

CS4632 Object Oriented Modeling and Simulation

Multi-Server Queuing Models with Priorities

Dr. José M. Garrido C.

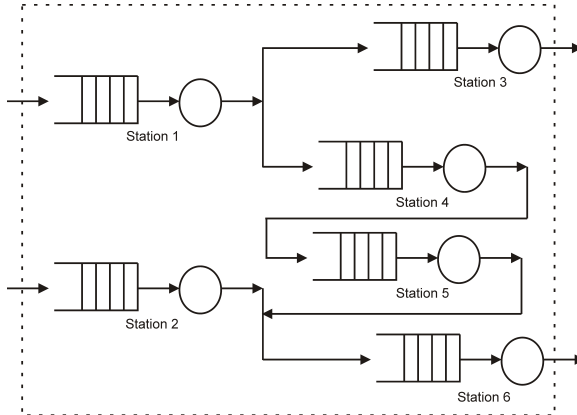
Department of Computer Science
Kennesaw State University

Spring 2017

Service Stations

- A queuing network is a more general model than the ones considered previously, it consists of a number of interconnected stations.
- The stations are interconnected in such a way that the outputs of one station are used as inputs to one or more other stations.
- Each station has its own server and queue, it can consist of one or more servers and a queue.
- Each service station in a queuing network provides a different type of service.

A Queuing Network Model with Six Stations



Queuing Network Models

- In queuing systems, customers request service from several service stations.
- A customer object will usually request service from several service stations.
- For example, in a data communication network the data packets are the customers and they travel from service station to service station until each packet reaches its final destination.

Performance Metrics

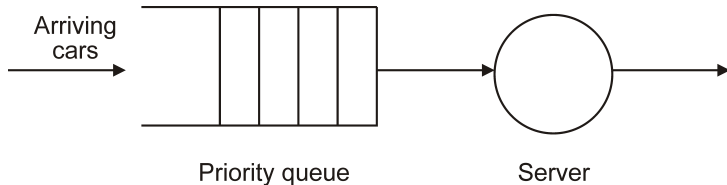
Some of the performance measures to be computed for queuing models are:

- For every server, the percentage of the total time that the server is actually carrying out the servicing of the customers, the server utilization
- The average number of customers waiting (i.e., the number of customers in the queue)
- The average time that a customer spends in the system, the average sojourn time
- The total number of customers serviced in a given interval, the system throughput
- The percentage of customers rejected because of the limited queue size. (These customers arrive when the queue is full.)

Customer Types

- In models with multiple types of customers, each customer type has its own mean arrival rate, mean service period, and other workload parameters.
- Usually, each customer type is given a **priority**.
- A priority is used by the server object to select a customer for service.
- The server normally selects the customer with the highest priority.
- The priority affects the order of removal from the queue—the queue discipline.

Model of a Simple System with Priorities



Applications with Priorities

- Real-time systems used to control power plants
- Computer communication networks, in which data may be routed in different ways, according to the message priorities
- Medical attention in a hospital, in which patients are attended according to the degree of the emergency
- Operating systems, in which processes are serviced according to the process' priority

Priority Queues

- Simulation models with priorities include at least one priority queue for arriving customers.
- A priority queue is a queue that stores customers in **order of priority** and uses FIFO ordering for customers with the same priority.
- Customers with the highest priority are always at the head of the queue. Computer communication networks, in which data may be routed in different ways, according to the message priorities

Priorities With Single-Server Models

- Customer objects of different types arrive and join a priority queue.
- The priority is assigned by the arrivals object.
- Since customer objects of the various types have different mean inter-arrival periods and different mean service periods, for each type of customers a different arrivals object is created.
- Each arrivals object creates customer objects of the same type.

Process Priorities

- The priority of a process is an integer value and represents its relative importance in relation to other processes.
- In OOSimL, the highest priority value is 0.
- To access the priority of a process, the statement **assign priority** is used.

Example Process Priorities

```
define mach_prio of type integer // priority of machine
define myprio of type integer    // priority of self
define machine of class Server   // reference to server
...
assign priority of machine to mach_prio
assign priority to myprio        // get priority self
```

Set Priority of Active Object

To set the priority of a process, the statement **fix priority** is used. The general structure of the statement follows.

```
fix priority < prio_var > [ to < ref_variable > ]
```

Example of Set Priority

The following example sets the priority 5 to the current process, then it sets priority 1 to another process referenced by *machine*.

```
fix priority 5 // to self  
fix priority 1 to machine
```

Example 2 with Priority

The following example increases the priority of a process referenced by *machine* by 1. The code defines an integer variable *m_priority* to store the priority of the process.

```
define m_priority of type integer
...
assign priority of machine to m_priority
increment m_priority // increment current priority
fix priority m_priority to machine // new priority
```

Priority Queues

- Priority queues allow the simulation model to retrieve or remove processes from the queue, based on the priority of the processes.
- Priority queues do **not** behave as FIFO queues.
- A priority queue is an object of class *Pqueue*.
- OOSimL includes statements that are used to define and manipulate priority queues.

Example Creating Objects of Class Pqueue

The following lines of code declare and create the priority queue with reference name *cust_queue*, with queue name *Customer Queue*, using 8 different priorities, and size (capacity) of 25 customers for every priority.

```
define cust_queue of class Pqueue
...
// create priority queue
create cust_queue of class Pqueue
    using "Customer Queue", 8, 25
```

Create Objects of Class Pqueue

- Objects of class *Pqueue* are passive objects.
- The constructor for creating a priority queue requires the name of the queue, the optional number of different priorities, and the queue size for all priorities, which is also optional.
- If the second argument is not included, the number of priorities is assigned to 300 (default).
- If the third argument is not included, the assigned default queue size is 1,000 for every priority.

Priority Queue Length

- This **assign** statement gets the current size (total number of processes) in the specified priority queue.
- An integer value that corresponds to the current length of the queue is assigned to the specified integer variable.
- The general structure of the statement follows.

assign length of `< queue_ref >` **to** `< variable >`

Example Length of Priority Queue

For example, the following lines of code get the current length of priority queue *cust_queue* (defined above) and assign this value to variable *mqlength*.

```
define mqlength of type integer
...
assign length of cust_queue to mqlength
```

Check Priority Queue Empty

- The **if** statement with the **empty** condition check if a queue is empty.
- For example, the following lines of code check if queue object *cust_queue* (defined above) is empty.
- The current process suspends itself if the queue is empty, otherwise, it proceeds to remove a process from the queue.

```
if cust_queue is empty then
    // if queue is empty, suspend here
    suspend self
    ...
else
    // dequeue process
    ...
endif
```

Processes With A Specified Priority

The following line of code gets the current number of processes in the priority queue *cust_queue*, with priority *l_prio*, and assigns this number to variable *num_proc*.

```
assign length of cust_queue with  
    priority l_prio to num_proc
```

Insert Process Into Priority Queue

- The **insert** (and **enqueue**) statement inserts the specified process into the priority queue.
- The size of the queue is increased by one.
- The priority of the process is automatically used to insert the process.

```
insert < ref_variable > into < queue_ref >  
insert self into < queue_ref >  
enqueue < ref_variable > into < queue_ref >  
enqueue self into < queue_ref >
```

Example Insert

The following lines of code insert (enqueue) process *cust_obj* into priority queue *pcust_queue*.

```
define cust_prio = 3 of type integer
define cust_obj of class Customer
...
// enqueue customer process
insert cust_obj into pcust_queue
```


Remove Process with Specified Priority

- The statements **remove** and **dequeue** remove (dequeue) a process of a specified priority, from a priority queue.
- After removal, the size of the queue is reduced by one, as a result of this operation.

```
remove [ object ] < ref_var > of class < cl_name >  
      from [ queue ] < queue_ref >  
      with priority < prio_variable >  
dequeue [ object ] < ref_var > of class < cl_name >  
      from [ queue ] < queue_ref >  
      with priority < prio_variable >
```

Example Remove Process

- The following example removes (dequeues) a customer process from queue *cust_queue* and assigns a reference *cust_obj* to it, given priority *cust_prio*.
- If no priority is specified, the statement removes the highest priority process from the priority queue.

```
define cust_prio of type integer
...
remove object cust_obj of class Customer from
    queue cust_queue with priority cust_prio
```

Carwash with Priorities

- Each car object includes its priority (type) in addition to its service period, as part of its relevant attributes.
- A car object joins the car queue according to its priority.
- The Arrivals object creates a sequence of car objects of a specified priority. One car object is created after every inter-arrival time interval.
- Every car object has a different service interval.

Output of Carwash with Priorities

End Simulation of Model of Carwash System
with Priorities date: 7/27/2016 time: 9:24

Total cars of type 0 serviced : 120
Car type 0 average wait period: 10.587397344561573
Total cars of type 1 serviced : 68
Car type 1 average wait period: 235.79209727808004
Total cars of type 2 serviced : 35
Car type 2 average wait period: 442.1166150740475
Total cars of type 3 serviced : 41
Car type 3 average wait period: 750.9129750062083
Total cars of type 4 serviced : 29
Car type 4 average wait period: 701.8389044542504

Total number of cars serviced: 293
Car average wait period: 286.414
Average period car spends in the shop: 289.953
Machine utilization: 0.994

Multi-Server Models with Priorities

- There are several car types, each with a specified priority
- There are several objects of class Arrivals, each one generates car objects of the specified type
- Each server dequeues and services the car with the highest priority
- A different configuration has each server servicing cars of a specified type

Model of a Multi-Server System with Priorities

