Queueing Networks - I

Classification

&

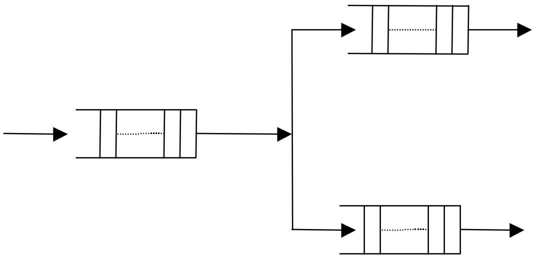
Basic Concepts

Queueing Network Interconnected queues with jobs flowing from one queue to another

Examples: Machine-shop
Communication Network
Computer System

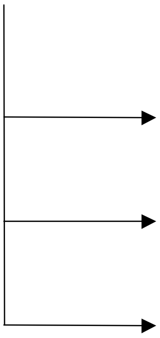
> *It may be sometimes easier to model a complicated service scenario as a queueing network in order to capture better the way the service is actually provided*

Routing of Jobs in a Queueing Network



- Probabilistic Routing (Single/Multiple classes)
- Class-Based Deterministic Routing

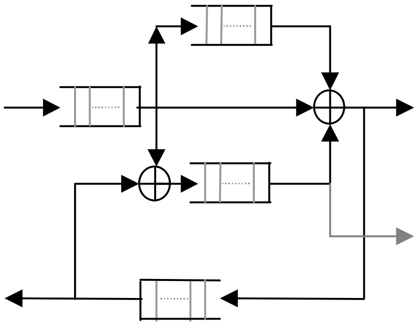
Classification of Queueing Networks



Open Networks

Closed Networks

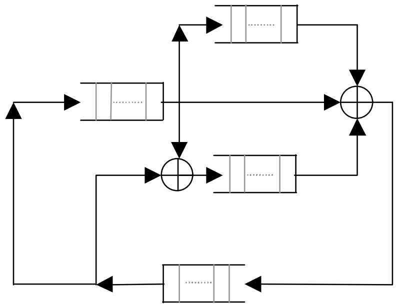
Mixed Networks



Open Queueing Network

If the network has multiple job classes then it must be open for each class of jobs.

Copyright 2002, Sanjay K. Bose

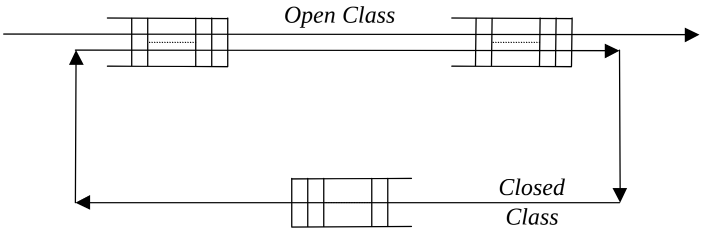


Closed Queueing Network

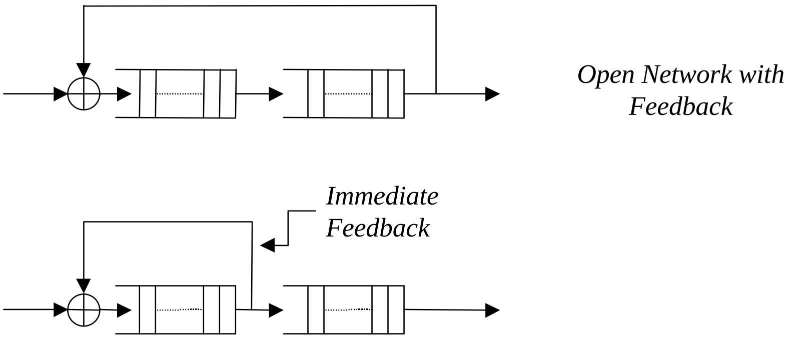
If the network has multiple job classes then it must be closed for each class of jobs.

Copyright 2002, Sanjay K. Bose

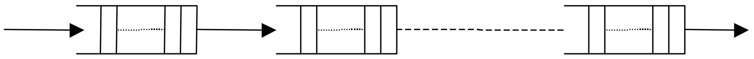
Mixed Network



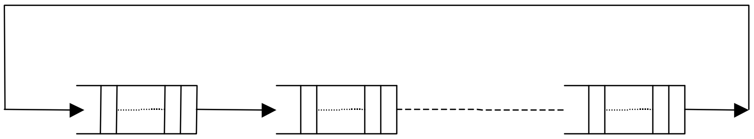
Network has multiple job classes and is open with respect to some classes but closed with respect to the others.



Acyclic or Feedforward Network *is an open queueing network with no feedback. In such a network, a queue may be visited at most once by a job entering the network.*



Tandem (Open) Queueing Network



Cyclic (Closed) Queueing Network

Fork-Join Queue

• In a Fork-Join Queue, an entering job is decomposed to be serviced in parallel by a number of *sibling queues*.

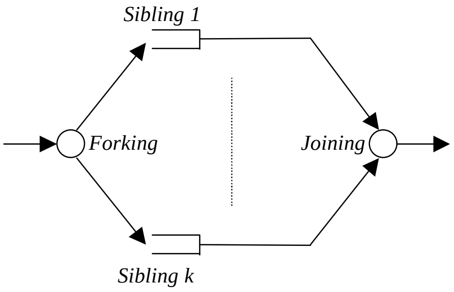
• **Forking Process -** A job is offered to two or more sibling queues for every job entering the Fork-Join Queue.

• **Joining Process -** Once all the (sub)jobs finish their service at each of the sibling queues, they are recombined and the job leaves the Fork-Join queue.

• For every job entering the Fork-Join queue, only one job leaves the queue. This is even though, the job may be decomposed into several subjobs, one each for each of the sibling queues.

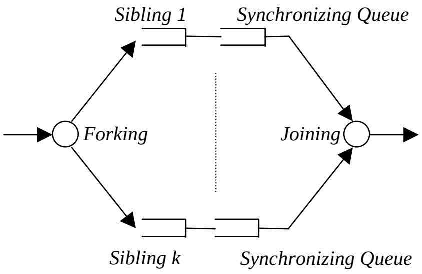
• It is possible to have a Fork-Join queue either with or without a synchronizing queue. (See the following slides.)

Fork-Join Queue without Synchronizing Queue

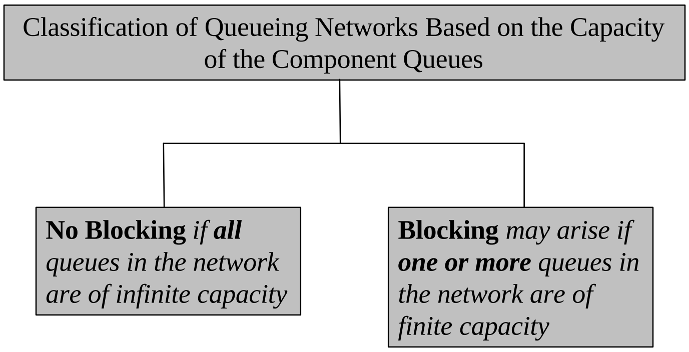


- *A sub-job finishing service at a sibling queue is forced to wait at that queue (blocking its server) until all the other sub-jobs also finish their service at their respective sibling queues.*
- *When this happens, the sub-jobs are combined and the actual job finally leaves the Fork-Join queue*

Fork-Join Queue with Synchronizing Queue



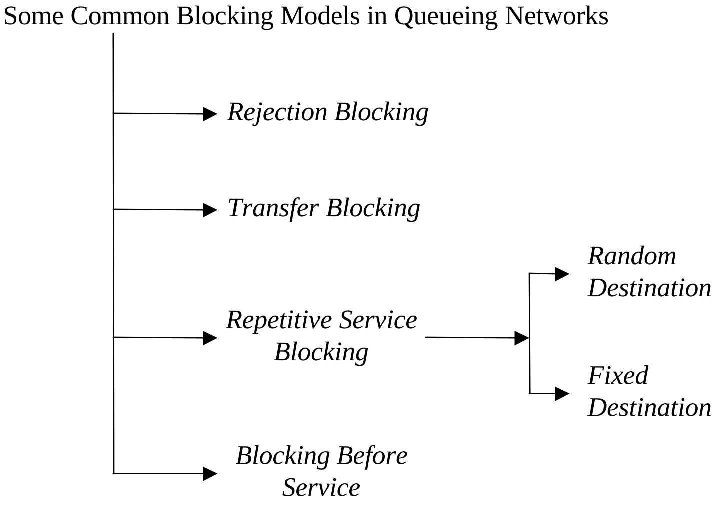
- *A sub-job finishing service at a sibling queue waits in its synchronizing queue until all the other sub-jobs also finish their service at their respective sibling queues.*
- *When this happens, the sub-jobs are combined and the actual job finally leaves the Fork-Join queue*
- *Note that in this case, the sibling queue's server is unblocked and can take on a sub-job of the next job entering the Fork-Join queue, if any.*



Classification of Queueing Networks Based on the Capacity of the Component Queues

No Blocking *if* ***all*** *queues in the network are of infinite capacity*

Blocking *may arise if* ***one or more*** *queues in the network are of finite capacity*



Some Common Blocking Models in Queueing Networks

Rejection Blocking

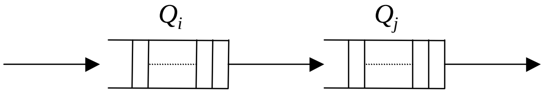
Transfer Blocking

Repetitive Service Blocking

Random Destination

Fixed Destination

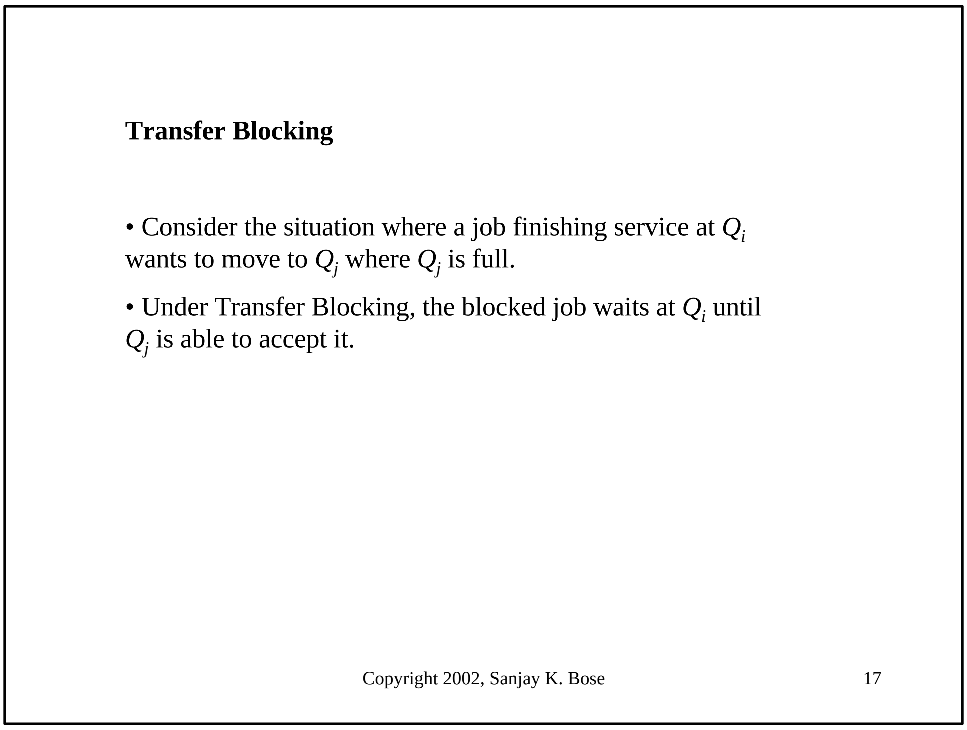
Blocking Before Service



- Consider a job which finishes service at Q_i and is then required to move Q_j.
- Blocking may occur if Q_j is a finite capacity queue and if all its waiting positions are full.
- The Blocking Model then decides the action that will be taken to handle this as far as the blocked job is concerned.

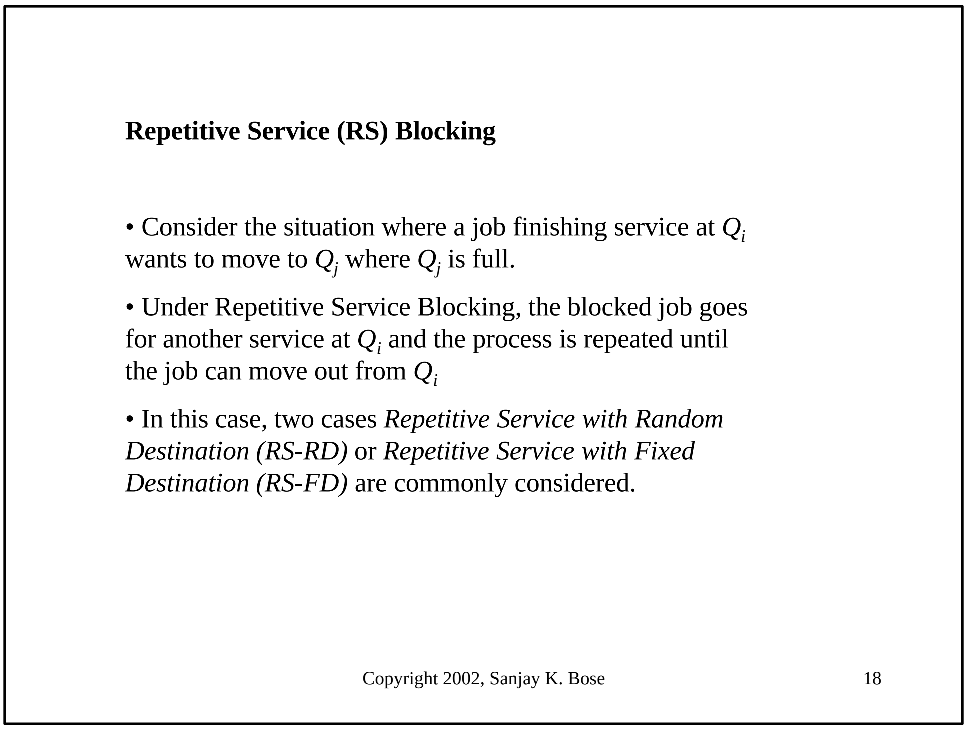
Rejection Blocking

- Consider the situation where a job finishing service at Q_i wants to move to Q_j where Q_j is full.
- Under Rejection Blocking, the blocked job is forced to leave the system.
- This model is only applicable for an open network of queues. (In a closed network, jobs circulate in the network without any departures from the system or new arrivals to the system.)



Transfer Blocking

- Consider the situation where a job finishing service at Q_i wants to move to Q_j where Q_j is full.
- Under Transfer Blocking, the blocked job waits at Q_i until Q_j is able to accept it.



Repetitive Service (RS) Blocking

- Consider the situation where a job finishing service at Q_i wants to move to Q_j where Q_j is full.
- Under Repetitive Service Blocking, the blocked job goes for another service at Q_i and the process is repeated until the job can move out from Q_i
- In this case, two cases *Repetitive Service with Random Destination (RS-RD)* or *Repetitive Service with Fixed Destination (RS-FD)* are commonly considered.

Repetitive Service with Fixed Destination (RS-FD) Blocking

- After finishing its repeated service, the blocked job tries to go to the *same destination* as the one where it was trying to go before.
- In case the destination queue is still full, it repeats the process (repeated service at the blocked queue). This is done until it can move to the destination queue at the end of a repeated service.

Repetitive Service with Random Destination (RS-RD) Blocking

- After finishing its repeated service, the blocked job chooses a new destination randomly according to the routing probabilities from the source node.
- This is done independently of the destination choice made earlier.
- In case the destination queue (possibly a new one) is still full, it repeats the process (repeated service at the blocked queue). This is done until it can move to the randomly selected destination queue at the end of a repeated service.

Blocking Before Service

- A job declares its destination queue, say Q_j, before it starts service at Q_i.
- If Q_j is full at that instant, the server at Q_i gets blocked, i.e. it cannot serve other jobs.
- Service to the job starts at Q_i only when the destination node Q_j gets unblocked and can accept the job after it finishes its service at the source queue Q_i.

Product-Form Queueing Networks

Consider an arbitrary network of K queues at equilibrium, with n_i jobs in the k^{th} queue, i.e. Q_k.

The queue is referred to as a *product-form queueing network* if the joint distribution of the number in each queue of the system may be written in the following form -

$$P\{n_1, n_2, ..., n_K\} = \prod_{i=1}^{K} f_i(n_i)$$

In a large number of queueing situations, and under fairly general conditions, this result is observed to hold either exactly or as a good approximation.