Multi-Server and Priority Queues

CS6323
Priority Queues

- Modelling systems with customers of different priority
  - Airline check-in, computer thread scheduling

- Approach
  - Differentiate customer types

- Protocols
  - Pre-emption vs. non-pre-emption
Reminder: Kendall’s notation

A/S/N/K gives a theoretical description of a system

- **A** is the arrival process
  - M = Markovian = Poisson Arrivals
  - D = deterministic (constant time between arrivals)
  - G = general (anything else)

- **S** is the service process
  - M,D,G same as above

- **N** is the number of parallel processors

- **K** is the buffer size of the queues
  - K term can be dropped when buffer size is infinite
The \( \text{M/M/1 Queue} \) (a.k.a., birth-death process)

- a.k.a., \( \text{M/M/1/}\infty \)
  - Poisson arrivals
  - Exponential service time
  - 1 processor, infinite length queue

- Can be modeled as a Markov Chain (because of memoryless behaviour)

- Distribution of time spent in state \( n \) the same for all \( n > 0 \)

Transition probabilities:

- \( \lambda/(\lambda+\mu) \)
- \( \lambda/(\lambda+\mu) \)
- \( \lambda/(\lambda+\mu) \)
- \( \lambda/(\lambda+\mu) \)

# pkts in system

(When \( > 1 \), is 1 larger than # pkts in queue)
M/M/1 cont’d

As long as $\lambda < \mu$, queue has following steady-state average properties

- **Defs:**
  - $\rho = \frac{\lambda}{\mu}$
  - $N = \#$ pkts in system
  - $T =$ packet time in system
  - $N_Q = \#$ pkts in queue
  - $W =$ waiting time in queue

- $P(N=n) = \rho^n(1-\rho)$
  - (indicates fraction of time spent w/ $n$ pkts in queue)
  - Utilization factor $= 1 - P(N=0) = \rho$

- $E[N] = \sum_{n=0}^{\infty} n P(N=n) = \frac{\rho}{(1-\rho)}$

- $E[T] = E[N] / \lambda$ (Little’s Law) $= \frac{\rho}{(\lambda(1-\rho))} = \frac{1}{(\mu - \lambda)}$

- $E[N_Q] = \sum_{n=1}^{\infty} (n-1) P(N=n) = \frac{\rho^2}{(1-\rho)}$

- $E[W] = E[T] - \frac{1}{\mu}$ (or $= E[N_Q] / \lambda$ by Little’s Law) $= \frac{\rho}{(\mu - \lambda)}$
Multi-Server ($M/M/1/K$) queue

- Also can be modeled as a Markov Model
  - requires $K+1$ states for a system (queue + processor) that holds $K$ packets (why?)
  - Stay in state $K$ upon a packet arrival
  - Note: $\rho \geq 1$ permitted (due to multiple servers)
M/M/1/K properties

\[ P(N=n) = \begin{cases} 
\frac{\rho^n(1-\rho)}{(1 - \rho^{K+1})}, & \rho \neq 1 \\
1 / (K+1), & \rho = 1 
\end{cases} \]

\[ E[N] = \begin{cases} 
\frac{\rho}{((1-\rho)(1 - \rho^{K+1}))}, & \rho \neq 1 \\
1 / (K+1), & \rho = 1 
\end{cases} \]

i.e., divide M/M/1 values by \((1 - \rho^{K+1})\)
Priority Queues

- Classes have different priorities
- May depend on explicit marking or other header info, e.g., IP source or destination, TCP Port numbers, etc.
- Transmit a packet from the highest priority class with a non-empty queue
Priority Queue Scheduling Policy

- 2 versions:
  - Preemptive: (postpone low-priority processing if high-priority pkt arrives)
  - non-preemptive: any packet that starts getting processed finishes before moving on
Modeling priority queues as $M/M/1/K$

- preemptive version ($K=2$): assuming preempted packet placed back into queue
  - state $w/ x,y$ indicates $x$ priority queued, $y$ non-priority queued
  - what are the transition probabilities?
  - what if preempted is discarded?
Modeling priority queues as $M/M/1/K$

- preemptive version (K=2 for each priority)
  - state w/ $x,y$ indicates $x$ priority queued, $y$ non-priority queued
M/M/1/K Priority Queue: Pre-empted
Job Discarded

- preemptive version (K=2): assuming preemted packet placed back into queue
  - state w/ x,y indicates x priority queued, y non-priority queued
Modeling priority queues as M/M/1/K

- Non-preemptive version (K=2)
  - yellow (solid border) = nothing or high-priority being proc’d
  - red (solid border) = low-priority being processed
  - red (dashed border) = nothing/high-priority being processed
  - what are the transition probabilities?
Scheduling Policies (more)

- **Round Robin:**
  - Each flow gets its own queue
  - Circulate through queues, process one pkt (if queue non-empty), then move to next queue
Weighted Fair Queuing: is a generalized Round Robin in which an attempt is made to provide a class with a differentiated amount of service over a given period of time.
Weighted Fair Queue details

- Each flow, \( i \), has a weight, \( W_i > 0 \)
- A Virtual Clock is maintained: \( V(t) \) is the “clock” at time \( t \)
- Each packet \( k \) in each flow \( i \) has
  - virtual start-time: \( S_{i,k} \)
  - virtual finish-time: \( F_{i,k} \)
- The Virtual Clock is restarted each time the queue is empty
- When a pkt arrives at (real) time \( t \), it is assigned:
  - \( S_{i,k} = \max\{F_{i,k-1}, V(t)\} \)
  - \( F_{i,k} = S_{i,k} + \text{length}(k) / W_i \)
  - \( V(t) = V(t') + (t-t') / \sum_{j} W_j \)
    - \( t' = \) last time virtual clock was updated
    - \( B(t',t) = \) set of sessions with pkts in queue during \( (t',t] \)
Scheduling And Policing Mechanisms

Scheduling: choosing the next packet for transmission on a link can be done following a number of policies:

- **FIFO** (First In First Out) a.k.a. **FCFS** (First Come First Serve): in order of arrival to the queue
  - packets that arrive to a full buffer are discarded
  - another option: discard policy determines which packet to discard (new arrival or something already queued)