BITS Pilani
K K Birla Goa Campus

Operating Systems

Dr. Lucy J. Gudino
Dept. of CS and IS

innovate achieve lead

BITS Pilani
K K Birla Goa Campus

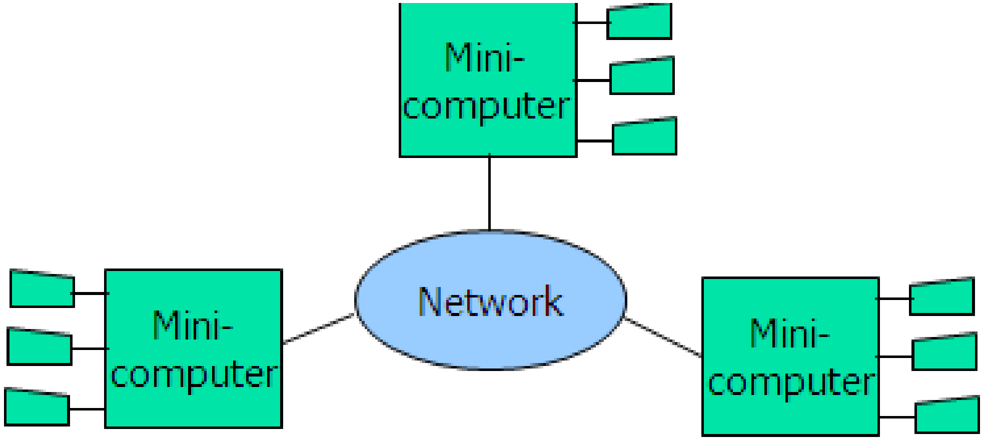
Process Synchronization

Introduction

- Scheduling algorithms
  - Non - Preemptive : FCFS, SJF, Priority
  - Preemptive : SRT, RR, Multilevel Queue
- Basic assumption
  - Uni-processor system

Introduction

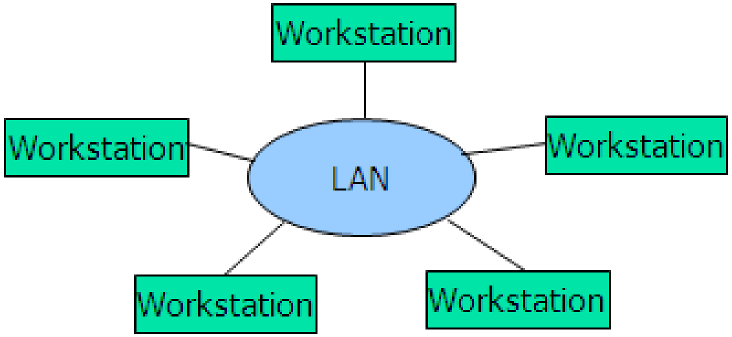
- Distributed systems: 3 types
  - Minicomputer model
  - Workstation model
  - Processor pool model
- Minicomputer model
  - Consists of several minicomputers
  - Supports multiple users and provides access to remote resources present in the communication network
  - The ratio of number of processors to the number of users is less than one



Courtesy : CSS434 System Models by Professor: Munehiro Fukuda

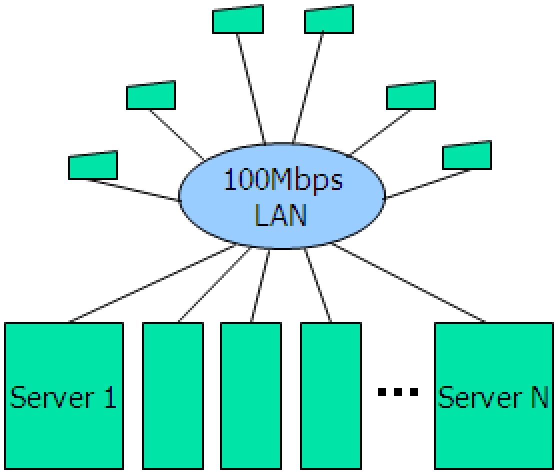
Contd…

- Workstation Model
  - Consists of several workstations and one user per workstation
  - workstation is equipped with powerful processor, memory, bit mapped display
  - may include math coprocessor
  - user can access data regardless of location of the data or location of workstation
  - The ratio of number of processors to the number of users is equal to one



Contd…

- Processor Pool Model
  - Pool of processors assigned to a user
  - The ratio of number of processors to the number of users is greater than one
- Processor allocation in multiprocessor system
  - Non-migratory and migratory
- Non-Migratory :
  - Static allocation: decide a processor for a process
  - Once decided, process will be executed on the allocated processor

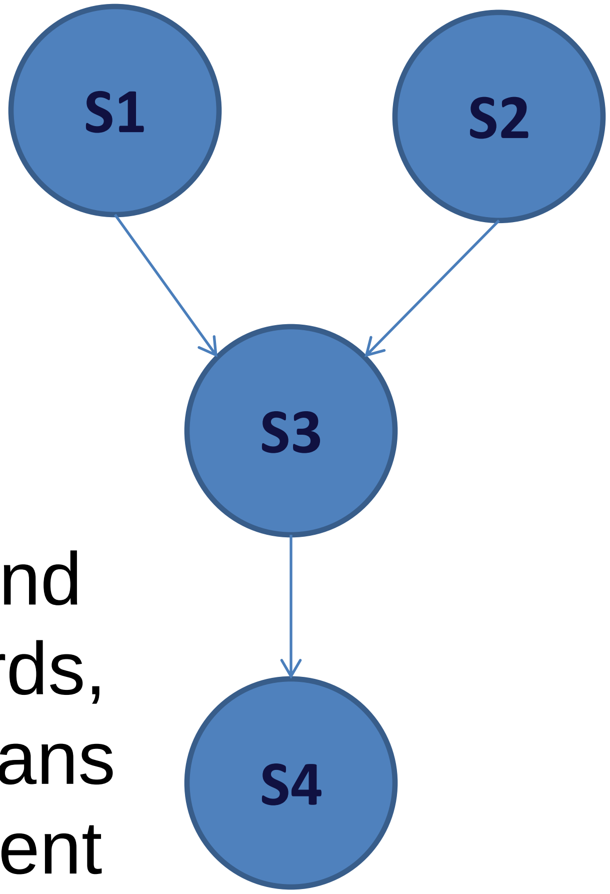


Contd…

- Migratory :
  - Dynamic allocation
  - processes can be moved after creation, which allows for better load balancing but is more complex.
- Multiple tasks and single task decomposed into subtasks
- Concurrent processing

Concurrent processing

- Example :
  - S1 : a ← w + x
  - S2 : b ← y + z
  - S3: c ← b – a
  - S4: d ← c – 1
- A precedence graph is a directed acyclic graph whose nodes correspond to individual statements. In other words, An edge from node Si to node Sj means Sj can be executed only after statement Si has completed execution.

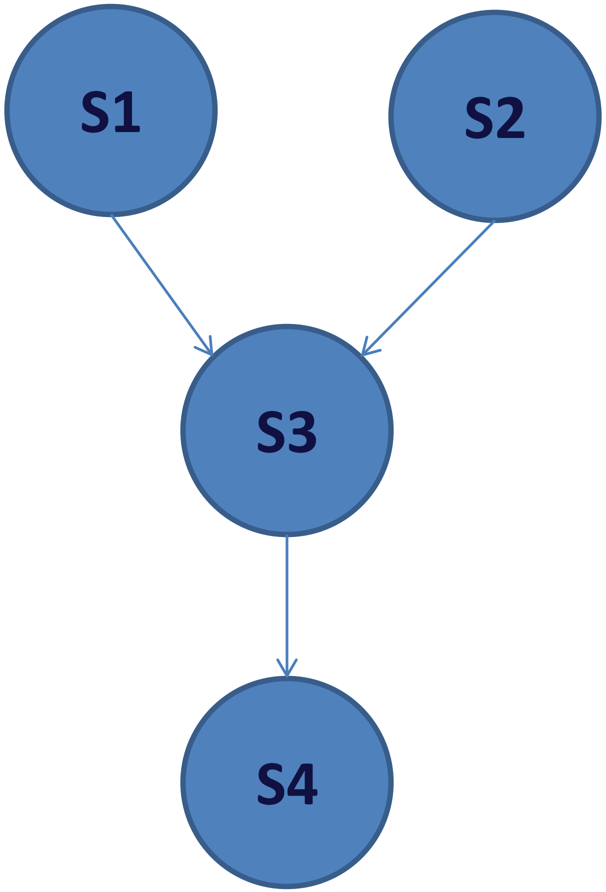


Concurrent processing…

- Two sets : Read set $R(S_i)$ and Write set $W(S_i)$
  - $R(S_i) = \{al, a2, a3 ....., a_n\}$, the read set for S_i, is the set of all variables whose values are referenced in statement S_i during the execution.
  - $W(S_i) = \{b1, b2, ..., b_n\}$. the write set for S_i, is the set of all variables whose values are changed (written) by the execution of statement S_i.

Concurrent processing…

- Example :
  - S1 : a $\leftarrow$ w + x
  - S2 : b $\leftarrow$ y + z
  - S3: c $\leftarrow$ b – a
  - S4: d $\leftarrow$ c – 1



- Read Set Write Set

Read Set	Write Set
R(S1) = {w, x}	W(S1) = {a}
R(S2) = {y, z}	W(S2) = {b}
R(S3) = {b, a}	W(S3) = {c}
R(S4) = {c}	W(S4) = {d}

Concurrent processing

- Concurrency conditions:

$$R(S_i) \cap W(S_j) = \phi$$

$$W(S_i) \cap R(S_j) = \phi$$

$$W(S_i) \cap W(S_j) = \phi$$

$$R(S_i) \cap R(S_j) \neq \phi$$

•Read Set	Write Set
R(S1) = {w, x}	W(S1) = {a}
R(S2) = {y, z}	W(S2) = {b}
R(S3) = {b, a}	W(S3) = {c}
R(S4) = {c}	W(S4) = {d}

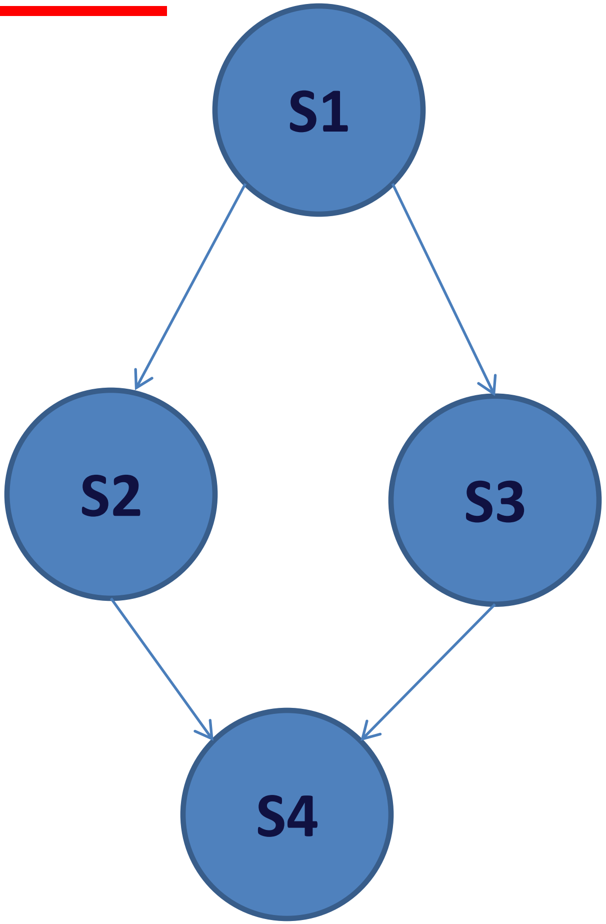
Concurrent processing…

- fork and join constructs:
  - fork L: produces two concurrent executions in a program
  - one execution begins at the statement labeled at L
  - other is the continuation of the execution at the statement following the fork instruction.
- join: recombine two concurrent computations into one
  - parameter count is used to specify the number of computations to be joined

Concurrent processing…

```
begin
        count = 2
        s1
        fork L1
        s2
    L2: join count
        s4
end

L1: s3
    goto L2
```



Join count implementation

count = count -1;

if count != 0 then

quit or terminate the process