Chapter 12: Multiprocessor Architectures

Lesson 13:

Deadlock and virtual channels, and Flow control strategies

Objective

- To understand the deadlock
- To learn how to use virtual channel for breaking deadlock
- To learn to select flow-control strategy

Deadlock in channel

Distributed memory multiprocessor system

- System handles two types of messages:
  1. Messages that are acknowledged by response, for example a read request and
  2. Messages that are not responded, just accepted, for example, token or invalidation command or write-back or read response or snoopy bus message

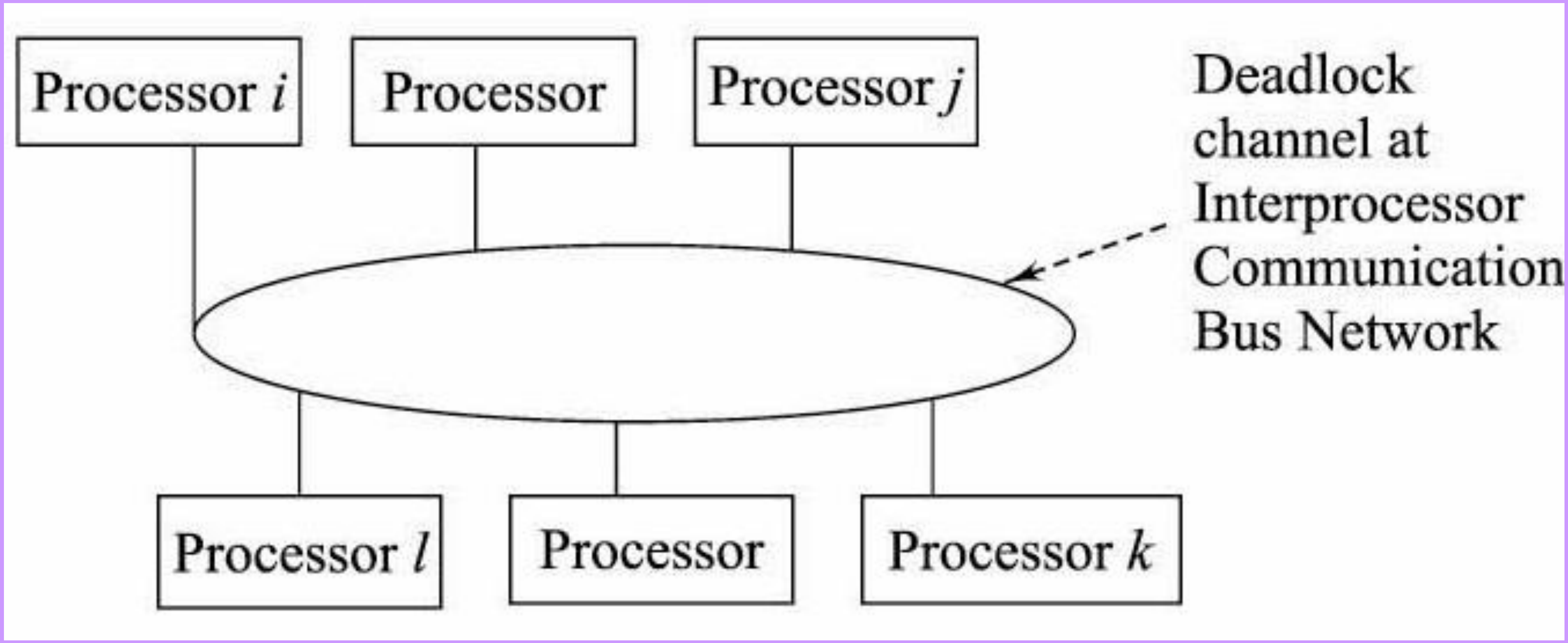
Deadlock

- Assume that a processor *i* is waiting for information from *j*, processor *j* is waiting for information from *k*, processor *k* is waiting for information from *l*, and *l* is waiting for information from *i*
- This circular dependency results in deadlock channel
- Deadlock can be for a fetch (read request), token, or response action

Deadlock channel

- A *deadlock channel* that arises because processors *i*, *j*, *k*, and *l* wait for RECEIVE and for a SEND command execution in another processor

Deadlock channel due to i, j, k and l waiting for RECEIVE and for SEND Command execution at other



Deadlock channel

- Arises because processors *i*, *j*, *k*, and *l* wait for RECEIVE and for a SEND command execution in another processor

Distributed memory multiprocessor system

- Ensures that deadlock will not occur

Virtual channel

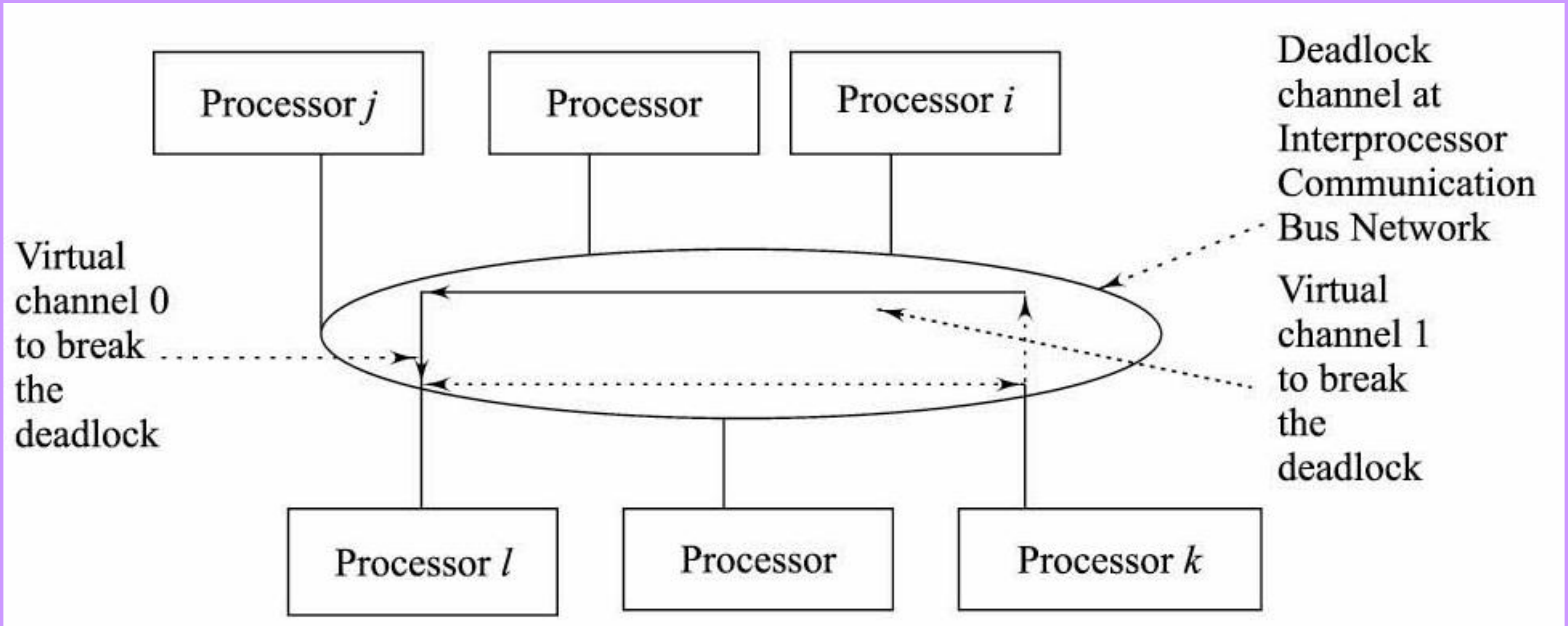
Virtual channel

- Used to divide the deadlock cycle
- Virtual channels used to avoid deadlock by systematically breaking cycles in the channel dependence graph
- A virtual channel after division of the deadlock channel to enable *j* to execute SEND and *i* to RECEIVE

Two or more virtual channels

- Can exist for each physical channel
- Messages at a processing node numbered higher (for example, *k*) than their destination (for example, *j*) are routed on a channel (for example, channel 1)
- Simultaneously, messages at a node numbered (for example, *i* or *j*) less than their destinations (for example, *i*) routed on another channel (for example, channel 0)

Virtual Channel after division of Deadlock Channel to Enable j to Execute SEND and i to RECEIVE



Flow Control Strategies

One flow control strategy

- Distributed memory systems use two separate queues for the requests and tokens to solve deadlocks
- Further, they maintain the order in which request type messages occur; the hardware gives priority to the token type messages and provides an alternate channel for messages of the latter type
- The queue of read request type messages is bounded

Negative acknowledgements only strategy

- Assumed that a request is granted unless referred back

A simple flow control mechanism

- Optimize the expected bandwidth needs as per the cache coherence protocol
- Each virtual channel path can have a queue sufficient to handle a limited number of successive requests, and from these other virtual channels remain unaffected

Highly degraded performance

- When many processors simultaneously send the remote requests for memory

Summary

We Learnt

- Deadlock for a fetch (read request), token, or response action
- Circular dependency in the channel
- Division of deadlocked channel into two or more virtual channels
- One flow control strategy is that distributed memory systems use two separate queues for the requests and tokens to solve deadlocks

End of Lesson 13 on

Deadlock and virtual channels, and Flow control strategies