ECE590-03
Enterprise Storage Architecture

Fall 2016

Network-Attached Storage (NAS)

Tyler Bletsch
Duke University

Adapted from the course "Information Storage and Management v2" (module 7, "Network-Attached Storage"), published by EMC corporation.

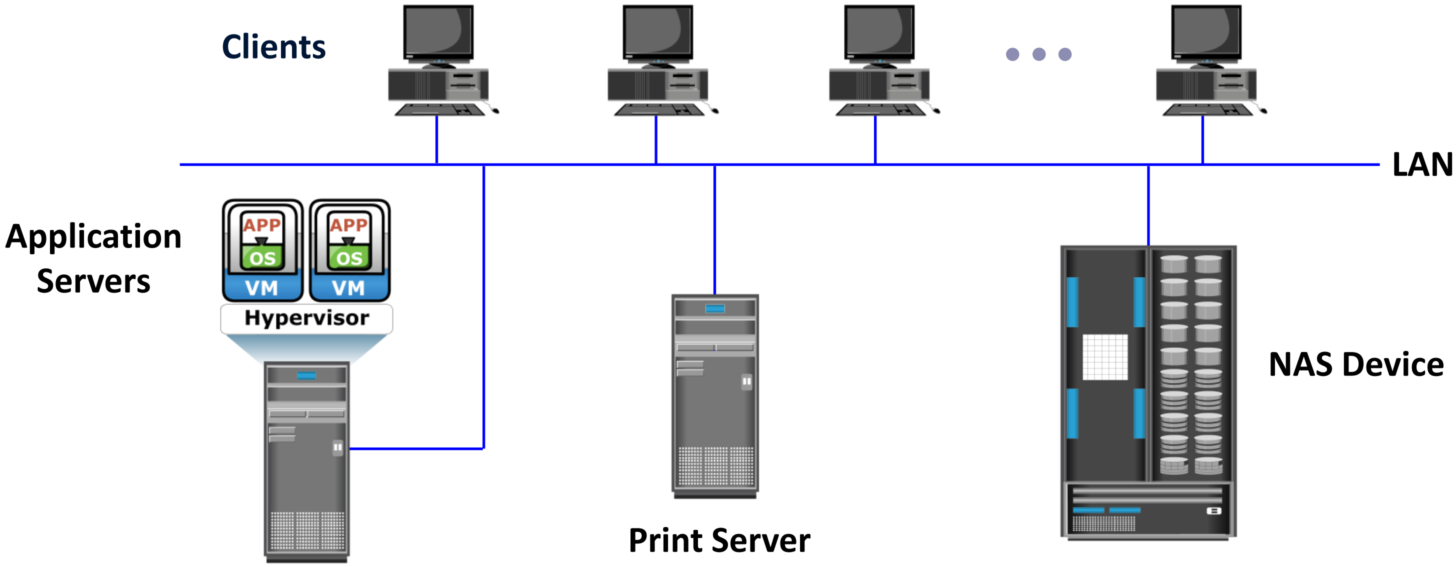
Includes additional content cited inline.

What is NAS?

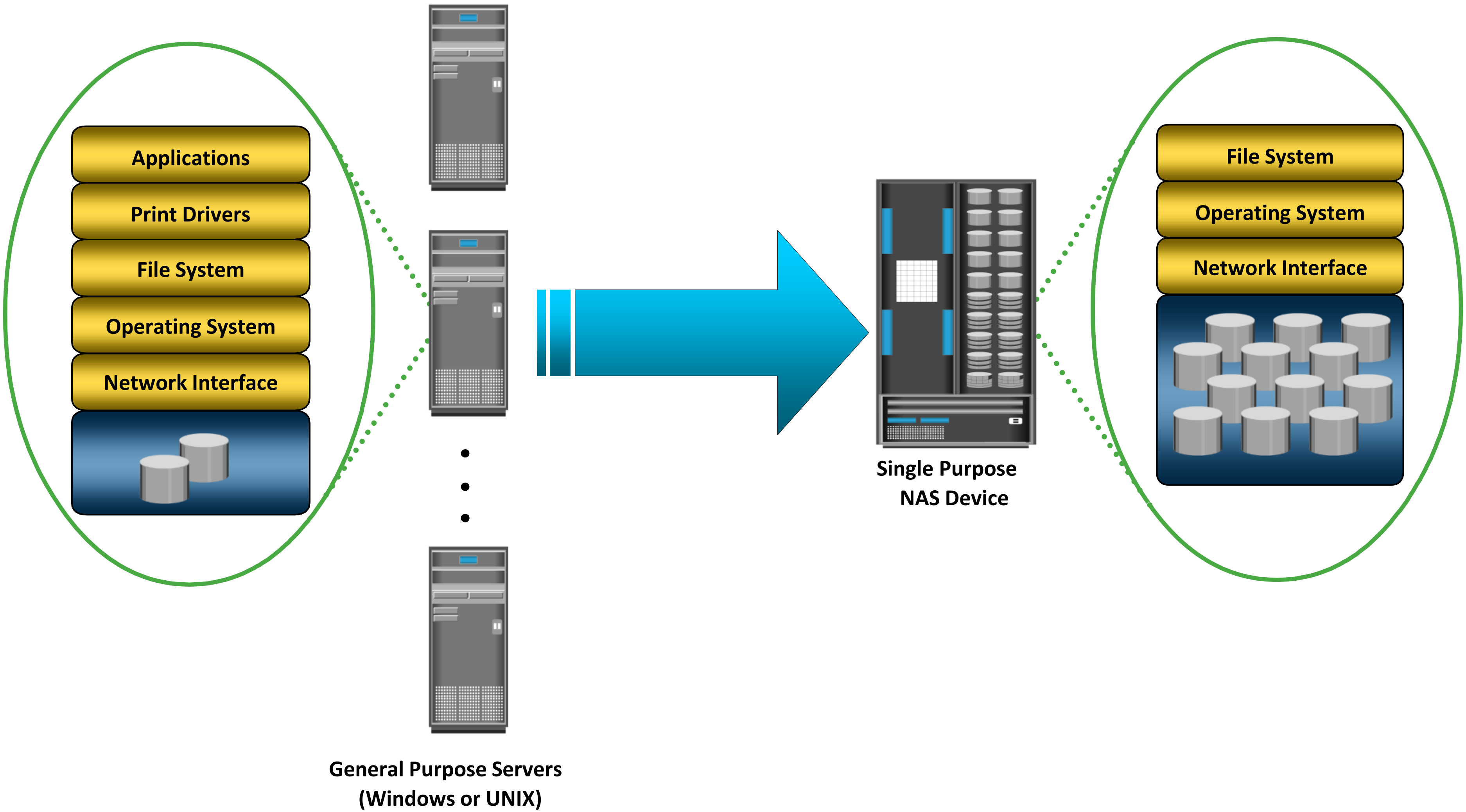
NAS

It is an IP-based, dedicated, high-performance file sharing and storage device.

- Enables NAS clients to share files over IP network
- Uses specialized operating system that is optimized for file I/O
- Enables both UNIX and Windows users to share data



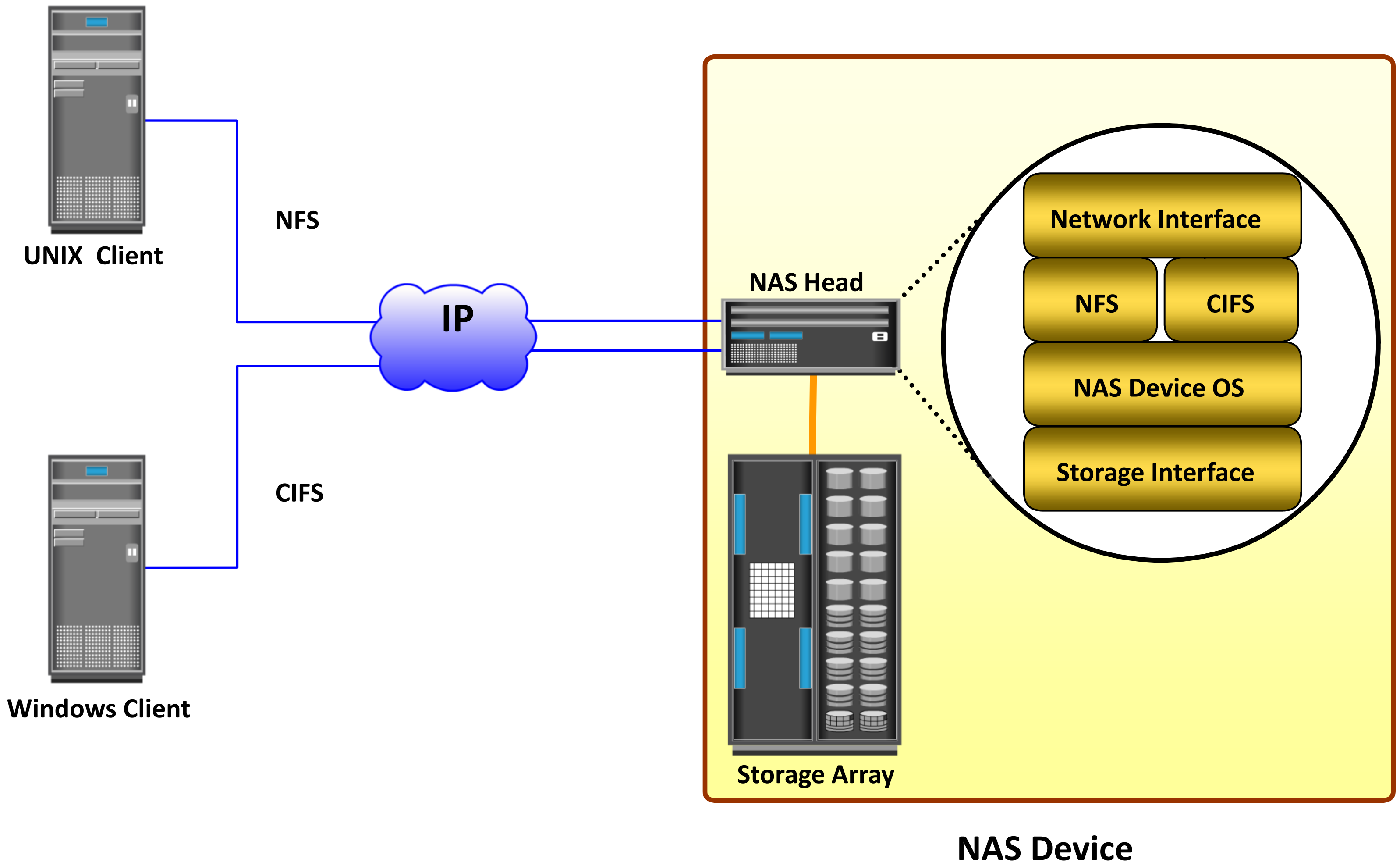
General Purpose Servers Vs. NAS Devices



Benefits of NAS

- Improved efficiency
- Improved flexibility
- Centralized storage
- Simplified management
- Scalability
- High availability – through native clustering and replication
- Security – authentication, authorization, and file locking in conjunction with industry-standard security
- Low cost
- Ease of deployment

Components of NAS



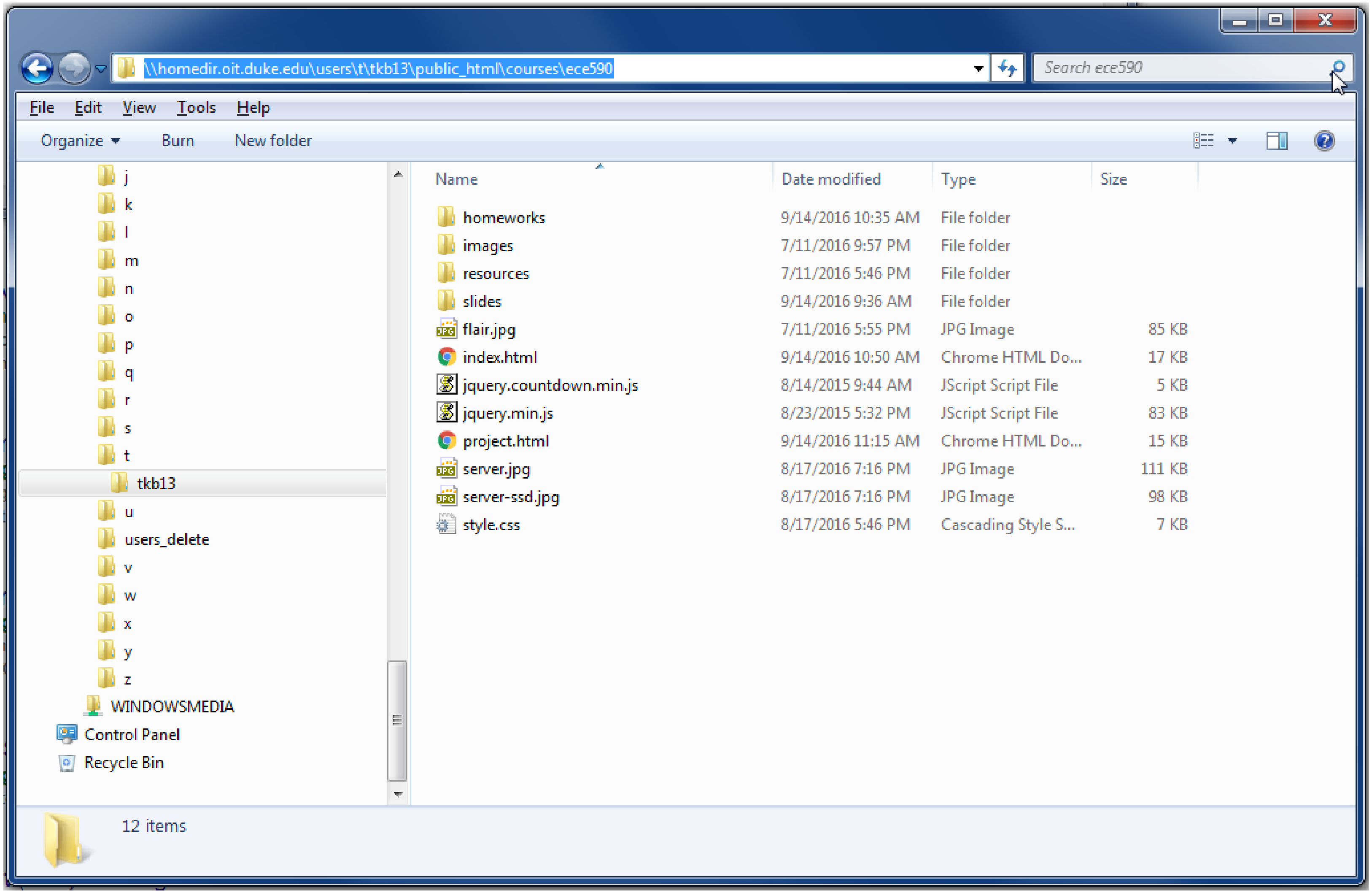
NAS File Sharing Protocols

- Two common NAS file sharing protocols are:
  - Common Internet File System (CIFS)
  - Network File System (NFS)

Common Internet File System

- Client-server application protocol
  - An open variation of the Server Message Block (SMB) protocol
- Enables clients to access files that are on a server over TCP/IP
- Stateful Protocol
  - Maintains connection information regarding every connected client
  - Can automatically restore connections and reopen files that were open prior to interruption

CIFS client example



Network File System

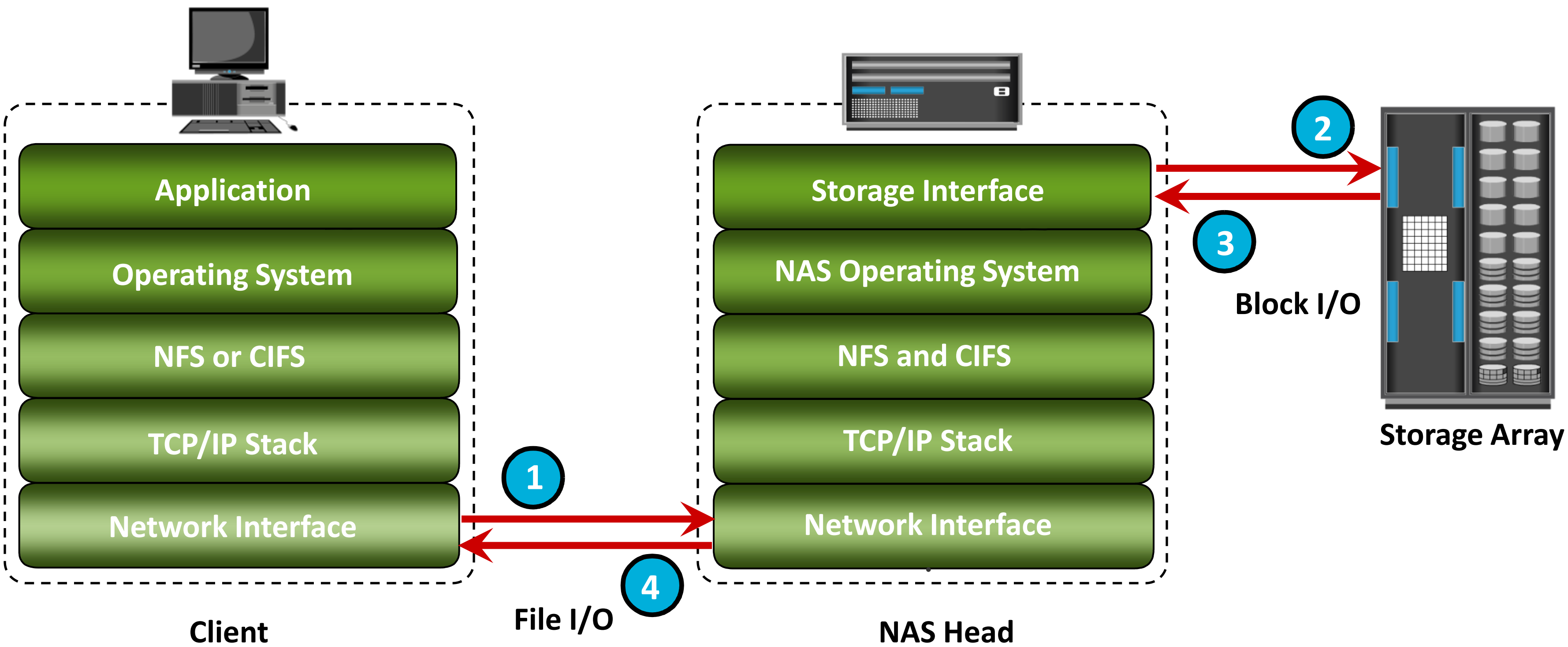
- Client-server application protocol
- Enables clients to access files that are on a server
- Uses Remote Procedure Call (RPC) mechanism to provide access to remote file system
- Currently, three versions of NFS are in use:
  - NFS v2 is stateless and uses UDP as transport layer protocol
  - NFS v3 is stateless and uses UDP or optionally TCP as transport layer protocol
  - NFS v4 is stateful and uses TCP as transport layer protocol

NFS mount example

tkb13@login-teer-02:~

```
tkb13@login-teer-02:~ $ mount
/dev/mapper/VolGroup00-LogVol00 on / type ext4 (rw)
proc on /proc type proc (rw)
sysfs on /sys type sysfs (rw)
devpts on /dev/pts type devpts (rw,gid=5,mode=620)
tmpfs on /dev/shm type tmpfs (rw)
/dev/sda1 on /boot type ext3 (rw)
none on /proc/sys/fs/binfmt_misc type binfmt_misc (rw)
sunrpc on /var/lib/nfs/rpc_pipefs type rpc_pipefs (rw)
oit-nas-fe03.oit.duke.edu:/nfshomedir/j on /nfshomes/j type nfs4 (rw,nodev,sec=krb5,vers=4,sloppy,addr=10.138.0.15,clientaddr=10.138.19.4)
oit-nas-fe03.oit.duke.edu:/nfshomedir/t on /nfshomes/t type nfs4 (rw,nodev,sec=krb5,vers=4,sloppy,addr=10.138.0.15,clientaddr=10.138.19.4)
//homedir.win.duke.edu/users/a/at200 on /winhomes/at200 type cifs (rw)
//homedir.win.duke.edu/users/a/apr17 on /winhomes/apr17 type cifs (rw)
//homedir.win.duke.edu/users/t/tn74 on /winhomes/tn74 type cifs (rw)
//homedir.win.duke.edu/users/m/mrg on /winhomes/mrg type cifs (rw)
//homedir.win.duke.edu/users/x/xc77 on /winhomes/xc77 type cifs (rw)
//homedir.win.duke.edu/users/m/mpd13 on /winhomes/mpd13 type cifs (rw)
//homedir.win.duke.edu/users/g/gsh15 on /winhomes/gsh15 type cifs (rw)
//homedir.win.duke.edu/users/i/ik51 on /winhomes/ik51 type cifs (rw)
//homedir.win.duke.edu/users/a/ap375 on /winhomes/ap375 type cifs (rw)
//homedir.win.duke.edu/users/c/cx17 on /winhomes/cx17 type cifs (rw)
//homedir.win.duke.edu/users/c/cf164 on /winhomes/cf164 type cifs (rw)
//homedir.win.duke.edu/users/p/pas44 on /winhomes/pas44 type cifs (rw)
//homedir.win.duke.edu/users/m/mtz3 on /winhomes/mtz3 type cifs (rw)
//homedir.win.duke.edu/users/p/pg96 on /winhomes/pg96 type cifs (rw)
//homedir.win.duke.edu/users/a/agp11 on /winhomes/agp11 type cifs (rw)
//homedir.win.duke.edu/users/l/lz107 on /winhomes/lz107 type cifs (rw)
//homedir.win.duke.edu/users/m/mm479 on /winhomes/mm479 type cifs (rw)
//homedir.win.duke.edu/users/j/jv96 on /winhomes/jv96 type cifs (rw)
//homedir.win.duke.edu/users/s/ss810 on /winhomes/ss810 type cifs (rw)
//homedir.win.duke.edu/users/w/wl150 on /winhomes/wl150 type cifs (rw)
//homedir.win.duke.edu/users/c/cml78 on /winhomes/cml78 type cifs (rw)
//homedir.win.duke.edu/users/f/fm87 on /winhomes/fm87 type cifs (rw)
//homedir.win.duke.edu/users/z/znz2 on /winhomes/znz2 type cifs (rw)
//homedir.win.duke.edu/users/j/jhc37 on /winhomes/jhc37 type cifs (rw)
//homedir.win.duke.edu/users/a/aer51 on /winhomes/aer51 type cifs (rw)
//homedir.win.duke.edu/users/d/dcb37 on /winhomes/dcb37 type cifs (rw)
//homedir.win.duke.edu/users/b/bas65 on /winhomes/bas65 type cifs (rw)
//homedir.win.duke.edu/users/c/cmg77 on /winhomes/cmg77 type cifs (rw)
//homedir.win.duke.edu/users/m/msh54 on /winhomes/msh54 type cifs (rw)
//homedir.win.duke.edu/users/m/mz93 on /winhomes/mz93 type cifs (rw)
//homedir.win.duke.edu/users/j/jcw71 on /winhomes/jcw71 type cifs (rw)
//homedir.win.duke.edu/users/k/ket26 on /winhomes/ket26 type cifs (rw)
//homedir.win.duke.edu/users/r/rw103 on /winhomes/rw103 type cifs (rw)
//homedir.win.duke.edu/users/c/cwh31 on /winhomes/cwh31 type cifs (rw)
//homedir.win.duke.edu/users/m/mt265 on /winhomes/mt265 type cifs (rw)
//homedir.win.duke.edu/users/i/ike on /winhomes/ike type cifs (rw)
//homedir.win.duke.edu/users/t/tkb13 on /winhomes/tkb13 type cifs (rw)
tkb13@login-teer-02:~ $
```

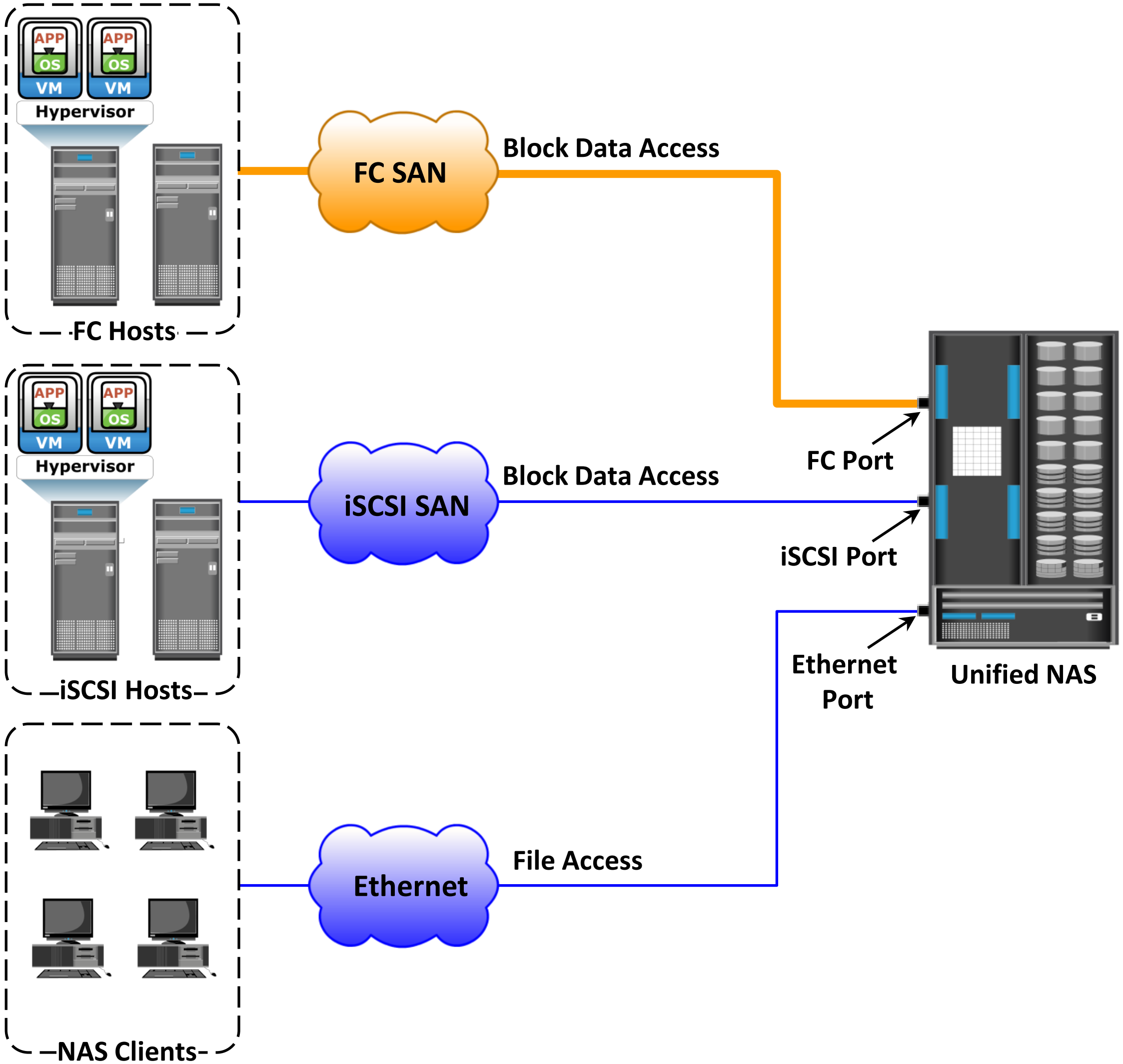
NAS I/O Operation



NAS Implementation – Unified NAS

- Consolidates NAS-based (file-level) and SAN-based (block-level) access on a single storage platform
- Supports both CIFS and NFS protocols for file access and iSCSI and FC protocols for block level access
- Provides unified management for both NAS head and storage

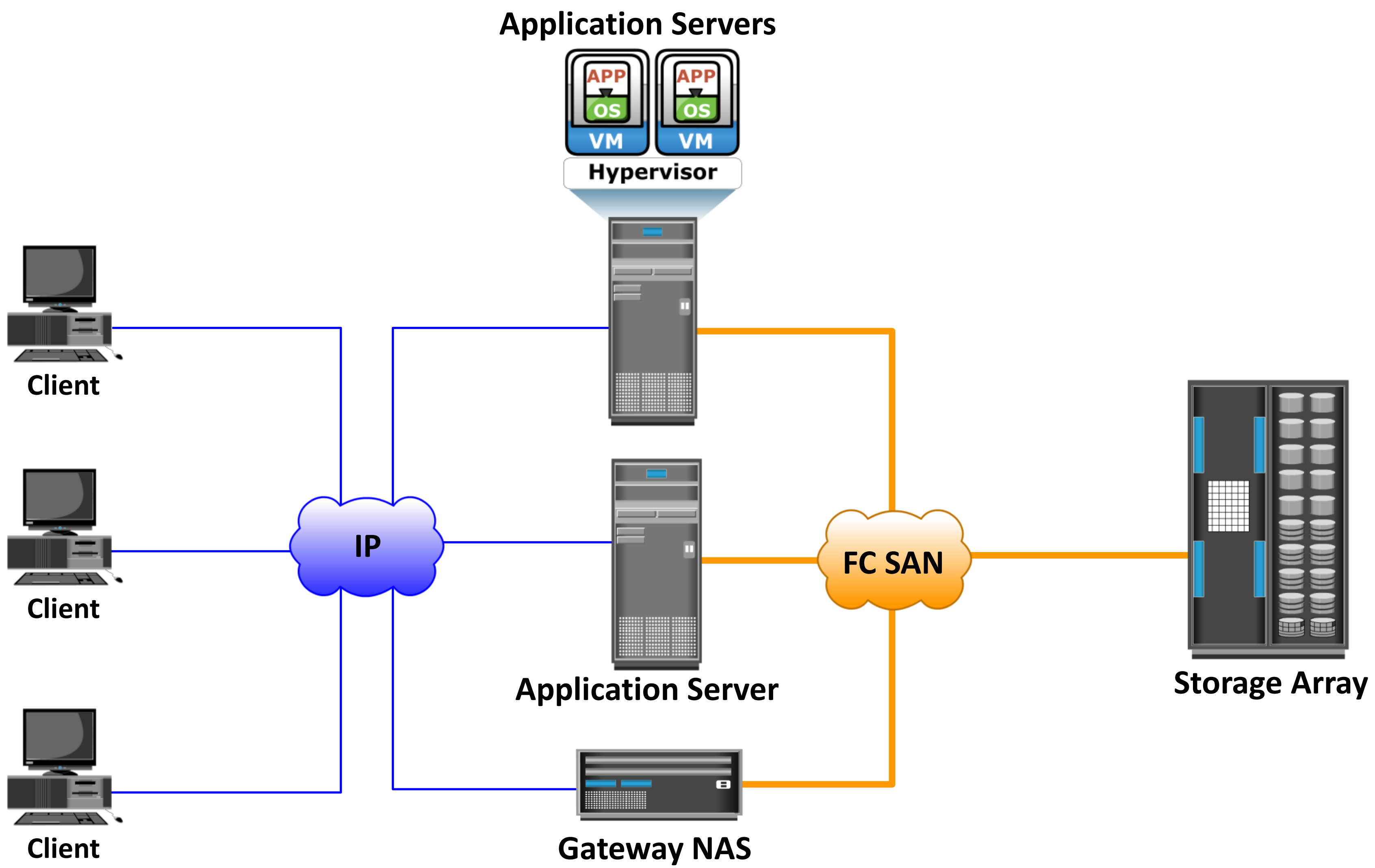
Unified NAS Connectivity



NAS Implementation – Gateway NAS

- Uses external and independently-managed storage
  - NAS heads access SAN-attached or direct-attached storage arrays
- NAS heads share storage with other application servers that perform block I/O
- Requires separate management of NAS head and storage

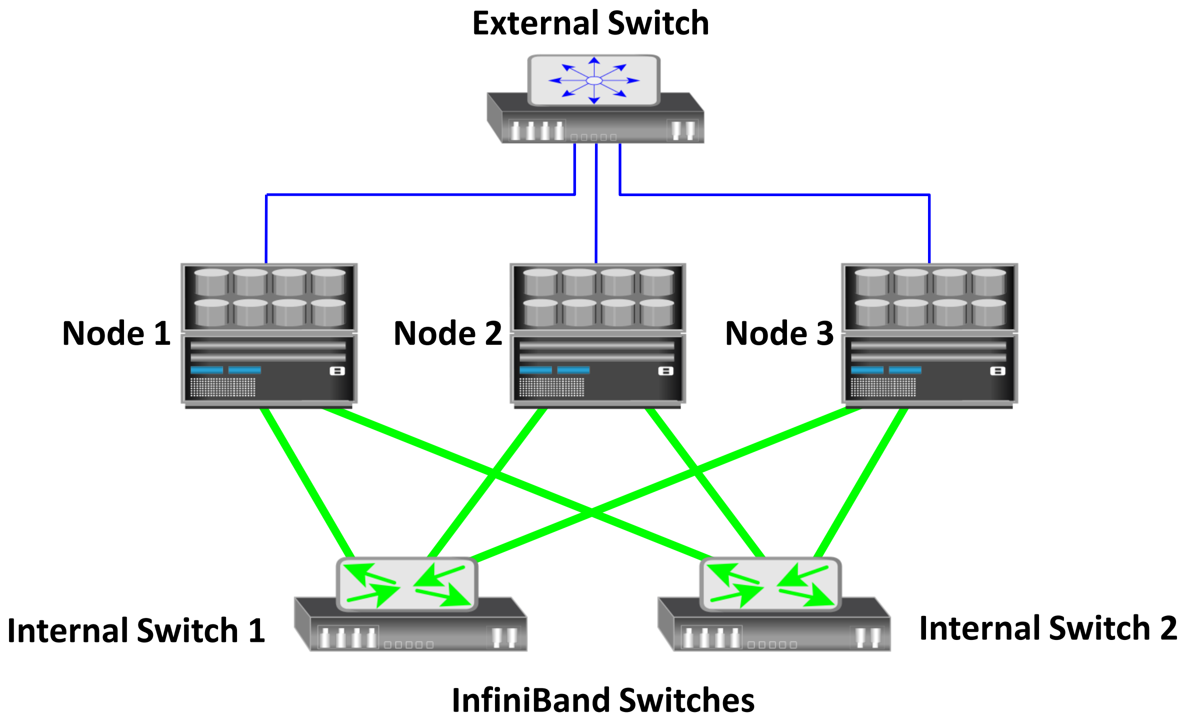
Gateway NAS Connectivity



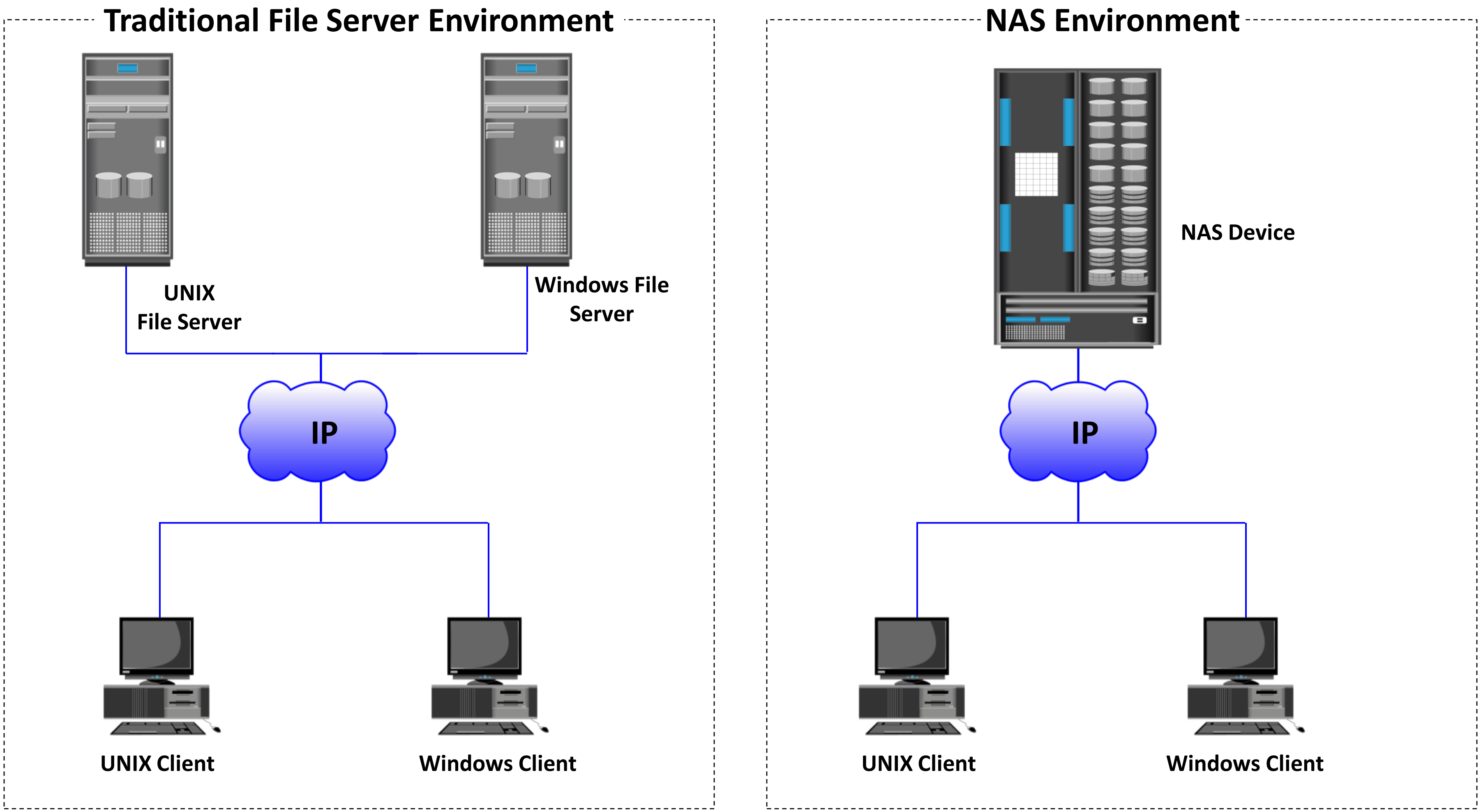
NAS Implementation – Scale-out NAS

- Pools multiple nodes together in a cluster that works as a single NAS device
  - Pool is managed centrally
- Scales performance and/or capacity with addition of nodes to the pool non-disruptively
- Creates a single file system that runs on all nodes in the cluster
  - Clients, connected to any node, can access entire file system
  - File system grows dynamically as nodes are added
- Stripes data across all nodes in a pool along with mirror or parity protection

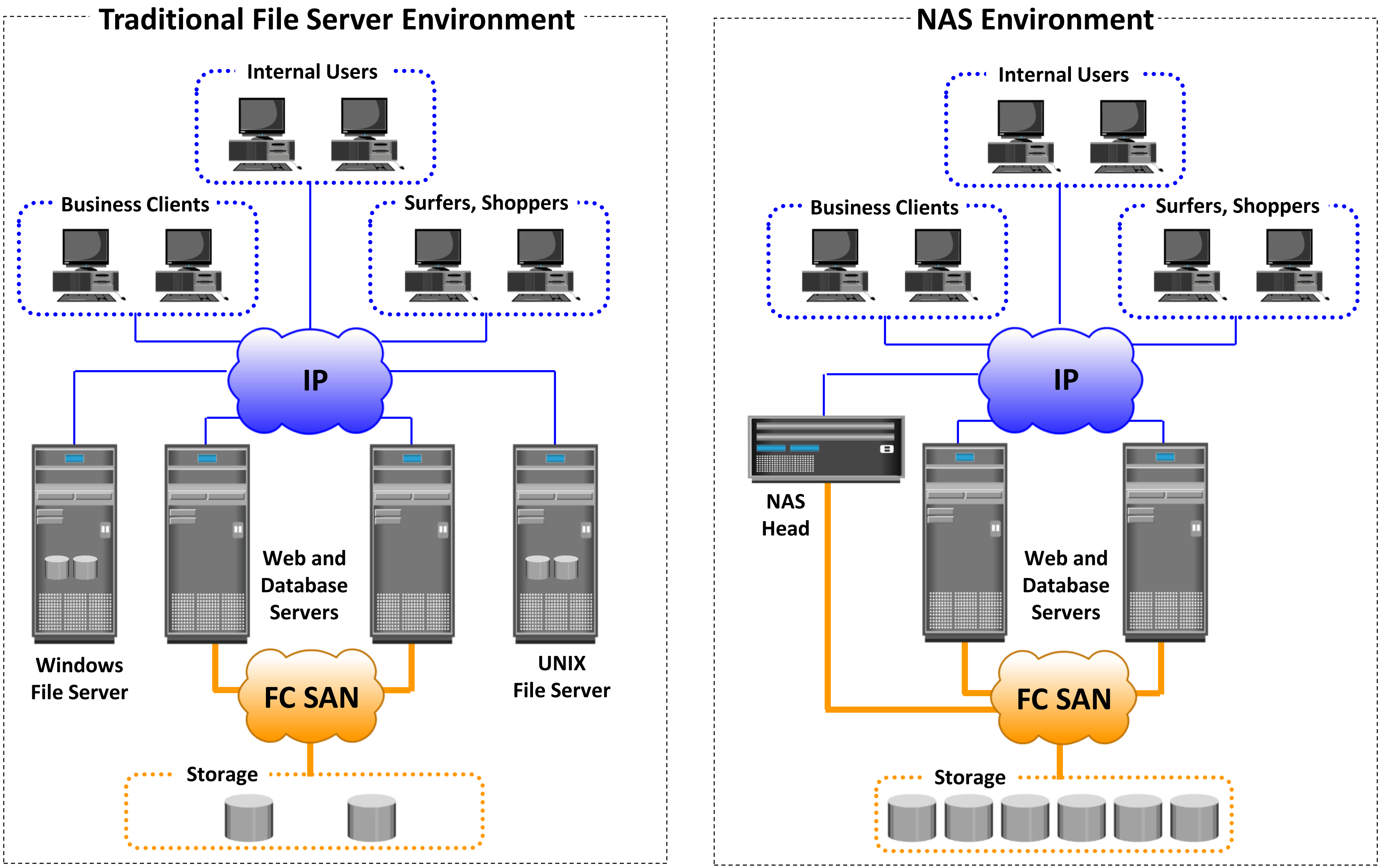
Scale-out NAS Connectivity



NAS Use Case 1 – Server Consolidation with NAS



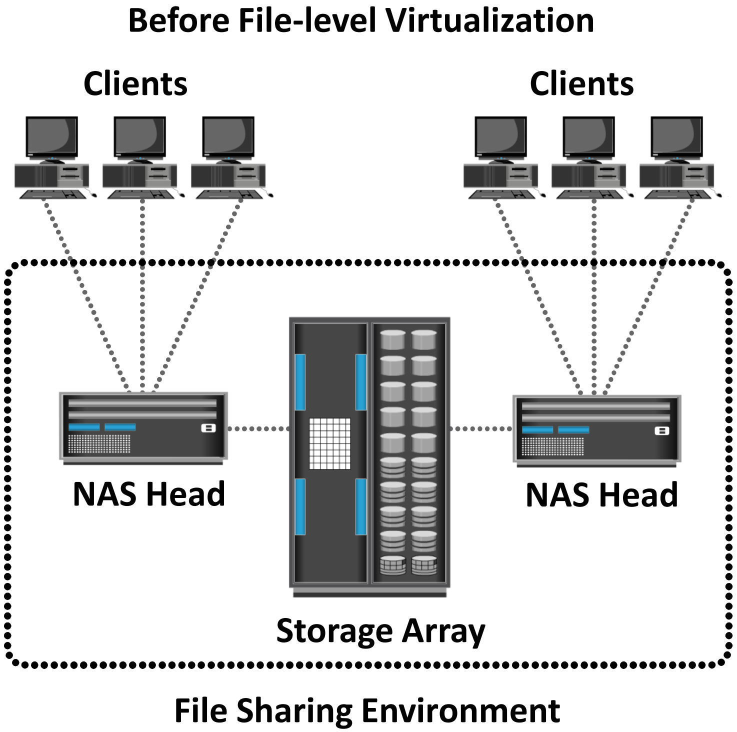
NAS Use Case 2 – Storage Consolidation with NAS

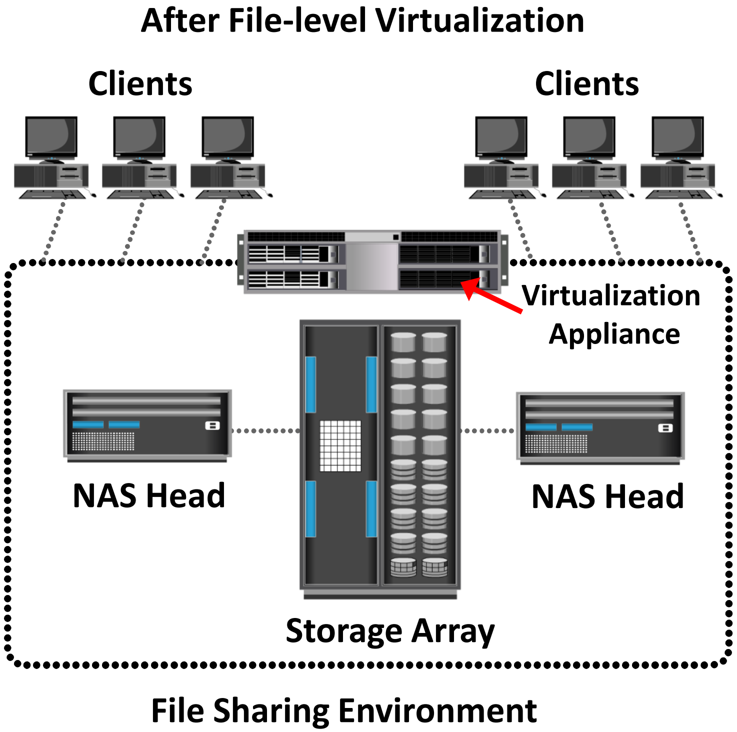


File-level Virtualization

- Eliminates dependency between data accessed at the file-level and the location where the files are physically stored
- Enables users to use a logical path, rather than a physical path, to access files
- Uses global namespace that maps logical path of file resources to their physical path
- Provides non-disruptive file mobility across file servers or NAS devices

Comparison: Before and After File-level Virtualization





- Dependency between client access and file location
- Underutilized storage resources
- Downtime is caused by data migrations

- Break dependencies between client access and file location
- Storage utilization is optimized
- Non-disruptive migrations

Module 7: Network-Attached Storage (NAS)

Concept in Practice:

- EMC Isilon
- EMC VNX Gateway

EMC Isilon

- Scale-out NAS solution
- Includes ‘OneFS’ operating system that creates a single file system across Isilon cluster
- Provides ability to nondisruptively add nodes to Isilon cluster
- Includes ‘SmartPools’ that enables different node types to be mixed in a single cluster
- Monitors component health and transparently reallocates files
- Uses ‘Autobalance’ that rebalances data automatically, when a new node is added to the cluster
- Uses ‘FlexProtect’ that protects from up to four simultaneous failures of either nodes or individual drives

EMC VNX Gateway

- Gateway NAS solution
- Provides multi-protocol network file system access, dynamic expansion of file systems, high availability, and high performance
- Comprises one or more NAS heads, called ‘X-Blades’ that run VNX operating environment
- Includes ‘Control Station’ that provides a single point for configuring X-Blades

NFS and CIFS in more detail

Based on "NFS, AFS, GFS" presentation by Yunji Zhong (ASU) and "Network File System" presentation by Joshua Caltagirone-Holzli (Univ. of Florida)

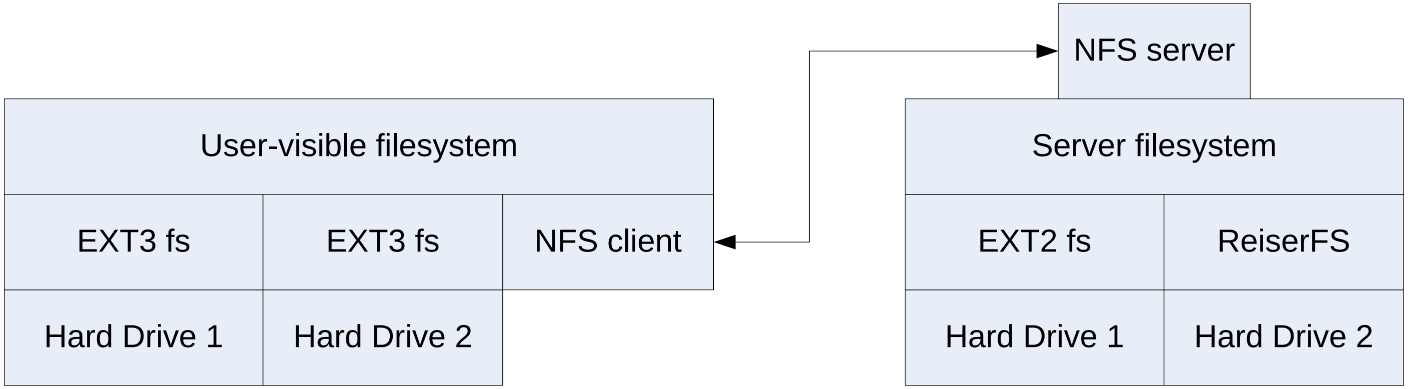
NFS (Network File System)

- First developed in 1980s by Sun
- Presented with standard UNIX FS interface
- Network drives are *mounted* into local directory hierarchy

- Initially completely stateless
  - Operated over UDP; did not use TCP streams
  - File locking, etc, implemented in higher-level protocols
- Modern implementations use TCP/IP & stateful protocols

Server-side Implementation

- NFS defines a *virtual file system*
  - Does not actually manage local disk layout on server
- Server instantiates NFS volume on top of local file system
  - Local hard drives managed by concrete file systems (EXT, ReiserFS, ...)



NFS Locking

- NFS v4 supports stateful locking of files
  - Clients inform server of intent to lock
  - Server can notify clients of outstanding lock requests
  - Locking is lease-based: clients must continually renew locks before a timeout
  - Loss of contact with server abandons locks

NFS Client Caching

- NFS Clients are allowed to cache copies of remote files for subsequent accesses
- Supports *close-to-open* cache consistency
  - When client A closes a file, its contents are synchronized with the master, and timestamp is changed
  - When client B opens the file, it checks that local timestamp agrees with server timestamp. If not, it discards local copy.
  - Concurrent reader/writers must use flags to disable caching

Security and NFS

- /etc/exports
  - This file enumerates the hostnames of systems who have access to directories in the local file system
- Export file systems only to clients you trust
- Access to NFS ports should be restricted

- File level access on NFS based on:
  - UID, GID, and file permissions
- NFS servers trust the client to tell who is accessing flies
- Example: if mary and bob have the same UID then they are able to access each other's files

- Root_squash – prevents root from changing the UID on the NFS server
  - Forces root to be a normal user on the server

Security and CIFS

- Clients log into CIFS server with end-user credentials
  - No client "trust" as with NFS
- Windows file systems support Access Control Lists (ACLs)
  - Fine-grained control of access permissions to users and groups

- Local account mode:
  - User account exists on the local client machine
  - When connecting to a remote CIFS share, user enters in credentials for remote user they want to connect as
- Active Directory mode:
  - All systems in the environment are "in the domain" (share credentials and policies via Active Directory)
  - User logs into system with a single-signon (like Duke NetID)
  - Attempts to access CIFS shares automatically use this account (can override with other credentials if desired)