System Analysis

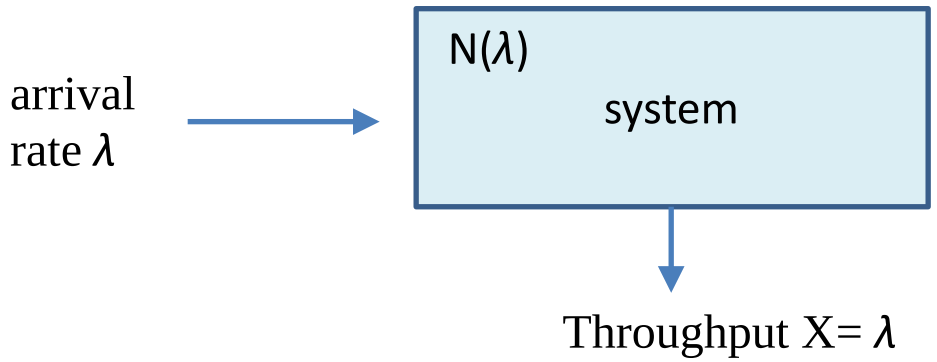
University of Tehran
Prepared by: Dr ahmad Khonsari
The University of Tehran

Excerpt from : **Performance Modeling and Design of Computer Systems**
Queueing Theory in Action
Mor Harchol-Balter

Synopsis of Systems

One of the first tasks in performance modeling is deciding how jobs arrive at the system. In an **open model**, jobs arrive at the system at some rate that is independent of system state, and leave when they are done — as illustrated in the Fig.

The input parameter to the model is the arrival rate λ , and
performance measures (that are functions of λ) to be determined by the model are:
average number of jobs in the system N(λ),
average time in the system T(λ),
utilization, etc.
Note that, in steady state, the throughput X would just be λ.

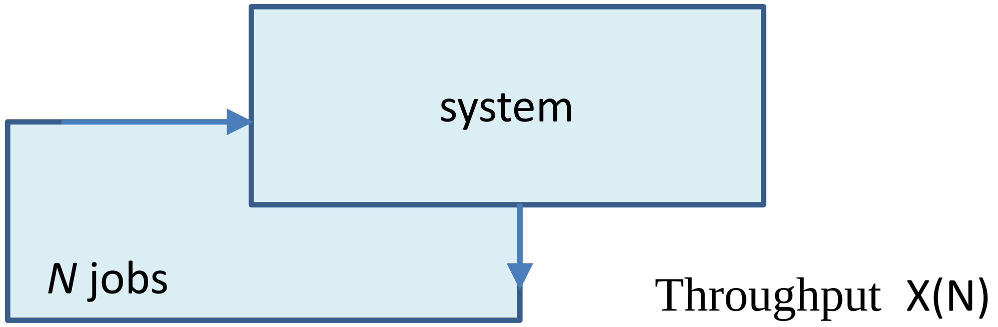


Open model: jobs arrive independently of system state.
Performance measures are functions of λ.

Synopsis of Systems

In a **closed model**, the system has N jobs, and every completed job is immediately replaced by a new job, as illustrated in the Fig. ; the input parameter N (sometimes called **population size**) is thus constant.
The performance measures (that are functions of N) to be determined are:
throughput X(N),
average time in the system T(N),
utilization, etc.

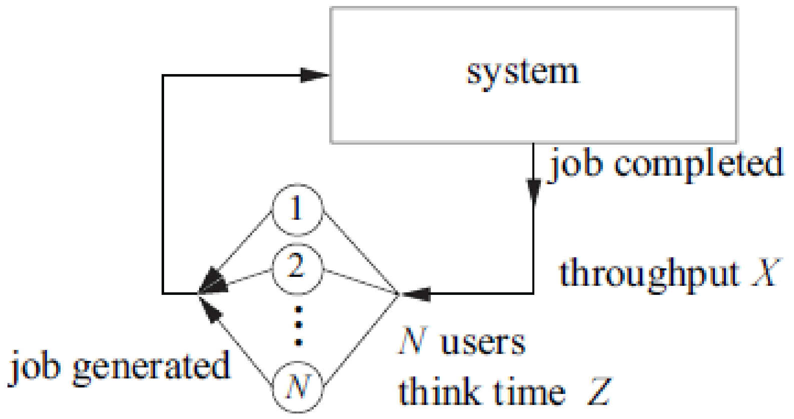
The performance measures are functions of N , e.g. if 2 arrive into the system, then throughput will be 2, ..



Closed model: a completing job is immediately replaced by an arriving job.
Performance measures are functions of population N.

Synopsis of Systems

Administrators: are interested in system performance metrics like **throughput** and **utilization**,
Human users: perceive performance mostly through **response times**. If one is particularly interested in how these depend on the number of users, then it would be more appropriate to use an **interactive model**. This is a special case of a closed system, in which each job is generated by a user who, after that job completes, sends another job after some **think time.** see Fig.



Interactive model: each job is generated by a user, with a think time between a completed job and a new job.
Response time is the time a job spends in the box labeled "system."

Synopsis of Systems

The input parameters are the number of users N — a constant,
closed system — a think time Z specified either as an average or with a density distribution (e.g., exponential).

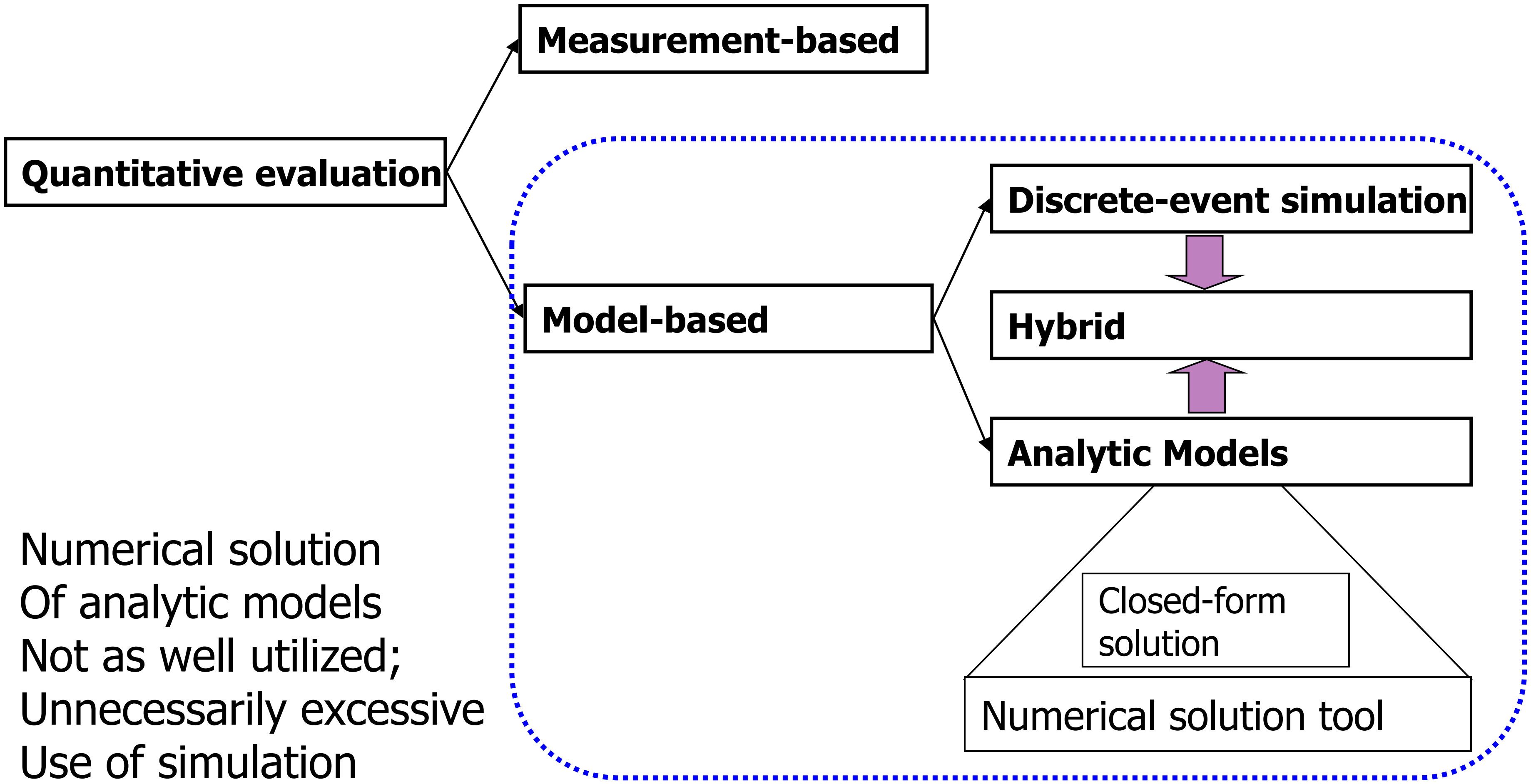
The central metric would be average <u>response time</u>.
Other performance measures include <u>throughput</u> and <u>utilization</u>.

Think time is sometimes referred to as **sleep time**. However, it may be useful to give them different meanings.

E.g., consider a <u>closed model</u> for the web surfing behavior of a fixed number of users. In each surfing **session**, a user may click one hyperlink after another,

“think time” : the time between the completion of a download and the next click;
“sleep time” : the time between the completion of a session and the next session.

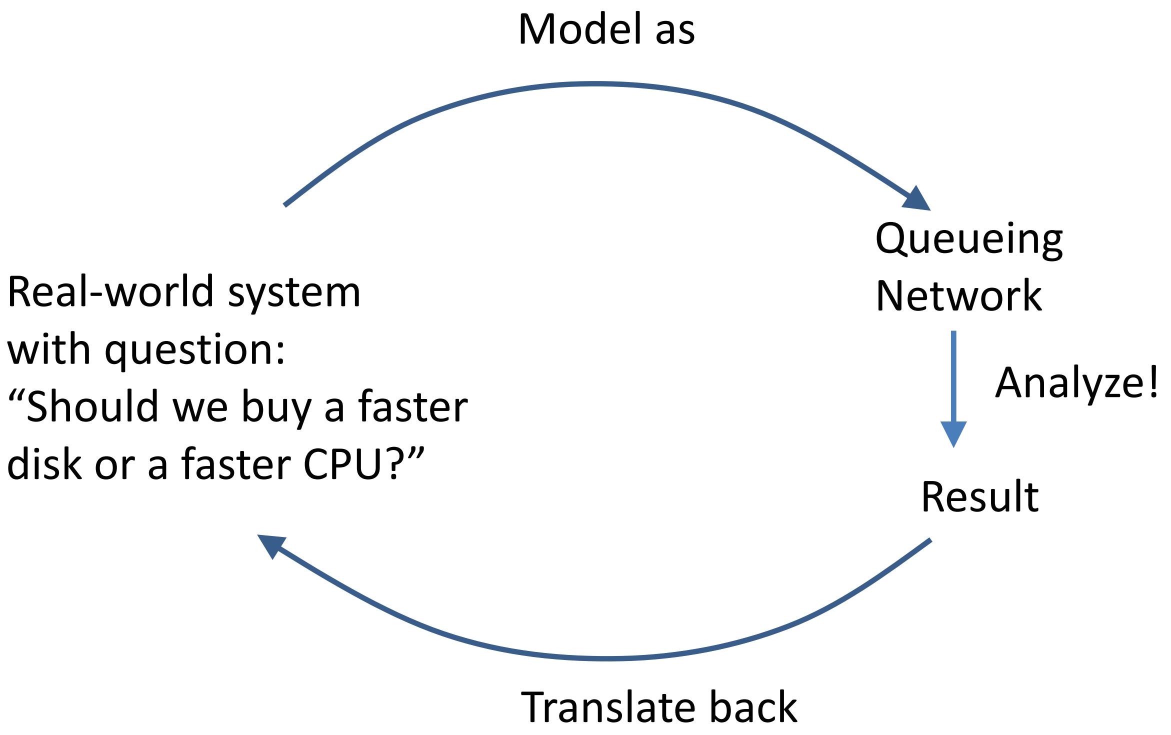
Dependability Evaluation Methods



Numerical solution
Of analytic models
Not as well utilized;
Unnecessarily excessive
Use of simulation

Solution Process

Queueing theory is the study of queueing behavior in networks and systems


Model as
Queueing Network
Analyze!
Result
Translate back
Real-world system with question: “Should we buy a faster disk or a faster CPU?”

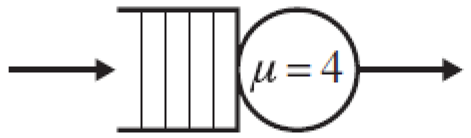
<u>solution process</u>

Single Server Queue-FCFS

A **queueing network** is made up of **servers**

FCFS

Arriving jobs
$\lambda = 3$



Single-server network

Single Server Queue-FCFS

parameters associated with the single-server network:

Service Order :

the order in which jobs will be served by the server. Unless otherwise stated, assume First-Come-First-Served (FCFS).

Average Arrival Rate :

the average rate, λ, at which jobs arrive to the server (e.g., λ = 3 jobs/sec).

Mean Interarrival Time :

the average time between successive job arrivals (e.g., $1/\lambda = \frac{1}{3}$ sec).

Size (Service Requirement):

the "size" of a job is a RV S.

S: the time it would take the job to run on this server if there were no other jobs around (no queueing).

In a queueing model, the size (a.k.a. service requirement) is typically associated with the server (e.g., this job will take 5 seconds on this server).

Single Server Queue-FCFS

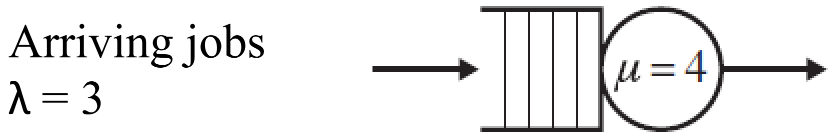
parameters associated with the single-server network:

Mean Service Time:

the expected value of *S*, namely the average time required to service a job on this CPU, where "service" does not include queueing time.

In Figure below , $\mathbf{E}[S] = \frac{1}{4}$sec.

FCFS



Average Service Rate:

the average rate, μ, at which jobs are served (e.g., $\mu = 4$ jobs/sec $= \frac{1}{\mathbf{E}[S]}$).

Single Server Queue-FCFS

<u>parameters associated with the single-server network</u>:

We consider the following common **performance metrics** in the context of a single-server system:

Response Time, Turnaround Time, Time in System, or Sojourn Time (*T*)

$T = t_{\text{depart}} - t_{\text{arrive}}$, job's response time

,where

t_{depart} = the time when the job leaves the system,

t_{arrive} = the time when the job arrived to the system.

We are interested in

$\mathbf{E}[T]$: the mean response time;

$\mathbf{Var}(T)$: the variance in response time;

$\mathbf{P}\{T > t\}$: the tail behavior of T,

Single Server Queue-FCFS

parameters associated with the single-server network: (cntd.)

Waiting Time or Delay (T_Q):

the time that the job spends in the queue, not being served.

It is also called the "time in queue" or the "wasted time."

Notice that $\mathbf{E}[T] = \mathbf{E}[T_Q] + \mathbf{E}[S]$.

Under ***FCFS*** service order, waiting time can be defined as the time from when a job arrives to the system until it first receives service.

Number of Jobs in the System (N):

those jobs in the queue, plus the one being served (if any).

Number of Jobs in Queue (N_Q):

only the number of jobs waiting (in queue).

observations that we can make about the single-server network:

- as λ, the mean arrival rate, increases, all the performance metrics mentioned earlier increase (get worse).
- as μ, the mean service rate, increases, all the performance metrics mentioned earlier decrease (improve).
- We require that $\lambda \leq \mu$ (we always assume $\lambda < \mu$).

Single Server Queue-FCFS

Question: If $\lambda > \mu$ what happens?
Answer: If $\lambda > \mu$ the queue length goes to infinity over time.

Question: Can you provide the intuition?
Answer:
let ***N*(*t*):** the number of jobs in the system at time *t*,
***A*(*t*) :** the number of arrivals at time *t*,
***D*(*t*):** the number of departures at time *t*,

Consider a large time *t*, then we have:
$\boldsymbol{E[N(t)] = E[A(t)] - E[D(t)] \geq \lambda t - \mu t = t(\lambda - \mu).}$
(The inequality comes from the fact that the expected number of departures by time *t* is actually smaller than μt, because the server is not always busy).

Now observe that if
$\boldsymbol{\lambda > \mu}$**, then** $\boldsymbol{t(\lambda - \mu) \rightarrow \infty}$**, as** $\boldsymbol{t \rightarrow \infty}$**.**
Throughout we assume $\lambda < \mu$, which is needed for stability (keeping queue sizes from growing unboundedly).
$\lambda \geq \mu$ are discussed in Ch 9.

Single Server Queue-FCFS

Question: Given the previous stability condition ($\lambda < \mu$), suppose that the interarrival distribution and the service time distribution are *Deterministic* (i.e., both are constants). What is T_Q? What is T?
Answer: $T_Q = 0$, and $T = S$.

queueing (waiting) results from *variability* in service time and/or interarrival time distributions.
an example of how variability leads to queues:
Let's discretize time.
Suppose at each time step, an arrival occurs with probability $p = 1/6$.
Suppose at each time step, a departure occurs with probability $q = 1/3$.
Then there is a non-zero probability that the queue will build up (temporarily) if several arrivals occur without a departure.

Classification of Queueing Networks

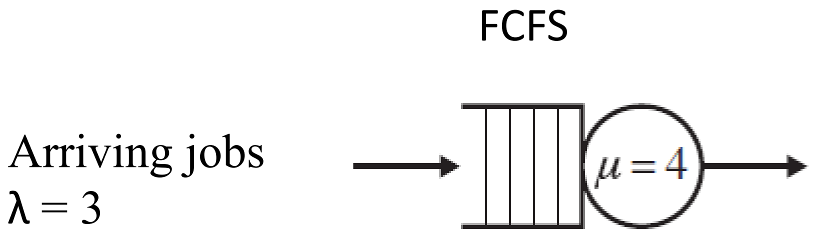
open networks and **closed networks**.

Stochastic processes books usually limit their discussion
to open networks.

By contrast, the systems performance analysis books almost exclusively discuss closed networks.

Single Server Queue-FCFS

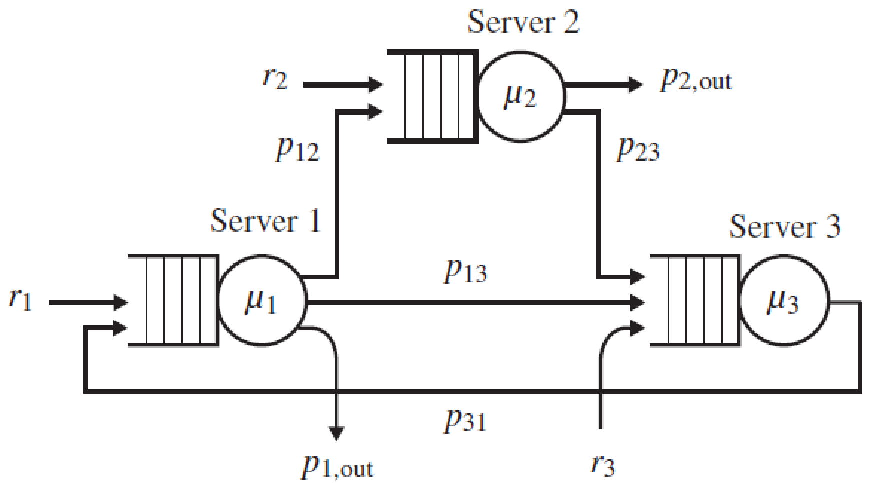
An open queueing network has external arrivals and departures.
4 examples of open networks are illustrated below.
Example1: The Single-Server System



Single-server network

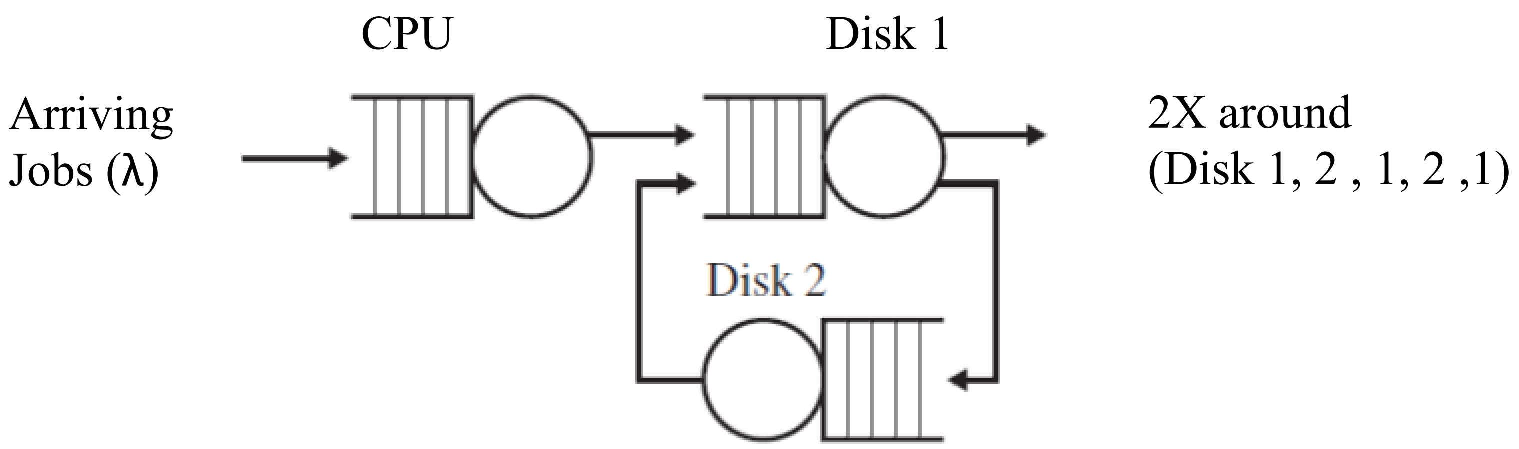
Open Networks

Example2: Network of Queues with Probabilistic Routing
server *i* receives external arrivals ("outside arrivals") with rate r_i.
Server *i* also receives internal arrivals from some of the other servers.
Departed packet at server *i* is next routed to server *j* with probability p_{ij} .
the probabilities may depend on the "class" of the packet, so that not all packets have to follow the same routing scheme.
Application: e.g. Modeling packet flows in the Internet: the class of the packet (and hence its route) depend on its source and destination IP addresses.
Modeling delays: transmission time of a wire= server with some latency. We like to calculate the mean round-trip times for packets on a particular route, given the presence of the other packets.



Network of queues with probabilistic routing

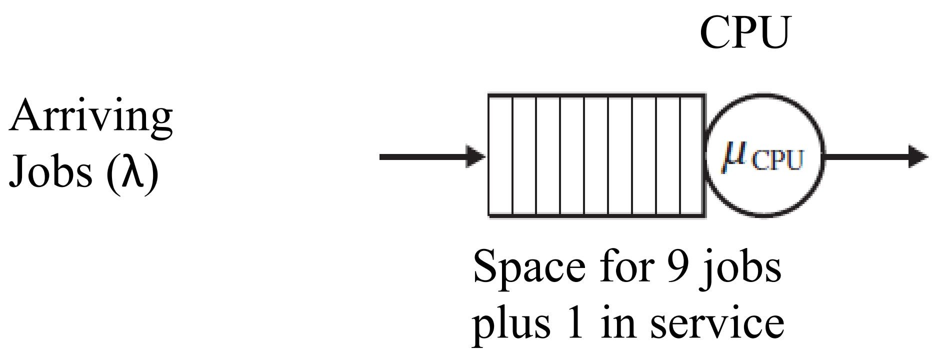
Example3: Network of Queues with Non-Probabilistic Routing
all jobs follow a predetermined route:
CPU to disk 1 to disk 2 to disk 1 to disk 2 to disk 1 and out.



Network of queues with non-probabilistic routing.

Example4: Finite Buffer

An example of a single-server network with finite buffer is shown in the Figure. Any arrival that finds no room is dropped.



Network of queues with non-probabilistic routing.

Performance Metrics:

Thus far we saw four performance metrics:
E[N], **E**[T], **E**[N_Q], and **E**[T_Q].

they can also be used to describe performance in a multi-server, multi-queue system.
E.g. , **E**[T] : the mean time a job spends in the whole system, including all time spent in various queues and time spent receiving service at various servers,
E[T_Q] : just the mean time the job "wasted" waiting in various queues.
We use index i to denote the *i*th queue in a multi-server system,
e.g. **E**[N_i] : the expected number of jobs both queueing and in service at server *i*,
E[T_i]: the expected time a job spends queueing and in service at server *i*.

More Metrics: Utilization and Throughput

Device Utilization (ρ_i **)** is the *fraction of time device i is busy*.
Note our current definition of utilization applies only to a single device (server).
When the device is implied, we simply write ρ (omitting the subscript).

Suppose we watch a device *i* for a long period of time.
Let:
***τ* :** the length of the observation period.
B : the total time during the observation period that the device is non-idle (busy).

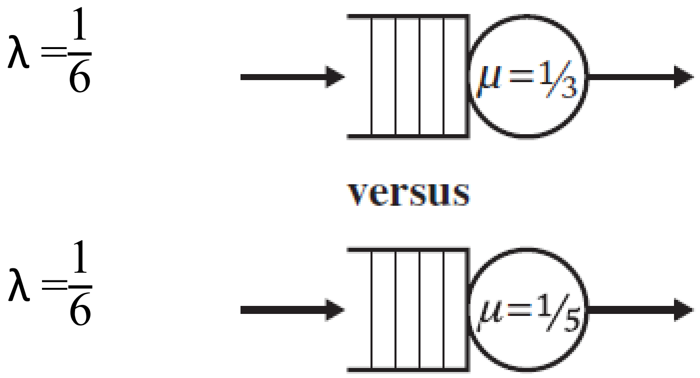
Then:

$$\rho_i = \frac{B}{\tau}$$

More Metrics: Throughput and Utilization

Throughput is arguably the performance metric most used.
Everyone wants higher throughput!

Question: How does maximizing throughput relate to minimizing response time? For example, in the Figure, which system has higher throughput?
Answer: We will see later.



Comparing throughput of two systems.

More Metrics: Throughput and Utilization

Device Throughput (X_i): the *rate of completions at device i* (e.g., jobs/sec). The throughput (X) of the system is the rate of job completions in the system.
Let C denote the total number of jobs completed at device i during time τ .
Then.

$$X_i = \frac{C}{\tau}$$

So how does X_i relate to ρ_i? Well,

$$\frac{C}{\tau} = \left(\frac{C}{B}\right)\frac{B}{\tau}$$

Question: what is $\frac{C}{B}$?

Answer: $\frac{B}{C} = \frac{\text{total time that device is non−idle during time } \tau}{\text{total number of jobs completed during time } \tau} = \mathbf{E}[S]$.

So $\frac{C}{B} = \frac{1}{\mathbf{E}[S]} = \mu_i$

So we have: $\boldsymbol{X_i = \mu_i \rho_i}$

More Metrics: Throughput and Utilization

another way to derive this expression by conditioning:

X_i = Mean rate of completion at server *i*
= **E**[Rate of completion at server *i* | server *i* is busy] · **P**{server *i* is busy}
+ **E**[Rate of completion at server *i* | server *i* is idle] · **P**{server *i* is idle}
= μ_i · **P**{server *i* is busy} + 0
= $\mu_i \cdot \rho_i$

Or, equivalently,

$\rho_i = X_i \cdot \mathbf{E}[S]$.
This is: ***the Utilization Law***.

Question: What is X?

Answer: $X = \rho \cdot \mu$. But what is ρ? we can prove that $\rho = \frac{\lambda}{\mu}$.

An intuitive way to see this, but *not* a proof!!

ρ = Fraction of time server is busy

$$= \frac{\text{Average service time required by a job}}{\text{Average time between arrivals}}$$

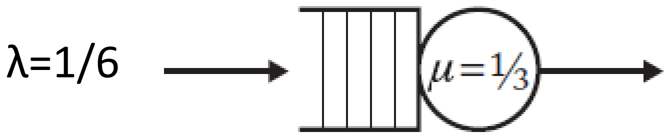
$$= \frac{1/\mu}{1/\lambda} = \frac{\lambda}{\mu}$$

So, this leaves us with

$$X = \rho \cdot \mu = \frac{\lambda}{\mu} \cdot \mu = \lambda.$$

So the throughput does not depend on the service rate whatsoever!

FCFS

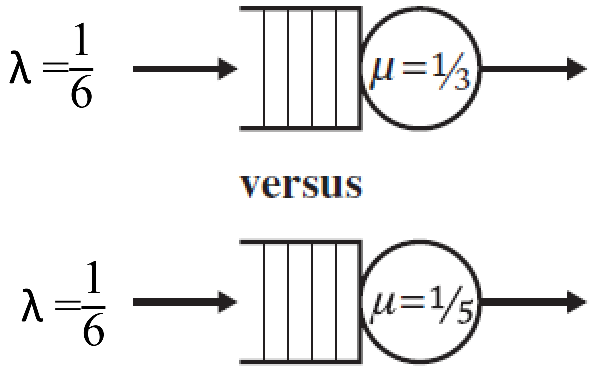


Single-server model.

More Metrics: Throughput and Utilization

Both systems below have the same throughput of 1/6 jobs/sec.
In the case of the faster processor, the response time drops and the queue length drops, but *X* does not change.
Therefore, lower <u>response time</u> is *not* related to higher <u>throughput</u>.

Question: Explain why X does not change.
Answer:
irrespective of the value of μ, the completion rate is still bounded by the arrival rate:
"rate in = rate out."
<u>Changing μ affects the maximum possible X, but not the actual X.</u>
Note that because we assume a stable system, then, for large t, the number of arrivals during t is approximately the number of completions during t.



Two systems with different values of μ.
Throughput, *X*, is the same in both.

More Metrics: Throughput and Utilization

Example: Probabilistic Network of Queues: What is the Throughput?

r_i = average outside arrival rate into server *i*, and μ_i = the average service rate at server *i*.

Question: What is the system throughput, *X*, in the Figure?

Answer: $X = \sum_i r_i$.

Question: What is the throughput at server *i*, X_i?

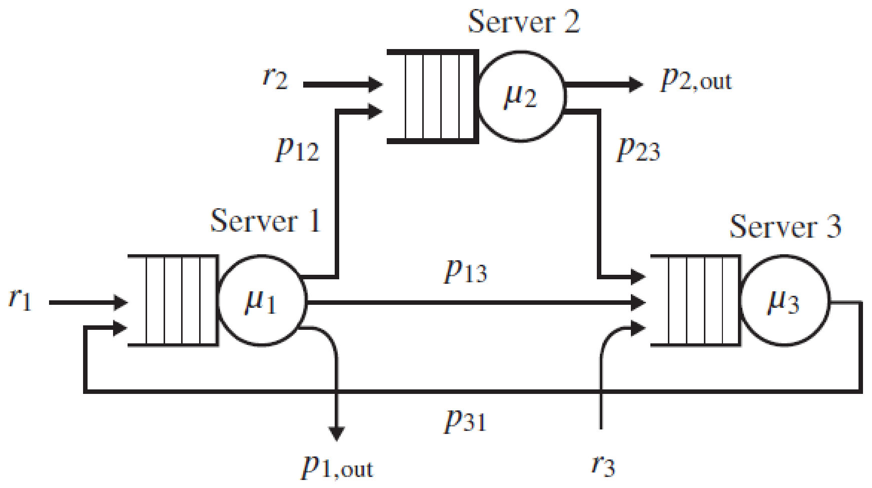
Answer: Let λ_i denote the total arrival rate into server *i*.

Then $X_i = \lambda_i$. But to get λ_i we need to solve the following simultaneous equations:

$\lambda_i = r_i + \sum_j \lambda_j P_{ji}$

Question: How are the r_i 's constrained in these equations?

Answer: For the network to reach "equilibrium" (flow into server = flow out of server), we must have $\lambda_i < \mu_i$, $\forall i$, and this constrains the r_i 's



More Metrics: Throughput and Utilization

Example: Network of Queues with Non-Probabilistic Routing: What is the Throughput?

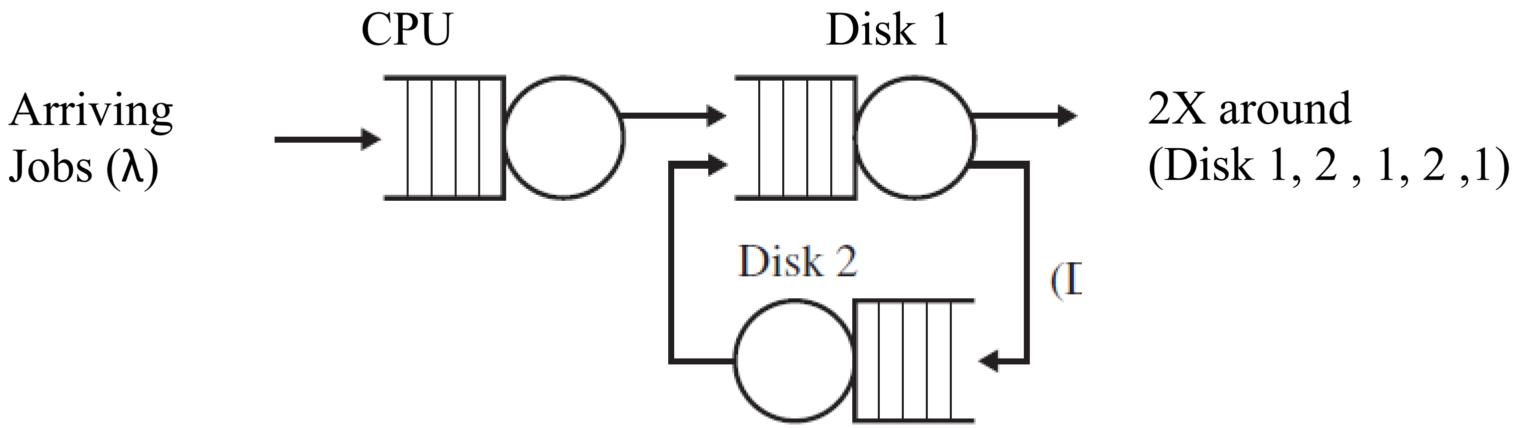
Question: What is *X* in the Figure ?

Answer: $X = \lambda$.

Question: What are X_{Disk1} and X_{Disk2}?

Answer: $X_{\text{Disk2}} = 2\lambda$ *and* $X_{\text{Disk1}} = 3\lambda$.

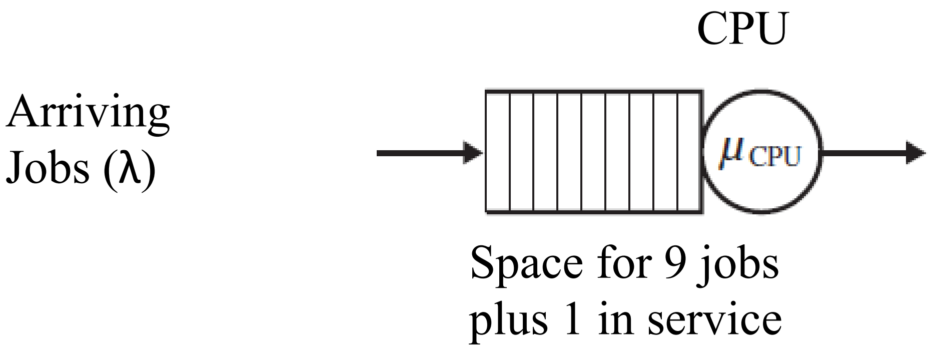
.



Network of queues with non-probabilistic routing.

More Metrics: Throughput and Utilization

Example: Finite Buffer: What is the Throughput?
In the Figure, the outside arrival rate is λ and the service rate is μ.
Question: What is X?
Answer:
$X = \rho\mu$.
But we need stochastic analysis to determine ρ because it is no longer simply λ/μ.
Observe that $X < \lambda$ because some arrivals get dropped.



Closed Networks

Closed queueing networks have no external arrivals or departures.

two types of **<u>Closed networks</u>**

- Interactive (terminal-driven)
- Batch system

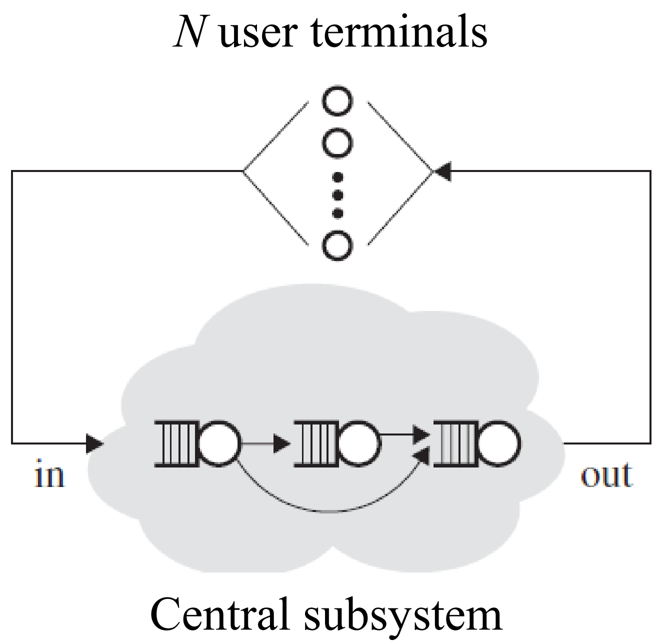
Interactive (terminal-driven)	Batch system

Figure 2.9. Closed network categories.

Closed Networks:
Interactive (Terminal-Driven) Systems

Terminals (users) send a job to the "central subsystem" and then wait for a response.
The central subsystem is a network of queues.
A user cannot submit her next job before her previous job returns.
Thus, **the number of jobs** [A.K.A **load** or **MPL** (multiprogramming level)] in the system is fixed (equal to the number of terminals).
load (**MPL**) not to be confused with device utilization.

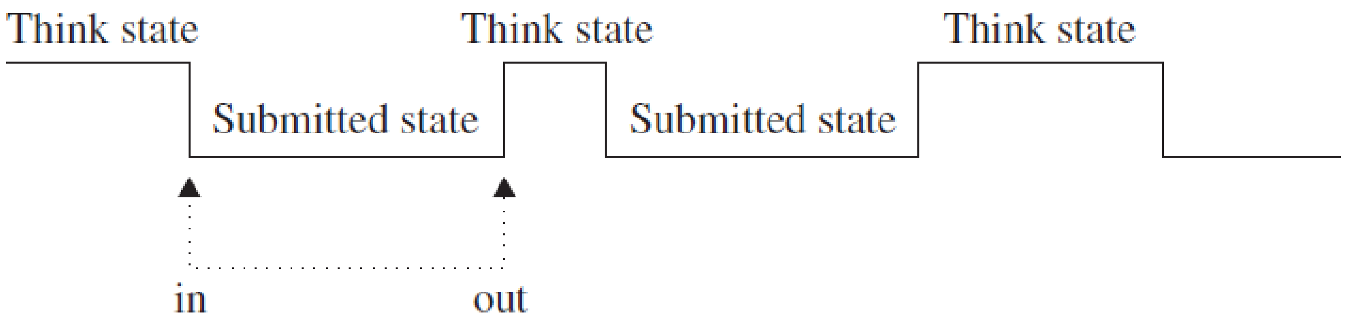
Z **(Think time)**= a RV that denotes the time at each terminal between receiving the result of one job and sending out the next job.
Note: the number of jobs in the central subsystem is at most the number of terminals, because some users might be in the "thinking" state.



Interactive system

Closed Networks:
Interactive (Terminal-Driven) Systems

Example: *N* users each at a terminal filling out a form with many fields .
Once filled out all field , screen is submitted to the central subsystem for update. A new screen cannot be filled out until the previous update is performed.
The "think time," *Z*, is the time to key data to the screen.



The user (terminal) alternates between thinking and waiting for the submitted job to return.

Closed Networks
Interactive (Terminal-Driven) Systems

Question: define the response time for the interactive system?
Answer: Response time is the time it takes a job to go between “in” and “out”.
$\mathbf{E}[\text{Response Time}]$ or $\mathbf{E}[R]$: the average time to get from “in” to “out”
$\mathbf{E}[T]$, is different from $\mathbf{E}[R]$ and is defined as
$\mathbf{E}[T] = \mathbf{E}[R] + \mathbf{E}[Z]$

Important: Response time:
Open system ***T=*** “response time” is a r.v. ($\mathbf{E}[T] = \mathbf{E}[T_Q] + \mathbf{E}[S]$)

closed interactive systems, . ***R*** *= response time*
And we refer to ***T*** as the *system time* (or “time in system”)

Closed Networks
Interactive (Terminal-Driven) Systems

Goal: maximize no. of users , while keeping their **E**[*R*] low.

Note: Note that interactive systems are very different from open systems in that a small change in *N* has a profound effect on the system behavior.

The typical questions asked by systems designers are:
 Given the original system, what is max. *N* such that **E**[*R*] stays below some threshold? That is, how does **E**[*R*] rise with *N*?
 Assume a fixed multiprogramming level, *N*.
 Given that we can make changes to the central subsystem (i.e., make certain devices faster), which changes will improve **E**[*R*] the most?

Closed Networks
Interactive (Terminal-Driven) Systems

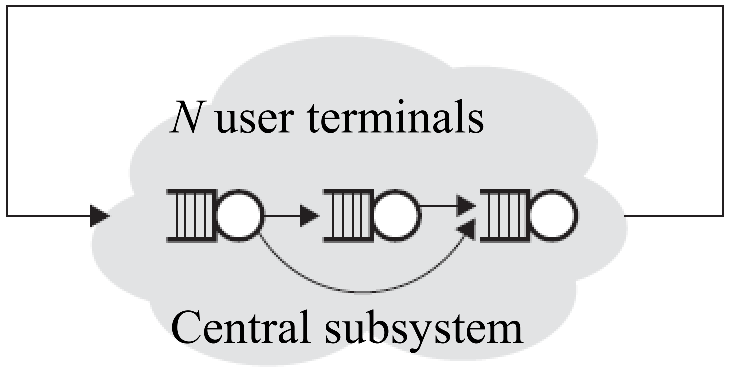
Question: modeling performance of a website. Is it as a closed interactive system or an open system?
Answer: There are research papers of both types.
closed system model:
once a user clicks on a link (submits a job), he typically waits to receive the result before clicking on another link.

Open system model:
a website may have a huge number of users, each of whom is very transient in his or her use of the website.

<u>**"partly-open"**</u> system:
users arrive from outside as in an open system, but make *k* requests to the system when they arrive, where each request can only be made when the previous request completes (as in a closed system).

Closed Networks
Batch Systems

A batch system looks like an interactive system with a think time of zero.
the goals are somewhat different for batch systems.
In a batch system, typically one is running many jobs overnight. As soon as one job completes, another one is started.
So there are always *N* jobs in the central subsystem.
The MPL is usually predetermined and fixed. E.g. the MPL might be the number of jobs that fit into memory.



Batch system.

Closed Networks
Batch Systems

Goal: obtain high *throughput*, so that as many jobs as possible are completed overnight.

The typical question asked by systems designers is:
"How can we improve the central subsystem so as to maximize throughput?"
we are typically constrained by some fixed maximum MPL (because only so
many jobs fit into memory or for some other reason).

The only method to increase throughput is changing the central subsystem, either by changing the routing or by speeding up some device.

Observation: in the batch system we are not concerned with response times because the jobs are running overnight.

Question: What does X mean in a closed system?
Answer: X is the number of jobs crossing "out" per second.
Note that "in" = "out" for the batch system.

Throughput in a Closed System

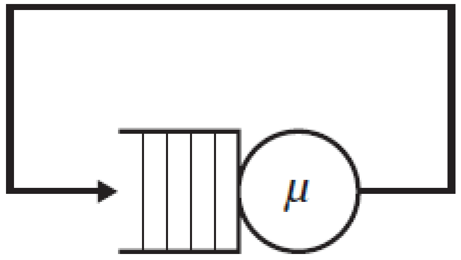
Example: Single Server
Figure shows a closed network consisting of a single server.

Question: What is the throughput, X, in the Figure?
Answer: $X = \mu$.
Observe that this is *very different* from the case of the open network where throughput was independent of service rate!

Question: What is the mean response time, **E**[R], in the Figure ?
Answer: For a closed batch system, **E**[R] = **E**[T], i.e. the response time and time in system are the same.
In this figure, **E**[T] = N/μ, because every "arrival" waits behind $N - 1$ jobs and then runs.

Note that X and **E**[R] are inversely related!
$X = \mu$, **E**[R] = **E**[T] $\propto N/\mu$

MPL = N



Single-server closed network.

Throughput in a Closed System

Example: Tandem Servers

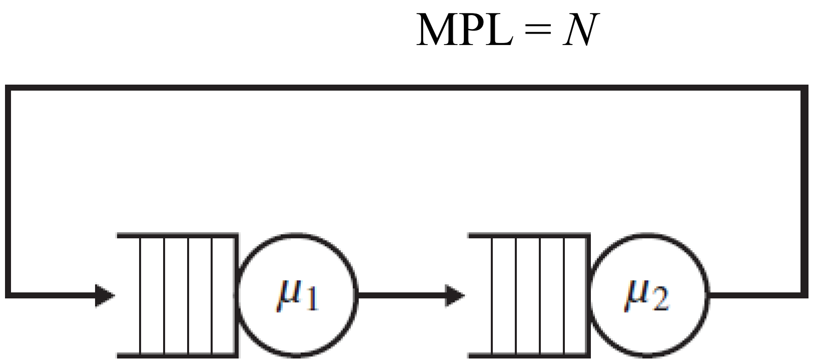


Figure 2.14. Tandem servers closed network.

Question: What is the throughput?
Answer: We may say $X = \min(\mu 1, \mu 2)$ . ...

Question: Why is this previous answer not necessarily correct?
Answer: The previous answer is correct if we know that the slower server is always busy, but that is not necessarily the case. Imagine $N = 1$. Then it is certainly not the case that the slower server is always busy.

Question: what happens when $N = 2$. Now it appears that there is always at least one job at the slow server, doesn't it?
Answer: No, the slower server is still not always busy. sometimes the slow server is faster than the fast server – since service rates are just averages!

So do we in fact need to take the job size distribution into account to get the exact answer? Does the job size distribution really affect the answer very much?

Throughput in a Closed System

Differences between Closed and Open Networks

Open Systems

Throughput, X, is independent of the μ_i's

X is not affected by doubling the μ_i's.

Throughput and response time are *not* related.

Closed Systems

X depends on μ_i's.

If we double all the μ_i's while holding N constant, then X changes.

for closed systems:
Higher throughput $\Longleftrightarrow$ Lower avg. response time.

Throughput in a Closed System

A Question on Modeling

Question: How do you obtain $\mathbf{E}[S]$ in practice for your single-server system?
Answer: you may say, as $\mathbf{E}[S]$ is the mean time required for a job in isolation, you should just send a single job into the system and measure its response time and repeat the experiment several times to get an average.
This makes sense in theory, but does not work well in practice, since cache conditions and other factors are very different for the scenario of just a single job compared with the case when the system has been loaded for some time.
<u>A better approach: </u>remember that $\mathbf{E}[S] = \frac{1}{\mu}$, so it suffices to think about the service rate of the server in jobs/second.
To get μ, assuming an open system, we can make λ higher and higher, which will increase the completion rate, until the completion rate levels off (saturates) at some value, which will be rate μ.
<u>A better idea : </u>put the server into a closed system, with zero think time. This guarantees the server to always be occupied with work. Now, if we measure the completion rate at the server (jobs completing per second), then that will give us μ for the server and $\mathbf{E}[S] = \frac{1}{\mu}$

Throughput in a Closed System

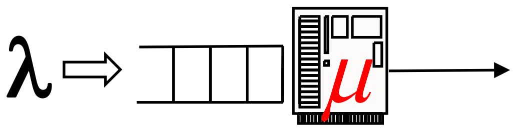
A Question on Modeling

Thus far, we have mentioned that ($\lambda < \mu$) is *necessary* for stability.
This condition will also be *sufficient* to ensure stability of the networks we consider in practice.
However, it is not generally a *sufficient* condition for stability in more complex queueing networks.
See the papers of Maury Bramson .

Open versus Closed Systems

Always:
Throughput= p(server busy)* Rate server

Open



X=ρμ

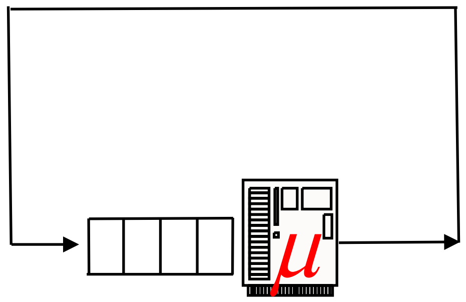
$$\rho = \lambda E[S] = \frac{\lambda}{\mu}$$

$$X = \lambda$$

Always:
Throughput= p(server busy)* Rate server

Closed Batch

MPL N: fixed #jobs



X=ρμ

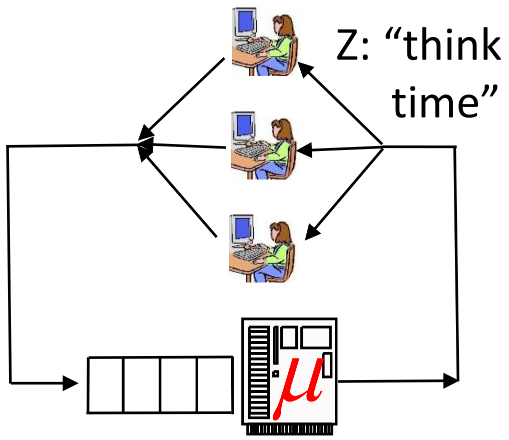
$$\rho = 1$$

$$X = \mu$$

Always:
Throughput= p(server busy)* Rate server

Closed Interactive

MPL N: fixed #users



X=ρμ

$$\rho = 1 - \Pr\{\text{All thinking}\}$$

$$X = \rho\mu$$