory hierarchy



A state of NAND flash