Overview – Little’s Result
- formalizes intuition
- applies to open or closed systems
- useful for:
  - computing measures e.g., given \( X \) and \( R \), compute \( N \)
  - verifying measured values
  - deriving new results & insights
  - providing new system functionality

Outline
- Little’s Result
  - Utilization law
  - Interactive response time law
  - Multiple Classes

Little’s Result: \( N = \lambda R \)
- \( \lambda = \text{arrival rate (customers/sec)} \)
- \( N = \text{average number of customers in the system} \)
- \( R = \text{average total system residence time (sec)} \)
- e.g., if \( \lambda = 3 \text{ customers/minute} \) and \( R = 2.5 \text{ minutes} \),
  then \( N = 3 \text{ customers/min} \times 2.5 \text{ min} = 7.5 \text{ customers} \)
  (holds for open or closed system)

\( N = \lambda R: \text{ M.S. students in C.S.} \)
- \( \lambda = 65 \text{ students/yr} \)
- \( N = \text{average number of students in the Dept} \)
- \( R = \text{is independent of } N \)

What is the value of \( N \)?
- if we know \( R \), then \( N = \lambda R \), but what if we’re not given \( \lambda \)?
- Assign a reasonable value for \( R \): e.g., \( R = 2 \) (modeling)
- \( N = 65 \text{ students/yr} \times 2 \text{ yrs} = 130 \text{ students} \)
- What if \( R = 2.25 \) (sensitivity analysis)

\( N = \lambda R: \lambda > X \)
- \( \lambda = \text{arrival rate (customers/sec)} \)
- \( N = \text{average number of customers in the system} \)
- \( R = \text{average total system residence time (sec)} \)

if \( \lambda > X \):
- then \( N = \infty \) (open system)
- \( R = \infty \)
\[ N = \lambda R: \lambda = X \]

\[ \gamma = \text{accumulated customer-seconds in } [0, T] \]

\[ \frac{\gamma}{T} = X R = \frac{C}{T} \times \left( \frac{1}{C} \right) \]

(holds for open or closed system).

\[ N \times R: \text{ Computing Measures} \]

Application:

compute from NCSA O2K job trace:
- \( \lambda, X \); average wait time (W); average runtime (T)
- if \( \lambda = X \), use Little's result to estimate:
  - average number waiting to run = \( X W \)
  - average number waiting & running = \( X(W+T) \)
- question: what if \( \lambda \neq X \)?

\[ N = XR: \text{ single-server queue} \]

- infeasible: \( S = 3 \text{ seconds}, \lambda = 0.5/\text{second}, X = \min(\lambda, 1/S) = 1/S \)
- feasible: \( \lambda < 1/S \)

\[ Q = \lambda R = XR \]

measure \( X = C/T \), \( R = \frac{1}{X} \sum R \)

and compute \( Q \).

\[ N = XR: \text{ server} \]

By Little's result,
\[ XS = U \]

\( "\text{Utilization Law}" \)

\[ U = XS: \text{ Numerical Examples} \]

if \( X = 5 \text{ customers/second} \) and \( U = 0.75 \),
what is the average service requirement, \( S \)?

if \( X = 5 \text{ customers/second} \) and \( S = 100 \text{ milliseconds} \),
what is the server utilization, \( U \)?
\[ U = XS: \text{ bound on feasible } \lambda \]

- infeasible: 
  \[ S = 3 \text{ seconds, } \lambda = 0.5/\text{second}: \lambda S = 1.5 > 100\% \]
- feasible: 
  \[ \lambda S < 1 \Rightarrow \lambda < 1/S \] (formalizes intuition)

**U = XS: Insight**

Application: system measures 
\[ Q = X(W + S) = q + U \neq q + 1 \]

**N = XR: machine repair model**

- \[ U = XS: X < 1/S \]
- \[ Q = XR: \]
  a) measure \[ X = C/T \] and \[ R = \frac{C}{Z} \] compute \[ Q \]
  b) what if we just measure \[ X \]?
  strategy: look for another bounding box

**Interactive Response Time Law**

by Little's result, 
\[ N = X(R + Z) \]

**N = X(R+Z): Application**

For a system with 64 active workstations, the average time between requests to the file server is 3 seconds, and the average response time is 0.2 seconds.

Question: given: \[ N = 64, Z = 3 \text{ sec, } R = 0.2 \text{ sec} \]

What is the average no. of requests at the file server?
\[ X = N/(R+Z) = 64/3.2 = 20 \text{ requests/second} \]
\[ Q = XR = 20 \times 0.2 = 4 \text{ requests} \]
N = XR: types of applications

key types of applications:
- compute measures, i.e., given X and R, compute N
- check the correctness of measured data
- system capacity planning
- insight — e.g., SRPT scheduling minimizes N ⇒ min. R
- new system functionality

Visit Counts - review

- \( V_k \) = average no. of visits to node k
  - e.g., \( V_{cpu} = 100 \), \( V_{disk} = 39.6 \)
- \( D_k = V_k S_k \) = average total demand at node k
  - e.g., \( S_{cpu} = 10 \text{ msec}, D_{cpu} = 1 \text{ sec} \)

Forced Flow Law: \( X_k = V_k X \)

e.g., if \( V_{cpu} = 5 \) and \( X = 10/\text{min} \)
\( X_{cpu} = 5 \times 50/\text{min} = 2,500 \times 5/\text{min} = 12,500 \text{ requests/sec} \)

System Capacity Planning

media server:
- 30-min videos, 300 Kb/s, 8 disks w. balanced load,
- 4K i/o blocks, projected λ: 5 customers/minute
- \( V_{disk} = (1800 \times 300 / 4 \times 8) / 8 = 2,100 \)
- \( X_{disk} = 2,100 \times 5 = 10,500 \text{ requests/sec} \)

New Functionality: TCP Vegas

- Algorithm for setting window (W)
  - backlog = \( W \frac{RTT}{RTT - \text{baseRTT}} \)
  - Slow Start: exit if backlog > \( \gamma \)
  - every RTT in congestion avoidance:
    - \( W^{++} \), if backlog < \( \alpha \)
    - \( W = W \), if \( \alpha < \text{backlog} < \beta \)
    - \( W^{--} \), if backlog > \( \beta \)
  - (i.e., reduce W before loss occurs)

Visit Counts - review

- \( V_k \) = average no. of visits to node k
  - e.g., \( V_{cpu} = 100 \), \( V_{disk} = 39.6 \)
- \( D_k = V_k S_k \) = average total demand at node k
  - e.g., \( S_{cpu} = 10 \text{ msec}, D_{cpu} = 1 \text{ sec} \)

Forced Flow Law: \( X_k = V_k X \)

e.g., if \( V_{cpu} = 5 \) and \( X = 10/\text{min} \)
\( X_{cpu} = 5 \times 50/\text{min} = 2,500 \times 5/\text{min} = 12,500 \text{ requests/sec} \)

System Capacity Planning

media server:
- 30-min videos, 300 Kb/s, 8 disks w. balanced load,
- 4K i/o blocks, projected λ: 5 customers/minute
- \( V_{disk} = (1800 \times 300 / 4 \times 8) / 8 = 2,100 \)
- \( X_{disk} = 2,100 \times 5 = 10,500 \text{ requests/sec} \)

New Functionality: TCP Vegas

- Algorithm for setting window (W)
  - backlog = \( W \frac{RTT}{RTT - \text{baseRTT}} \)
  - Slow Start: exit if backlog > \( \gamma \)
  - every RTT in congestion avoidance:
    - \( W^{++} \), if backlog < \( \alpha \)
    - \( W = W \), if \( \alpha < \text{backlog} < \beta \)
    - \( W^{--} \), if backlog > \( \beta \)
  - (i.e., reduce W before loss occurs)
Consider a computer system with a disk that serves page fault requests. Suppose the utilization of the paging disk is 80% and that each page fault request requires an average of 10 milliseconds of service at the paging disk.

If the system throughput is 5 transactions/sec, what is the average number of page faults per transaction?

\[ \text{given: } U_{\text{disk}} = 0.8, \ S_{\text{disk}} = 0.01 \text{ sec} \]

\[ X_{\text{disk}} = \frac{U_{\text{disk}}}{S_{\text{disk}}} = \frac{0.8}{0.01} = 80 \text{ page faults/second} \]

\[ V_{\text{disk}} = \frac{X_{\text{disk}}}{X} = \frac{80}{5} = 16 \text{ page faults per transaction} \]
Summary

- Forced Flow Law, \( X_k = V_k X \)
- Little's Result, \( N = XR, \ N_A = X_A R_A \)
- Utilization Law: \( U_k = X_k S_k = X D_k \)
- Interactive Response Time Law: \( N = X(R+Z) \)

- Many useful applications
  - System design and capacity planning
  - Computing measures
  - Obtaining new insight
  - Checking the correctness of measured parameters

Questions?

CS 547: next steps

- Read LZGS, Chapter 5 for Wednesday
- Homework #2 due 9/29
- Keep up with reading & homeworks

Memory Constrained Systems

- Requests queue for a memory partition prior to competing for CPU and I/O resources
  - \( R = \text{total average residence time in the system} \)
  - \( N = X(R+Z) \) (interactive response time law still holds)
  - \( U_k = X D_k \) (utilization law still holds)
- See the example on pages 49-50 of LZGS