Lecture 1

Prof. Krishna R. Pattipati

Dept. of Electrical and Computer Engineering
University of Connecticut
Contact: krishna@engr.uconn.edu (860) 486-2890

ECE 336
Stochastic Models for the Analysis of Computer Systems and Communication Networks

Introduction

- Course Overview
- Queuing Models
- Little's Theorem
- Applications

Course Mission or Goal

- Provide systems analysts with central concepts of widely used **performance and reliability models** of <u>complex</u> computer systems and communication networks

- State of the art algorithms and theoretical results in **<u>queuing networks</u>** and **<u>Markov chain</u> models of reliability**

- Numerous applications in **complex computer systems and communication networks**

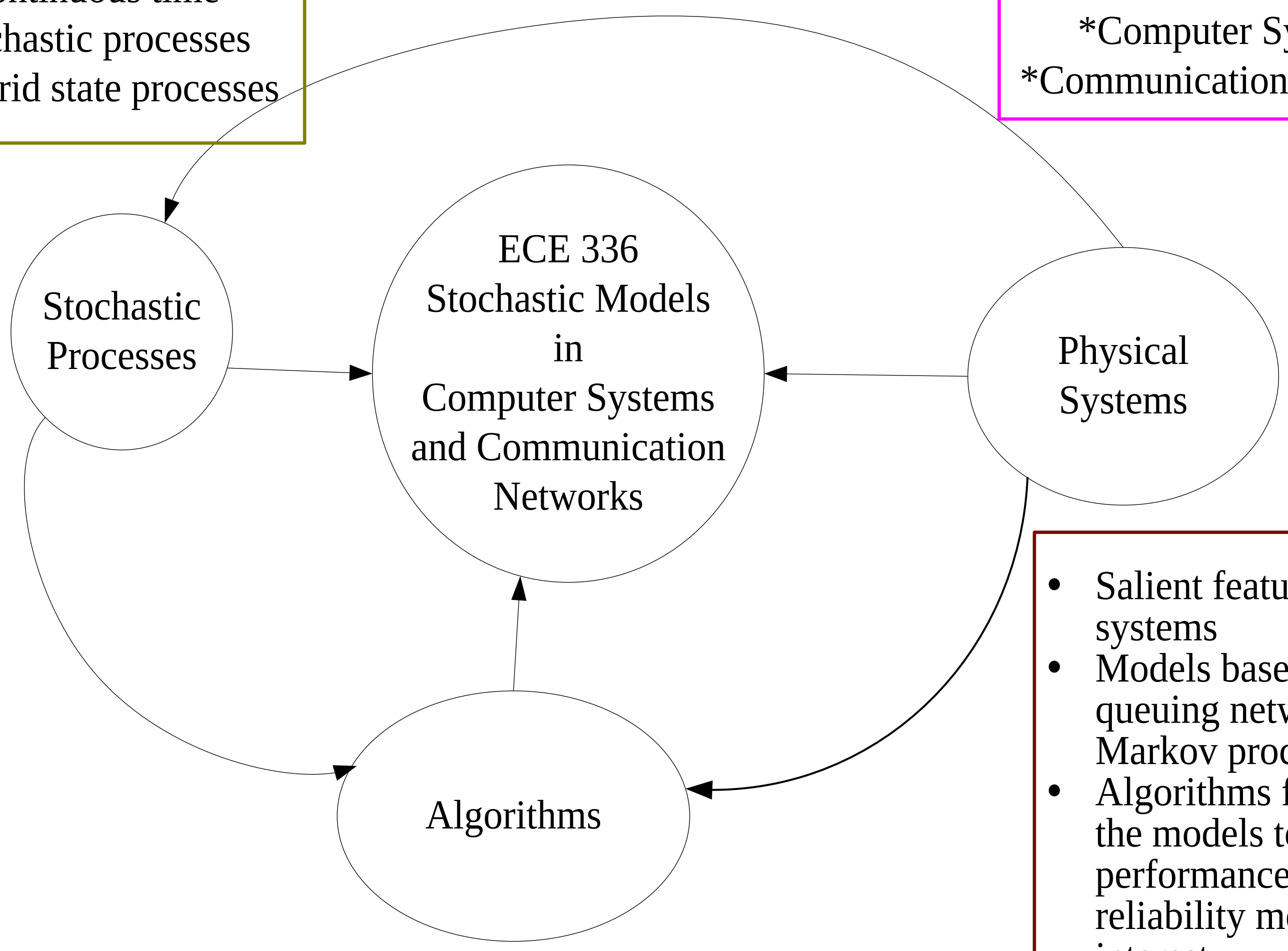
Background Requirements:

Stochastic processes and probability theory (ECE 313)

Three Recurrent Themes

*Discrete state
continuous time
stochastic processes
*Hybrid state processes

*Computer Systems
*Communication Networks



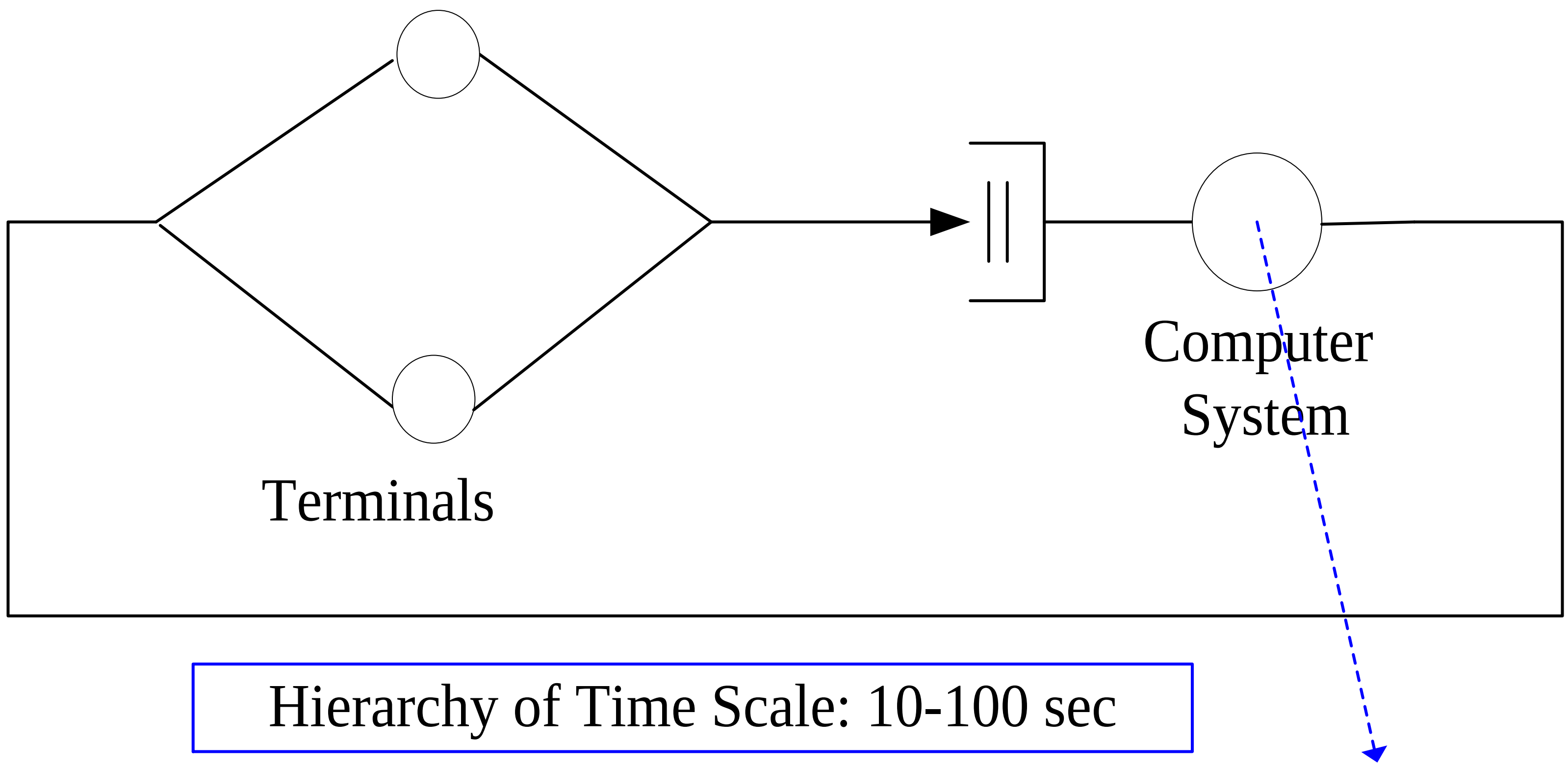
- Salient features of systems
- Models based on queuing networks and Markov processes
- Algorithms for solving the models to obtain performance and reliability measures of interest

What do we mean by complexity?

Complexity is in the eye of the beholder. For us, it means that the system has four basic attributes.

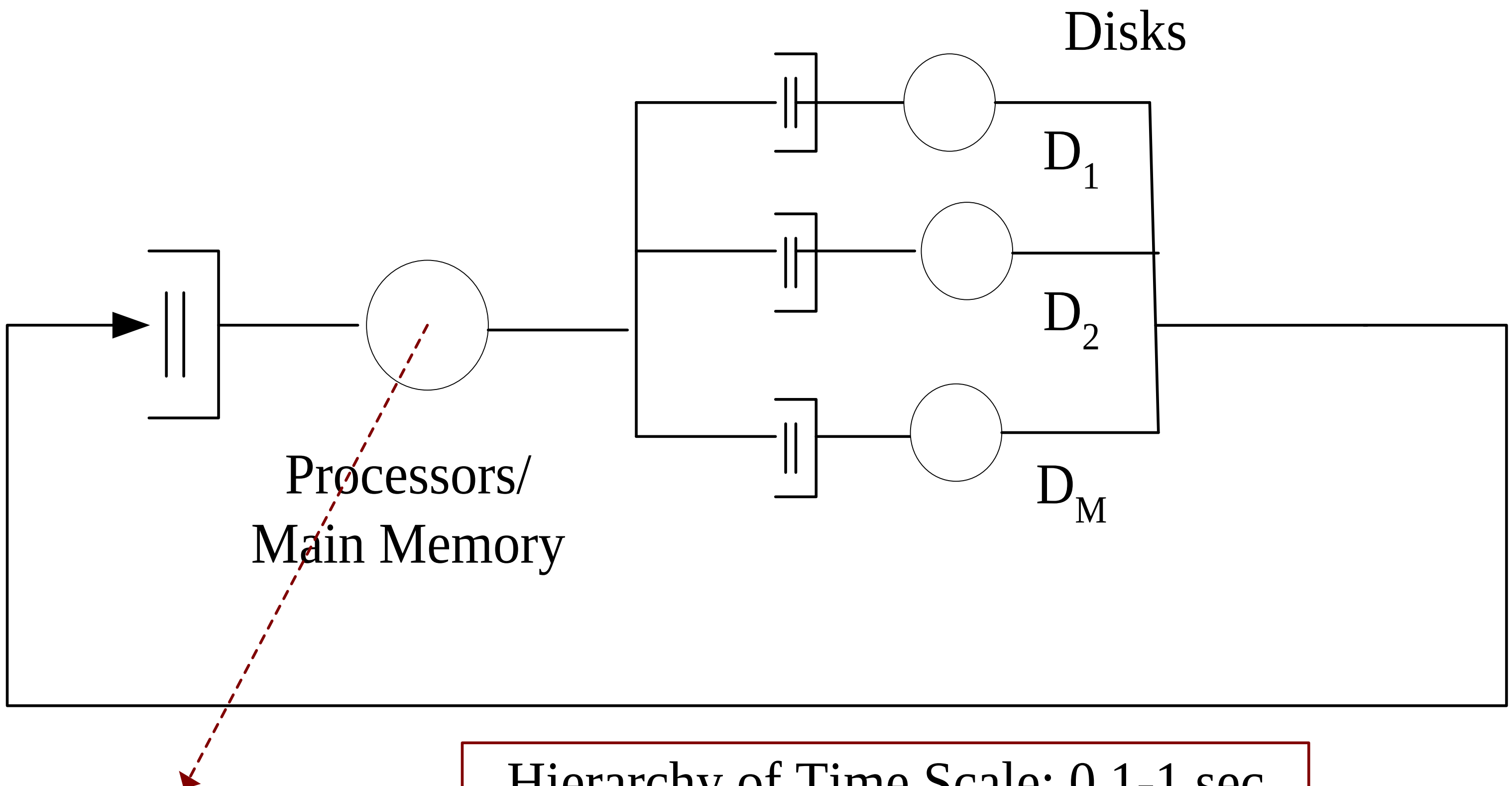
- A collection of **interconnected components or resources** (i.e., a network of resources). Resources are also termed **nodes** or **vertices**
- Provide **service** to a community of users (users can be humans or non-humans)
- Users **compete** for the network (system) resources (i.e., contention or **queuing for limited resources**). **Need to model queuing Delays**
- **Redundancy** for fault-tolerance. **Need to model reliability**

Hierarchical Network of Resources -1



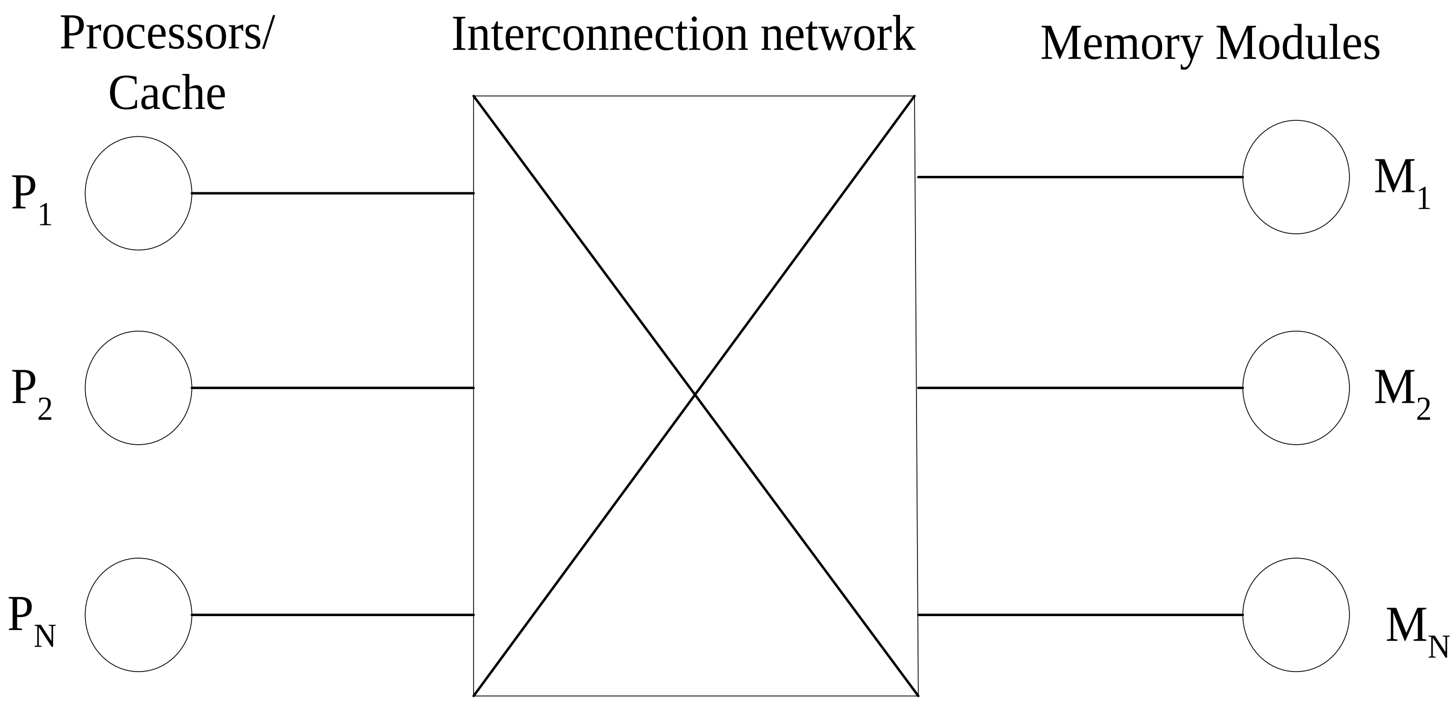
Hierarchy of Time Scale: 10-100 sec

Hierarchical Network of Resources - 2



Hierarchy of Time Scale: 0.1-1 sec

Hierarchical Network of Resources - 3



Hierarchy of Time Scale: 1-100 μsec

Examples of Networks

Physical Systems		Resources or Nodes
Computing Systems	⟷	Hardware and software resources, such as CPU, cache, main memory, disks, OS
Computer-Comm. Networks	⟷	Communication links or channels, buffers, buses, computers, transmission control units
Manufacturing	⟷	Tools, storage areas, NC machines, computers, personnel, material handling systems

Primary focus of this course

Course Topics - 1

- Queuing models, Little’s theorem and applications
- Review of discrete-time and continuous-time Markov chains, Geometric and exponential distributions, the Poisson process, Uniformization
- Birth-Death Processes, M/M/1, M/M/1/N, M/M/m, M/M/∞, M/M/m/m queues and applications
- Control of M/M/1 queues: controlled service rate, controlled arrival rate, priority assignment and the μc rule
- Open (Jackson) networks and applications to capacity assignment in communication networks
- Closed (Gordon-Newell) networks, computational algorithms and applications to computer systems and flow-controlled virtual circuits
- Multi-class (closed, open, and mixed) networks, computational algorithms and applications

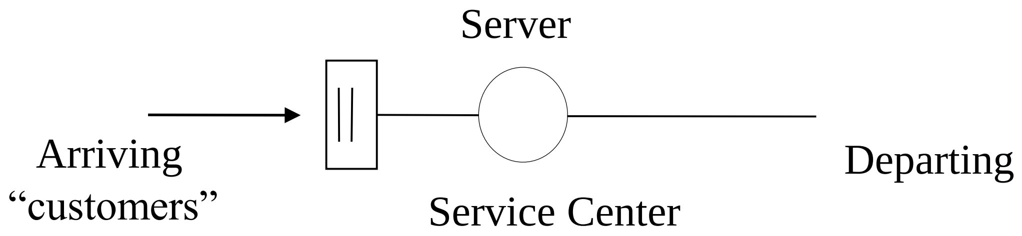
Course Topics - 2

- Approximation techniques for non-product form networks
- M/G/1 queue, M/G/1 queue with vacations and applications to reservations and polling
- Priority queues, batch arrivals, G/G/1 queue, approximations to networks
- Random access communications (Aloha, Slotted Aloha, CSMA/CD)
- Reliability models of computer systems and communication networks
- Performability (combined performance and reliability) models of fault-tolerant computer systems

Focus of Lecture1

1. How to characterize a **resource** or **node** or a **simple queue** ?
2. What are the **measures of system performance** ?
3. Fundamental Accounting identity…………….**Little's Theorem**

Basic Structure of a Node



Customers: Jobs (transactions, requests, tasks, processes, algorithms) in a computer system, messages or packets in a communication network, Parts in manufacturing

Specification of a Node

Need to specify four items:

1. Arrival pattern: customer description
    - Population size (finite for closed systems)
    - Statistical pattern of arrivals (e.g., Poisson)
    - Classes of customers
2. Service mechanism: service demand (work), processing capacity or service rate (work/unit time)

$$\text{Service Time} = \frac{\text{Service Demand}}{\text{Service Rate}}$$

3. Queuing discipline: order in which customers are served
4. Storage capacity of the server ~ buffer or holding area

Arrival Pattern -1

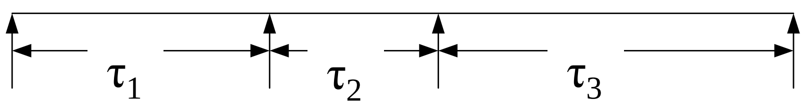
1. Customers are drawn from a population or input source

Customers

- Come from a very large population ("infinite source") also known as OPEN SYSTEM
- Come from a finite population (e.g., Interactive terminals). Also known as CLOSED SYSTEM

We will treat both cases: finite => Number in the system affects the number remaining in the input source

2. Statistical pattern of arrivals: specified by the density of inter-arrival times (e.g., exponential inter-arrival times ⇔ Poisson arrivals)



Arrival Pattern - 2

Inter-arrival Time Density

- Deterministic (or regular) arrivals
  $\tau_i = \tau$ constant $\Rightarrow$ impulse at τ
- Exponential inter-arrival time density (Poisson process)
- General independent arrivals (regenerative or renewal) processes
- General distribution (correlated arrival)

3. Different types of customers (e.g., batch and interactive, priority)

Server Mechanism

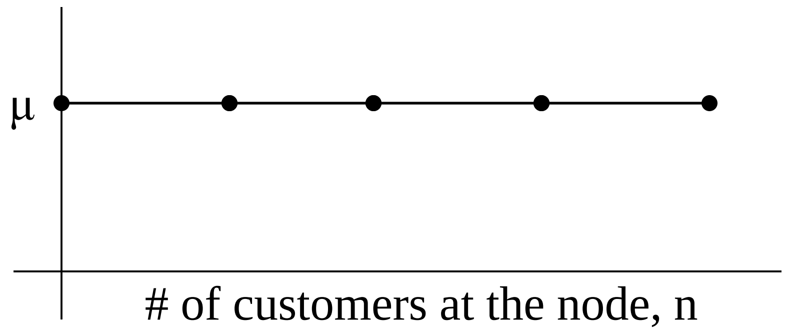
1. Service demand: amount of service required by a customer at a service center $\Rightarrow$ work to be performed, S (e.g., . computer : # of instructions to be executed; communication network: # of bits to be transmitted from node i to node j = [# of packets][# of bits/packet]
   - Different customer classes can have different service demands
   - Service demands of a given class are i.i.d.

2. Service rate or processing capacity: How fast the service center (node) processes the work
   - CPU: # of instructions/ sec
   - Comm. Link: transmission rate = # of bits/sec
   - Memory: transfer rate = # of words/sec (or) # of bits/sec

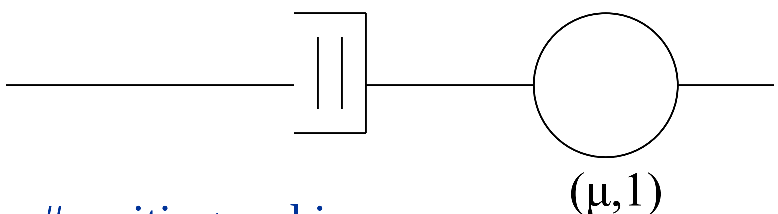
We will consider four forms of service rate functions
$\Rightarrow$ four types of nodes

Single Server Node

Single server node: service rate is a constant μ , i.e., independent of the number of customers present. Also known as fixed rate service center.

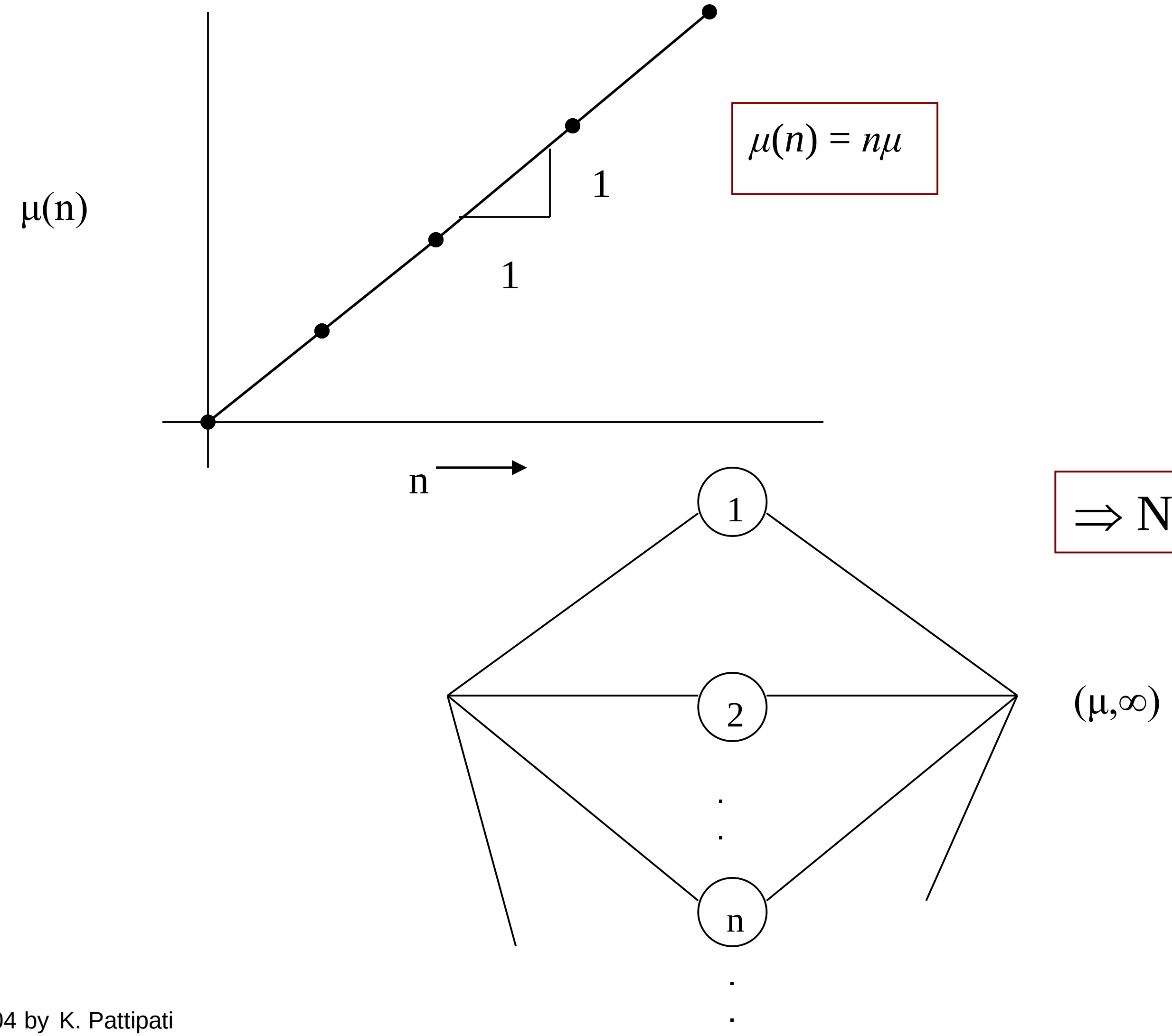


$$\mu(n) = \mu$$



n= # waiting and in service

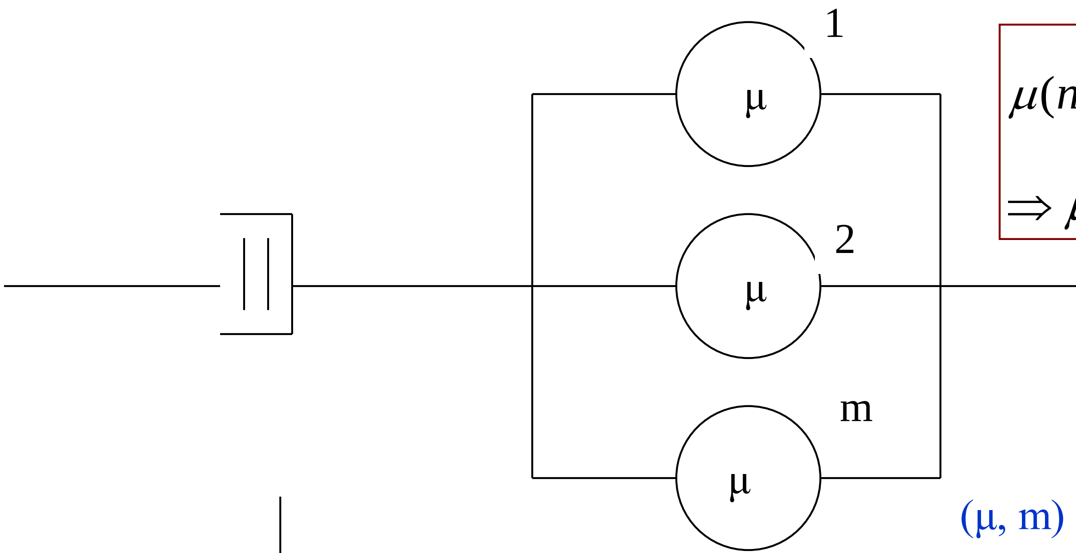
Infinite Server Node



$\mu(n) = n\mu$

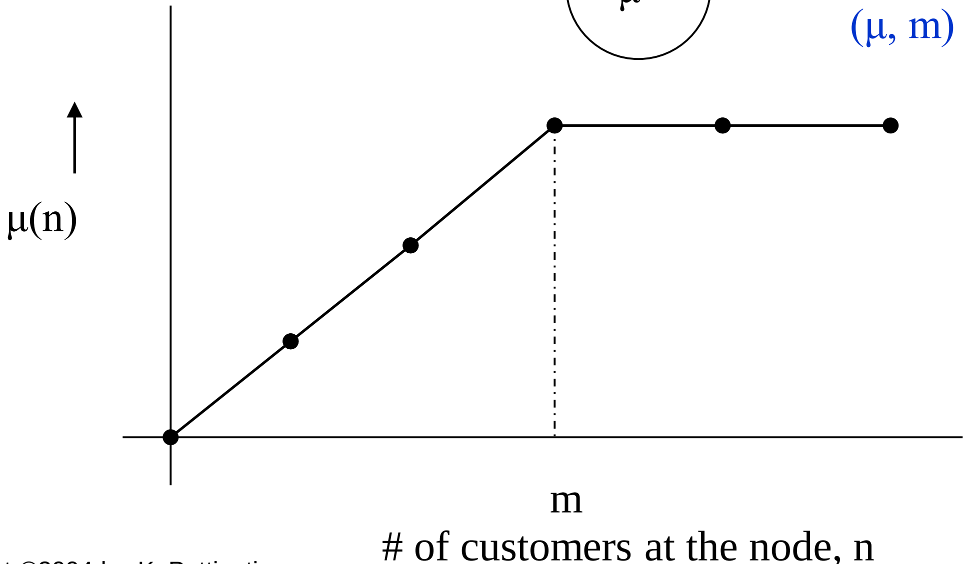
$\Rightarrow$ No waiting

Multiple Server Node

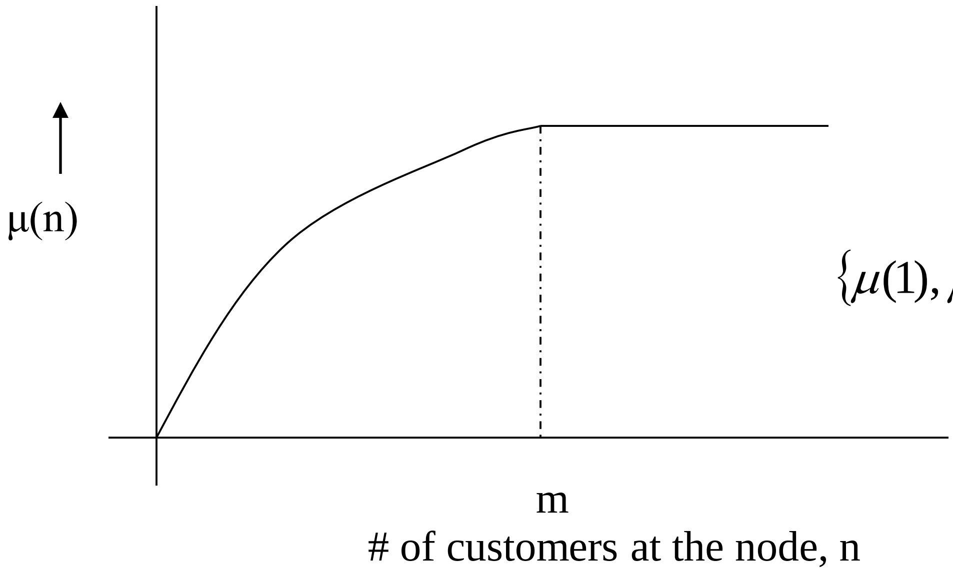


$$\mu(n) = \begin{cases} n\mu, n < m \\ m\mu, n \geq m \end{cases}$$

$$\Rightarrow \mu(n) = \min(n, m)\mu = (n \wedge m)\mu$$



State-Dependent Node



$$\{\mu(1), \mu(2), \ldots., \mu(m); m\}$$

Very useful in Hierarchical Queuing Network modeling using flow equivalent node (FEN) concept

Queuing (or Scheduling) Discipline

- An algorithm that determines the order in which customers are served

Key assumption: if there are customers waiting to be served, the server is never idle (the so called work-conserving queuing discipline)

FCFS: customers are served in the order of their arrival (LIST)

LCFS: Last come, first served (STACK)

SPT: Shortest processing time rule (minimizes the average completion time)

Priority Scheduling: Procedure that differentiates among customer types. Select the next customer to be served as one having the highest priority among all the customers waiting to be served.

Priority Scheduling -1

Two Types of priority

- Static (i.e., fixed a priori)
- Dynamic
  - State-dependent $\Rightarrow f$ (# of customers of each type waiting at the node)
  - Time-dependent $\Rightarrow f$ (elapsed time since each customer enters the system)

For both types of priority, there exist two further distinctions depending what we do with the customer being served while a higher priority customer enters the system (preemptive versus non-preemptive)

Types of Preemption

Type of preemption

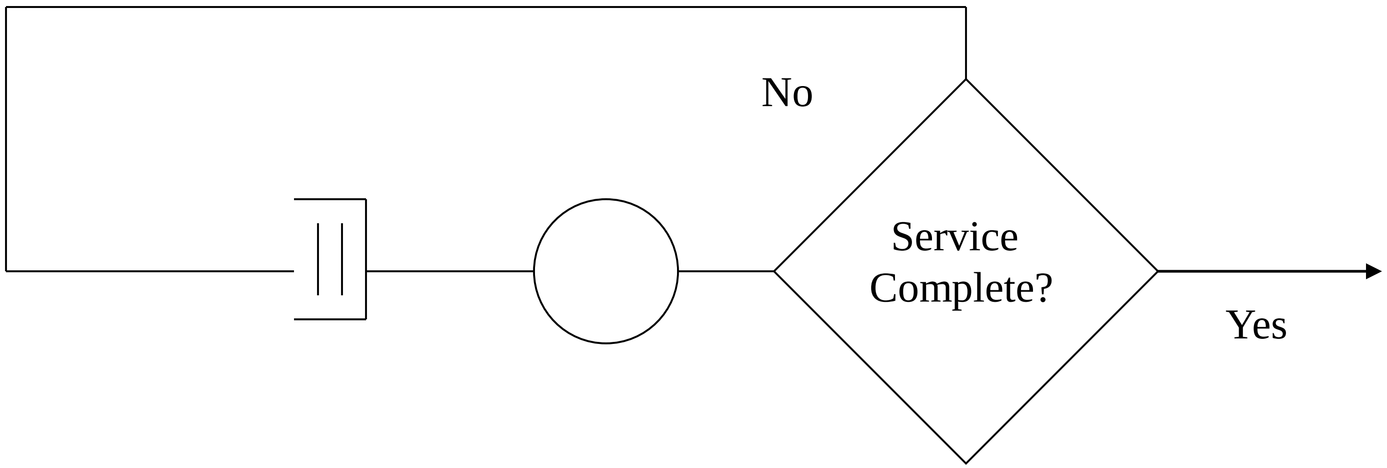
- Non-preemptive (service once begun for a low priority customer is never interrupted)
- preemptive (high priority customer always interrupts low priority customers)

preemptive

- Resume (resumes service at the point where service was interrupted)
- Repeat (start from scratch)

Round Robin

Round Robin: A customer is given continuous service for an amount of time called a “quantum” or “time slice”.



It is FCFS, but server gives service for a “time slice” only and at the end customer has to join the end of the queue. Widely used in computer systems, since it provides fast service to customers with small service demand at the expense of customers with large service demands. Can be thought of as “shortest-in first-out” policy.

Processor Sharing

Processor sharing:

- ❑ Analytic approximation of round robin. Make quantum $\longrightarrow 0 \Rightarrow$ if n customers (including those in service) are at the node, then service rate μ is divided equally among n customers (each gets μ/n)
- ❑ In PS, all customers receive service from a single server simultaneously with equal service rate.

Buffer Capacity

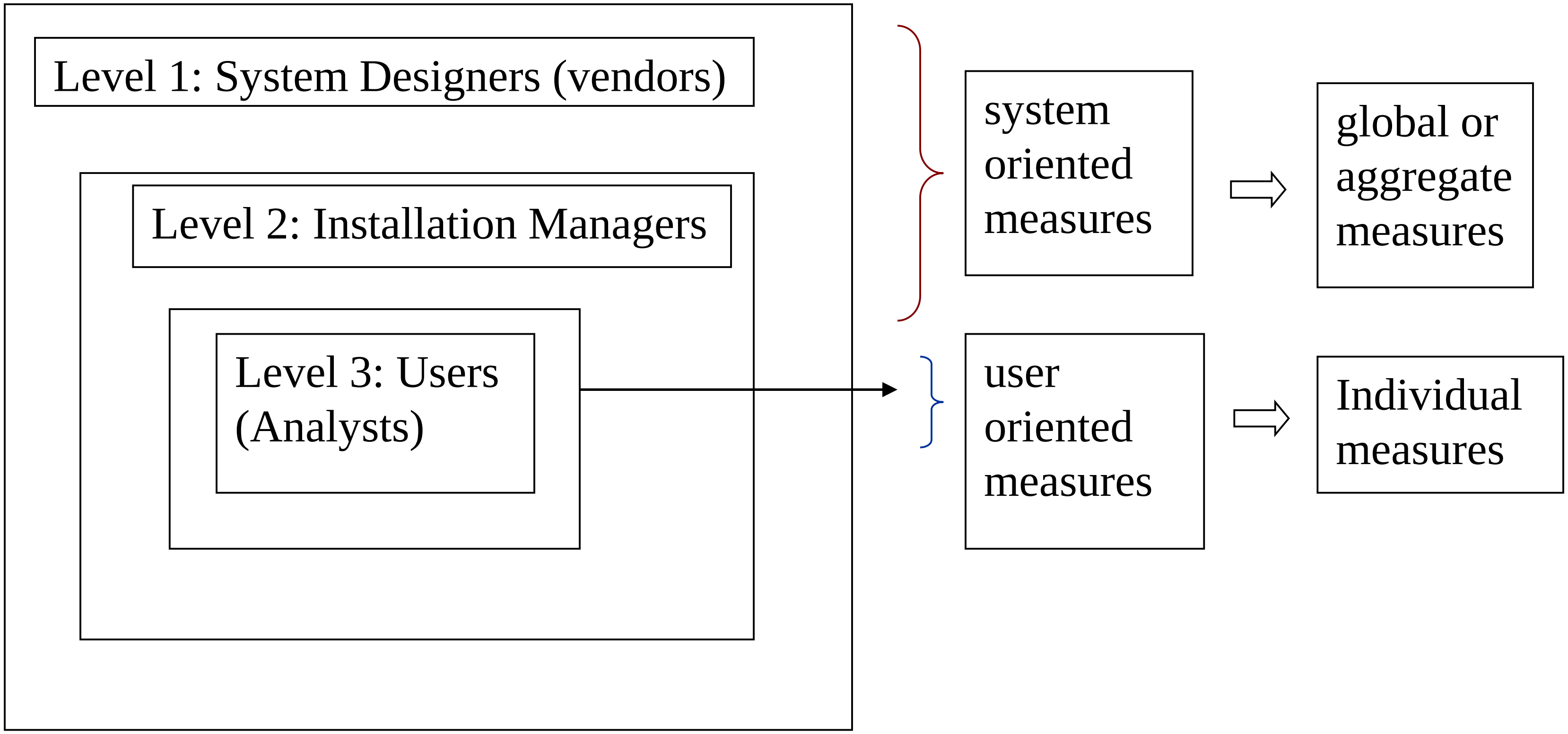
Storage: Buffer Capacity

$\Rightarrow$ **number of customers that can wait at the node**

$\Rightarrow$ **Determines the blocking probability**

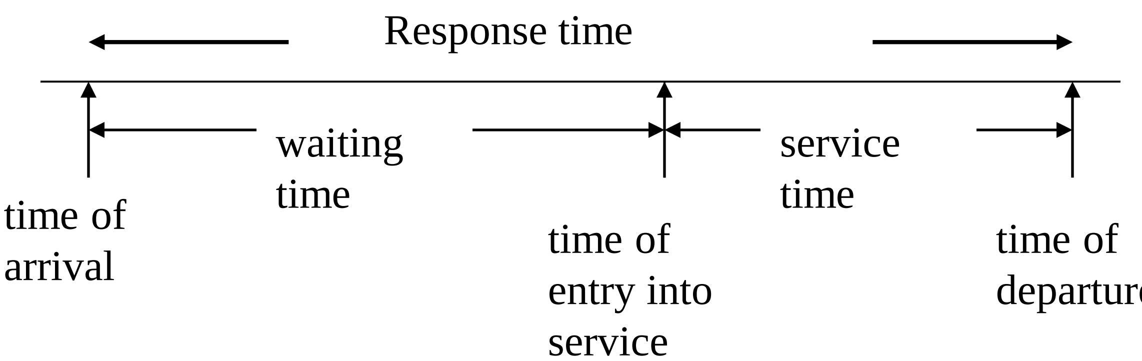
$\Rightarrow$ **Limits throughput**

Performance Measures



User-oriented Measures

1. Response time:
   E [time of departure of a customer - time of arrival of a customer]
   = Average time a customer spends at each node
   = Average waiting time + Average service time



$$R = W + \bar{t} = W + \frac{s}{\mu}$$

(Assuming a single server node)
Can also talk about system response time
$= \sum_i$ Response time at node i

2. Queue length

   Average number of customers at each node (including the customer in service) = Average number waiting + Average number in service

$$Q = Q_W + \text{Average number in service}$$

System-oriented Measures

1. Throughput

   Average number of jobs processed per unit time ⇒ a measure of productivity of the system

$$X = \frac{\text{Number of jobs completed during } (t_o\,,\, t_f)}{\text{Observation interval } (t_f - t_o)} = \frac{C}{T}$$

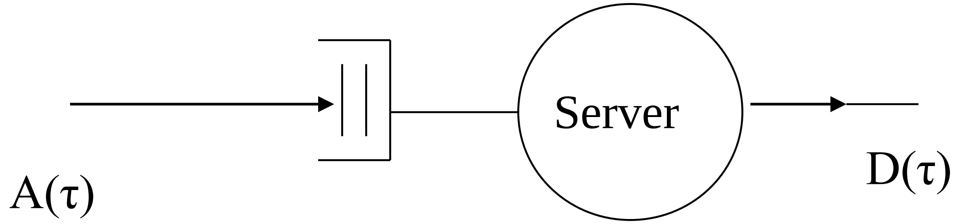
   You can also talk of nodal and system throughputs.

2. Utilization of a node

   Fraction of the time (or the probability that) the node is busy

Little's Theorem - 1

Little's Theorem (formula) is simply an accounting identity.



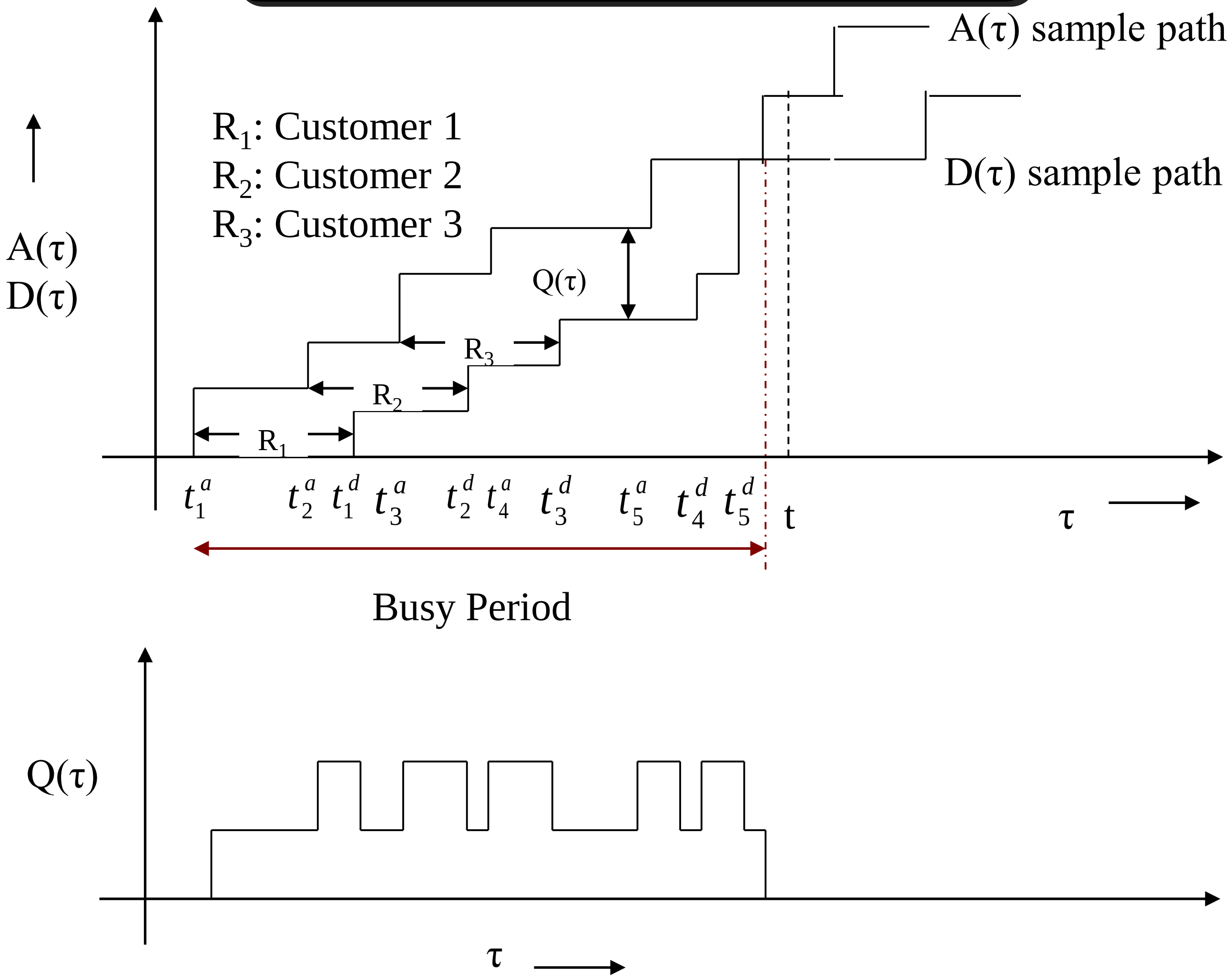
t_j^a Arrival time of j[th] customer

t_j^d Departure time of j[th] customer

$$\boxed{Q(\tau) = A(\tau) - D(\tau)}$$

Let us look at the sample paths of $A(\tau), D(\tau)$ and $Q(\tau)$

Little's Theorem - 2



Little's Theorem - 3

Note that no assumption is made on the arrival or departure distributions. Also, no assumption is necessary on the scheduling discipline. Figure assumes FCFS, but is valid for any queuing system that reaches statistical equilibrium $\Rightarrow$ busy periods must be finite or Q(τ) is "ergodic."

Little's theorem relates:

- The average number of customers in the system (i.e., the "typical" # of customers either waiting in the queue or undergoing service), *Q* system Length
- The average response time per customer (i.e., the "typical" time a customer spends waiting in the queue plus the service time), *R* in sec.
- Customer throughput in customers/sec. For open systems, we use the notation λ. For closed systems, we use the notation X.

Little's Law: $\boxed{Q = \lambda R \text{ for open systems}}$

$$\boxed{Q = X\ R \text{ for closed systems}}$$

Proof of Little's Law - 1

$Q\ (\tau)$ is ergodic $\Longrightarrow$ Time averages = Ensemble Averages

Time averaging interpretation

$$\overline{Q}(t) = \frac{1}{t}\int_0^t Q(\tau)\,d\tau$$

and steady-state average $\Longrightarrow$ $\overline{Q}(t) = \lim_{t\to\infty} \frac{1}{t}\int_0^t Q(\tau)\,d\tau$

Probabilistic interpretation ….. ensemble average

Let $p_k(t)$ = Probability {k customers in the queue at time t (waiting or in service)}

$\overline{Q}(t)$ = Average number of customers in the system at time t

$$= \sum_{k=1}^{\infty} k p_k(t)$$

In steady state $\Rightarrow \lim_{t\to\infty} p_k(t) \to p_k$

$$\lim_{t\to\infty} \overline{Q}(t) \to Q = \sum_{k=0}^{\infty} k p_k$$

Proof of Little's Law - 2

Similarly, for response time, let $\overline{R}_k$ denote the average delay of k^{th} customer. In the steady state:

Probabilistic interpretation:

$$R = \lim_{k\to\infty} \overline{R}_k$$

From sample paths

Also,

$$R = \lim_{k\to\infty} \frac{1}{k}\sum_{i=1}^{k} R_i = \lim_{k\to\infty} \overline{R}_k$$

Time average interpretation

R_i = Delay (Response time) of i^{th} customer

Also

λ = Average customer arrival rate

$$= \lim_{t\to\infty} \frac{\text{Expected number of arrivals in } [0, t]}{t}$$

Ensemble interpretation

Proof of Little's Law - 3

$$= \lim_{t\to\infty} \frac{A(t)}{t}$$

Time average interpretation

Need to prove $Q = \lambda R$

We will show for FCFS only (LCFS and arbitrary service HW problem # 2). In fact, it is valid for any scheduling discipline. Proof involves computing the area under the sample path curve in two ways:

One way: $\int_0^t Q(\tau)d\tau$

Second way: $\sum_{i=1}^{D(t)} R_i + \sum_{i=D(t)+1}^{A(t)} (t - t_i^a)$

Define $\overline{Q}(t) = \frac{1}{t}\int_0^t Q(\tau)d\tau$

= Time average of number of customers in the system in the interval [0,t]

Proof of Little's Law - 4

$$\bar{\lambda}(t) = \frac{A(t)}{t}$$ = Time average of customer arrival rate in the interval [0,t]

$$\bar{R}_{D(t)}(t) = \frac{\sum_{i=1}^{D(t)} R_i + \sum_{i=D(t)+1}^{A(t)} (t - t_i^a)}{A(t)} =$$ Time average of response time

$$\bar{Q}(t) = \bar{\lambda}(t)\bar{R}_{D(t)}(t)$$

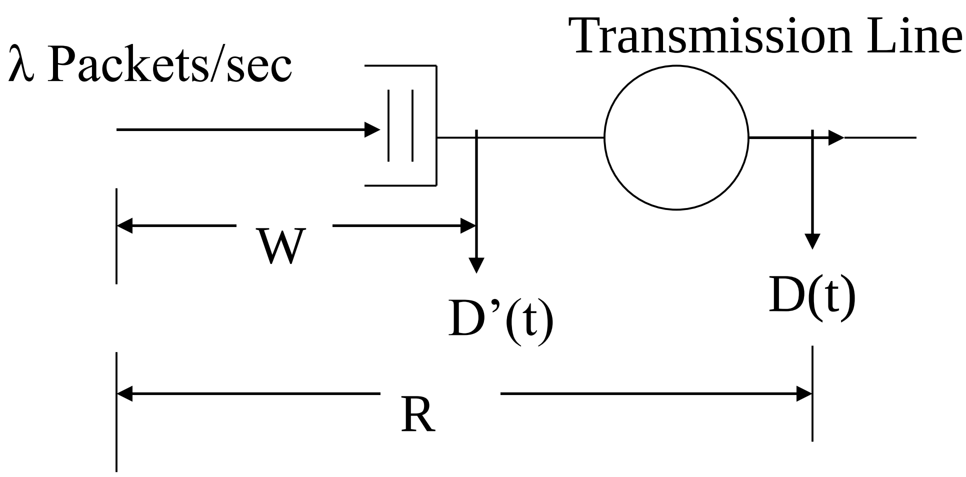
Taking $\lim_{t\to\infty}$ $\boxed{Q = \lambda\, R}$

$$\boxed{\lim_{t\to\infty} \frac{A(t)}{t} = \lim_{t\to\infty} \frac{D(t)}{t}}$$

$\boxed{\text{Arrivals = Departures as } t \to \infty}$

Applications of Little's Theorem -1

Example 1: Single server node



$$Q_W = \lambda W$$

$$U = \lambda \bar{t}$$

Utilization law is a special case of Little's formula.

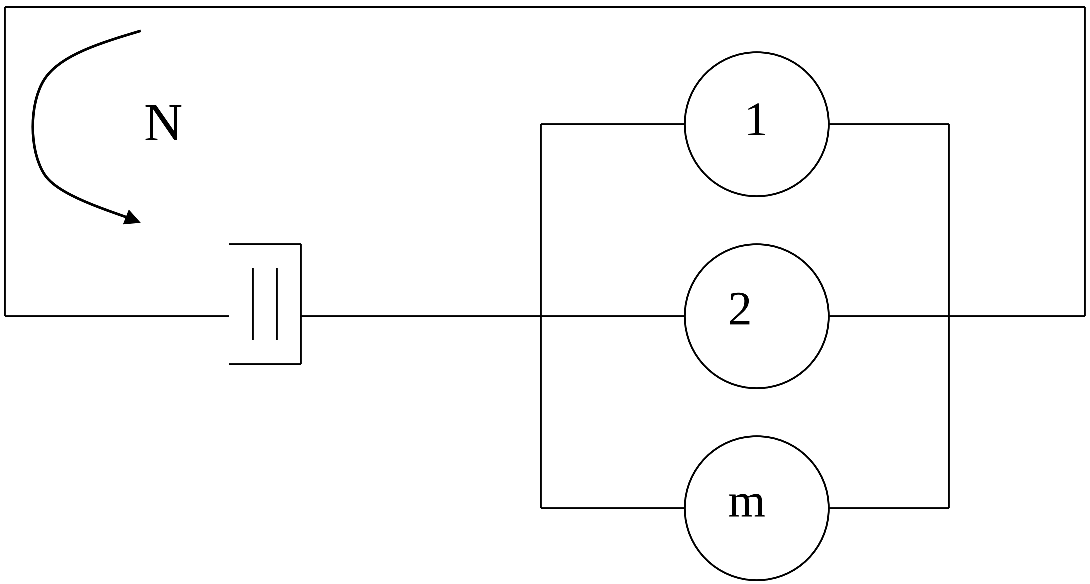
$$Q = \lambda R = Q_W + U$$

$$U \le 1 \Rightarrow \lambda < \frac{1}{\bar{t}}$$ for stability

Throughput = $\boxed{\min(\lambda, \frac{1}{\bar{t}})}$

Applications of Little's Theorem - 2

Example 2 : A closed system with a multi-server node



$N > m$ system is always full

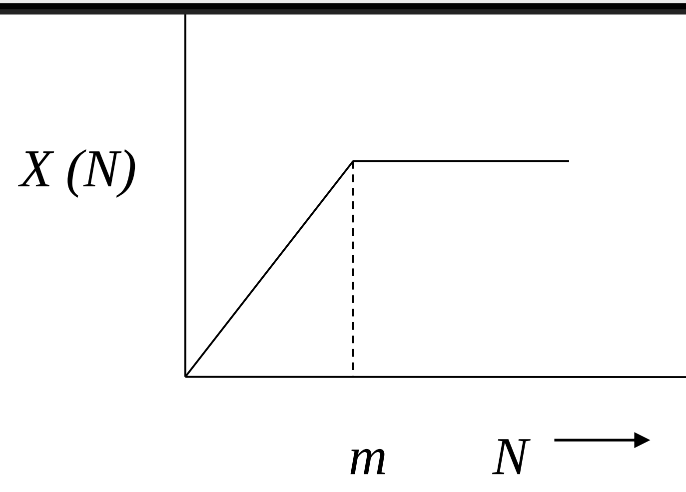
$$X(N)\bar{t} = \min(m, N)$$

$$X(N)R(N) = N$$

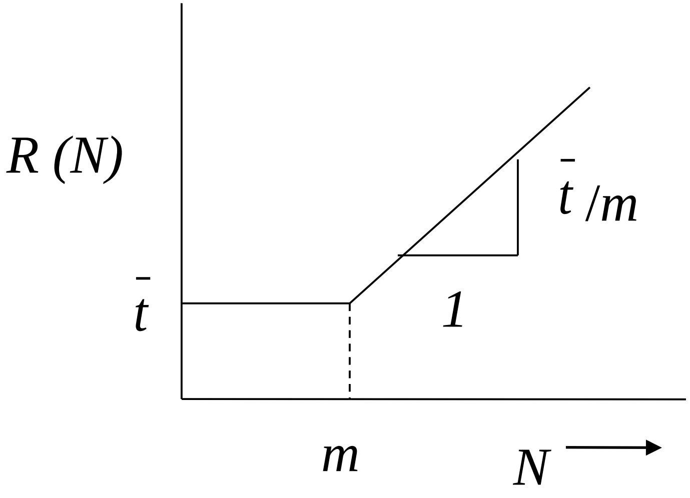
so,

$$R(N) = \frac{N}{\min(m, N)}\bar{t}$$

A Closed System with a Multi-server Node



Throughput versus N



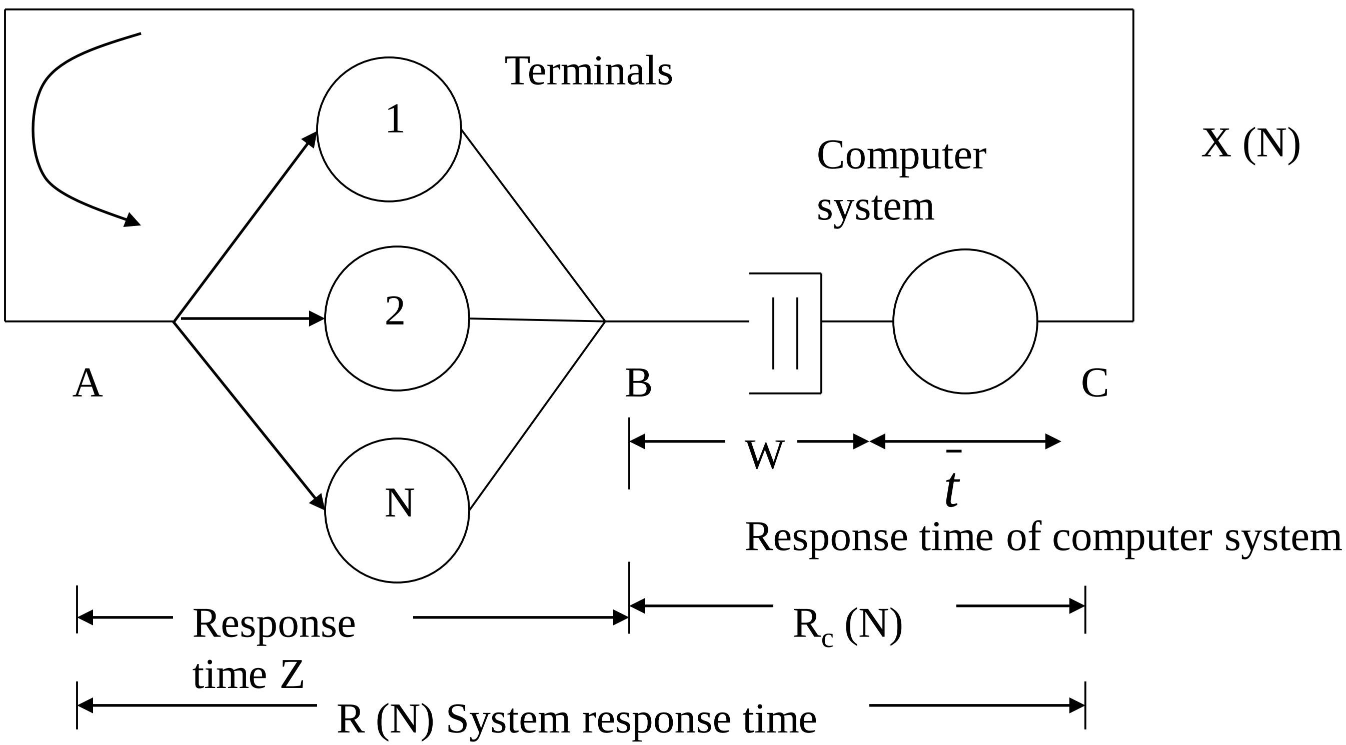
Response time versus N

$$m \to \infty \Rightarrow R(N) = \bar{t}$$

$$\Rightarrow \textit{no waiting}$$

Applications of Little's Theorem - 3

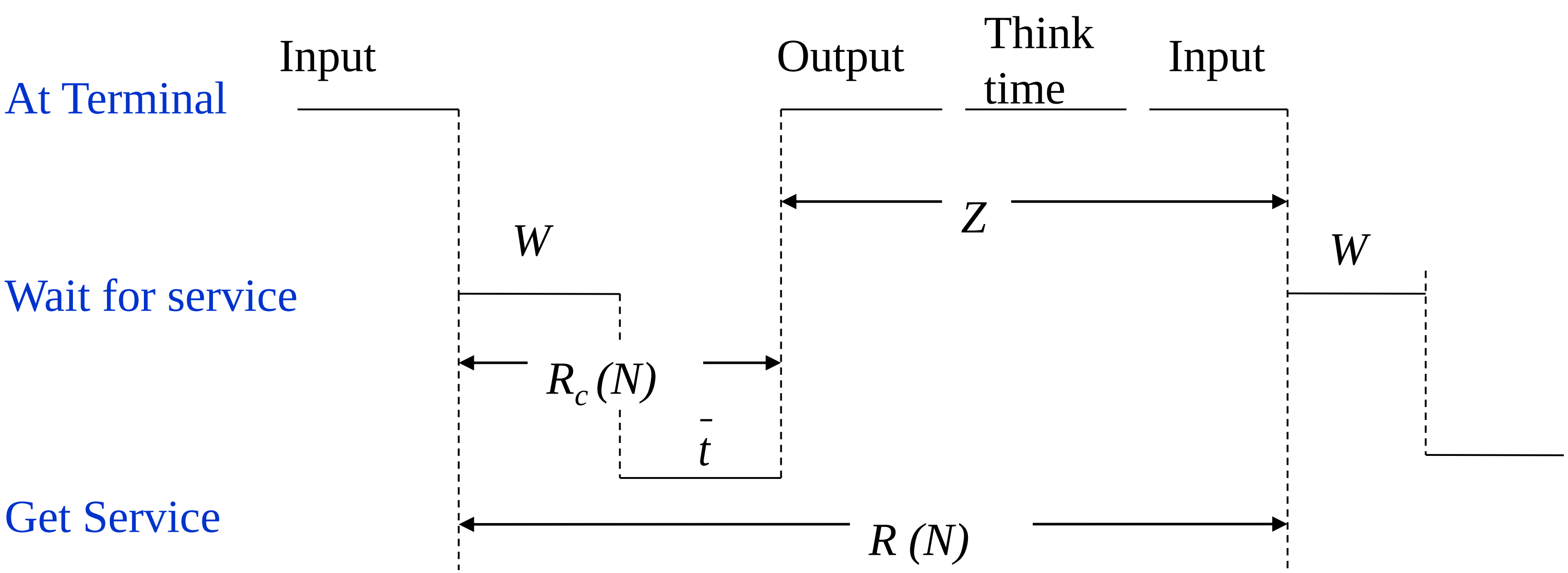
Example 3: Machine Repairman Model or Machine interference model (also models a multi-access communication channel or a time-sharing computer system)



Machine Repairman Model

Machines ⟷ Terminals ⟷ Messages or packets

Repairman ⟷ CPU+I/O ⟷ communication channel



Machine Repairman Model

Points A and C $\quad X(N)R(N) = N \quad \Rightarrow X(N) = \dfrac{N}{R(N)}$

Also $\quad R(N) = R_c(N) + Z$

We will obtain bounds on $R_c(N)$ via the so called Asymptotic Bounding Analysis (ABA).

Let us consider two extreme cases:

No waiting $\Rightarrow R_c(N) = \bar{t}$

Wait for (N-1) customers $\Rightarrow R_c(N) = (N-1)\bar{t} + \bar{t} = N\bar{t}$

Note: if multiple servers: $\quad R_c(N) = (N-m+1)\bar{t}$

So, $\quad \bar{t} \le R_c(N) \le N\bar{t}$

$$\boxed{\bar{t} + Z \le R(N) \le N\bar{t} + Z}$$

$$\frac{1}{N\bar{t}+Z} \le \frac{1}{R(N)} \le \frac{1}{\bar{t}+Z}$$

Machine Repairman Model

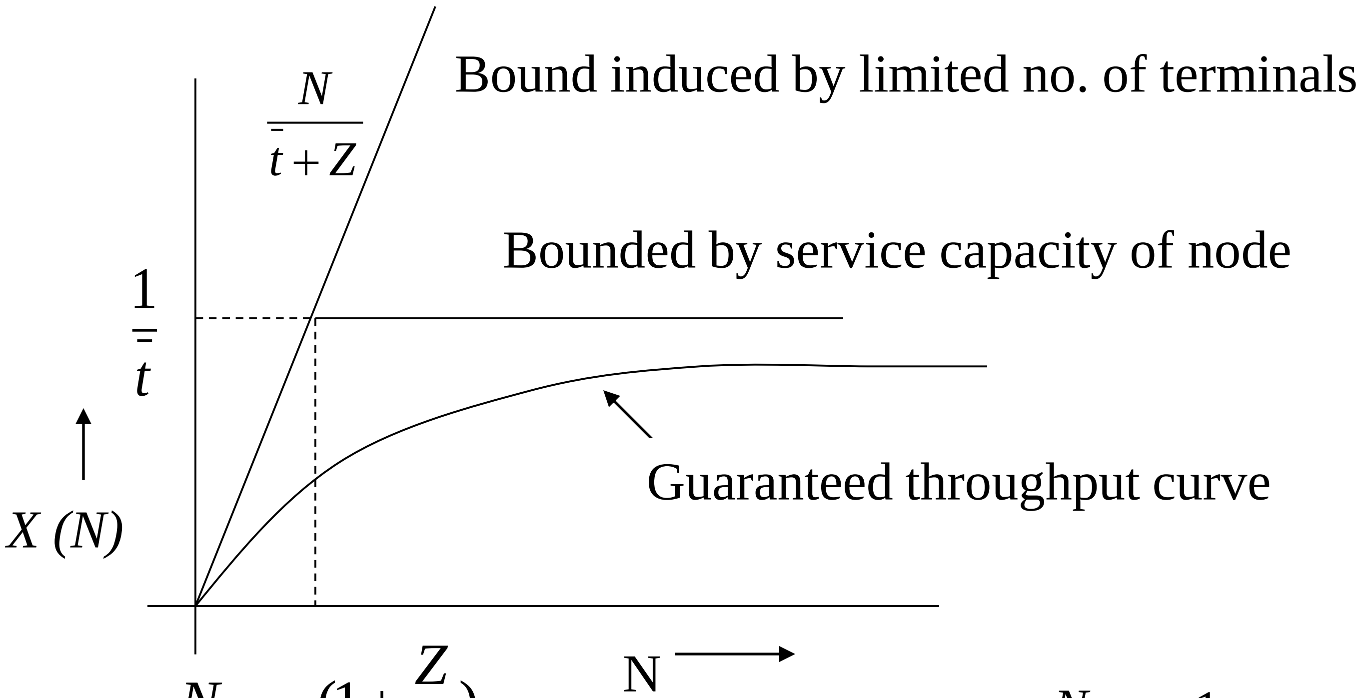
So, $\dfrac{N}{N\bar{t}+Z} \le X(N) \le \dfrac{N}{\bar{t}+Z}$

Also, since $X(N) \le \dfrac{1}{\bar{t}}$ (note for multi-server $X(N) \le \dfrac{m}{\bar{t}}$)

$$\boxed{\frac{N}{N\bar{t}+Z} \le X(N) \le \min\left[\frac{N}{\bar{t}+Z}, \frac{1}{\bar{t}}\right]}$$ ABA bounds

So $$\boxed{\max\left(N\bar{t}, Z+\bar{t}\right) \le R(N) \le Z + N\bar{t}}$$

Machine Repairman Model



$$N_{cr} = (1 + \frac{Z}{\bar{t}})$$

$$\frac{N_{cr}}{Z+\bar{t}} = \frac{1}{\bar{t}}$$

$$\boxed{\Rightarrow N_{cr} = 1 + \frac{Z}{\bar{t}}}$$

Machine Repairman Model

$N < 1 + \frac{Z}{\bar{t}} \Rightarrow$ Throughput limited by number of terminals

$\Rightarrow$ Service center is idle or most users in reflection.

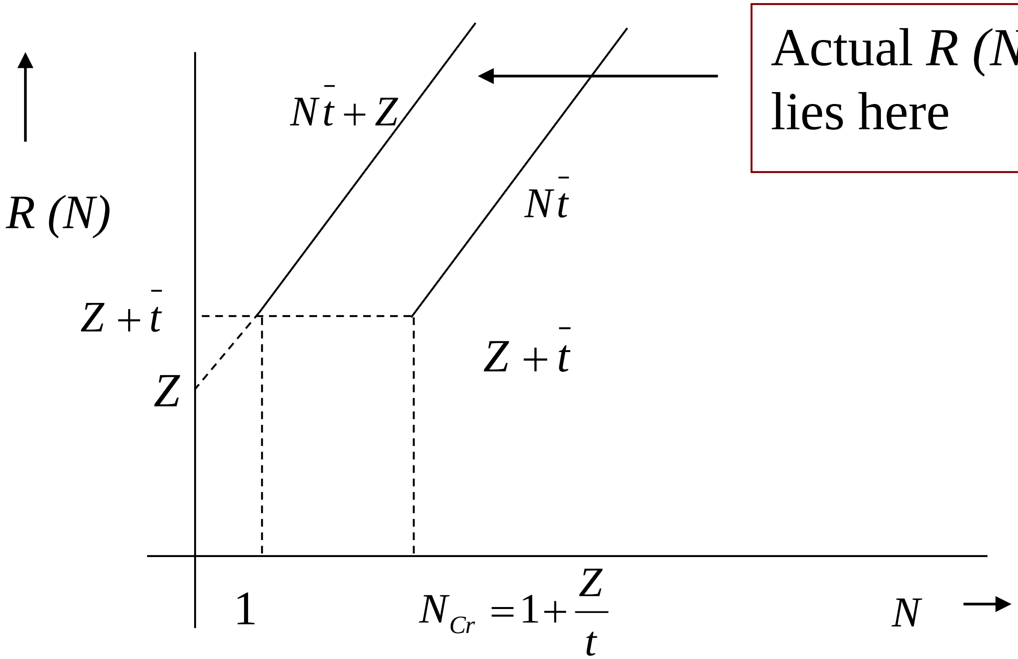
$N > 1 + \frac{Z}{\bar{t}} \Rightarrow$ Throughput is limited by service capacity of service center

$\Rightarrow$ Service center is saturated and linear increase in response time

$\boxed{1 + \frac{Z}{\bar{t}} =}$ $1 + \frac{\text{Think Time}}{\text{Service Time}}$ is called saturation point.

$\Rightarrow$ Suggests a method of selecting # of users, terminals and machines.

Machine Repairman Model

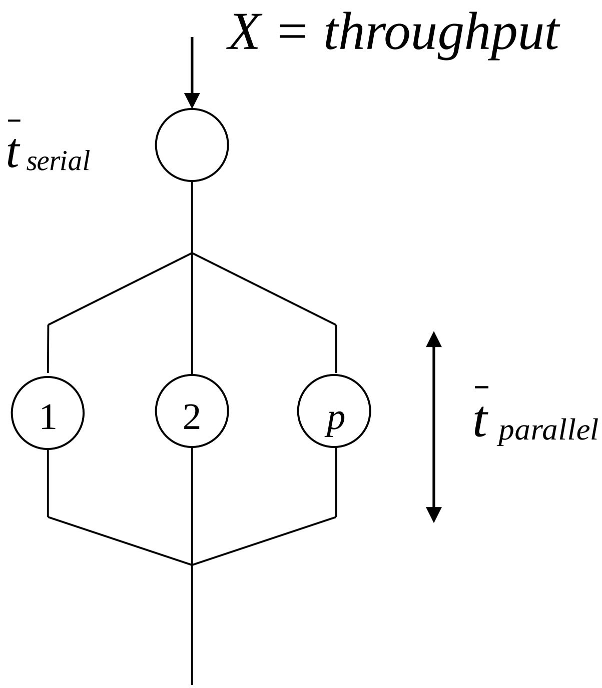


Actual $R\ (N)$
lies here

Applications of Little's Theorem - 4

Example 4: Amdahl's Law and Problem Scaling

Suppose that we have an algorithm with the following task structure (series-parallel graph).



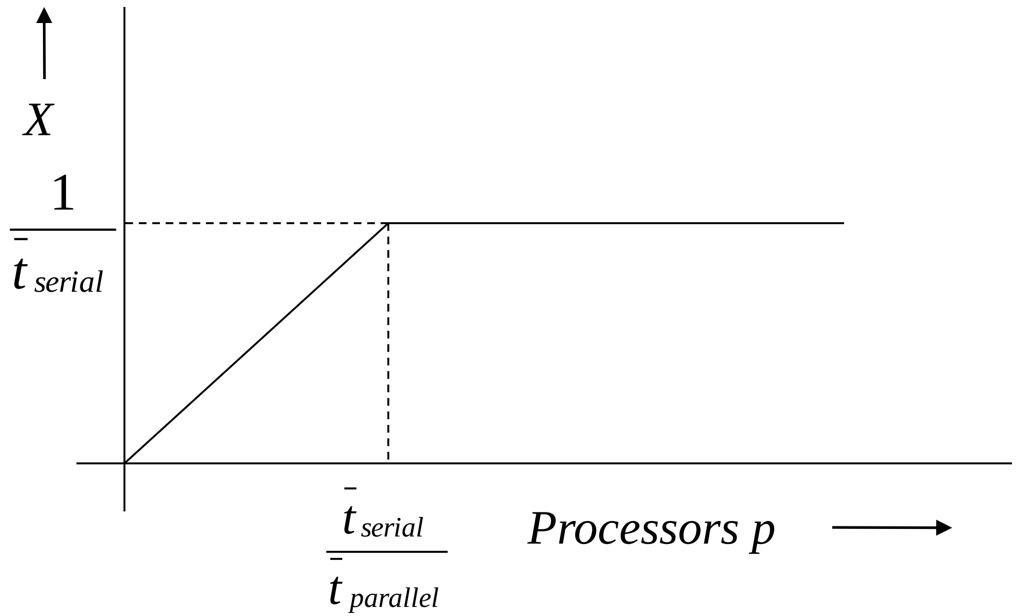
Need:

$$X \cdot \bar{t}_{serial} \leq 1$$

$$X \cdot \frac{\bar{t}_{parallel}}{p} \leq 1$$

$$\Rightarrow X \leq \min\left[\frac{1}{\bar{t}_{serial}}, \frac{p}{\bar{t}_{parallel}}\right]$$

Amdahl's Law and Problem Scaling



$$\text{Speed up} = \frac{\text{Serial execution time}}{\text{Parallel execution time}} = \frac{\bar{t}_{serial} + \bar{t}_{parallel}}{\bar{t}_{serial} + \dfrac{\bar{t}_{parallel}}{p}}$$

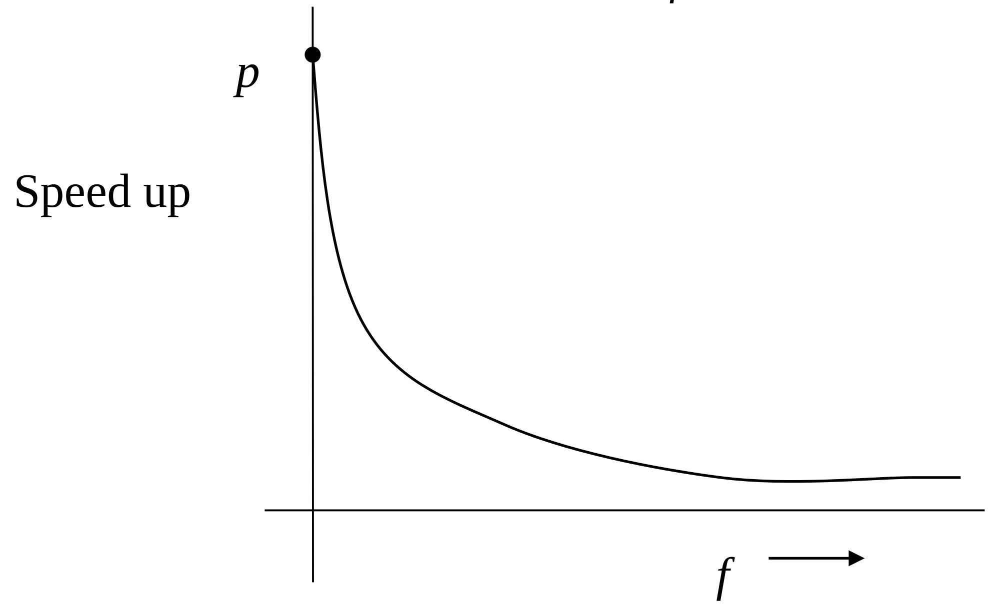
Amdahl's Law and Problem Scaling

Let $$f = \frac{\bar{t}_{serial}}{\bar{t}_{parallel}} \Rightarrow Speedup = \frac{p(1+f)}{pf+1}$$

As $f \to 1$ Speed up $\to \dfrac{2p}{p+1}$

As $f \to 0$ Speed up $\to p$

As $p \to \infty$ Speed up $\to 1 + \dfrac{1}{f}$



Amdahl's Law and Problem Scaling

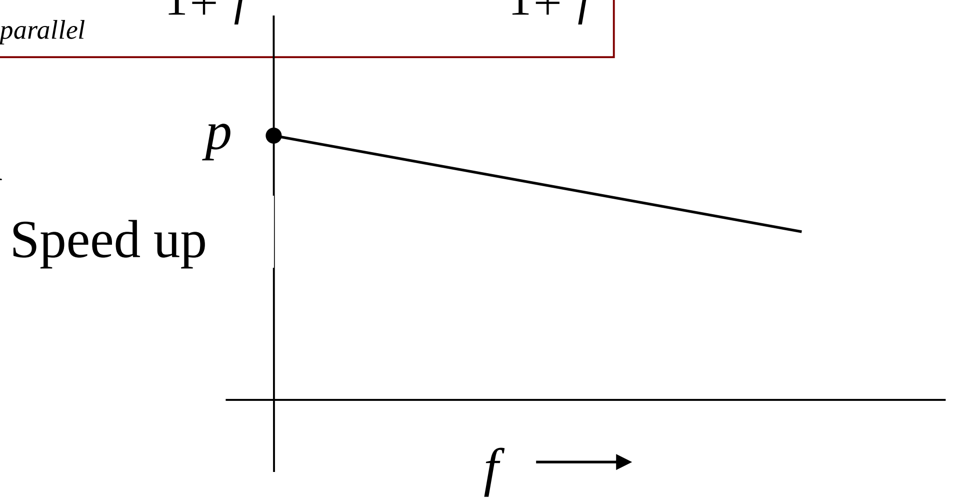
In research at Scandia labs (SIAM Journal of scientific computing, July 1988), It was found that as the problem was scaled up, t_{serial} was found to be relatively constant.

$\Rightarrow$ time on parallel processor: $\bar{t}_{serial} + \bar{t}'_{parallel}$

time on a single processor: $\bar{t}_{serial} + p.\bar{t}'_{parallel}$

Speed up = $\boxed{\dfrac{\bar{t}_{serial} + p.\bar{t}'_{parallel}}{\bar{t}_{serial} + \bar{t}'_{parallel}} = \dfrac{p+f}{1+f} = p + (1-p)\dfrac{f}{1+f}}$

Have been able to obtain speed ups of more than 500 on a 1024 processor system.



Summary & Reading Assignment

1. How to characterize a **resource** or **node** or a **simple queue** ?
2. What are the **measures of system performance** ?
3. Fundamental Accounting identity……………**Little's Theorem**
4. Applications of Little's Theorem

Read:

- Bertsekas and Gallager, section 3.2
- Kobayashi, section 3.6
- Kleinrock vol.1, ch.2, section 2.1
- Gustafson, J.L., Amdahl's law revisited, CACM, 1988.
- Stuck and Arthurs, Chapters 2 and 3