Queueing Theory and Simulation, lecture 4

Nicky van Foreest

June 22, 2021

Contents

1 General things

1.1 General things

- Assignments: please sign up in time, we have some 100 groups...
- Assignments: if too late, max grade a 6.
- Tutorials

2 Concepts: Rate, stability, load, section 3.3

2.1 Why long-run characterization?

- We have recursions to characterization of the state of a queueing at a particular time
- To know the state at any time, compute the full recursion up to that point. This is a lot of work
- There is no simple expression for the state as a function of time
- we focus on time-averages

2.2 Arrival Rate

See the book for the details. This is a sketch of ideas.

$$\begin{aligned}
\{X_k\} &\quad \text{Basis data} && (1)\\
A_k &= A_{k-1} + X_k, && (2)\\
A(t) &= \max\{n : A_n \leq t\} && (3)\\
\lambda &= \lim_{t\to\infty} \frac{A(t)}{t} && (4)\\
\frac{A_n}{n} &= \frac{A_n}{A(A_n)} \approx \frac{t}{A(t)} && (5)\\
\mathsf{E}\left[X\right] &= \lim_{n\to\infty} \frac{1}{n}\sum_{k=1}^{n} X_k = \lim_{n\to\infty} \frac{A_n}{n} = \frac{1}{\lambda}. && (6)
\end{aligned}$$

2.3 Departure rate

$$\begin{aligned}
D(t) &\leq A(t) && (7)\\
&\Longrightarrow && (8)\\
\delta &= \lim_{t\to\infty} \frac{D(t)}{t} \leq \lim_{t\to\infty} \frac{A(t)}{t} = \lambda. && (9)
\end{aligned}$$

2.4 Service rate

$$\begin{aligned}
\{S_k\} &\quad \text{Basis data} && (10)\\
U_k &= U_{k-1} + U_k, && (11)\\
U(t) &= \max\{n : U_n \leq t\} && (12)\\
\mu &= \lim_{t\to\infty} \frac{U(t)}{t} && (13)\\
\mathsf{E}\left[S\right] &= \lim_{n\to\infty} \frac{1}{n}\sum_{k=1}^{n} U_k = \frac{1}{\mu}. && (14)
\end{aligned}$$

2.5 Load and utilization

- load: $\lambda\,\mathsf{E}\left[S\right]$, rate at which work arrives
- utilization: $\rho = \lambda\,\mathsf{E}\left[S\right]/c$ for the $G/G/c$ queue
- only when $c = 1$: ρ is equal to the load.

3 KPIs, section 3.4

3.1 KPIs sampled at/observed by job arrival times.

$$\mathsf{E}\left[W\right] = \lim_{n\to\infty} \frac{1}{n} \sum_{k=1}^{n} W_k \tag{15}$$

$$\mathsf{P}\left[W \leq x\right] = \lim_{n\to\infty} \frac{1}{n} \sum_{k=1}^{n} I_{W_k \leq x} \tag{16}$$

$$\tag{17}$$

3.2 KPIs time averages

$$L(s) = A(s) - D(s) \tag{18}$$

$$\mathsf{E}\left[L\right] = \lim_{t\to\infty} \frac{1}{t} \int_0^t L(s)\,\mathrm{d}s, \tag{19}$$

We can also define the average as seen by job arrivals:

$$L(k) = A(A_k) - D(A_k) \tag{20}$$

$$\mathsf{E}\left[L\right] = \lim_{n\to\infty} \frac{1}{n} \sum_{k=1}^{n} L_k \tag{21}$$

$$\tag{22}$$

These two $\mathsf{E}\left[L\right]$ are not necessarily the same. As yet I haven't found good notation to make a clear distinction. Be *very* careful!

4 Convergence to steady state

4.1 Dealing with random variables

Python Code

```python
W = RV({5: 1})
X = RV({1: 1 / 3, 2: 1 / 3, 4: 1 / 3,})
S = RV({1: 1 / 3, 2: 1 / 3, 3: 1 / 3,})

for n in range(1, 21):
    W += S - X
    W = W.plus()
```

Recall:

$$\mathsf{P}\left[S - X = i\right] = \sum_k \mathsf{P}\left[S = i + k\right] \mathsf{P}\left[X = k\right].$$

Thus, I need a $+$ operator for $X+Y$, a $-$ operator for $X-Y$, and a function $[X]^+ = \max\{X, 0\}$.

4.2 Python inheritance to the rescue

Python Code

```python
from collections import defaultdict
import operator


class RV(defaultdict):
    def __init__(self, p=None):
        super().__init__(float)
        if p:
            for (i, pi,) in p.items():
                self[i] = pi
```

4.3 Addition

Python Code

```python
    def apply_operator(self, Y, op):
        R = RV()
        for (i, pi,) in self.items():
            for (j, pj,) in Y.items():
                R[op(i, j)] += pi * pj
        return R

    def __add__(self, X):
        return self.apply_operator(X, operator.add)
```

4.4 Plus

Python Code

```python
    def apply_function(self, h):
        R = RV()
        for (i, pi,) in self.items():
            R[h(i)] += pi
        return R


    def plus(self):
        return self.apply_function(lambda x: max(x, 0))
```

4.5 Confession

- For fully working code, see the book.
- With some python knowledge it's fairly easy to compute $\{W_k\}$.
- Now you have to confess that all this abstraction in python is fantastic. So, please type in the chat box how much you like this :-)