Module-5
Distributed Shared Memory (DSM)

Introduction

- Basically there are TWO IPC paradigms in DOS

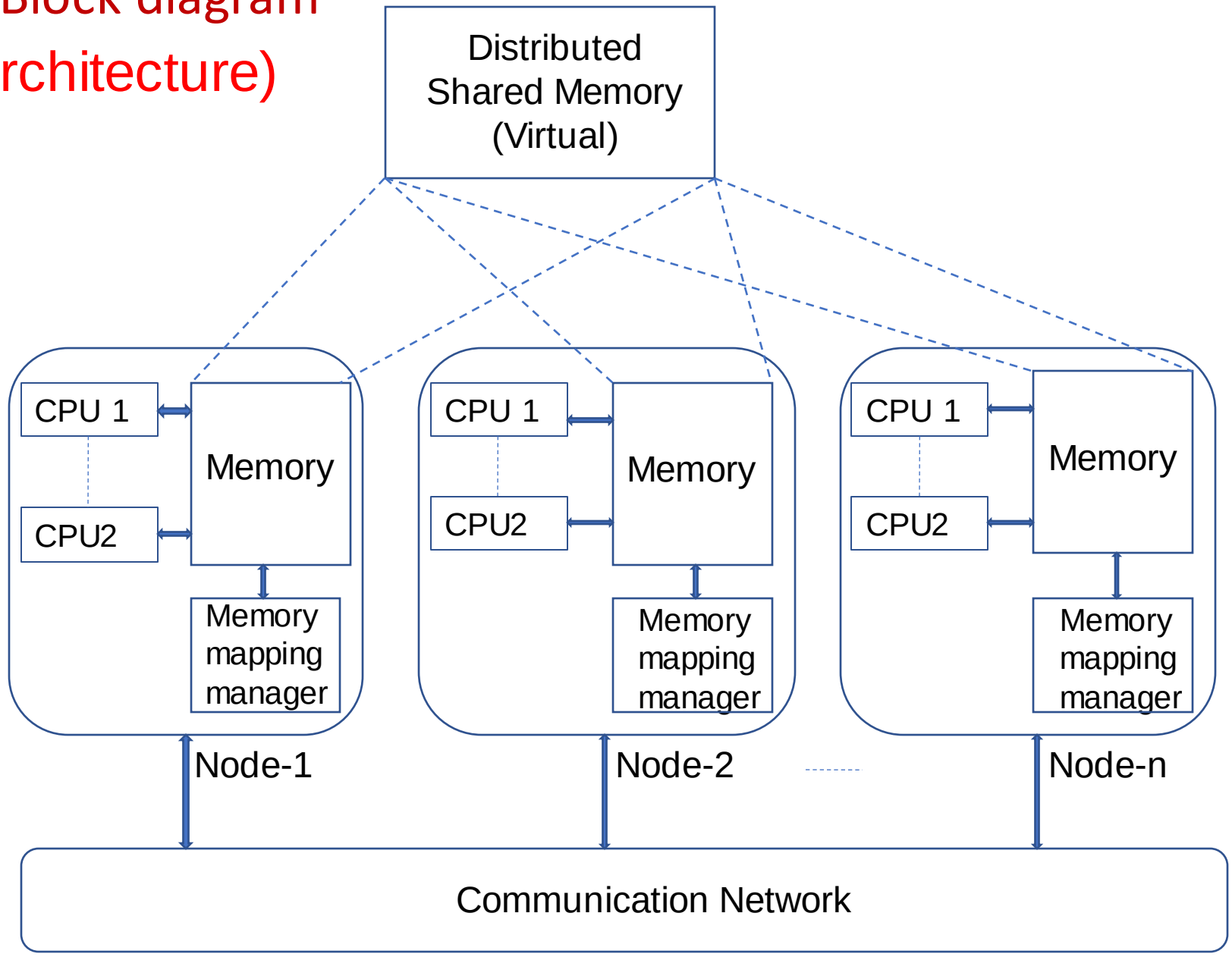
1. Message passing and RPC
2. Shared data Approach

Distributed Shared Memory (DSM)

- The DSM provides to processes in a system with a shared address space
- Processes use this address space in the same way as they use local memory
- Primitives used are
  1. data= Read(address)
  2. Write(address, data)
- The DSM in tightly coupled system is natural.
- In loosely coupled systems there is no physically shared memory available to support DSM paradigm.

The term DSM refers to the shared-memory paradigm applied to loosely coupled distributed-memory systems (Virtual Memory)

DSM Block diagram (Architecture)



DSM architecture

- DSM provides a virtual address space shared among processes on loosely coupled processors
- *DSM is basically an abstraction that integrates the local memory of different machines in a network environment into a single logical entity shared by cooperating processes executing on multiple sites.*
- The shared memory exists only virtually, hence it is also called (DSVM)
- In DSM each node of the system has one or more CPUs and a memory unit.
- The nodes are connected by a high-speed communication network.
- The DSM abstraction presents a large shared-memory space to the processors of all nodes.
- A software memory-mapping manager routine in each node maps the local memory onto the shared virtual memory.
- To facilitate the mapping operation, the shared-memory space is partitioned into blocks.
- The idea of data caching is used to reduce network latency
- The main memory of individual nodes is used to cache blocks of the shared-memory space.

DSM operation

when a process on a node wants to access some data from a memory block of the shared memory space

1. The local memory-mapping manager takes charge of its request.
2. If the memory block containing the accessed data is resident in the local memory, the request is satisfied.
3. Otherwise, a network block fault is generated and the control is passed to the operating system.
4. The OS then sends a message to the node on which the desired memory block is located to get the block.
5. The missing block is migrated from the remote node to the client process's node and the OS maps it into the application address space.
6. Data blocks keep migrating from one node to another on demand basis, but no communication is visible to the user processes.
7. Copies of data cached in local memory eliminate network traffic.
8. DSM allows *replication/migration* of data blocks

Real Time examples of DSMs

- Telecare, e.g. remote health care and specialist consultation
- E-learning
- Call center support
- Collaborative work, e.g. shared whiteboard or other collaborative editing systems
- Real-time Chat lets you chat with your friends and see what they are typing in real-time. The application uses the Distributed Shared Memory enabler to synchronize chat messages.

Design and Implementation issues of DSM

1. Granularity:

- Refers to the block size of a DSM system
- Unit of sharing data or unit of data transfer across network.
- Proper block size explores the granularity of parallelism and the amount of network traffic generated by network block faults.

2. Structure of shared-memory space

- Refers to the layout of the shared data in memory.
- Depending on the type of application that the DSM system is intended to support

3. Memory coherence and access synchronization

- In DSM system, the replication/copies of shared data is available in all nodes, coherence/consistency of accessing of data in particular node is very difficult.
- Concurrent access of shared data requires synchronization primitives such as semaphores, event count, lock etc.

4. Data location and access

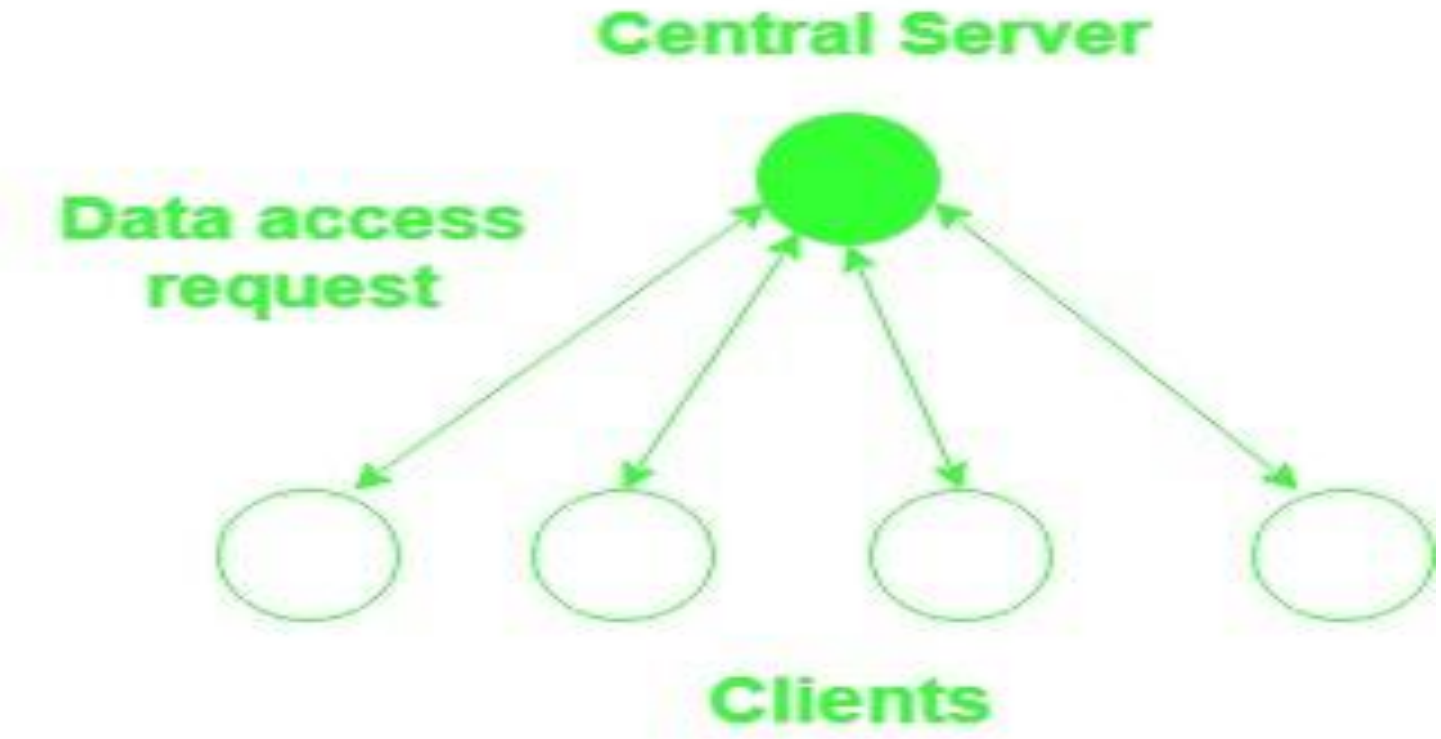
- To share data in DSM, it is necessary to locate and retrieve the data accessed by a user process

5. Replacement strategy
   - ❖ If the local memory of a node is full, the data block of local memory is replaced with the new data block.
   - ❖ Cache replacement strategy is necessary in DSM
6. Thrashing
   - ❖ In a DSM system, data blocks migrate between nodes on demand.
   - ❖ If two nodes compete for write access to a single data item, the corresponding data block is forced to transfer back and forth at such a high rate that no real work can get done.
   - ❖ The DSM which use a policy to avoid this situation.
7. Heterogeneity
   - ❖ The DSM system is designed to take care of the heterogeneity so that it functions properly with machines with different architectures.

Algorithms for implementing DSM

- **Central Server Algorithm**
- **Migration Algorithm**
- **Read Replication Algorithm**
- **Full Replication Algorithm**

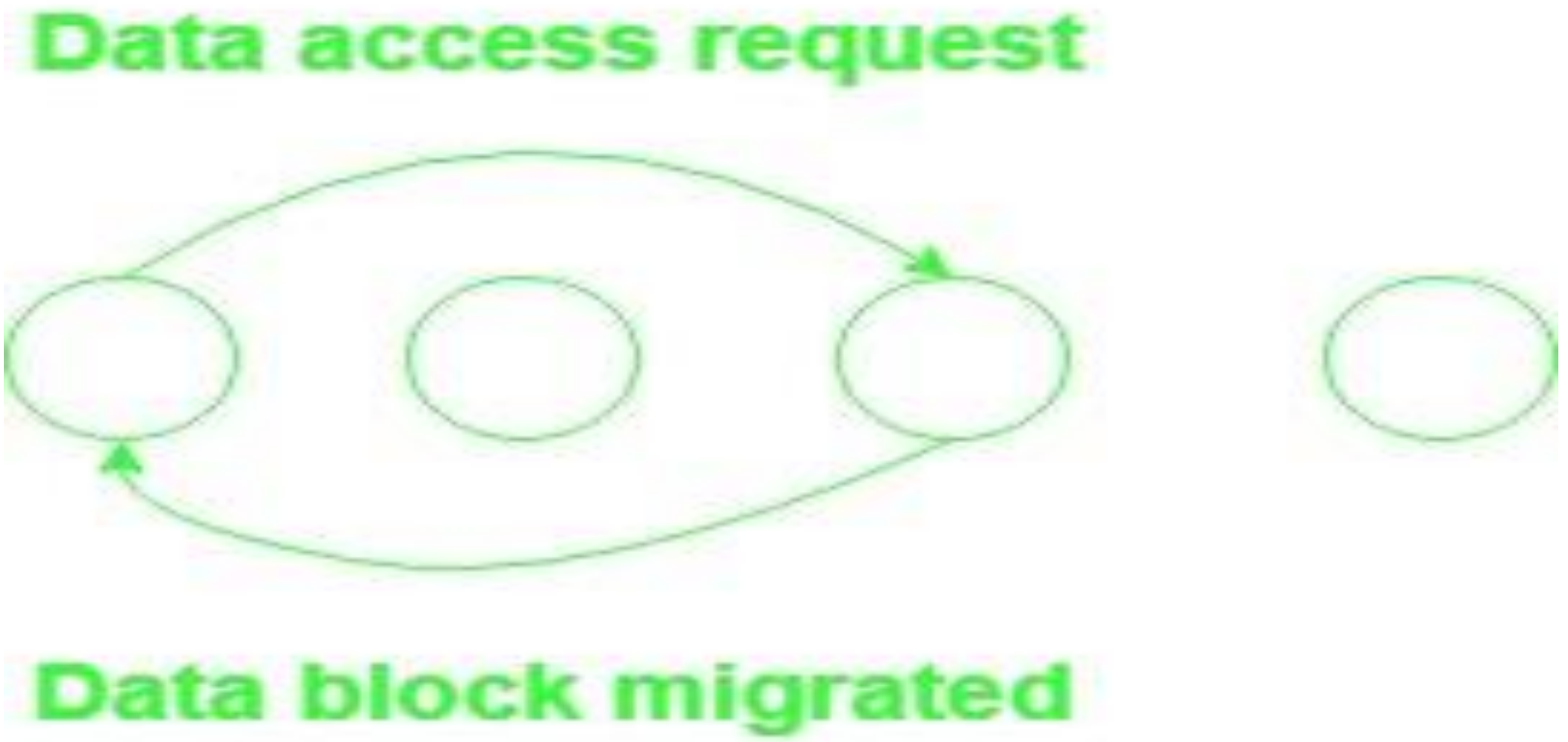
Central Server Algorithm



Central Server Algorithm

- In this, a central server maintains all shared data. It services read requests from other nodes by returning the data items to them and write requests by updating the data and returning acknowledgement messages.
- Time-out can be used in case of failed acknowledgement while sequence number can be used to avoid duplicate write requests.
- It is simpler to implement but the central server can become bottleneck and to overcome this shared data can be distributed among several servers.
- This distribution can be by address or by using a mapping function to locate the appropriate server.

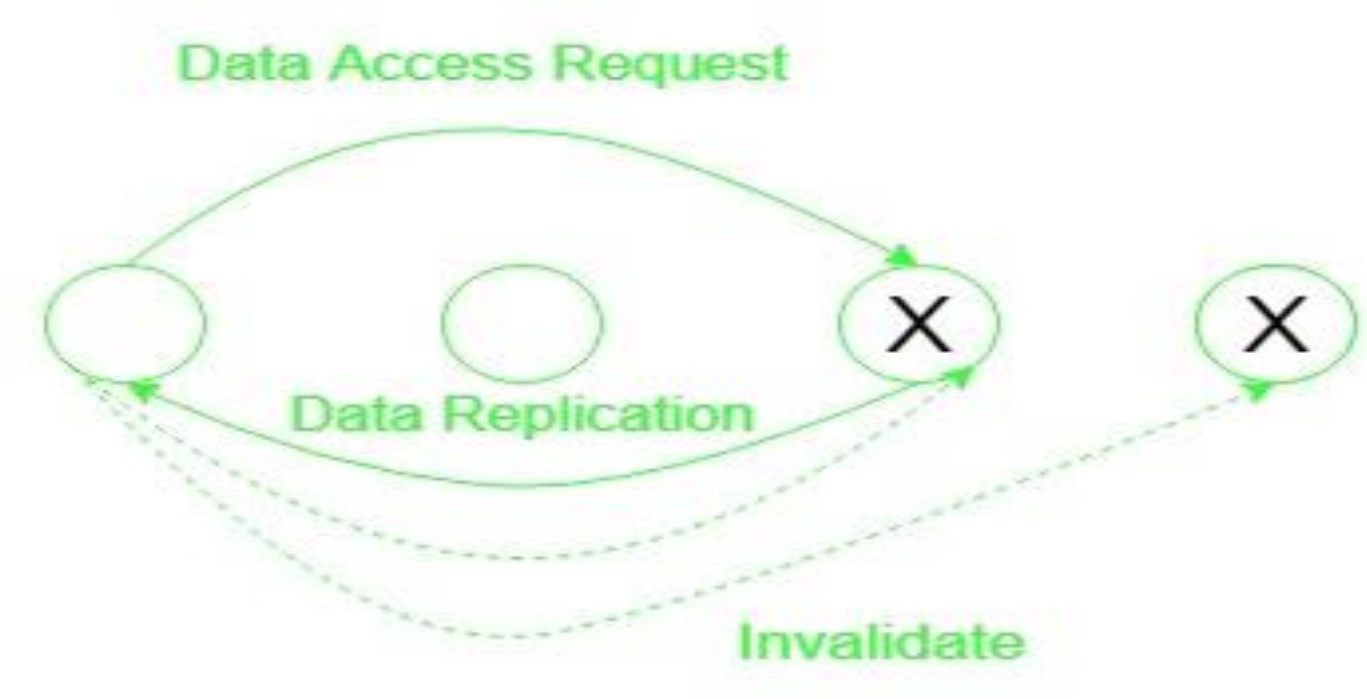
Migration Algorithm



Migration Algorithm

- In contrast to central server algo where every data access request is forwarded to location of data while in this data is shipped to location of data access request which allows subsequent access to be performed locally.
- It allows only one node to access a shared data at a time and the whole block containing data item migrates instead of individual item requested.
- It is susceptible to thrashing where pages frequently migrate between nodes while servicing only a few requests.
- This algo provides an opportunity to integrate DSM with virtual memory provided by operating system at individual nodes.

Read Replication Algorithm

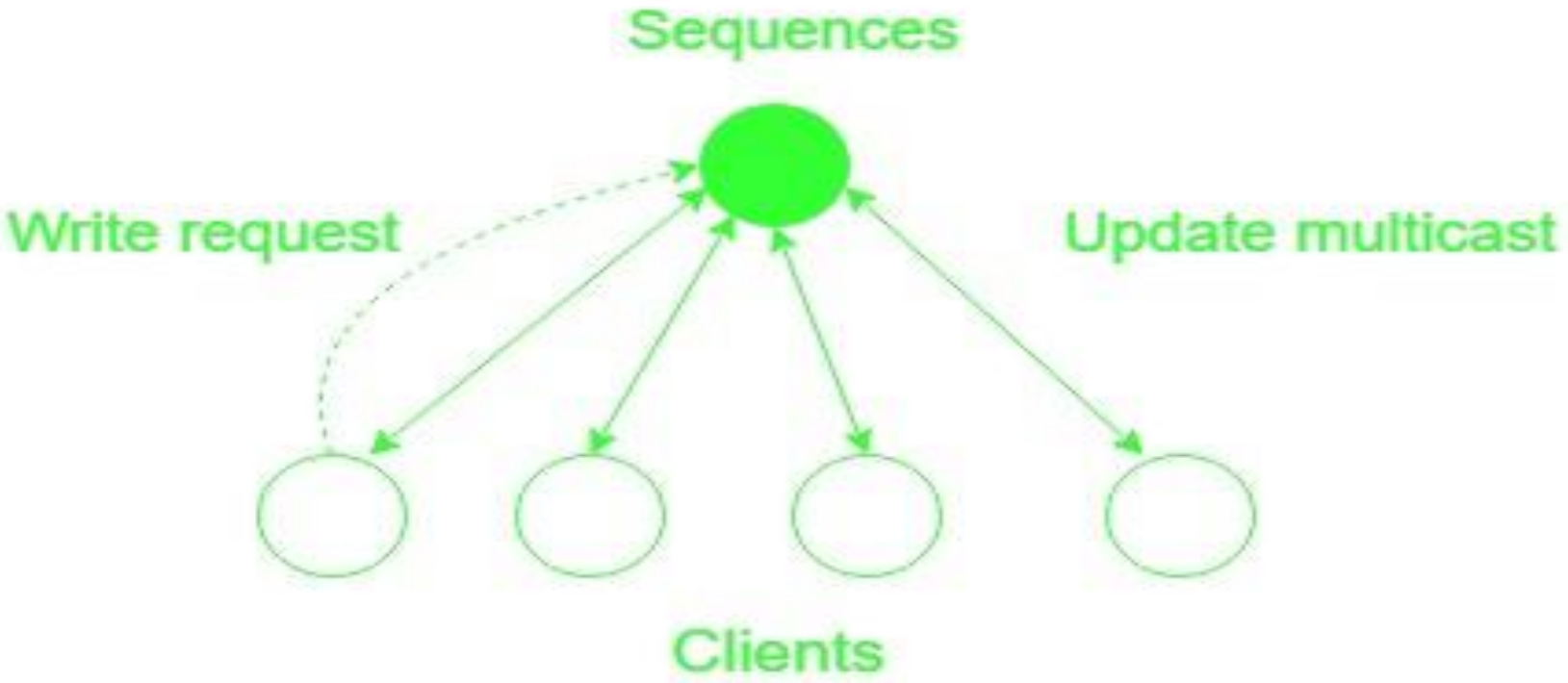


Write operation in Read Replication algorithm

Read Replication Algorithm

- This extends the migration algorithm by replicating data blocks and allowing multiple nodes to have read access or one node to have both read write access.
- It improves system performance by allowing multiple nodes to access data concurrently.
- The write operation in this is expensive as all copies of a shared block at various nodes will either have to invalidated or updated with the current value to maintain consistency of shared data block.
- DSM must keep track of location of all copies of data blocks in this.

Full Replication Algorithm



Write operation in Full Replication algorithm

Full Replication Algorithm

- It is an extension of read replication algorithm which allows multiple nodes to have both read and write access to shared data blocks.
- Since many nodes can write shared data concurrently, the access to shared data must be controlled to maintain it's consistency.
- To maintain consistency, it can use a gap free sequences in which all nodes wishing to modify shared data will send the modification to sequencer which will then assign a sequence number and multicast the modification with sequence number to all nodes that have a copy of shared data item.

Memory coherence

Consistency Models

- Refers to the degree of consistency that has to be maintained for the shared-memory data to a set of applications.

1. Sequential consistency models

- If all processes see the same order of all memory access operations on the shared memory. Ex: (r1,w1,r2)
- No new memory operation is started until all the previous ones have been completed.
- Uses single-copy(one-copy) semantics – all the processes share a memory location always see exactly the same contents stored in it.
- Acceptable consistency model for Distributed applications

2. Causal consistency model

- A memory reference operation (read/write) is said to be potentially causally related to another memory reference operation if the first one might have been influenced in any way by the second one.
- Eg: *Read* operation followed by *write* operation
- Write operations are not potentially causally related (w1, w2)
- It keep tracks of which memory reference operation is dependent on which other memory reference operations. Can be done by constructing and maintaining dependency graph for the memory access operations.

3. Processor consistency model

- It ensures that all write operations performed on the same memory location (no matter by which process they are performed) are seen by all the processes in the same order.
- Enhances memory coherence- For any memory location all processes agree on the same order of all write operations to that location.

  Ex: w12 and w22 are memory operations performed on the same memory location x, all processes must see them in the same order – w12 before w22 or w22 before w12.

4. Weak consistency model

- It is not necessary to show the change in memory done by every write operation to other processes.
- The results of several write operations can be combined and sent to other processes only when they needed it.
- Isolated accesses to shared variables are rare.
- Very difficult to show and keep track of the changes at time to time
- It uses a special variable called a synchronization variable (SV)for memory synchronization (visible to all processes)
- Memory synchronization in DSM will involve propagating memory updates done at a node to all other nodes having a copy of the same memory addresses.

5. Release consistency model

- All changes made to the memory by the process are propagated to other nodes.
- Uses two synchronization variables *(Acquire and Release)*
- *Acquire*-used by a process to tell the system about entering the critical section.
- *Release-* used by a process to tell about exit from critical section.
- Release consistency model uses synchronization mechanism based on barriers.
- It defines the end of a phase of execution of a group of concurrent processes before any process is allowed to proceed.

Coherence Protocols

- The needs to make the data replicas consistent
- Two types of basic protocols
  - Write-Invalidate Protocol: a write to a shared data causes the invalidation of all copies except one before the write.
  - Write-Update Protocol: A write to a share data causes all copies of that data to be updated.

Cache coherence in the PLUS system

PLUS write-update protocol

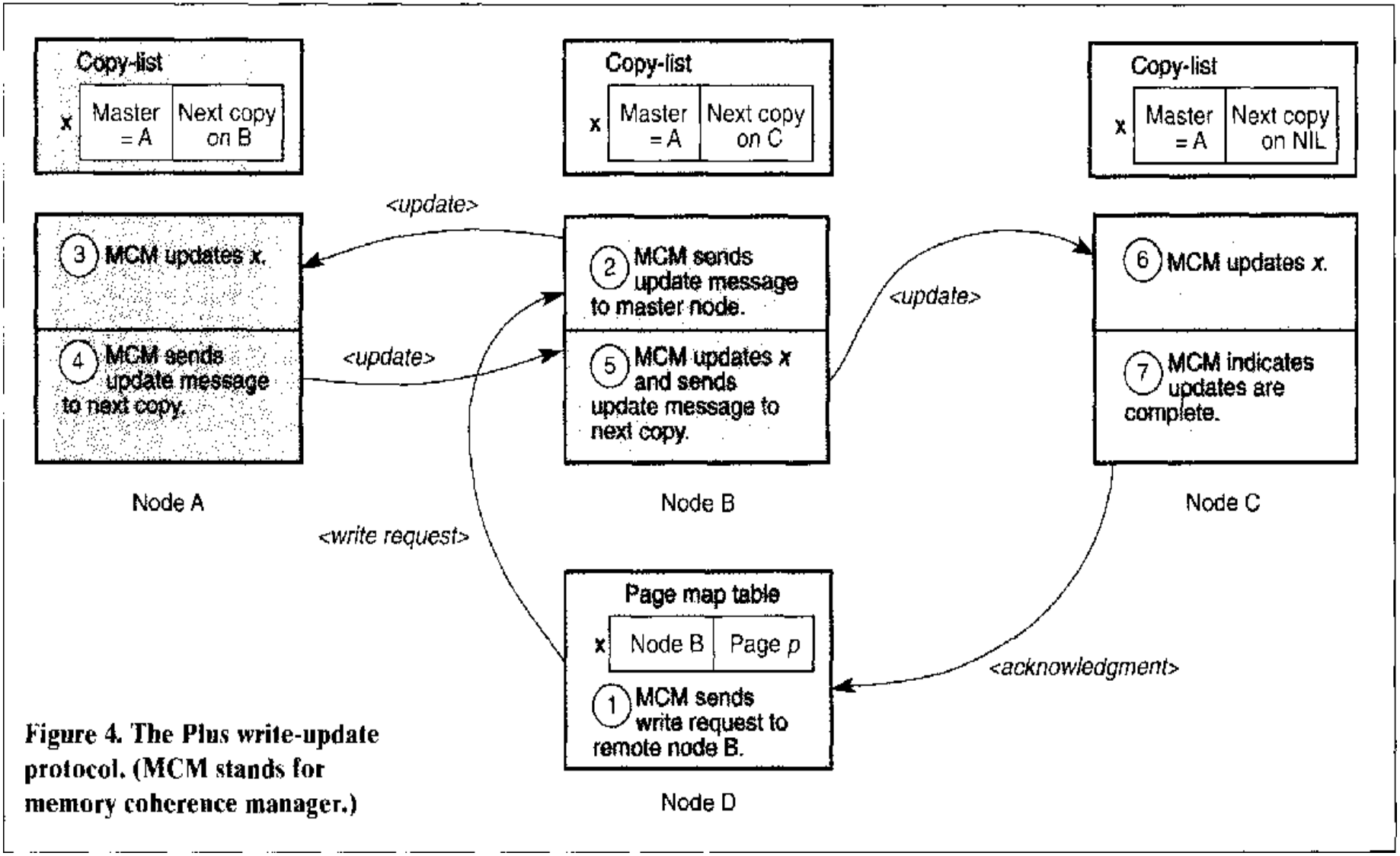


Figure 4. The Plus write-update protocol. (MCM stands for memory coherence manager.)

Unifying synchronization and data transfer in clouds

*Writer process

```
loop
Wait(empty);
Writelock(buffer);
Update buffer;
Unlock (buffer);
Signal(full);
End loop;
```

*Reader process

```
loop
Wait(full);
readlock(buffer);
read buffer;
Unlock (buffer);
Signal(empty);
End loop;
```

Type specific memory coherence in the Munim system

- Write-once objects
- Private objects
- Write-many objects
- Result objects
- Synchronization objects
- Migratory objects
- Producer-consumer objects
- Read-mostly objects
- General read-write objects
    - * invalid
    - * unowned
    - * owned exclusively
    - * owned nonexclusively

Case studies

- IVY
- Mirage

IVY(Integrated Shared Virtual Memory at Yale)

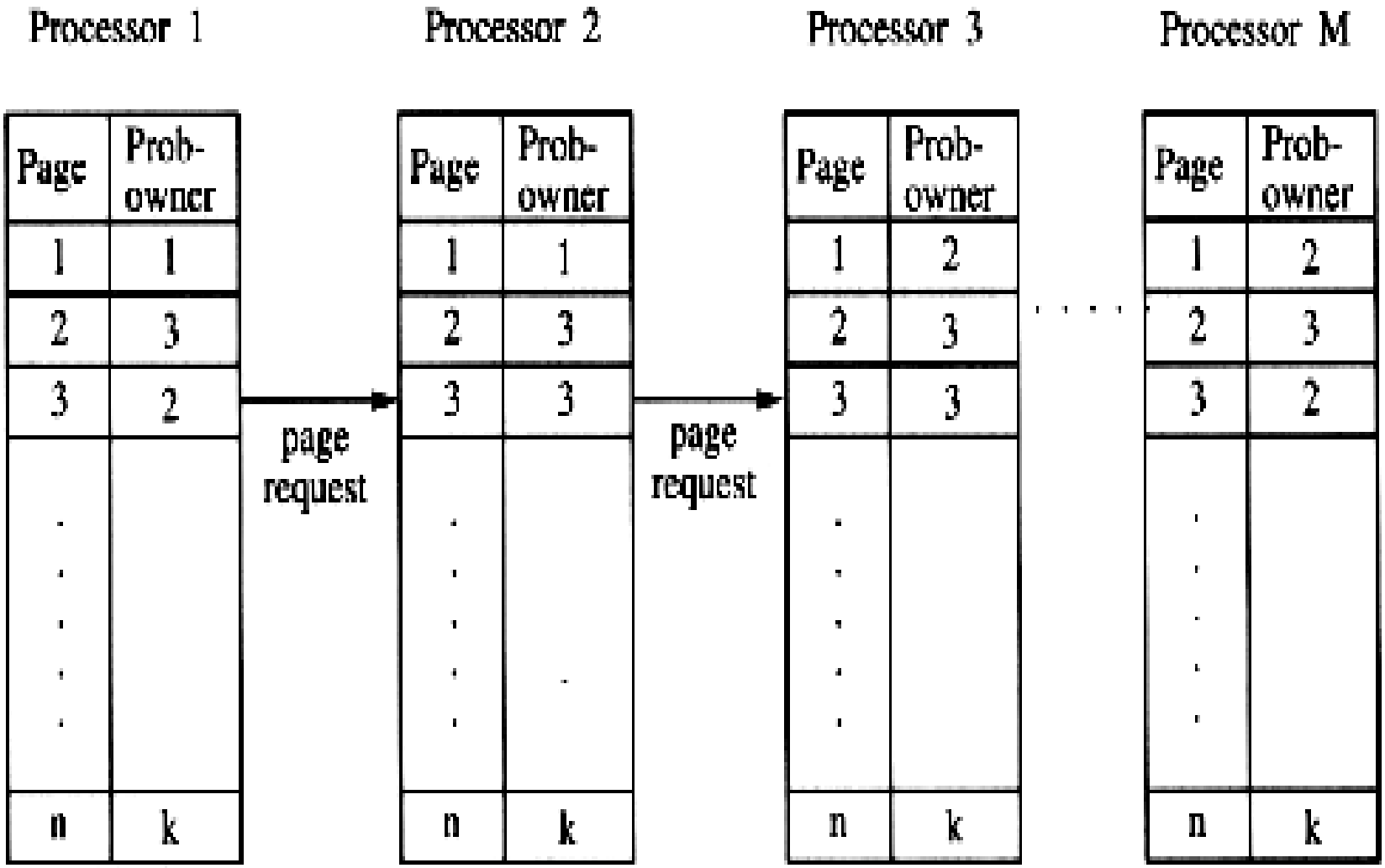
THE COHERENCE PROTOCOL:

- When a processor i has a write fault to a page p:
  - Processor i finds the owner page p.
  - The owner of page p sends the page and its copy set to I and marks its page table entry for page p as nil.
  - The faulting processor(i) sends out the invalidation messages to all the processors contained in the copy set.
- When a processor i has a read fault to a page p:
  - Processor i finds the owner page p.
  - The owner of page p sends a copy of page p to i and adds i to the copy set of p .processor I has read-only access to page p.
  - The owner marks its page table entry for p as read-only.

IVY PROTOCOLS

- The centralized manager scheme
- The fixed distributed manager scheme
- The dynamic distributed manager scheme

The dynamic distributed manager scheme



MIRAGE

- **Mirage** extends the IVY mechanisms by introducing a time interval, a page is pinned to a certain processor.
- During this interval, the ownership of the page will not be forwarded to another processor.
- This avoids **page thrashing** if two processors reference a single page repeatedly.