



OpenVMS Cluster Internals and Data Structures

Keith Parris
Multivendor Systems Engineering

HP Services



Overview



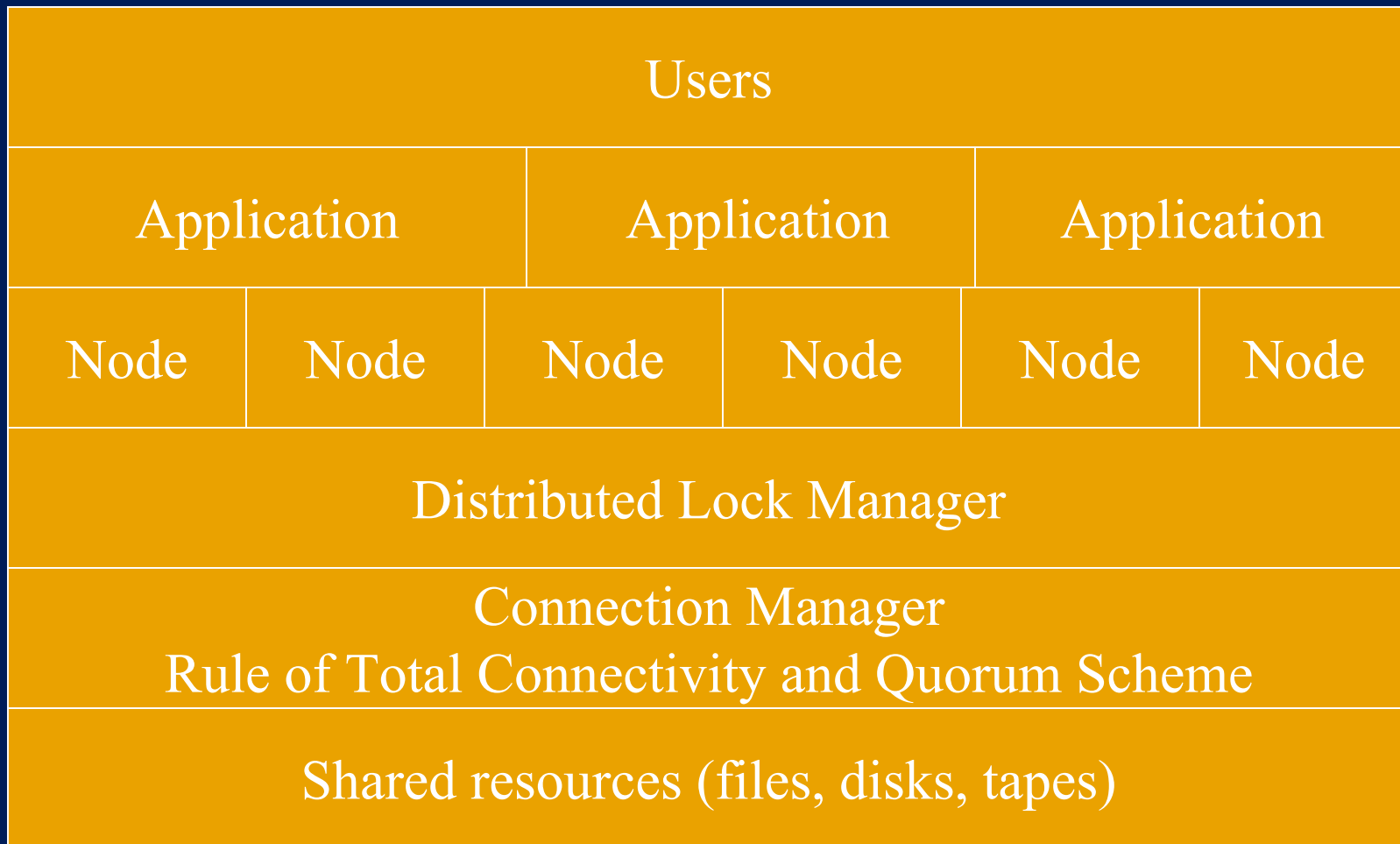
- Emphasis will be on Architecture & Design, Algorithms, and Data
- Data structure study focus will be on the functions and meaning of the data
 - Not the physical layout of a block of data in memory
 - Actual data field names are usually not included
- Focus in the area of algorithms will be on practical applications of OpenVMS Cluster Internals knowledge
- Information is as of OpenVMS version 7.3-1, with some 7.3-2 info

OpenVMS Cluster Overview



- An OpenVMS Cluster is a set of distributed systems which cooperate
- Cooperation requires coordination, which requires communication

Foundation for Shared Access



Foundation Topics



- SCA and its guarantees
 - Interconnects
- Connection Manager
 - Rule of Total Connectivity
 - Quorum Scheme
- Distributed Lock Manager
- MSCP/TMSCP Servers

OpenVMS Cluster Data Structures



- SCS
 - SB (System Block)
 - PB (Path Block)
 - PDT (Port Descriptor Table)
 - CDT (Connection Descriptor Table)
 - RDT (Request Descriptor Table)
 - BDT (Buffer Descriptor Table)

OpenVMS Cluster Data Structures



- Connection Manager
 - CLUB (Cluster Block)
 - CLUDCB (Cluster [Quorum] Disk Control Block)
 - CSB (Cluster System Block)
 - CSV (Cluster System Vector)

OpenVMS Cluster Data Structures



- Distributed Lock Manager
 - RSB (Resource Block)
 - Resource Hash Table
 - LKB (Lock Block)
 - Lock ID Table

System Communications Architecture (SCA)



- SCA governs the communications between nodes in an OpenVMS cluster

System Communications Services (SCS)



- System Communications Services (SCS) is the name for the OpenVMS code that implements SCA
 - The terms SCA and SCS are often used interchangeably
- SCS provides the foundation for communication from OpenVMS nodes on a cluster interconnect to:
 - Other OpenVMS systems
 - Storage controllers

Cluster Interconnects



- SCA has been implemented on various types of hardware:
 - Computer Interconnect (CI)
 - Digital Storage Systems Interconnect (DSSI)
 - Fiber Distributed Data Interface (FDDI)
 - Ethernet (10 megabit, Fast, Gigabit)
 - Asynchronous Transfer Mode (ATM) LAN
 - Memory Channel
 - Galaxy Shared Memory

Cluster Interconnects



Interconnect	MB/sec	Distance	Nodes
CI	2 x 8.75	90 m	32
DSSI	3.75	6 m	8
Ethernet	1.25	500 m	100s
Fast Ethernet	12.5	100 m	100s
Gigabit Ethernet	125	30 m/100 km	100s
FDDI	12.5	2 km/100 km	100s
Memory Channel	100	3 m/3 km	8

Cluster Interconnects: Host CPU Overhead



Interconnect	Host CPU Overhead
Local on node	Very Low
Galaxy SMCI	High
Memory Channel 2/1.5	High/High
Gigabit Ethernet	Medium
FDDI	Medium
DSSI	Low
CI	Low

Interconnects (Storage vs. Cluster)

- Originally, CI was the *one and only* Cluster Interconnect
- CI allowed connection of both OpenVMS nodes and storage controllers
- LANs allowed connections to OpenVMS nodes and LAN-based Storage Servers
- SCSI and Fibre Channel allowed only connections to storage – no communications to other OpenVMS nodes
- So now we must differentiate between Cluster Interconnects and Storage Interconnects

Storage-only Interconnects



- Small Computer Systems Interface (SCSI)
- Fibre Channel (FC)

Cluster & Storage Interconnects



- CI
- DSSI
- LAN

Cluster-only Interconnects (No Storage)



- Memory Channel
- Galaxy Shared Memory Cluster Interconnect (SMCI)
- ATM LAN

System Communications Architecture (SCA)



- Each node must have a unique:
 - SCS Node Name
 - SCS System ID
- Flow control is credit-based

System Communications Architecture (SCA)



- Layers:
 - SYSAPs
 - SCS
 - Ports
 - Interconnects

SCA Architecture Layers



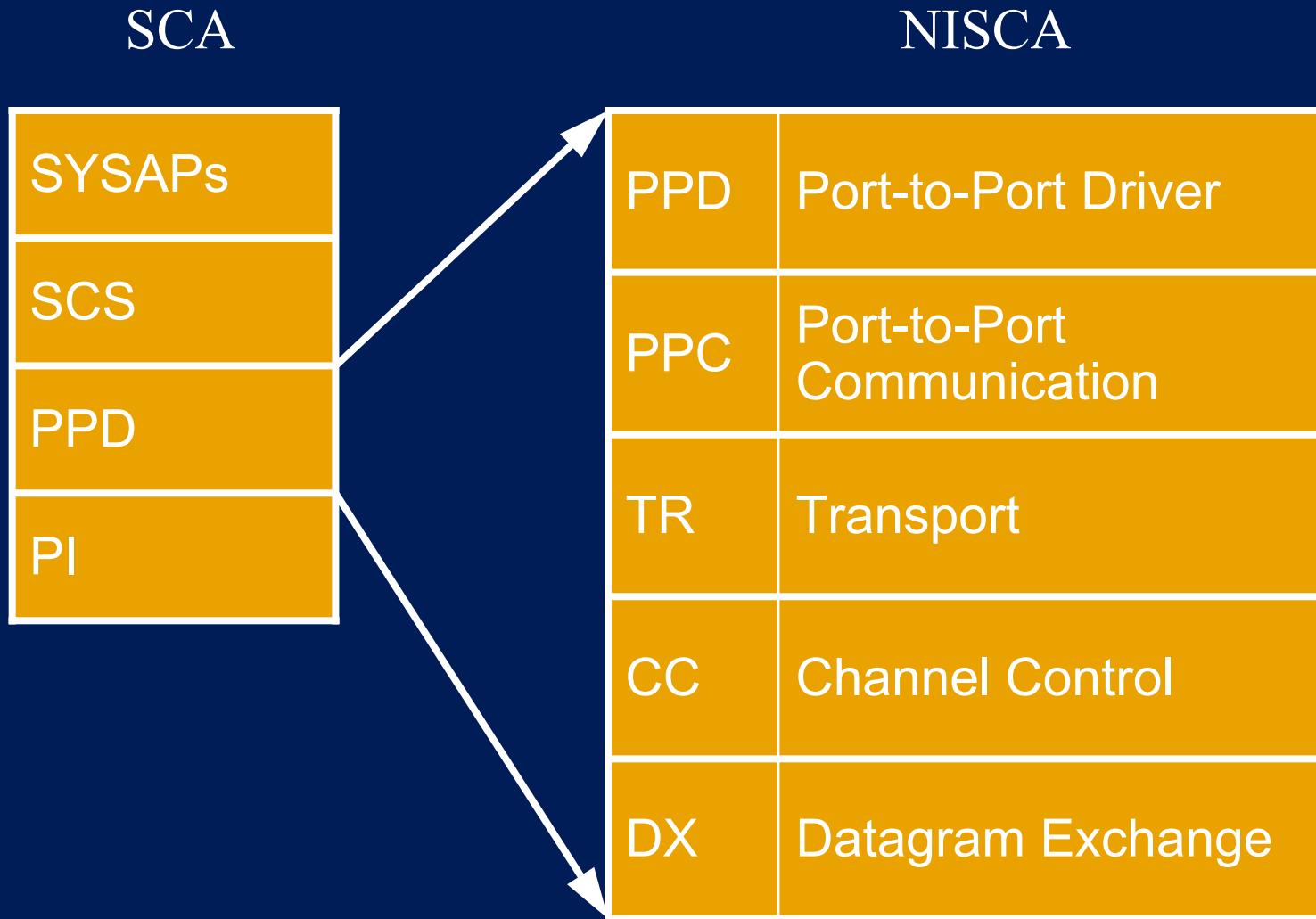
SYSAPs	System Applications
SCS	System Communications Services
PPD	Port-to-Port Driver
PI	Physical Interconnect

LANs as a Cluster Interconnect



- SCA is implemented in hardware by CI and DSSI port hardware
- SCA over LANs is provided by Port Emulator software (PEDRIVER)
- SCA over LANs is referred to as NISCA
 - NI is for Network Interconnect, (an early name for Ethernet within DEC, in contrast with CI, the Computer Interconnect)
- SCA over LANs and storage on SANs is presently the focus for future directions in OpenVMS Cluster interconnects
 - Although InfiniBand or RDMA on IP are potentially promising cluster interconnects in the Itanium timeframe

NISCA Layering



SCS with Bridges and Routers



- If compared with the 7-layer OSI network reference model, SCA has no Routing (what OSI calls Network) layer
- OpenVMS nodes cannot route SCS traffic on each other's behalf
- SCS protocol can be bridged transparently in an extended LAN, but not routed

OSI Network Model



Layer 7	Application
Layer 6	Presentation
Layer 5	Session
Layer 4	Transport
Layer 3	Network
Layer 2	Data Link
Layer 1	Physical

OSI Network Model



7	Application	FAL, CTERM; Telnet, FTP, HTTP, etc.
6	Presentation	Data representation; byte ordering
5	Session	Data exchange between two presentation entities
4	Transport	Reliable delivery: duplicates, out-of-order packets, retransmission; e.g. TCP
3	Network	Routing; packet fragmentation/ reassembly; e.g. IP
2	Data Link	MAC addresses; bridging
1	Physical	LAN adapters (NICs), Twisted-pair cable, Coaxial cable, Fiber optic cable

SCS on LANs



- Because multiple independent clusters might be present on the same LAN, each cluster is identified by a unique Cluster Group Number, which is specified when the cluster is first formed.
- As a further precaution, a Cluster Password is also specified. This helps protect against the case where two clusters inadvertently use the same Cluster Group Number. If packets with the wrong Cluster Password are received, errors are logged.

SCS on LANs



- The cluster group number can be determined by:
 - `SYSMAN> CONFIG SHOW CLUSTER_AUTHORIZATION`
- The cluster password is encrypted with a 1-way trapdoor algorithm, and cannot be retrieved. If it is lost, either:
 - It has to be re-set everywhere using `SYSMAN> CONFIG SET CLUSTER` and the cluster rebooted, or else
 - The file `SYS$COMMON:[SYSEXEC]CLUSTER_AUTHORIZE.DAT`, which contains the cluster group number and (encrypted) password, has to be copied to any new system disk added to the cluster

Interconnect Preference by SCS



- When choosing an interconnect to a node, SCS chooses one “best” interconnect type, and sends all its traffic down that one
- If the “best” interconnect type fails, it will fail over to another

Interconnect Preference by SCS



- Historically, SCS used this priority order by default:
 1. Galaxy Shared Memory Cluster Interconnect (SMCI)
 2. Memory Channel
 3. CI
 4. DSSI
 5. NI (FDDI, Ethernet, and ATM)

Port Load Class Values



From [LIB.LIS]PDTDEF.SDL, or \$PDTDEF:

Symbolic load class name	Interconnect type	Port Load Class Value (Mbits/sec)
PDT\$C_SM_CLASS	Galaxy SMCI	%x7FFF
PDT\$C_MC_CLASS	Memory Channel	800
PDT\$C_CI_CLASS	CI	140
PDT\$C_DSSI_MEDIUM_CLASS	DSSI	48
PDT\$C_NI_CLASS	Network Interconnect (LANs)	10

SCS Interconnect Selection Inequities



- Because LAN interconnects have grown in speed over time, the low Port Class value for the “NI” class became outdated
 - e.g. DSSI would be chosen over Gigabit Ethernet
- OpenVMS 7.3-1 sets priority values better, and also allows the default priorities to be overridden with the SCACP utility
 - For earlier versions of OpenVMS, the PORT_CLASS_SETUP tool available from CSC allows you to change order of priority if needed

Interconnect Preference by SCS



- Within the “NI” class, PEDRIVER handles path selection for LAN adapters (FDDI, Ethernet, and ATM), using:
 - 1) Maximum transfer size
 - 2) Lowest latency
 - 3) Packet loss rate

Interconnect load balancing



- PEDRIVER balances load among redundant LAN paths
 - More about this in a moment
- On VAX, OpenVMS would automatically shift connections around if a CI or DSSI port got overloaded. This was known as CI Port Load Sharing.
 - Based on average data volumes, ports might be placed into Yellow or Red zones and connections off-loaded from the port

Interconnect load balancing



- Unfortunately, CI Port Load Sharing code didn't get ported from VAX to Alpha
 - MOVE_REMOTENODE_CONNECTIONS tool available from CSC allows you to statically balance VMS\$VAXcluster SYSAP connections across multiple CI (or DSSI) adapters

PEDRIVER Load Distribution



Prior to 7.3:

- PEDRIVER transmitted packets on the one path to a given remote node which had lowest latency for received multicast Hello packets
 - Problem: Latency in reverse direction might be different from latency in forward direction
- PEDRIVER took advantage of DEC 10/100 (Ethernet/FDDI) bridges clearing the priority field in the header to detect packets which went from FDDI onto (slower) Ethernet
 - Problem: Non-standard -- non-DEC FDDI/Ethernet bridges didn't do this; no analogous field in Fast Ethernet or Gigabit Ethernet
- PEDRIVER only transmitted on one LAN adapter to a given node for a given load-sharing interval
 - Problem: Can't fully utilize multiple LAN adapters, especially when talking to one remote node

PEDRIVER Load Distribution



Improved behavior in 7.3 and above:

- PEDRIVER tracks latency in forward direction, not reverse
- PEDRIVER dynamically probes and adapts maximum packet size based on what actually gets through at a given point in time
- PEDRIVER selects a set of equivalent paths, and transmits over all of them on a round-robin basis

SCS Flow Control



- SCS flow control is credit-based
- Connections start out with a certain number of credits
 - Credits are used as messages are sent, and
 - Message cannot be sent unless a credit is available
 - Credits are returned as messages are acknowledged
- This prevents one system from over-running another system's resources
 - But insufficient credits can adversely affect performance

SCS Flow Control

- How to detect credit waits
 - \$ SHOW CLUSTER/CONTINUOUS
 - ADD CONNECTIONS
 - ADD CR_WAITS
 - Watch for large values which increase relatively rapidly over time
 - Or see DECps Performance Analysis report

Example of SCS Credit Waits



**\$ SHOW CLUSTER/CONTINUOUS
ADD CONNECTIONS
ADD CR_WAITS**

View of Cluster from system ID 12345 node: NODE01 6-MAY-2002 14:45:16

SYSTEMS		MEMBERS	CONNECTIONS		COUNTERS
NODE	SOFTWARE	STATUS	LOC_PROC_NAME	CON_STA	CR_WAITS
NODE01	VMS E7.3	MEMBER	SCS\$DIRECTORY	LISTEN	
			MSCP\$DISK	LISTEN	
			MSCP\$TAPE	LISTEN	
			VMS\$SDA_AXP	LISTEN	
			VMS\$VAXcluster	LISTEN	
			SCA\$TRANSPORT	LISTEN	
NODE02	VMS V7.2	MEMBER	SCA\$TRANSPORT	OPEN	0
			VMS\$DISK_CL_DRVR	OPEN	0
			MSCP\$DISK	OPEN	0
			MSCP\$TAPE	OPEN	0
			VMS\$VAXcluster	OPEN	1024
			VMS\$DISK_CL_DRVR	OPEN	0
NODE03	VMS V7.3	MEMBER	SCA\$TRANSPORT	OPEN	0
			VMS\$TAPE_CL_DRVR	OPEN	0
			MSCP\$DISK	OPEN	0
			MSCP\$TAPE	OPEN	0
			VMS\$VAXcluster	OPEN	3
			SCA\$TRANSPORT	OPEN	0
NODE04	VMS V7.2	MEMBER	SCA\$TRANSPORT	OPEN	0

SCS Flow Control

- How to alleviate credit waits:
 - For wait on VMS\$VAXcluster SYSAP on another OpenVMS node:
 - Raise SYSGEN parameter CLUSTER_CREDITS
 - Balance load among OpenVMS nodes
 - For wait on VMS\$DISK_CL_DRVR SYSAP to MSCP\$DISK SYSAP on an HSJ or HSD controller, redistribute credits
 - Lower MAXIMUM_HOSTS parameter on HSx controller to actual number of OpenVMS host nodes
 - Balance load across controllers
 - For wait on VMS\$DISK_CL_DRVR SYSAP to MSCP\$DISK SYSAP on another OpenVMS node:
 - Raise SYSGEN parameter MSCP_CREDITS on serving node
 - Balance load among OpenVMS MSCP-serving nodes
- Note that alleviating SCS credit waits may allow driving resources farther into saturation!

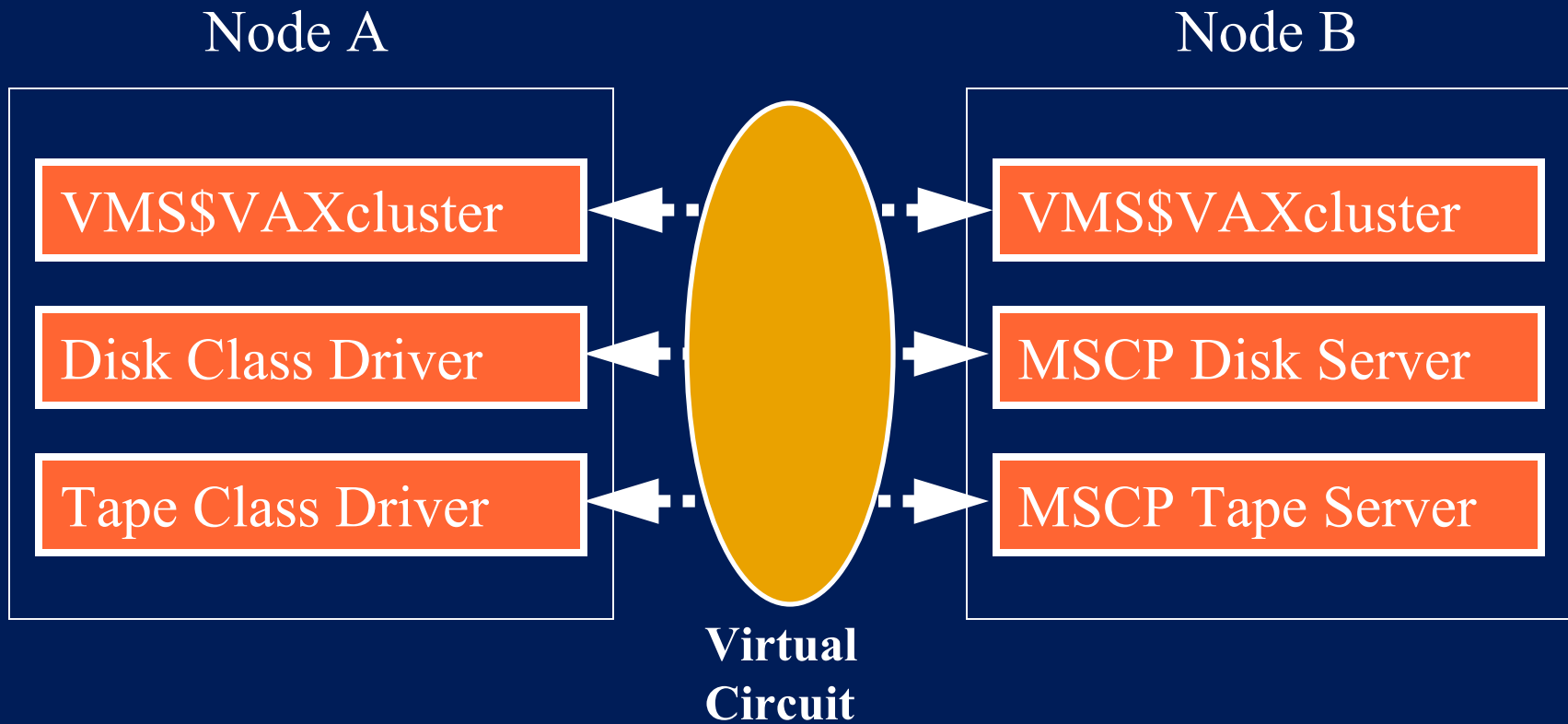
- SCS provides “reliable” port-to-port communications
- SCS multiplexes messages and data transfers between nodes over Virtual Circuits
- SYSAPs communicate via Connections over Virtual Circuits

Virtual Circuits



- Formed between ports on a Cluster Interconnect of some flavor
- Can pass data in 3 ways:
 - Datagrams
 - Sequenced Messages
 - Block Data Transfers

Connections over a Virtual Circuit



Datagrams



- “Fire and forget” data transmission method
- No guarantee of delivery
 - But high probability of successful delivery
- Delivery might be out-of-order
- Duplicates possible
- Maximum size typically 576 bytes
 - SYSGEN parameter SCSMAXDG (max. 985)

Sequenced Messages



- Guaranteed delivery (no lost messages)
- Guaranteed ordering (first-in, first-out delivery; same order as sent)
- Guarantee of no duplicates
- Maximum size presently 216 bytes
 - SYSGEN parameter SCSSMAXMSG (max. 985)

Block Data Transfers

- Used to move larger amounts of bulk data (too large for a sequenced message):
 - Disk or tape data transfers
 - OPCOM messages
- Data is mapped into “Named Buffers”
 - which specify location and size of memory area
- Data movement can be initiated in either direction:
 - Send Data
 - Request Data

Example Uses



Datagrams

Polling for new nodes; Virtual Circuit formation; logging asynchronous errors

Sequenced Messages

Lock requests; MSCP I/O requests and MSCP End messages with I/O status

Block Data Transfers

Disk and tape I/O data; OPCOM messages

System Applications (SYSAPs)



- Despite the name, these are pieces of the operating system, not user applications
- Work in pairs for specific purposes
- Communicate using a Connection formed over a Virtual Circuit between nodes
- Although unrelated to OpenVMS user processes, each is given a “Process Name”

SYSAPs



- SCS\$DIR_LOOKUP → SCS\$DIRECTORY
 - Allows OpenVMS to determine if a node has a given SYSAP
- VMS\$VAXcluster → VMS\$VAXcluster
 - Connection Manager, Distributed Lock Manager, OPCOM, etc.
- VMS\$DISK_CL_DRVR → MSCP\$DISK
 - Disk drive remote access
- VMS\$TAPE_CL_DRVR → MSCP\$TAPE
 - Tape drive remote access
- SCA\$TRANSPORT → SCA\$TRANSPORT
 - Queue Manager, DECdtm (Distributed Transaction Manager)

SYSAPs



Local Process Name	Remote Process Name	Function
VMS\$VAXcluster	VMS\$VAXcluster	Connection Manager, Lock Manager, CWPS, OPCOM, etc.
VMS\$DISK_CL_DRVR	MSCP\$DISK	MSCP Disk Service
VMS\$TAPE_CL_DRVR	MSCP\$TAPE	MSCP Tape Service
SCA\$TRANSPORT	SCA\$TRANSPOR T	Old \$IPC, queue manager, DECdtm
SCS\$DIR_LOOKUP	SCS\$DIRECTORY	SCS process lookup
	MSCP\$DUP	\$SET HOST/DUP
	VMS\$SDA_AXP	Remote SDA

SCS Data Structures: SB (System Block)



- Data displayed by
SHOW CLUSTER/CONTINUOUS
ADD SYSTEMS/ALL
- One SB per SCS node
 - OpenVMS node or Controller node which uses SCS

SCS Data Structures: SB (System Block)



- Data of interest:
 - SCS Node Name and SCS System ID
 - CPU hardware type/model, and HW revision level
 - Software type and version
 - Software incarnation
 - Number of paths to this node (virtual circuits)
 - Maximum sizes for datagrams and sequenced messages sent via this path

SCS Data Structures: PB (Path Block)



- Data displayed by
\$ SHOW CLUSTER/CONTINUOUS
ADD CIRCUITS/ALL
- One PB per virtual circuit to a given node

SCS Data Structures: PB (Path Block)



- Data of interest:
 - Virtual circuit status
 - Number of connections over this path
 - Local port device name
 - Remote port ID
 - Hardware port type, and hardware revision level
 - Bit-mask of functions the remote port can perform
 - Load class value (so SCS can select the optimal path)

SCS Data Structures: PDT (Port Descriptor Table)



- Data displayed by
\$ SHOW CLUSTER/CONTINUOUS
ADD LOCAL_PORTS/ALL
- One PDT per local SCS port device
 - LANs all fit under one port: PEA0

SCS Data Structures: PDT (Port Descriptor Table)



- Data of interest:
 - Port device name
 - Type of port
 - i.e. VAX CI, NPORT CI, DSSI, NI, MC, Galaxy SMCI
 - Local port ID, and maximum ID for this interconnect
 - Count of open connections using this port
 - Pointers to routines to perform various port functions
 - Load Class value
 - Counters of various types of traffic
 - Maximum data transfer size

SCS Data Structures: CDT (Connection Descriptor Table)



- Data displayed by
\$ SHOW CLUSTER/CONTINUOUS
ADD CONNECTIONS/ALL
ADD COUNTERS/ALL

SCS Data Structures: CDT (Connection Descriptor Table)



- Data of interest:
 - Local and remote SYSAP names
 - Local and remote Connection Ids
 - Connection status
 - Pointer to System Block (SB) of the node at the other end of the connection
 - Counters of various types of traffic

SCS Data Structures: RDT (Request Descriptor Table)



- Request Descriptor Table entries are used to keep track of outstanding requests where a response is expected
 - i.e. Lock Requests, or MSCP I/O operations
 - i.e. not a \$DEQ operation

SCS Data Structures: BDT (Buffer Descriptor Table)



- Buffer Descriptor Table entries are used to keep track of Named Buffers used for SCS Block Data Transfers

Connection Manager



- The Connection Manager is code within OpenVMS that coordinates cluster membership across events such as:
 - Forming a cluster initially
 - Allowing a node to join the cluster
 - Cleaning up after a node which has failed or left the cluster
- all the while protecting against uncoordinated access to shared resources such as disks

Rule of Total Connectivity



- Every system must be able to talk “directly” with every other system in the cluster
 - Without having to go through another system
 - Transparent LAN bridges are considered a “direct” connection

Quorum Scheme



- The Connection Manager enforces the Quorum Scheme to ensure that all access to shared resources is coordinated
- A majority of the potential cluster systems must be present in the cluster before any access to shared resources (i.e. disks) is allowed

Quorum Scheme

- Nodes are assigned values for the number of votes they have in determining a majority
- The total number of votes possible is called the “Expected Votes” – the number of votes to be expected when all cluster members are present
- Quorum is calculated as “just over half” of the total possible (the Expected) votes

Quorum Scheme



- As in human parliamentary procedure, requiring a quorum before doing business prevents two or more subsets of members from meeting simultaneously and doing conflicting business

Quorum Scheme



- If a cluster member is not part of a cluster with quorum, OpenVMS keeps it from doing any harm by:
 - Putting all disks into Mount Verify state, thus stalling all disk I/O operations
 - Requiring that all processes have the QUORUM capability before they can run
 - Clearing the QUORUM capability bit on all CPUs in the system, thus preventing any process from being scheduled to run on a CPU and doing any work

Quorum Scheme



- Primarily to provide a tie-breaking vote in 2-node clusters, a “quorum disk” can be assigned votes
 - OpenVMS periodically writes cluster membership info into the QUORUM.DAT file on the quorum disk and later reads it to re-check it; if all is well, OpenVMS can treat the quorum disk as a virtual voting member of the cluster

Quorum Scheme



- If two non-cooperating subsets of cluster nodes both achieve quorum and access shared resources, this is known as a “partitioned cluster”
- When a partitioned cluster occurs, the disk structure on shared disks is quickly corrupted

Quorum Scheme



- Avoid a partitioned cluster by:
 - Proper setting of EXPECTED_VOTES parameter to the total of all possible votes
 - Taking care if DECcmds (or Availability Manager) Quorum Fix is used to force quorum to be achieved in a cluster [subset] that is hung due to lack of quorum
 - Care in using the REMOVE_NODE option with SHUTDOWN.COM to lower quorum to a point below half the expected votes

Connection Manager and Transient Failures



- Some failures are temporary and transient
 - Especially in a LAN environment
- To prevent the disruption of unnecessary removal of a node from the cluster, when a communications failure is detected, the Connection Manager waits for a time in hopes of the problem going away by itself
 - This time is called the Reconnection Interval
 - SYSGEN parameter RECNXINTERVAL
 - RECNXINTERVAL is dynamic and may thus be temporarily raised if needed for something like a scheduled LAN outage

LAN reliability



- Try to avoid saturation of any portion of LAN hardware
- Bridge implementations must not drop small packets under heavy loads
 - SCS Hello packets are small packets

Connection Manager and Communications or Node Failures



- If the Reconnection Interval passes without connectivity being restored, or if the node has “gone away”, the cluster cannot continue without a reconfiguration
- This reconfiguration is called a **State Transition**, and one or more nodes will be removed from the cluster

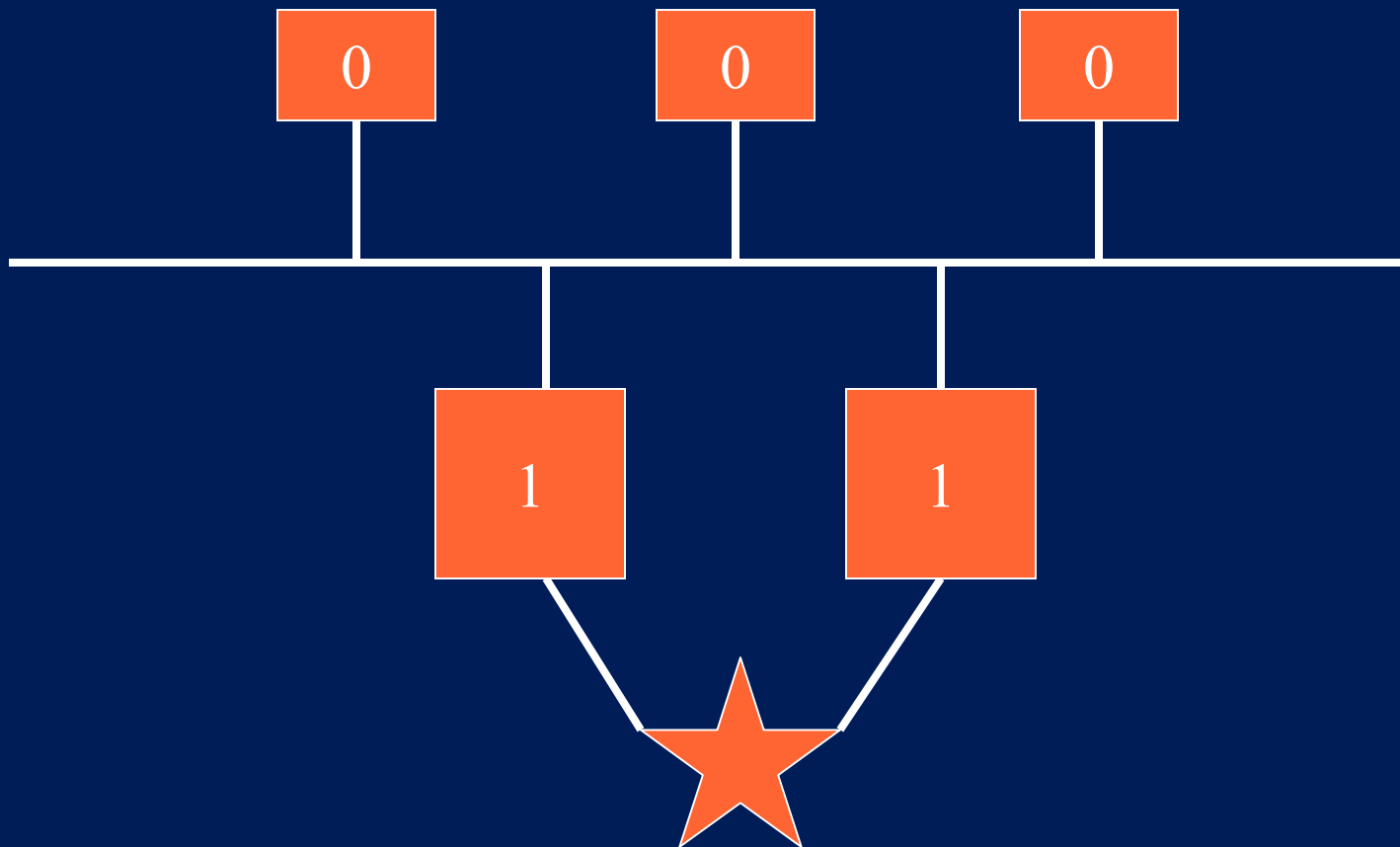
Optimal Sub-cluster Selection



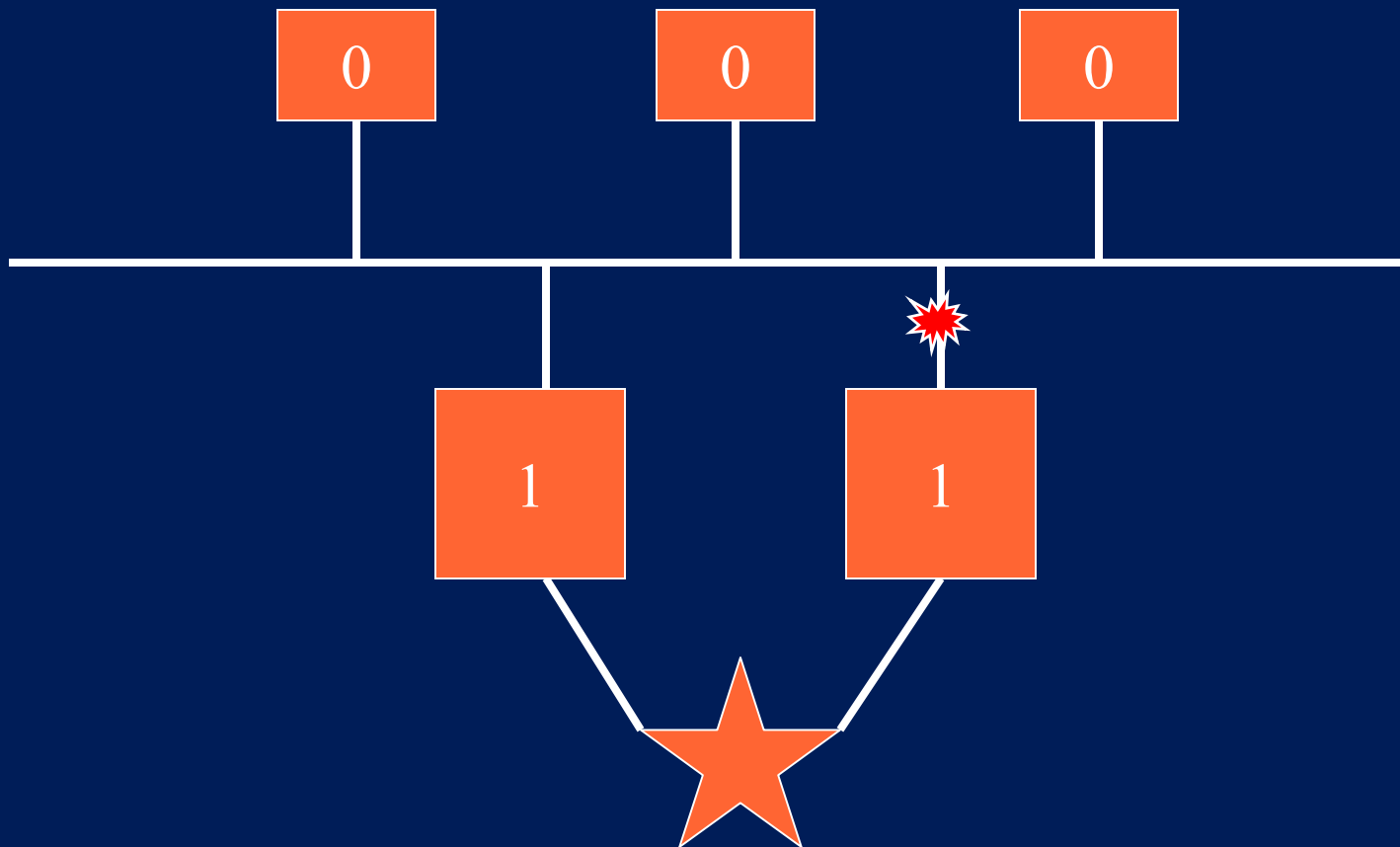
- Connection manager compares potential node subsets that could make up surviving portion of the cluster
- Pick sub-cluster with the most votes
- If votes are tied, pick sub-cluster with the most nodes
- If nodes are tied, arbitrarily pick a winner
 - based on comparing SCSSYSTEMID values of set of nodes with most-recent cluster software revision

- VOTES:
 - Most configurations with satellite nodes give votes to disk/boot servers and set VOTES=0 on all satellite nodes
 - If the sole LAN adapter on a disk/boot server fails, and it has a vote, ALL satellites will CLUEXIT!
 - Advice: give at least as many votes to node(s) on the LAN as any single server has, or configure redundant LAN adapters

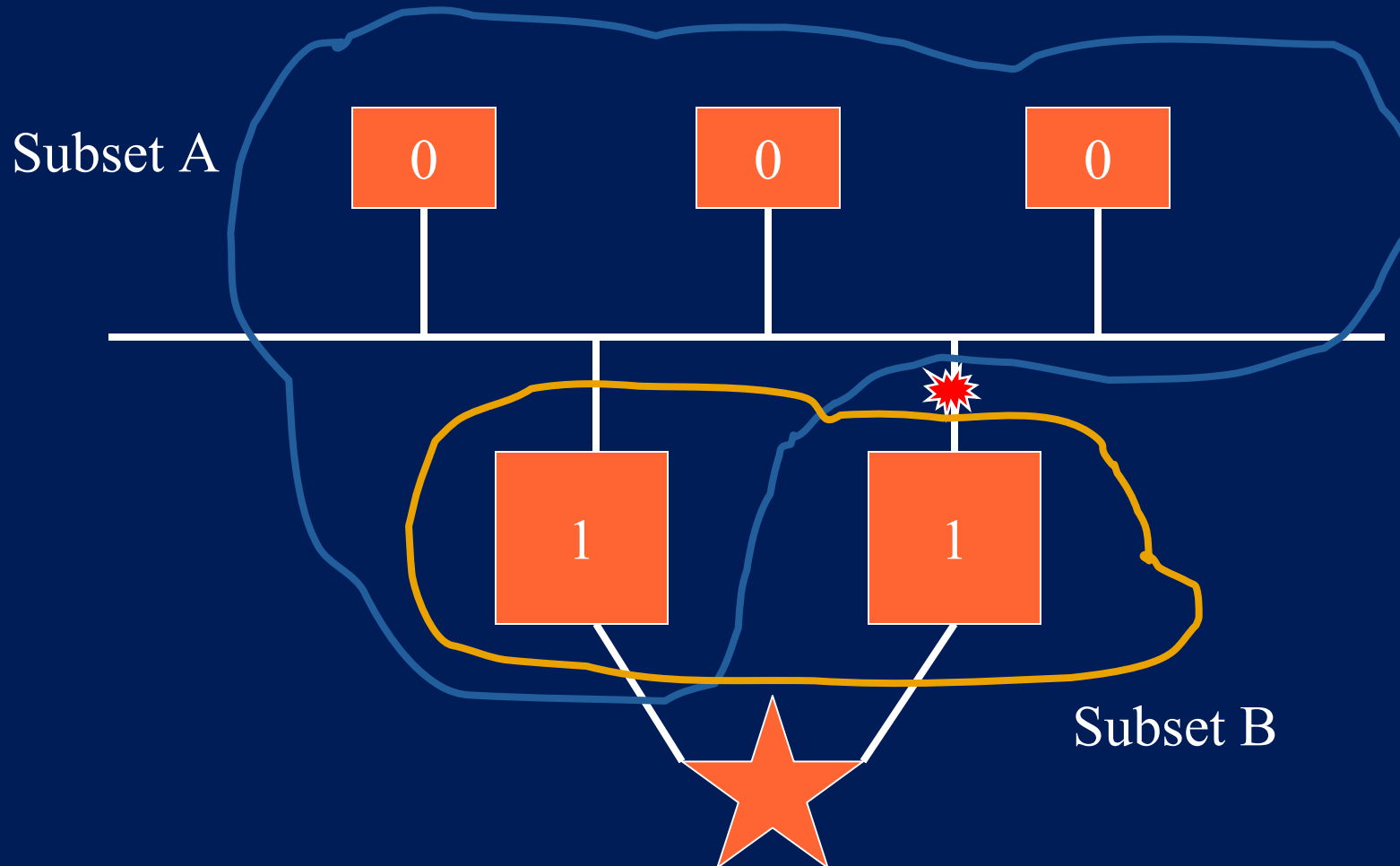
LAN redundancy and Votes



LAN redundancy and Votes

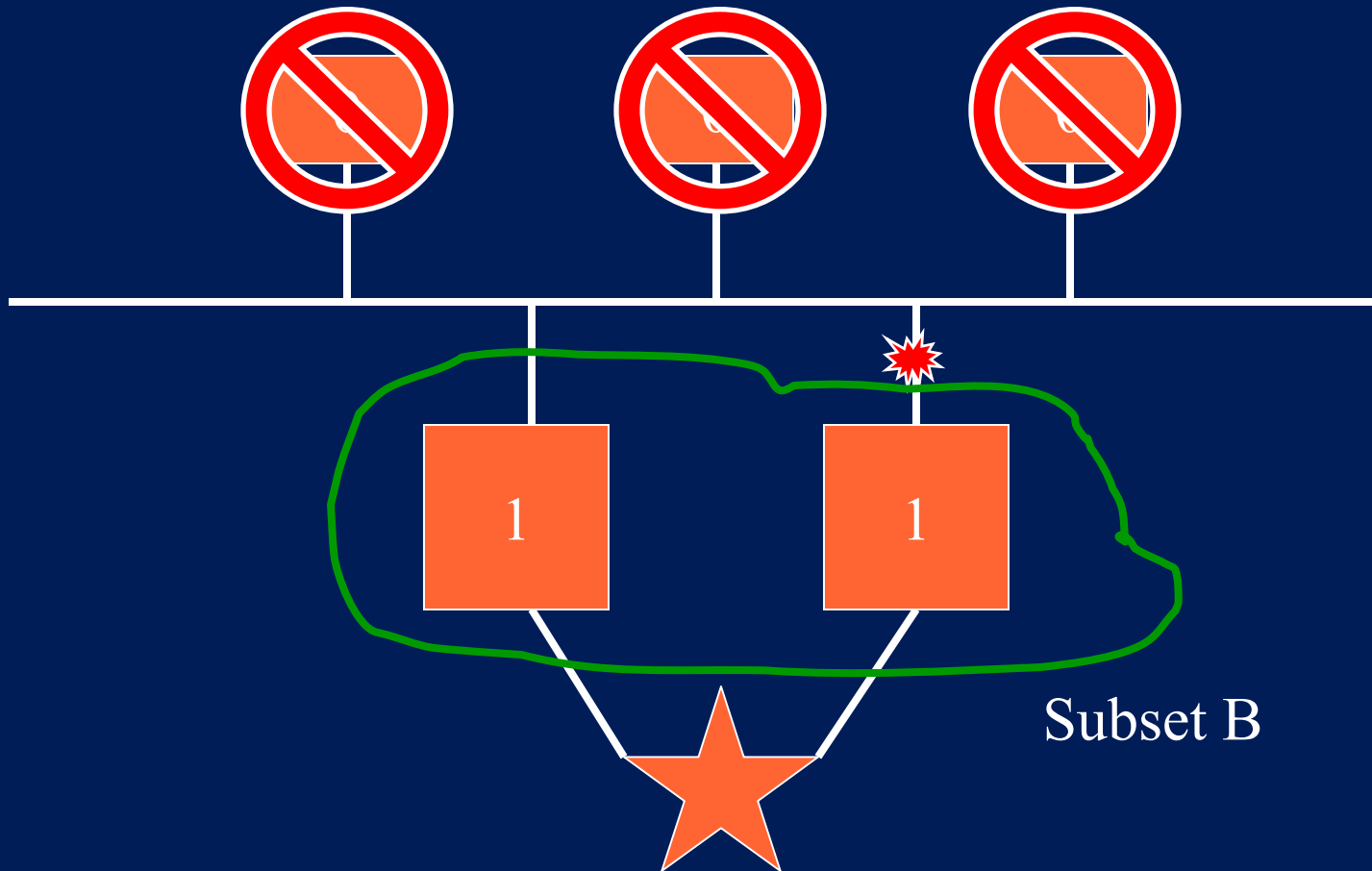


LAN redundancy and Votes

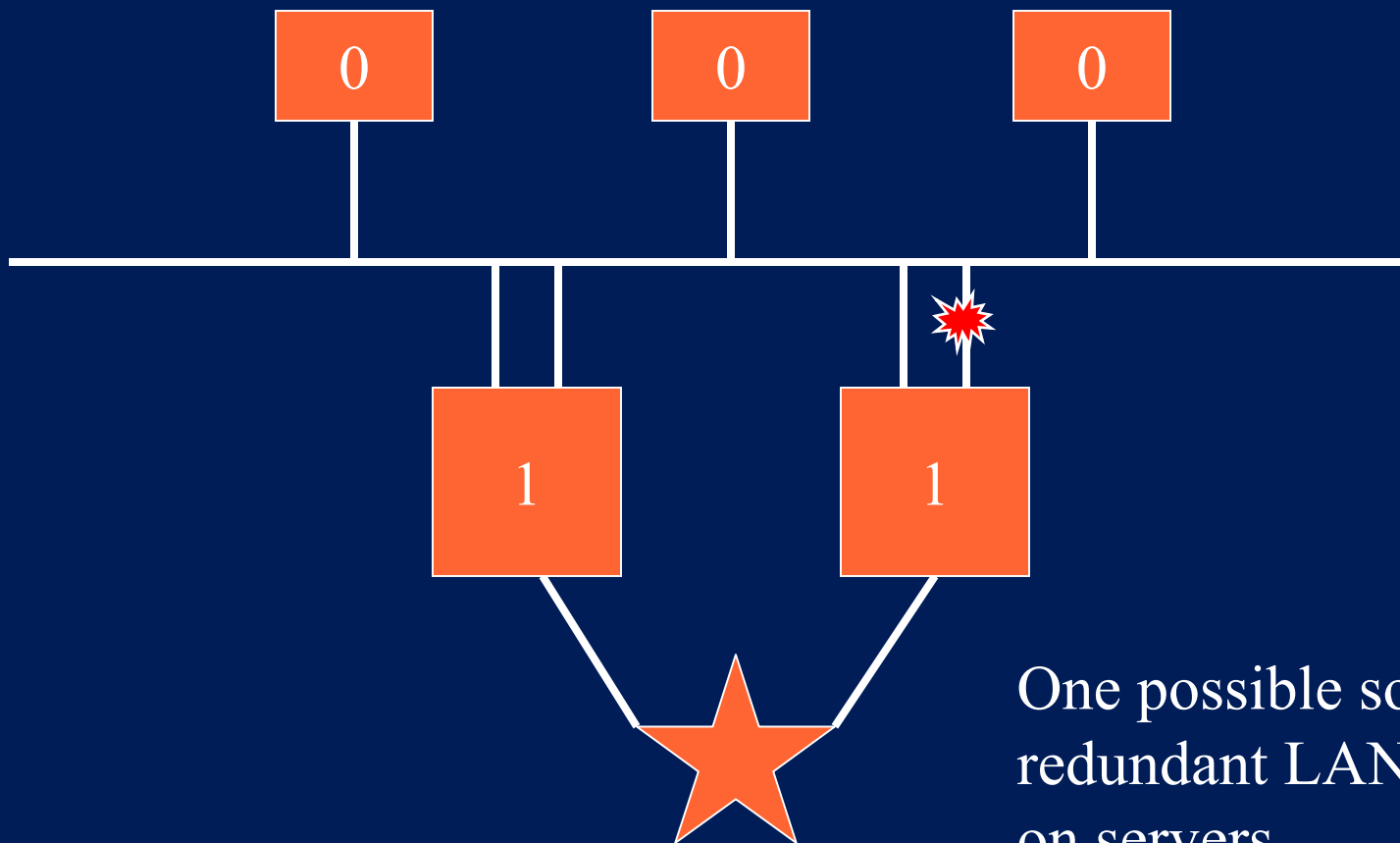


Which subset of nodes is selected as the optimal sub-cluster?

LAN redundancy and Votes

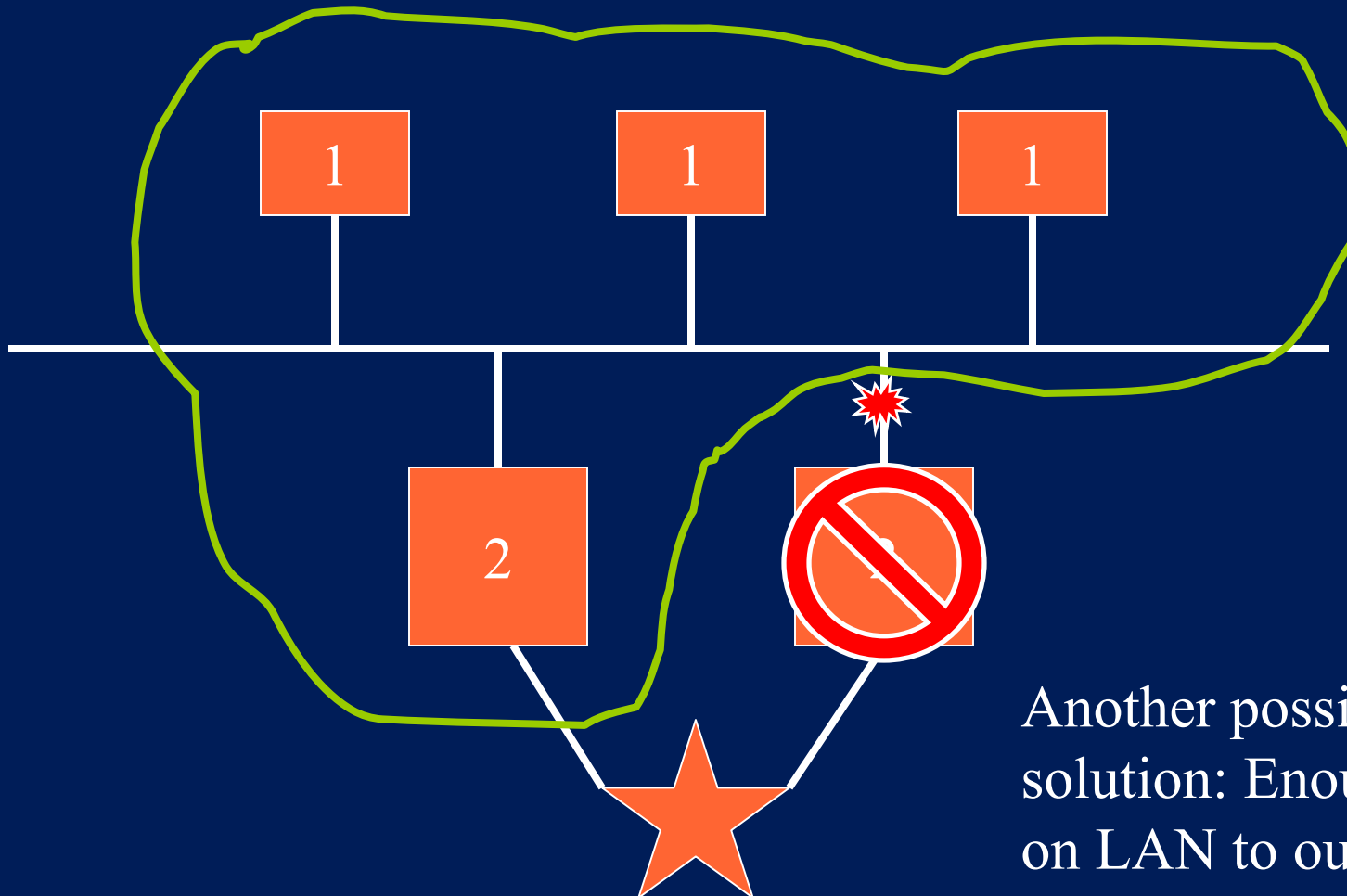


LAN redundancy and Votes



One possible solution:
redundant LAN adapters
on servers

LAN redundancy and Votes



Another possible solution: Enough votes on LAN to outweigh any single server node

Quorum Loss



- If too many systems leave the cluster, there may no longer be a quorum of votes left
- It is possible to manually force the cluster to recalculate quorum and continue processing if needed

Quorum Recovery Methods

- Software interrupt at IPL 12 from console
 - IPC> Q
- DECams or Availability Manager:
 - System Fix; Adjust Quorum
- DTCS or BRS integrated tool, using same RMDRIVER (DECams client) interface as DECams / AM

Connection Manager Data Structures: CLUB (Cluster Block)



- Only one CLUB on a given node
- Pointed to by CLU\$GL_CLUB
- Data displayed by
\$ SHOW CLUSTER/CONTINUOUS
ADD CLUSTER/ALL

Connection Manager Data Structures: CLUB (Cluster Block)



- Data of interest:
 - Number of cluster members
 - Total number of votes
 - Current cluster-wide effective values of EXPECTED_VOTES and QUORUM
 - Various flags, like whether the cluster has quorum or not
 - Pointer to the Cluster System Block (CSB) for the local system
 - Data used during state transitions, like the identity of State Transition Coordinator Node, and what phase the transition is in
 - Time the cluster was formed, and time of last state transition

Connection Manager Data Structures: CLUDCB (Cluster [Quorum] Disk Control Block)



- Contains info about the quorum disk

Connection Manager Data Structures: CSB (Cluster System Block)



- Data displayed by
\$ SHOW CLUSTER/CONTINUOUS
ADD MEMBERS/ALL
- One CSB for each OpenVMS node this node is aware of

Connection Manager Data Structures: CSB (Cluster System Block)



- Data of interest:
 - Cluster System ID (CSID) of this node
 - Status of this node (e.g. cluster member or not)
 - State of our connection to this node
 - e.g. Newly discovered node, normal connection, communications lost but still within RECNXINTERVAL wait period, lost connection too long so node is no longer in cluster
 - This node's private values for VOTES, EXPECTED_VOTES, QDSKVOTES, LOCKDIRWT
 - Cluster software version, and bit-mask of capabilities this node has (to allow support of cluster rolling upgrades despite adding new features)

Connection Manager Data Structures: CSV (Cluster System Vector)



- The CSV is effectively a list of all systems
- The Cluster System ID (CSID) acts as an index into the CSV
- Incrementing sequence numbers within CSIDs allows re-use of CSV slots without confusion
 - This is similar to how the sequence number is incremented before a File ID is re-used

Distributed Lock Manager



- The Lock Manager provides mechanisms for coordinating access to physical devices, both for exclusive access and for various degrees of sharing

Distributed Lock Manager



- System services \$ENQ and \$DEQ allow new lock requests, conversion of existing locks to different modes (or degrees of sharing), and release of locks, while \$GETLKI allows the lookup of lock information

Distributed Lock Manager



- Physical resources are mapped to symbolic resource names, and locks are taken out and released on these symbolic resources to control access to the real resources

Distributed Lock Manager



- Physical resources that the Lock Manager is used to coordinate access to include:
 - Tape drives
 - Disks
 - Files
 - Records within a fileas well as internal operating system cache buffers and so forth

Mapping symbolic lock resource names to real entities



- Techniques for mapping resource names to lock types
 - Common prefixes:
 - SYS\$ for OpenVMS executive
 - F11B\$ for XQP, file system
 - RMS\$ for Record Management Services
 - See Appendix H in Alpha V1.5 IDSM or Appendix A in Alpha V7.0 version

Resource names

- Example: Device Lock
 - Resource name format is
 - “SYS\$” {Allocation-Class Device Name}
 - Extract device name from resource name

Resource names

- Example: XQP File Serialization Lock
 - Resource name format is
 - “F11B\$s” {Lock Basis}
 - Parent lock is the Volume Allocation Lock “F11B\$v” {Lock Volume Name}
 - Calculate File ID from Lock Basis
 - Lock Basis is RVN and File Number from File ID (ignoring Sequence Number), packed into 1 longword
 - Identify disk volume from parent resource name

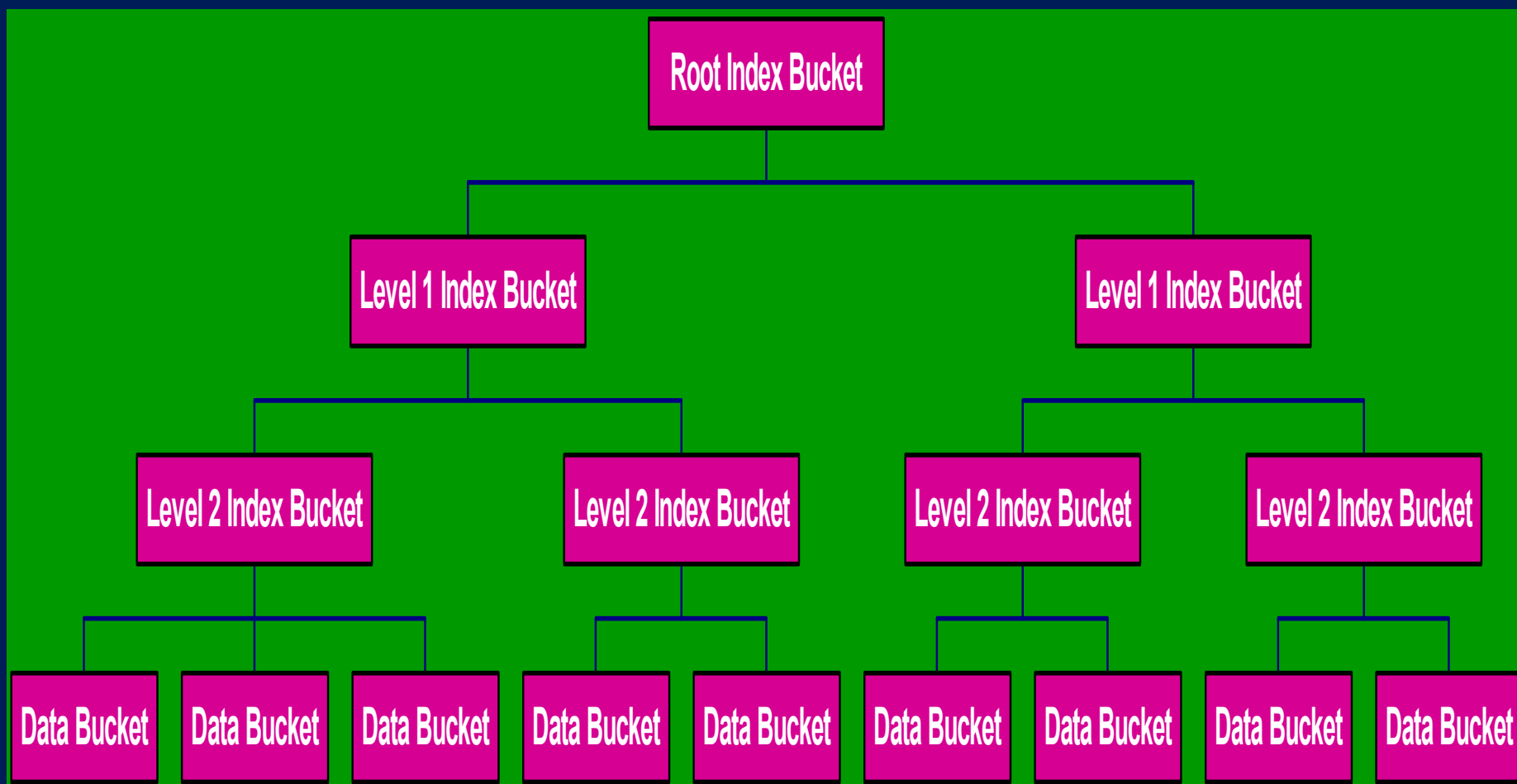
Resource names

- Identifying file from File ID
 - Look at file headers in Index File to get filespec:
 - Can use DUMP utility to display file header (from Index File)
 - `$ DUMP /HEADER /IDENTIFIER=(file_id) /BLOCK=COUNT=0 disk:[000000]INDEXF.SYS`
 - Follow directory backlinks to determine directory path
 - (or use LIB\$FID_TO_NAME routine to do all this, if sequence number can be obtained)

Resource names

- Example: RMS lock tree for an RMS indexed file:
 - Resource name format is
 - “RMS\$” {File ID} {Flags byte} {Lock Volume Name}
 - Identify filespec using File ID
 - Flags byte indicates private (1) or shared (2) disk mount
 - Pick up disk volume name
 - This is label as of time disk was mounted
- Sub-locks are used for buckets and records within the file

Internal Structure of an RMS Indexed File





RMS Data Bucket Contents

Data Bucket

Data Record	Data Record
Data Record	Data Record
Data Record	Data Record
Data Record	Data Record
Data Record	Data Record



RMS Indexed File Bucket and Record Locks

- Sub-locks of RMS File Lock
 - Have to look at Parent lock to identify file
- Bucket lock:
 - 4 bytes: VBN of first block of the bucket
- Record lock:
 - 8 bytes: Record File Address (RFA) of record

Distributed Lock Manager



- The Distributed Lock Manager works closely with the Connection Manager during State Transitions, when a node joins or leaves the cluster:
 - Any locks which were held by a node before it left the cluster are automatically freed
- We'll look at this in more detail later

Lock Request Latencies

- Latency depends on several things:
 - Directory lookup needed or not
 - Local or remote directory node
 - \$ENQ or \$DEQ operation
 - Local or remote lock master
 - If remote, type of interconnect

Directory Lookups

- This is how OpenVMS finds out which node is the lock master
- Only needed for 1st lock request on a particular resource tree on a given node
 - Resource Block (RSB) remembers master node CSID
- Basic conceptual algorithm: Hash resource name and index into lock directory vector, which has been created based on LOCKDIRWT values

Directory Lookups

- Conceptual algorithm:
 - Use Lock Directory Weight values to create table of node IDs (one row per unit of LOCKDIRWT)
 - No entry for node with LOCKDIRWT=0
 - Hash resource name to produce index into table to pick directory node

Directory Lookups

- “Ask” directory node which node is the lock master
 - by optimistically sending lock request there
 - 3 possible responses:
 - 1) No other node has interest -- you’re now the lock master
 - 2) Directory node also happens to be the lock master -- here’s the response to your lock request
 - 3) Another node is the lock master -- go ask them

Performance Hints

- Avoid \$DEQing a lock which will be re-acquired later
 - Instead, convert to Null lock, and convert back up later
 - Or, create locks as sub-locks of a parent lock that remains held
- This avoids directory lookups
 - and also avoids losing the activity counts in the root RSB used for lock-remastering decisions

Performance Hints



- For lock trees which come and go often:
 - A separate program could take out a Null lock on the root resource on each node at boot time, and just hold it “forever”
 - This avoids lock directory lookup operations for that tree

\$ENQueue

- New lock request (0-2 round trips)
 - No off-node traffic if this node is lock master
 - 1 round trip if:
 - no other node has interest, or
 - directory node is also lock master, or
 - local node is directory node
 - 2 round trips if:
 - directory node is not also the lock master node

\$ENQueue

- Conversion or sub-lock (0 or 1 round-trip)
 - No off-node traffic if this node is lock master
 - 1 round trip to lock master (except 2PC)
 - RSB already contains CSID of lock master node, so we never need to do a directory lookup

\$DEQueue

- 1-way message
- No response expected
- Client doesn't wait
- SCS message guarantee ensures eventual arrival
 - SCS credits may limit number of these in-flight at once

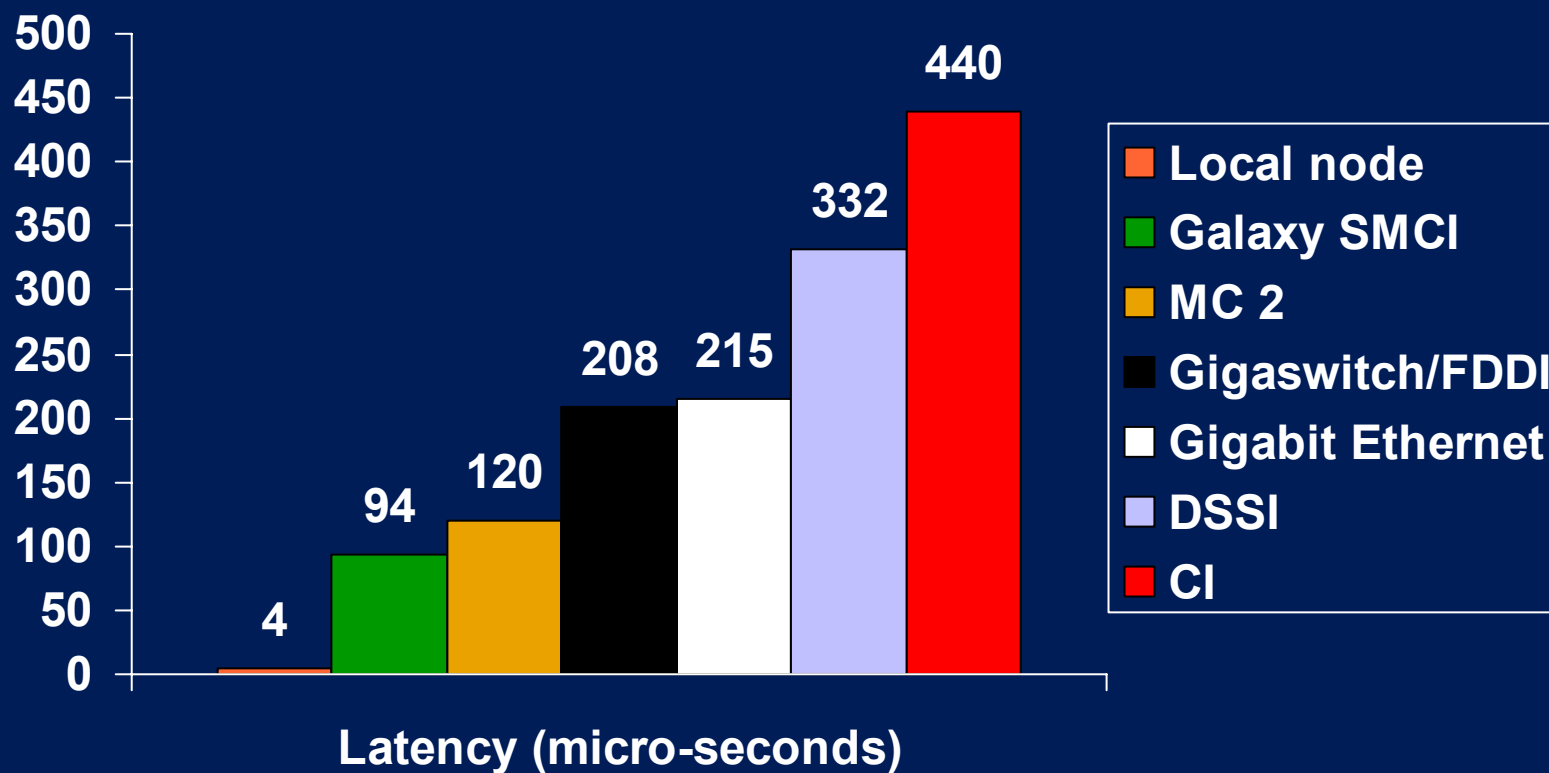
Lock Request Latencies

- Local requests are fastest
- Remote requests are significantly slower:
 - Code path ~20 times longer
 - Interconnect also contributes latency
 - Total latency up to 2 orders of magnitude higher than local requests

Lock Request Latencies

- I used Roy Davis' LOCKTIME programs to measure latency of lock requests locally and across different interconnects
- LOCKTIME algorithm:
 - Take out 5000 locks on remote node, making it lock master for each of 5000 lock trees (requires ENQLM > 5000)
 - Do 5000 \$ENQs, lock converts, and \$DEQs, and calculate average latencies for each
 - Lock conversion request latency is equivalent to round-trip time between nodes

Lock Request Latencies



Lock Mastership (Resource Mastership) concept



- One lock master node is selected by OpenVMS for a given resource tree at a given time
- Different resource trees may have different lock master nodes

Lock Mastership (Resource Mastership) concept



- Lock master remembers all locks on a given resource tree for the entire cluster
- Each node holding locks also remembers the locks it is holding on resources, to allow recovery if lock master node dies

Lock Mastership

- Lock mastership node may change for various reasons:
 - Lock master node goes down -- new master must be elected
 - OpenVMS may move lock mastership to a “better” node for performance reasons
 - LOCKDIRWT imbalance found, or
 - Activity-based Dynamic Lock Remastering
 - Lock Master node no longer has interest

Lock Mastership

- Lock master selection criteria:
 - Interest
 - Only move resource tree to a node which is holding at least some locks on that resource tree
 - Lock Directory Weight (LOCKDIRWT)
 - Move lock tree to a node with interest and a higher LOCKDIRWT
 - Activity Level
 - Move lock tree to a node with interest and a higher average activity level

How to measure locking activity

- OpenVMS keeps counters of lock activity for each resource tree
 - but not for each of the sub-resources
- So you can see the lock rate for an RMS indexed file, for example
 - but not for individual buckets or records within that file
- SDA extension LCK in OpenVMS V7.2-2 and above can show lock rates, and even trace all lock requests if needed

Measuring Lock Activity

- \$ANALYZE/SYSTEM
 - New SDA extension LCK for lock tracing in OpenVMS 7.2-2 and above
 - SDA> LCK !Shows help text with command summary
 - Can display various additional lock manager statistics:
 - SDA> LCK STATISTIC !Shows lock manager statistics
 - Can show busiest resource trees by lock activity rate:
 - SDA> LCK SHOW ACTIVE !Shows lock activity
 - Can trace lock requests:
 - SDA> LCK LOAD !Load the debug execlet
 - SDA> LCK START TRACE !Start tracing lock requests
 - SDA> LCK STOP TRACE !Stop tracing
 - SDA> LCK SHOW TRACE !Display contents of trace buffer
 - Can even trigger remaster operations:
 - SDA> LCK REMASTER !Trigger a remaster operation

Lock Remastering

- Circumstances under which remastering occurs, and does not:
 - LOCKDIRWT values
 - OpenVMS tends to remaster to node with higher LOCKDIRWT values, never to node with lower LOCKDIRWT
 - Shifting initiated based on activity counters in root RSB
 - PE1 parameter being non-zero can prevent movement or place threshold on lock tree size
 - Shift if existing lock master loses interest

Lock Remastering

- OpenVMS rules for dynamic remastering decision based on activity levels:
 - assuming equal LOCKDIRWT values
 - 1) Must meet general threshold of at least 80 lock requests so far (LCK\$GL_SYS_THRSH)
 - 2) New potential master node must have at least 10 more requests per second than current master (LCK\$GL_ACT_THRSH)

Lock Remastering

- OpenVMS rules for dynamic remastering:
 - 3) Estimated cost to move (based on size of lock tree) must be less than estimated savings (based on lock rate)
 - except if new master meets criteria (2) for 3 consecutive 8-second intervals, cost is ignored
 - 4) No more than 5 remastering operations can be going on at once on a node (LCK\$GL_RM_QUOTA)

Lock Remastering

- OpenVMS rules for dynamic remastering:
 - 5) If PE1 on the current master has a negative value, remastering trees off the node is disabled
 - 6) If PE1 has a positive, non-zero value on the current master, the tree must be smaller than PE1 in size or it will not be remastered

Lock Remastering

- Implications of dynamic remastering rules:
 - LOCKDIRWT must be equal for lock activity levels to control choice of lock master node
 - PE1 can be used to control movement of lock trees OFF of a node, but not ONTO a node
 - RSB stores lock activity counts, so even high activity counts can be lost if the last lock is DEQueued on a given node and thus the RSB gets deallocated

Lock Remastering

- Implications of dynamic remastering rules:
 - With two or more large CPUs of equal size running the same application, lock mastership “thrashing” is not uncommon:
 - 10 more lock requests per second is not much of a difference when you may be doing 100s or 1,000s of lock requests per second
 - Whichever new node becomes lock master may then see its own lock rate slow somewhat due to the remote lock request workload

Lock Remastering

- Lock mastership thrashing can result in user-visible delays
 - Lock operations on a tree are stalled during a remaster operation
 - Locks and Resources were sent over 1 per SCS message
 - Remastering large lock trees could take a long time
 - e.g. 10 to 50 seconds for 15K lock tree size, prior to 7.2-2
 - Improvement in OpenVMS in version 7.2-2 and above gives very significant performance gain
 - by using 64 Kbyte block data transfers instead of sending 1 SCS message per RSB or LKB

How to Detect Lock Mastership Thrashing

- Detection of remastering activity
 - MONITOR RLOCK in 7.3 and above (not 7.2-2)
 - SDA> SHOW LOCK/SUMMARY in 7.2 and above
 - Change of mastership node for a given resource
 - Check message counters under SDA:
 - SDA> EXAMINE PMS\$GL_RM_RBLD_SENT
 - SDA> EXAMINE PMS\$GL_RM_RBLD_RCVD
 - Counts which increase suddenly by a large amount indicate remastering of large tree(s)
 - SENT: Off of this node
 - RCVD: Onto this node
 - See example procedures WATCH_RBLD.COM and RBLD.COM

SDA> SHOW LOCK/SUMMARY



```
• $ analyze/system

• OpenVMS (TM) Alpha system analyzer

• SDA> show lock/summary
• ...
• Lock Manager Performance Counters:
• -----
• ...
• Lock Remaster Counters:
• Tree moved to this node 0
• Tree moved to another node 0
• Tree moved due to higher Activity 0
• Tree moved due to higher LOCKDIRWT 0
• Tree moved due to Single Node Locks 0
• No Quota for Operation 0
• Proposed New Manager Declined 0
• Operations completed 0
• Remaster Messages Sent 0
• Remaster Messages Received 0
• Remaster Rebuild Messages Sent 0
• Remaster Rebuild Messages Received 0
```

How to Prevent Lock Mastership Thrashing



- Unbalanced node power
- Unequal workloads
- Unequal values of LOCKDIRWT
- Non-zero values of PE1

Impact of Non-zero PE1 Values



- Concern: Locking down remastering with PE1 (to avoid lock mastership thrashing) can result in sub-optimal lock master node selections over time

Mitigating Impact of Non-zero PE1 Values



- Possible ways of mitigating side-effects of preventing remastering using PE1:
 - Adjust PE1 value as high as you can without producing noticeable delays
 - Upgrade to 7.2-2 or above for more-efficient remastering
 - Set PE1 to 0 for short periods, periodically

Deadlock searches

- The OpenVMS Distributed Lock Manager automatically detects lock deadlock conditions, and generates an error to one of the programs causing the deadlock

Deadlock searches

- Deadlock searches can take lots of time and interrupt-state CPU time
- DECps Performance Analysis report can identify when these are occurring
- DEADLOCK_WAIT parameter controls how long we wait before starting a deadlock search

Interrupt-state/stack saturation

- Too much lock mastership, MSCP-serving, etc. workload can saturate the primary CPU on a node
- See with `MONITOR MODES/CPU=n/ALL`
 - where “n” is the number of the Primary CPU as shown by `$SHOW CPU`

Interrupt-state/stack saturation

- FAST_PATH:
 - Can shift interrupt-state workload off primary CPU in SMP systems
 - IO_PREFER_CPUS value of an even number avoids sending interrupts from FAST_PATH devices to the Primary CPU
 - Consider limiting interrupts to a subset of non-primary CPU
 - FAST_PATH for MC “probably never”
 - FAST_PATH for SCSI and FC is in 7.3 and above
 - FAST_PATH for LANs (e.g. FDDI & Ethernet) in 7.3-2
 - Even with FAST_PATH enabled, the Primary CPU still received the device interrupt, but handed it off immediately via an inter-processor interrupt
 - 7.3-1 allows FAST_PATH interrupts to bypass Primary CPU entirely and go directly to a non-primary CPU on hardware platforms which support this

Dedicated-CPU Lock Manager

- With 7.2-2 and above, you can choose to dedicate a CPU to do lock management work. This may help reduce MP_SYNC time.
- LCKMGR_MODE parameter:
 - 0 = Disabled
 - >1 = Enable if at least this many CPUs are running
- LCKMGR_CPUID parameter specifies which CPU to dedicate to LCKMGR_SERVER process

Distributed Lock Manager Data Structures: RSB (Resource Block)



- Listhead for Root Resource Blocks is LCK\$GQ_RRSFL / BL
- Data of interest:
 - Resource name and length
 - UIC Group (0 means System)
 - Access Mode (Kernel, Executive, Supervisor, User)
 - Resource Master (aka Lock Master) node's CSID (0=local mastership)
 - Count of locks on this resource
 - Parent resource block, and depth within tree (0 for a Root RSB)
 - Count and queue of sub-resources, if any
 - Hash value for lookups via resource hash table
 - Activity counters used for dynamic lock remastering

Distributed Lock Manager Data Structures: Resource Hash Table



- Pointer is LCK\$GQ_HASHTBL
- Helps look up RSBs faster than by comparing resource names as text
- Size is determined by SYSGEN parameter RESHASHTBL
- Each table entry is a chain of RSBs with a given hash value

Distributed Lock Manager Data Structures: LKB (Lock Block)



- Data of interest:
 - Pointer to associated resource block (RSB)
 - Lock mode (K/E/S/U)
 - Process ID
 - State (e.g. granted, waiting, in conversion queue)
 - Parent lock, if any
 - Number of sub-locks
 - Lock ID of this lock
 - CSID of lock master node, and remote Lock ID there

Distributed Lock Manager Data Structures: Lock ID Table



- Pointer is LCK\$GQ_IDTBL
- Keeps track of all locks by Lock ID
- As with CSIDs and File IDs, sequence number portion of Lock ID is incremented to avoid any possibility of confusion upon subsequent re-use of a particular entry in the Lock ID Table

Cluster State Transition Topics



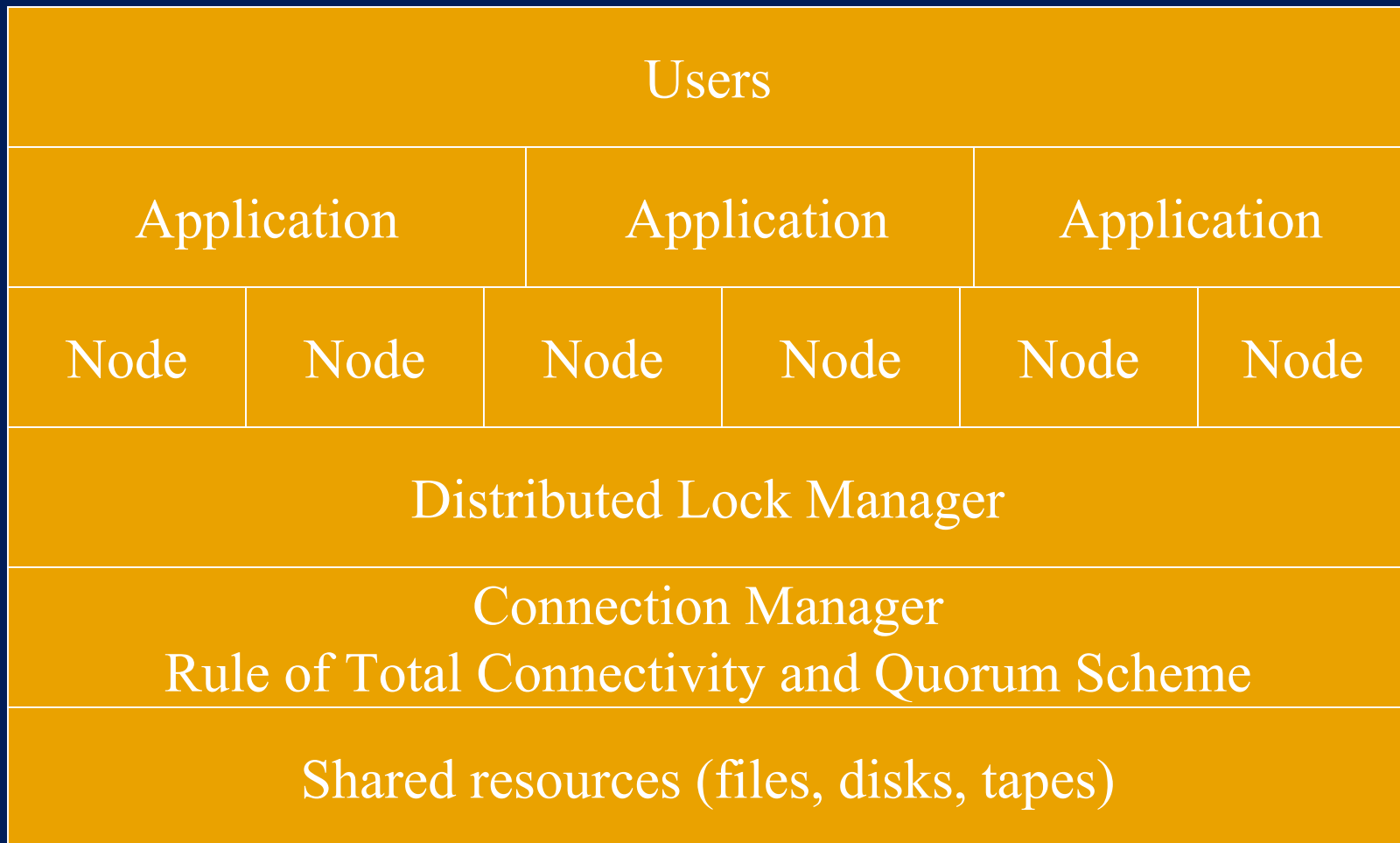
- Purpose of cluster state transitions
- The Distributed Lock Manager, and its role in State Transitions
- Things which trigger state transitions, and how to avoid them
- Failures and failure detection mechanisms
- Reconnection Interval

Cluster State Transition Topics

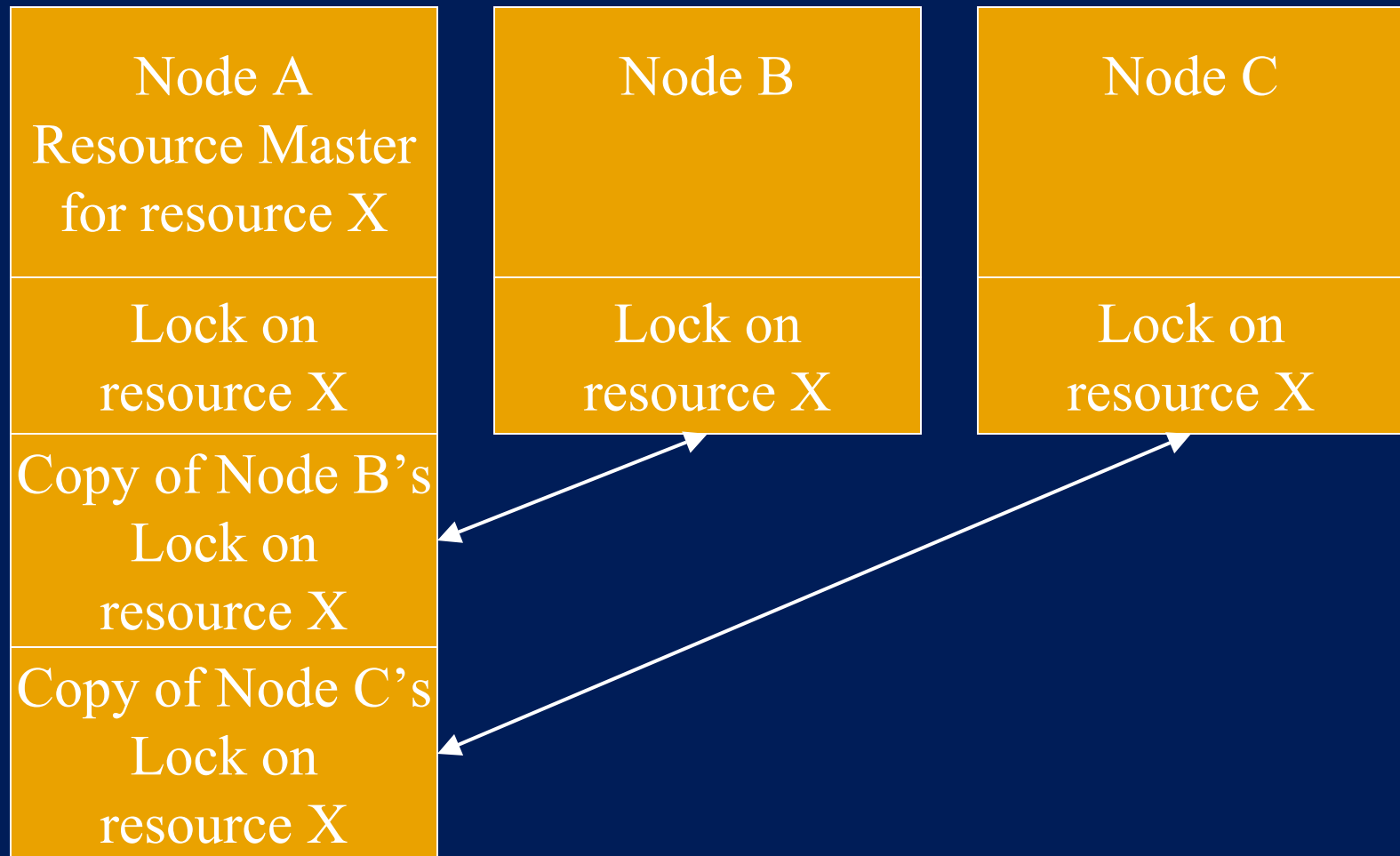


- Effects of a state transition
- Sequence of events in a state transition
- State-transition-related issues
- Minimizing impact of state transitions

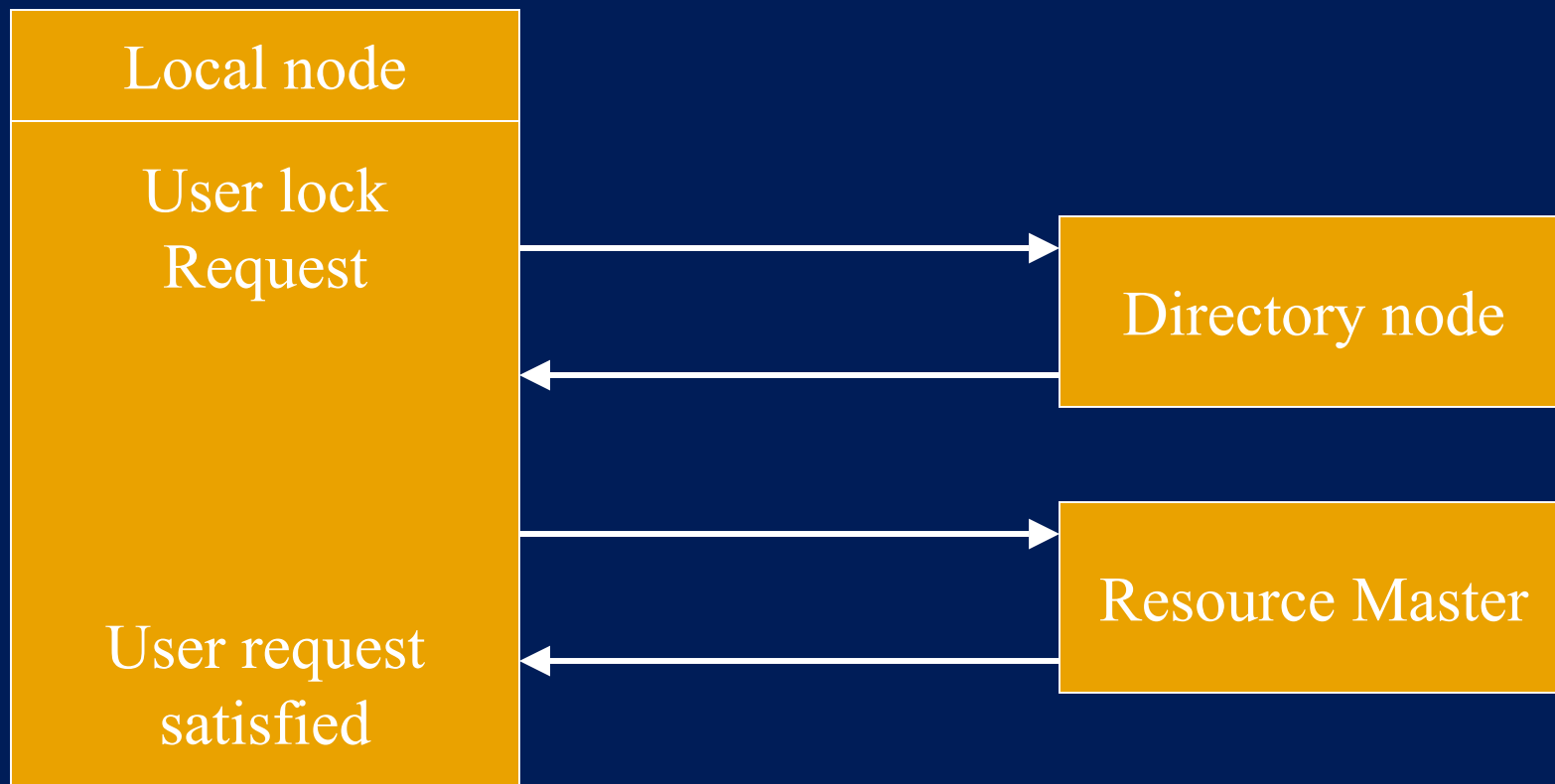
Foundation for Shared Access



Distributed Locks



Lock Directory Lookup

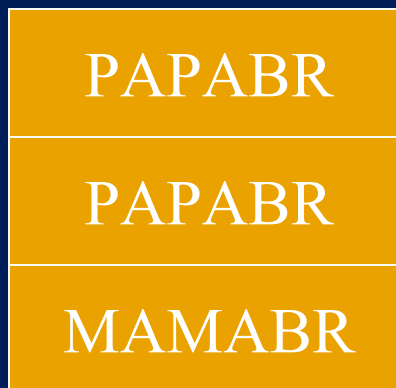


Lock Directory Vector



Big node:	PAPABR	Lock Directory Weight = 2
Middle-sized node:	MAMABR	Lock Directory Weight = 1
Satellite node:	BABYBR	Lock Directory Weight = 0

Resulting lock directory vector:



Implications for State Transitions



Node leaving or joining	Effect on directory vector
LOCKDIRWT zero:	No effect
LOCKDIRWT non-zero:	Must change

Lock Rebuild Types



- Merge
- Partial
- Directory
- Full

Lock Rebuild Types



- Merge:
 - Enter already-held lock tree into directory on directory node
 - Lock requests not stalled
- Partial:
 - Release locks held by departed node, and remaster any resource trees it was resource master for

Lock Rebuild Types



- Directory:
 - Do what a Partial rebuild does (clean up after departed node), and also rebuild the directory information
- Full:
 - Throw away all master copies of locks, select new resource masters, and then re-acquire locks, rebuilding directory information in the process

Lock Rebuild Types



Node's LOCKDIRWT	State change	Type of rebuild
Zero	Node joining	Merge
Zero	Node leaving	Partial
Non-zero	Node joining or leaving	Directory
N/A	Cluster formation	Full

State Transition Triggers

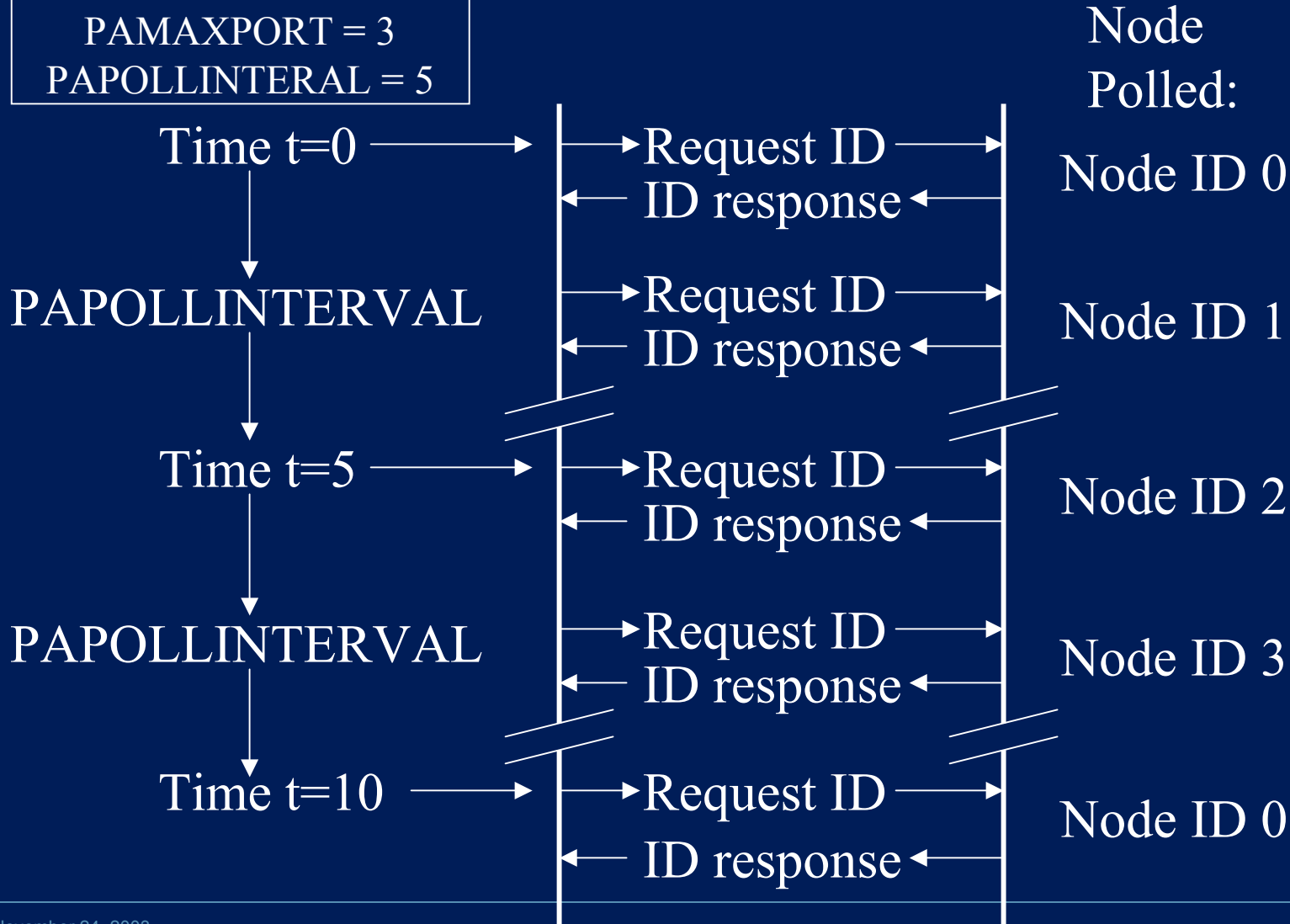


- A state transition is needed when:
 - A new member is added to the cluster
 - An existing member is removed from the cluster
 - An adjustment related to votes is needed:
 - shutdown with REMOVE_NODE option
 - quorum disk validation or invalidation
 - quorum is adjusted manually via:
 - IPC> Q routine
 - DECams/AM Quorum Fix via RMDRIVER
 - \$SET CLUSTER/EXPECTED_VOTES command in DCL

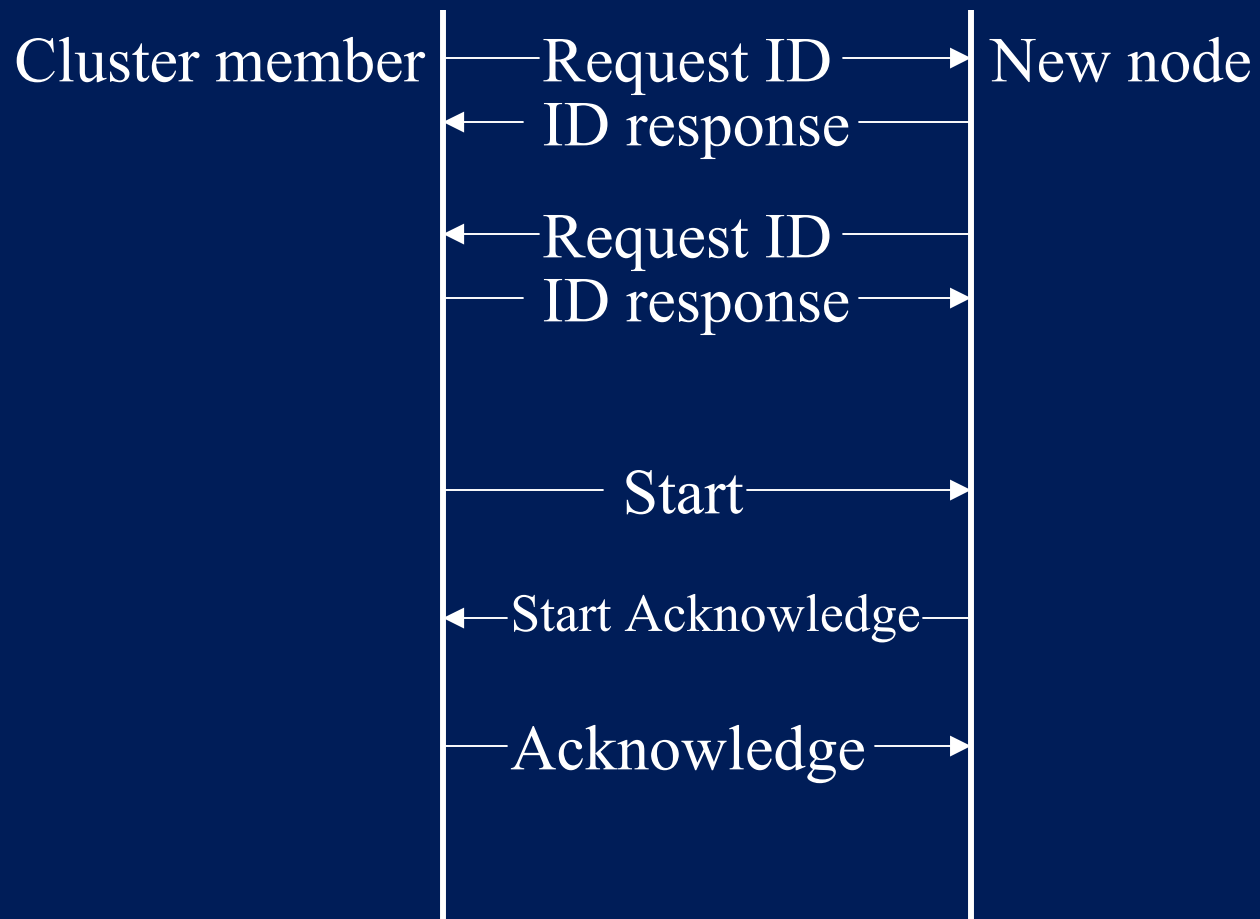
New-member detection on CI or DSSI



PANUMPOLL = 2
PAMAXPORT = 3
PAPOLLINTERVAL = 5



Virtual Circuit Formation on CI or DSSI



New-member detection on Ethernet or FDDI



Remote node

Local node

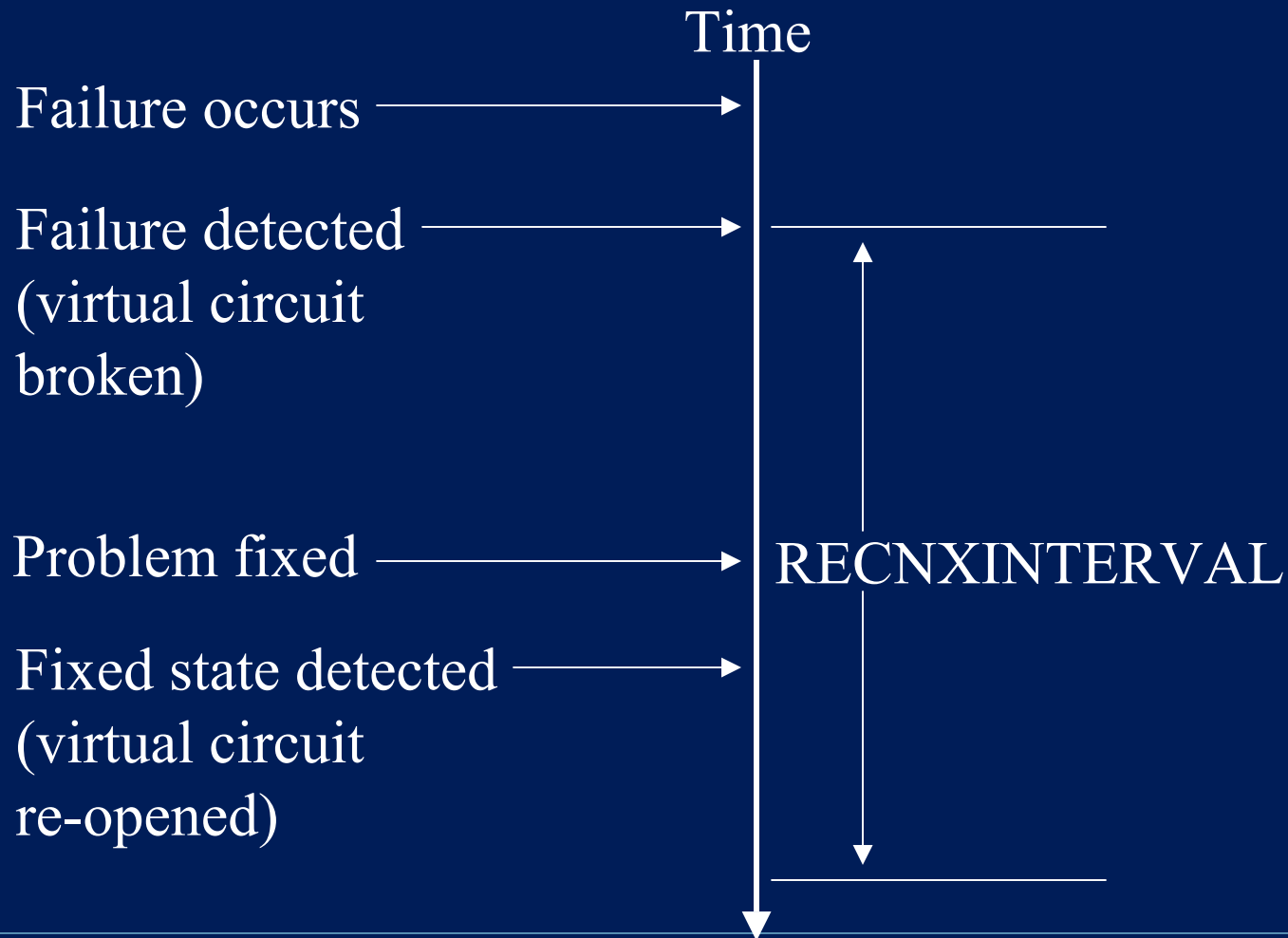


Failures

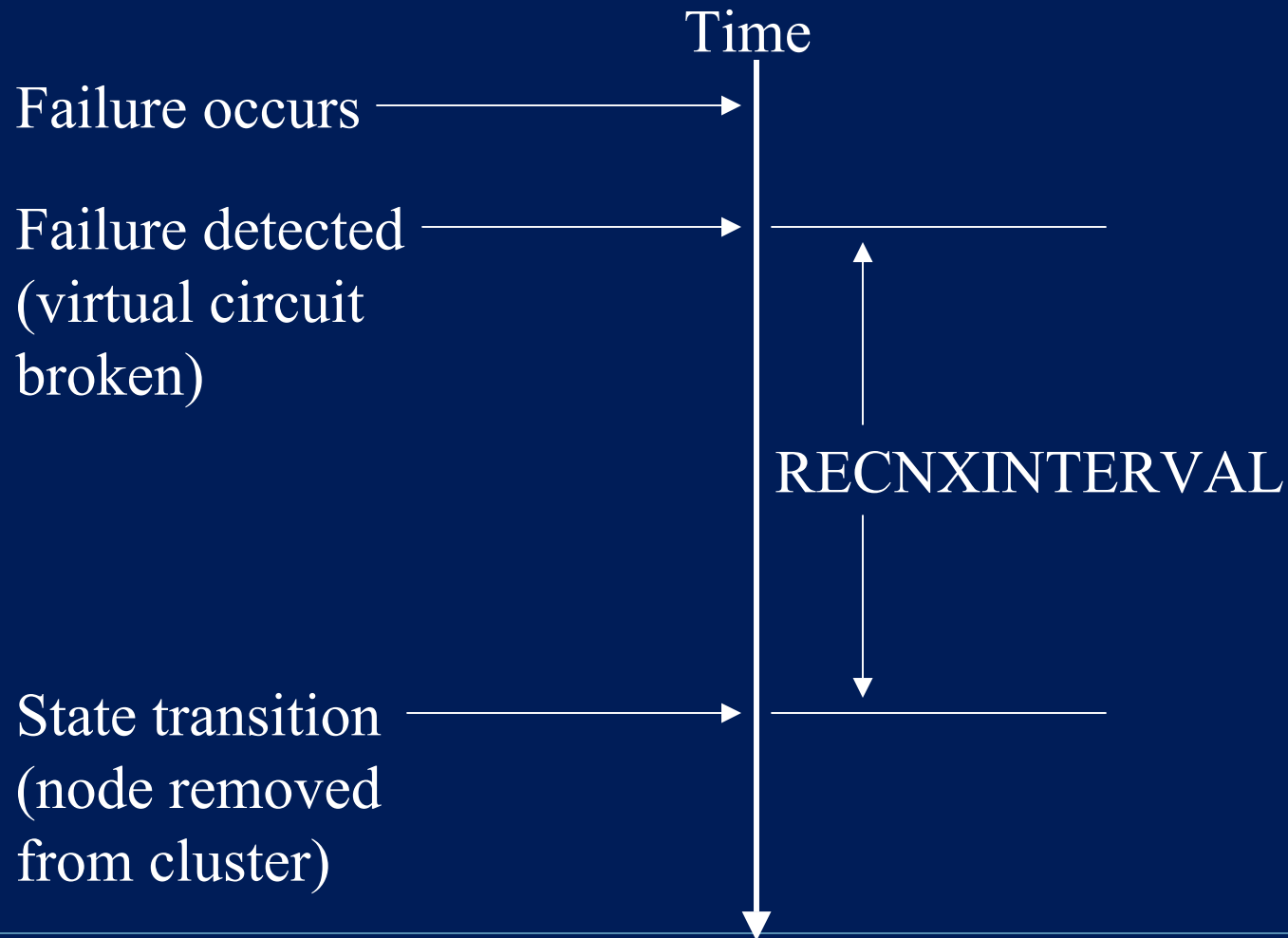


- Types of failures:
 - Node failure
 - Adapter failure
 - Interconnect failure
- Failure examples:
 - Power supply failure in network equipment
 - Operating system crash
 - Bridge reboot for firmware update, and spanning-tree reconfiguration which results

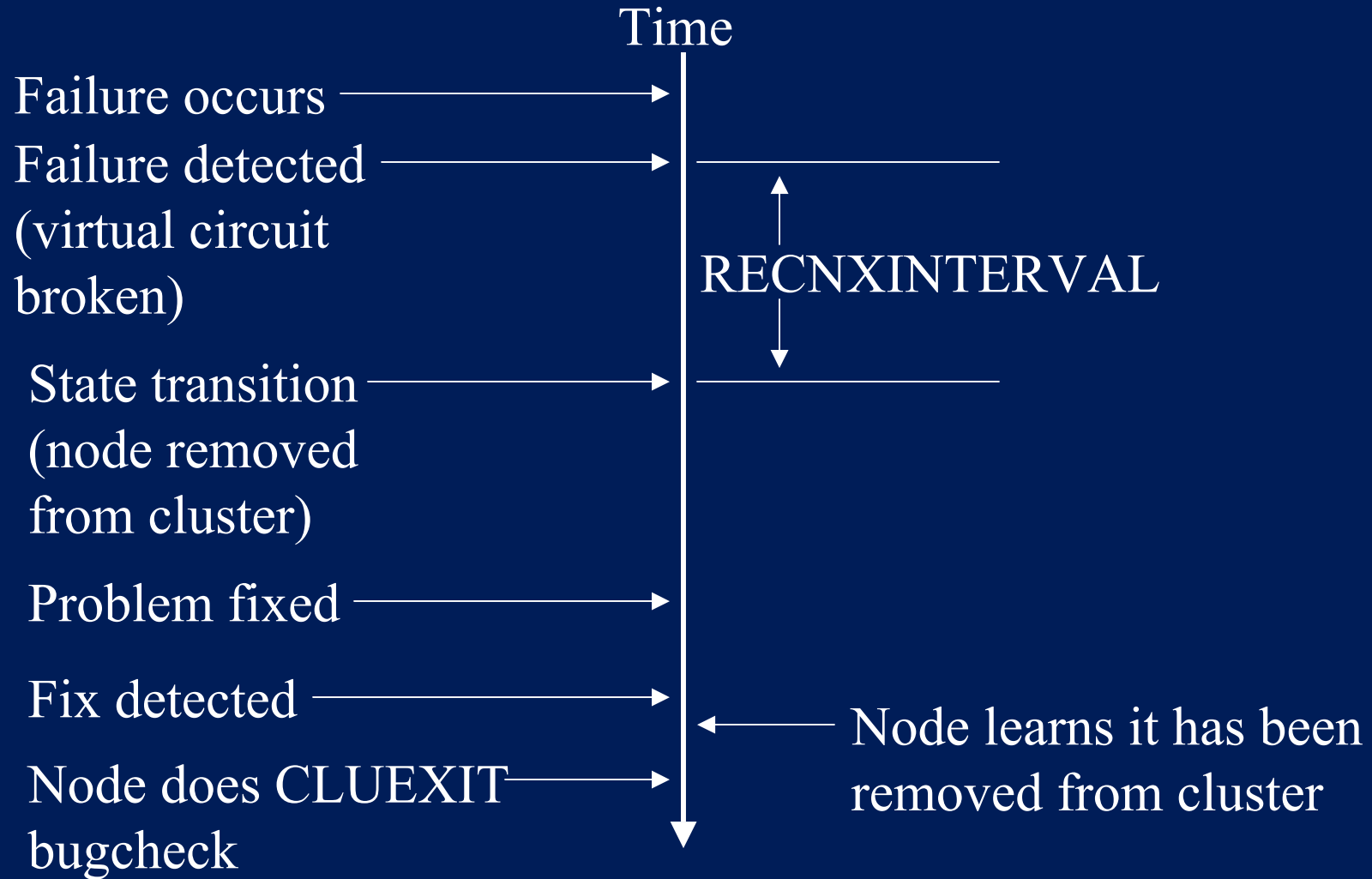
Failure and Repair/Recovery within Reconnection Interval



Hard Failure



Late Recovery

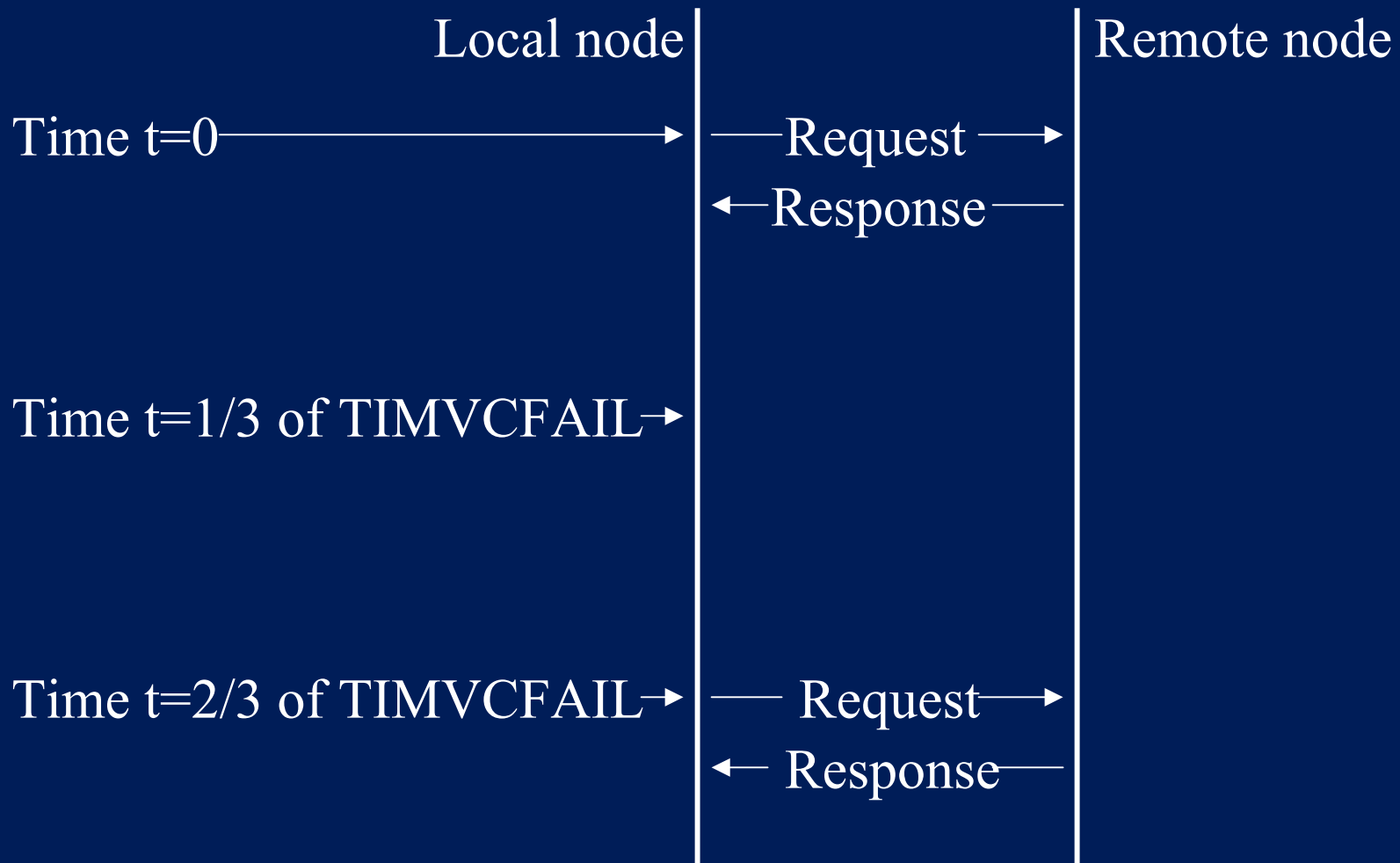


Failure Detection Mechanisms

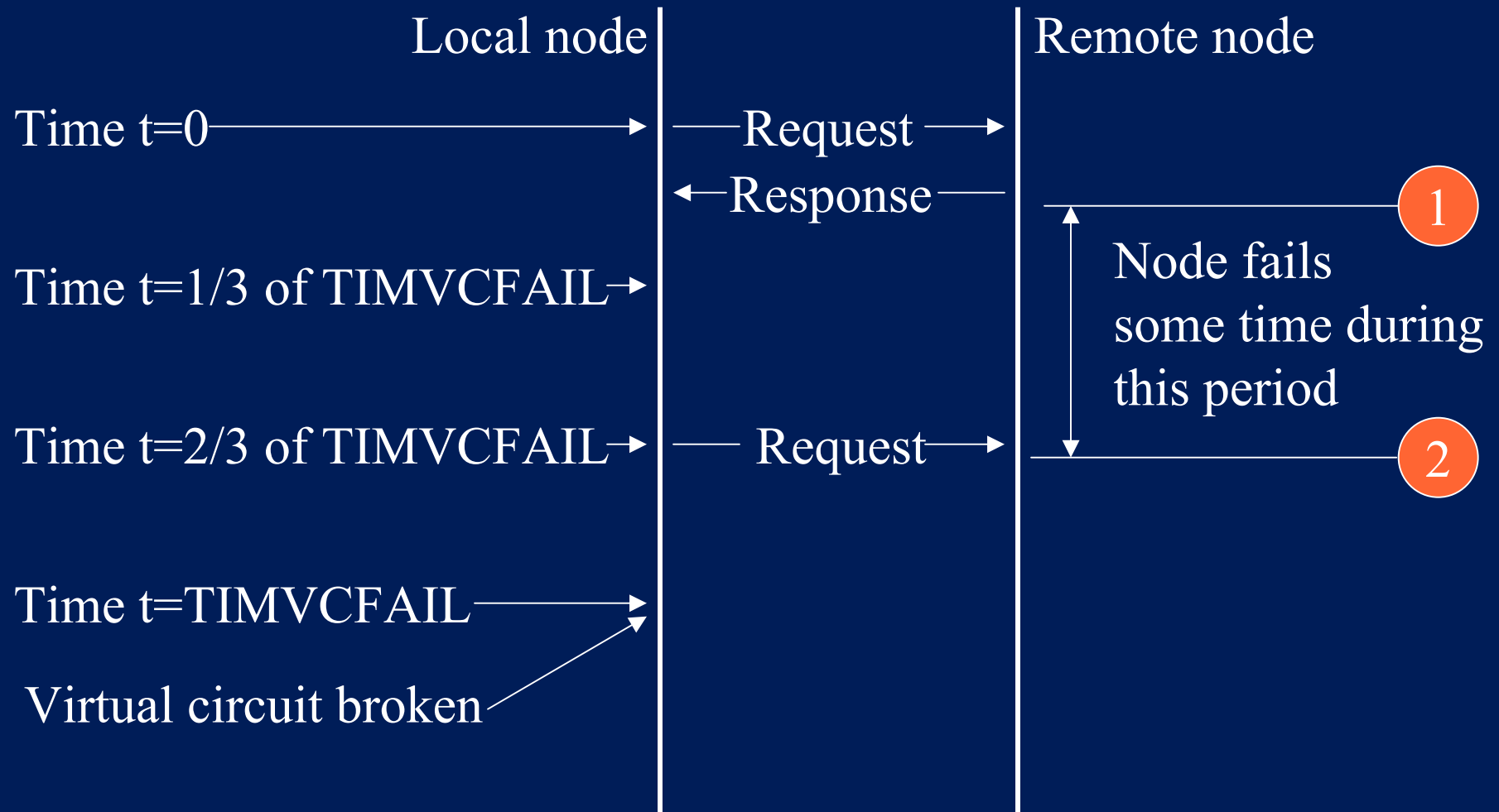


- Mechanisms to detect a node or communications failure
 - Last-Gasp Datagram
 - Periodic checking
 - Multicast Hello packets on LANs
 - Polling on CI and DSSI
 - TIMVCFAIL check

TIMVCFAIL Mechanism



TIMVCFAIL Mechanism



Sequence of events During a State Transition



- Determine new cluster configuration
 - If quorum is lost:
 - QUORUM capability bit removed from all CPUs
 - no process can be scheduled to run
 - Disks all put into mount verification
 - If quorum is not lost, continue...
- Rebuild lock database
 - Stall lock requests
 - I/O synchronization
 - Do rebuild work
 - Resume lock handling

Measuring State Transition Effects



- Determine the type of the last lock rebuild:

```
$ ANALYZE/SYSTEM
```

```
SDA> READ SYS$LOADABLE_IMAGES:SCSDEF
```

```
SDA> EVALUATE @(@CLU$GL_CLUB + CLUB$B_NEWRBLD_REQ) & FF
```

```
Hex = 00000002   Decimal = 2   ACP$V_SWAPPRV
```

- Rebuild type values:
 1. Merge (locking not disabled)
 2. Partial
 3. Directory
 4. Full

Measuring State Transition Effects



- Determine the duration of the last lock request stall period:

```
SDA> DEFINE TOFF = @(@CLU$GL_CLUB+CLUB$L_TOFF)
SDA> DEFINE TON = @(@CLU$GL_CLUB+CLUB$L_TON)
SDA> EVALUATE TON-TOFF
Hex = 0000026B   Decimal = 619   PDT$Q_COMQH+00003
```

Minimizing Duration of State Transitions



- Configurations issues:
 - Few (e.g. exactly 3) nodes
 - Quorum node; no quorum disk
 - Isolate cluster interconnects from other traffic
- Operational issues:
 - Avoid disk rebuilds on reboot
 - Let Last-Gasp work

System Parameters which affect State Transition Impact



- Failure detection time:
 - TIMVCFAIL can be set as low as 100 (1 second) as needed
- Reconnection interval after failure detection:
 - RECNXINTERVAL can be set as low as 1 second as needed

MSCP/TMSCP Servers



- Implement Mass Storage Control Protocol
- Provide access to disks [MSCP] and tape drives [TMSCP] for systems which do not have a direct connection to the device, through a node which does have direct access and has [T]MSCP Server software loaded

MSCP/TMSCP Servers



- MSCP disk and tape controllers (e.g. HSJ80, HSD30) also include a [T]MSCP Server, and talk the SCS protocol, but do not have a Connection Manager, so are not cluster members

Cluster Server process (CSP)



- Assists with cluster-wide operations that require process context on a remote node, such as:
 - Mounting and dismounting volumes:
 - \$MOUNT/CLUSTER and \$DISMOUNT/CLUSTER
 - \$BRKTHRU system service and \$REPLY command
 - \$SET TIME/CLUSTER
 - Distributed OPCOM communications
 - Interface between SYSMAN and SMISERVER on remote nodes
 - Cluster-Wide Process Services (CWPS)
 - Cluster-Wide Logical Names



Speaker Contact Info:

- Keith Parris
- E-mail: parris@encompasserve.org
- or keithparris@yahoo.com
- or Keith.Parris@hp.com
- Web: <http://encompasserve.org/~parris/>
- **and** <http://www.geocities.com/keithparris/>



i n v e n t