Tyler's Grill

Dine in or grab and go, Tyler's is fast, fresh and made for you. Located on the ground level of the Tyler Hanes Commons, Tyler's offers flavorful wraps, freshly prepared salads, sweet potato...
Tyler’s Grill: A Multi-Server Queue
Model Development At Three Levels

1. **Conceptual**: big picture, what questions to ask

2. **Specification**: equations, (pseudocode) algorithms

3. **Computational**: implement in code
Goal/Objective: *How fast must we service jobs?*

Assumptions:
- FIFO queue
- server takes no break
- non-preemptive: once service begins, must be completed
- conservative: if queue is not empty, no server can be idle
Model Development At Three Levels

1. *Conceptual*: big picture, what questions to ask (done!)

2. *Specification*: equations, (psuedocode) algorithms

3. *Computational*: implement in code
For a customer $i$:

- $a_i$: *arrival time*
- $w_i$: *wait* (time in the queue)
- $b_i$: time service begins  $b_i = a_i + w_i$
- $s_i$: *service time*
- $o_i$: *sojourn* (time in the system)  $o_i = w_i + s_i$
- $c_i$: *completion* (departure time)  $c_i = a_i + o_i$
Algorithmic Question

- How do we algorithmically process a job through the system?
  - For some queue disciplines, difficult
  - For FIFO, easy...

wait \( w_i \) determined by \( a_i \) relative to \( c_{i-1} \)

job \( i \) completes at \( c_i = a_i + w_i + s_i \)
Algorithmic Question

- How do we algorithmically process a job through the system?
  - For some queue disciplines, difficult
  - For FIFO, easy...

- wait $w_i$ determined by $a_i$ relative to $c_{i-1}$
- job $i$ completes at $c_i = a_i + w_i + s_i$
Algorithmic Question

- How do we algorithmically process a job through the system?
  - For some queue disciplines, difficult
  - For FIFO, easy...

wait \( w_i \) determined by \( a_j \) relative to \( c_{i-1} \)

job \( i \) completes at \( c_i = a_i + w_i + s_i \)

(Illustration by Ben Kirchner, Bloomberg Businessweek, Oct. 2010)
Algorithmic Question

- How do we algorithmically process a job through the system?
  - For some queue disciplines, difficult
  - For FIFO, easy...

- wait $w_i$ determined by $a_i$ relative to $c_{i-1}$
- job $i$ completes at $c_i = a_i + w_i + s_i$
Two Cases For Computing Wait

- Job $i$ arrives before job $i-1$ completes ($a_i < c_{i-1}$)

![Diagram showing scheduling and waiting times](attachment:diagram.png)
Two Cases For Computing Wait

- **Job $i$ arrives before job $i - 1$ completes ($a_i < c_{i-1}$)**

  ![Diagram for Job $i$ arrives before job $i - 1$ completes]

- **Job $i$ arrives after job $i - 1$ completes ($a_i \geq c_{i-1}$)**

  ![Diagram for Job $i$ arrives after job $i - 1$ completes]
**Process-Oriented:** Consider Each Customer Individually

<table>
<thead>
<tr>
<th>$i$</th>
<th>$r_i$</th>
<th>$a_i$</th>
<th>$n(a_i + \epsilon)$</th>
<th>$w_i$</th>
<th>$b_i$</th>
<th>$s_i$</th>
<th>$c_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>15</td>
<td>15</td>
<td>1</td>
<td>0</td>
<td>15</td>
<td>43</td>
<td>58</td>
</tr>
<tr>
<td>2</td>
<td>32</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>36</td>
</tr>
<tr>
<td>3</td>
<td>24</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>34</td>
</tr>
<tr>
<td>4</td>
<td>40</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>30</td>
</tr>
<tr>
<td>5</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>38</td>
</tr>
<tr>
<td>6</td>
<td>29</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>30</td>
</tr>
<tr>
<td>7</td>
<td>80</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>31</td>
</tr>
<tr>
<td>8</td>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>29</td>
</tr>
</tbody>
</table>

$r_i$: interarrival time  
$a_i$: arrival time  
$w_i$: wait (in the queue)  

$n(a_i + \epsilon)$: number in system @ time $a_i + \epsilon$ (for small $\epsilon$)  

$b_i$: time begins service  
$s_i$: service time  
$c_i$: time service is completed
Arrivals

\[ r_i \]

\[ a_{i-2} \quad a_{i-1} \quad a_i \quad a_{i+1} \]

- The interarrival time between job \( i - 1 \) and job \( i \):
  \[ r_i = a_i - a_{i-1} \]
  where \( a_0 = 0 \)

- Note that \( a_i = a_{i-1} + r_i \) and so (by induction)
  \[ a_i = r_1 + r_2 + \ldots + r_i \]
  \[ i = 1, 2, 3, \ldots \]

- single-server queue model: needs interarrival & service times
Process-Oriented Algorithm

\begin{align*}
a_0 &= 0.0 \\
c_0 &= 0.0 \\
i &= 0 \\
\text{while ( more jobs to process ) } &\{ \\
&\quad i = i + 1 \\
&\quad a_i = a_{i-1} + \text{getInterarrival()} \\
&\quad \text{if ( } a_i < c_{i-1} \text{ )} \\
&\quad\quad w_i = c_{i-1} - a_i \\
&\quad\text{else} \\
&\quad\quad w_i = 0.0 \\
&\quad s_i = \text{getService()} \\
&\quad c_i = a_i + w_i + s_i \\
&\} 
\end{align*}
Process-Oriented Algorithm

\[ a_0 = 0.0 \]
\[ c_0 = 0.0 \]
\[ i = 0 \]

\[
\text{while ( more jobs to process )} \{ \\
\quad i = i + 1 \\
\quad a_i = a_{i-1} + \text{getInterarrival}() \\
\quad \text{if (} a_i < c_{i-1} \text{)} \\
\quad \quad w_i = c_{i-1} - a_i \\
\quad \text{else} \\
\quad \quad w_i = 0.0 \\
\quad s_i = \text{getService}() \\
\quad c_i = a_i + w_i + s_i \\
\}
\]
Process-Oriented Algorithm For Single-Server Queue

**Process-Oriented Algorithm**

\[
a_0 = 0.0 \\
c_0 = 0.0 \\
i = 0
\]

while (more jobs to process) {
    \[i = i + 1\]
    \[a_i = a_{i-1} + \text{getInterarrival()}\]
    if (\[a_i < c_{i-1}\])
        \[w_i = c_{i-1} - a_i\]
    else
        \[w_i = 0.0\]
    \[s_i = \text{getService()}\]
    \[c_i = a_i + w_i + s_i\]
}
Process-Oriented Algorithm

\[ a_0 = 0.0 \]
\[ c_0 = 0.0 \]
\[ i = 0 \]

while ( more jobs to process ) {
    \[ i = i + 1 \]
    \[ a_i = a_{i-1} + \text{getInterarrival()} \]
    if ( \[ a_i < c_{i-1} \] )
        \[ w_i = c_{i-1} - a_i \]
    else
        \[ w_i = 0.0 \]
    \[ s_i = \text{getService()} \]
    \[ c_i = a_i + w_i + s_i \]
}
Process-Oriented Algorithm For Single-Server Queue

### Process-Oriented Algorithm

\[
\begin{align*}
a_0 &= 0.0 \\
c_0 &= 0.0 \\
i &= 0
\end{align*}
\]

\[
\text{while ( more jobs to process ) } \{ \\
i &= i + 1 \\
a_i &= a_{i-1} + \text{getInterarrival()} \\
\text{if } (a_i < c_{i-1}) \\
\quad w_i &= c_{i-1} - a_i \\
\text{else} \\
\quad w_i &= 0.0 \\
s_i &= \text{getService()} \\
c_i &= a_i + w_i + s_i
\}
\]
Gaining Insight From The Model

- Purpose of simulation: insight ⇔ requires system statistics

- Importance of different statistics varies:
  - Job perspective: wait time
  - Manager perspective: system utilization

- Two main categories:
  - *Job-averaged*: statistics based on observations (BOO)
  - *Time-averaged*: time-persistent statistics (TPS)
## Job-Averaged Statistics

<table>
<thead>
<tr>
<th>$i$</th>
<th>$r_i$</th>
<th>$a_i$</th>
<th>$n(a_i + \epsilon)$</th>
<th>$w_i$</th>
<th>$b_i$</th>
<th>$s_i$</th>
<th>$c_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>15</td>
<td>15</td>
<td>1</td>
<td>0</td>
<td>15</td>
<td>43</td>
<td>58</td>
</tr>
<tr>
<td>2</td>
<td>32</td>
<td>47</td>
<td>2</td>
<td>11</td>
<td>58</td>
<td>36</td>
<td>94</td>
</tr>
<tr>
<td>3</td>
<td>24</td>
<td>71</td>
<td>2</td>
<td>23</td>
<td>94</td>
<td>34</td>
<td>128</td>
</tr>
<tr>
<td>4</td>
<td>40</td>
<td>111</td>
<td>2</td>
<td>17</td>
<td>128</td>
<td>30</td>
<td>158</td>
</tr>
<tr>
<td>5</td>
<td>12</td>
<td>123</td>
<td>3</td>
<td>35</td>
<td>158</td>
<td>38</td>
<td>196</td>
</tr>
<tr>
<td>6</td>
<td>29</td>
<td>152</td>
<td>3</td>
<td>44</td>
<td>196</td>
<td>30</td>
<td>226</td>
</tr>
<tr>
<td>7</td>
<td>80</td>
<td>232</td>
<td>1</td>
<td>0</td>
<td>232</td>
<td>31</td>
<td>263</td>
</tr>
<tr>
<td>8</td>
<td>13</td>
<td>245</td>
<td>2</td>
<td>18</td>
<td>263</td>
<td>29</td>
<td>292</td>
</tr>
</tbody>
</table>

- **Average interarrival time:**
  \[ \bar{r} = \frac{r_1 + r_2 + \cdots + r_n}{n} = \frac{245}{8} = 30.625 \text{ seconds per job} \]

- **Arrival rate:** \(1/\bar{r} \approx 0.033\) jobs per second
### Job-Averaged Statistics

<table>
<thead>
<tr>
<th>$i$</th>
<th>$r_i$</th>
<th>$a_i$</th>
<th>$n(a_i + \epsilon)$</th>
<th>$w_i$</th>
<th>$b_i$</th>
<th>$s_i$</th>
<th>$c_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>15</td>
<td>15</td>
<td>1</td>
<td>0</td>
<td>15</td>
<td>43</td>
<td>58</td>
</tr>
<tr>
<td>2</td>
<td>32</td>
<td>47</td>
<td>2</td>
<td>11</td>
<td>58</td>
<td>36</td>
<td>94</td>
</tr>
<tr>
<td>3</td>
<td>24</td>
<td>71</td>
<td>2</td>
<td>23</td>
<td>94</td>
<td>34</td>
<td>128</td>
</tr>
<tr>
<td>4</td>
<td>40</td>
<td>111</td>
<td>2</td>
<td>17</td>
<td>128</td>
<td>30</td>
<td>158</td>
</tr>
<tr>
<td>5</td>
<td>12</td>
<td>123</td>
<td>3</td>
<td>35</td>
<td>158</td>
<td>38</td>
<td>196</td>
</tr>
<tr>
<td>6</td>
<td>29</td>
<td>152</td>
<td>3</td>
<td>44</td>
<td>196</td>
<td>30</td>
<td>226</td>
</tr>
<tr>
<td>7</td>
<td>80</td>
<td>232</td>
<td>1</td>
<td>0</td>
<td>232</td>
<td>31</td>
<td>263</td>
</tr>
<tr>
<td>8</td>
<td>13</td>
<td>245</td>
<td>2</td>
<td>18</td>
<td>263</td>
<td>29</td>
<td>292</td>
</tr>
</tbody>
</table>

- **Average service time:** $\bar{s} = 33.875$ seconds per job
  - **Service rate:** $1/\bar{s} \approx 0.030$ jobs per second

- **Arrival rate:** $1/\bar{r} \approx 0.033$ jobs per second
  - **Service rate:** $1/\bar{s} \approx 0.030$ jobs per second
### Traffic Intensity

<table>
<thead>
<tr>
<th>$i$</th>
<th>$r_i$</th>
<th>$a_i$</th>
<th>$n(a_i + \epsilon)$</th>
<th>$w_i$</th>
<th>$b_i$</th>
<th>$s_i$</th>
<th>$c_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>15</td>
<td>15</td>
<td>1</td>
<td>0</td>
<td>15</td>
<td>43</td>
<td>58</td>
</tr>
<tr>
<td>2</td>
<td>32</td>
<td>47</td>
<td>2</td>
<td>11</td>
<td>58</td>
<td>36</td>
<td>94</td>
</tr>
<tr>
<td>3</td>
<td>24</td>
<td>71</td>
<td>2</td>
<td>23</td>
<td>94</td>
<td>34</td>
<td>128</td>
</tr>
<tr>
<td>4</td>
<td>40</td>
<td>111</td>
<td>2</td>
<td>17</td>
<td>128</td>
<td>30</td>
<td>158</td>
</tr>
<tr>
<td>5</td>
<td>12</td>
<td>123</td>
<td>3</td>
<td>35</td>
<td>158</td>
<td>38</td>
<td>196</td>
</tr>
<tr>
<td>6</td>
<td>29</td>
<td>152</td>
<td>3</td>
<td>44</td>
<td>196</td>
<td>30</td>
<td>226</td>
</tr>
<tr>
<td>7</td>
<td>80</td>
<td>232</td>
<td>1</td>
<td>0</td>
<td>232</td>
<td>31</td>
<td>263</td>
</tr>
<tr>
<td>8</td>
<td>13</td>
<td>245</td>
<td>2</td>
<td>18</td>
<td>263</td>
<td>29</td>
<td>292</td>
</tr>
</tbody>
</table>

- **Average service time**: $\bar{s} = 33.875$ seconds per job
- **Service rate**: $1/\bar{s} \approx 0.030$ jobs per second

**Arrival rate**: $1/\bar{r} \approx 0.033$ jobs per second
- **Service rate**: $1/\bar{s} \approx 0.030$ jobs per second
**Time-averaged statistics**: area under a curve (integration)

- $n(t)$: number in the system
  - $\bar{n} = 1.435$
- $q(t)$: number in the queue
  - $\bar{q} = 0.507$
- $x(t)$: number in service
  - $\bar{x} = 0.928$
Given the algorithms, what are the \textit{stochastic} components?
Given the algorithms, what are the *stochastic* components?
> data(tylersGrill)
> arrivals = tylersGrill$arrivalTimes

> plot((arrivals / 3600) + 7.5, 1:length(arrivals), type = "s")
> abline(v = seq( 8.25, 11.50, by = 1 + (5 / 60)), lty = "dashed")
> abline(v = seq(12.50, 15.75, by = 1 + (5 / 60)), lty = "dashed")
Model Development At Three Levels

1. **Conceptual**: big picture, what questions to ask (done!)

2. **Specification**: equations, (psuedocode) algorithms (done!)

3. **Computational**: implement in code
Program `ssq()` implemented in R package `simEd`:
- To use, type `library(simEd)` on R/RStudio startup
- Interarrival times: `exponential(1.0)` `vexp(1, rate = 1)`
- Service times: `exponential(0.9)` `vexp(1, rate = 10/9)`

- Investigate **transient** behavior
  - Fix # of processed jobs, replicate using same initial state
  - Each replication uses different initial seed

- Investigate **steady-state** behavior
  - Will the statistics converge independent of initial seed?
  - How many jobs until steady-state?
Running \texttt{ssq} via R

```
> ssq(maxArrivals=1000, seed=1234567)
$customerArrivals
[1] 1000

$customerDepartures
[1] 1000

$simulationEndTime
[1] 996.19928

$averageWait
[1] 6.2324

$averageSojourn
[1] 7.1231

$avgNumInSystem
[1] 7.1503

$avgNumInQueue
[1] 6.2562

$utilization
[1] 0.89415

> ssq(maxArrivals=1000, seed=8675309)
$customerArrivals
[1] 1000

$customerDepartures
[1] 1000

$simulationEndTime
[1] 1022.81829

$averageWait
[1] 4.7171

$averageSojourn
[1] 5.6327

$avgNumInSystem
[1] 5.507

$avgNumInQueue
[1] 4.6118

$utilization
[1] 0.89518

?ssq gives R help
```
Convergence To Steady-State

- The accumulated $\bar{\sigma}$ printed every 100 jobs

- Initial Seed
  - 1234567
  - 5551212
  - 8675309

- Steady state average sojourn time is
  $$\frac{1}{\mu - \lambda} = \frac{1}{10/9 - 1} = 9$$

- Convergence is slow, erratic, and dependent on initial seed
Queue discipline: used to select a job for service

- **FIFO** – first in, first out (FCFS)
- **LIFO** – last in, first out
- **SIRO** – serve in random order
- **Priority** – typically shortest job first (SJF)

Assumptions:

- Service is *non-preemptive*
- Service is *conservative*
- Server takes no breaks
- Queue can be infinite length (for now)
Queue Terminology

- An $M/G/k$ service node:
  - $M$: distribution of interarrival times
  - $G$: distribution of service times
  - $k$: number of servers

- Standard Notation:
  - $M$: exponential
  - $D$: deterministic (not stochastic)
  - $E$: Erlang
  - $G$: general (any others)

- Examples:
  - $M/M/1$: exponential interarrival and service times, one server
  - $M/G/4$: exponential interarrivals, four (identical) servers, each with same general service time distribution
  - ssq: $M/M/1$ service node by default
1. Write R functions for three different gamma service processes:

   ```r
   getSvc1 = function() { rgamma(1, shape = 1.0, scale = 0.9) }
   getSvc2 = function() { rgamma(1, shape = 1.05, scale = 0.9) }
   getSvc3 = function() { rgamma(1, shape = 1.1, scale = 0.9) }
   ```

2. Experiment using the original and new service processes:

   ```r
   ssq(maxArrivals = 10000, seed = 1234567)
   ssq(maxArrivals = 10000, seed = 1234567, serviceFcn = getSvc1)
   ssq(maxArrivals = 10000, seed = 1234567, serviceFcn = getSvc2)
   ssq(maxArrivals = 10000, seed = 1234567, serviceFcn = getSvc3)
   ```

   What effects do the new service processes have on the output statistics? What if you increase the number of arrivals?