Module-4:

Statistical Models in Simulation- Terms and concepts. Statistical Models. Review of discrete and continuous distributions. Review of Poisson (stationary and non-stationary) processes. Empirical Distributions; Elementary Queuing Theory- Basic Structure of Queuing Models. Input Source (Calling Population). Queue, Queue Discipline, Service Mechanisms. Notations and relationships between *L, W, Lq,* and *Wq.* Little's Formula. Role of Exponential Distribution and Properties. Birth and Death Processes. M/M/s queues. Finite queue variation in M/M/s/K models with different s values. Finite Calling Population cases. Queuing Models involving Non-Exponential Distributions: M/G/1, M/D/s, M/Ek/s (involving Erlang distribution), Models without a Poisson Input, Models involving hyper exponential distributions, Priority Discipline Queuing Models: Preemptive and Non- Preemptive with results, properties and server number variations, Queuing Networks: Equivalence Property. Infinite Queues in Series and Product Form Solutions. Jackson Networks.

According to Kenneth Bollen "A model is a formal representation of a theory". A statistical model is a mathematical model that uses a set of assumptions for the generation of the observed data, and similar data from a larger population. . A statistical model is usually specified as a mathematical relationship between one or more random variables and other non-random variables.

Distribution is a statistical concept used in data research. Statisticians seeking to identify the outcomes and probabilities of a particular study will chart measurable data points from a data set, resulting in a probability distribution diagram.

Statisticians can identify the development of either a discrete or continuous distribution by the nature of the outcomes to be measured. Statisticians can identify the development of either a discrete or continuous distribution by the nature of the outcomes to be measured. Discrete distributions have a finite number of outcomes.

For example, when studying the probability distribution of a die with six numbered sides there can only be six possible outcomes, so the finite value is six. Another example can include flipping a coin. Flipping a coin can only result in two outcomes so the finite value is two.

Types of Distribution

1) **Discrete Distribution**
2) **Continuous Distribution**

What is a discrete distribution?

A discrete distribution is a statistical distribution that shows the probabilities of outcomes with finite values or it cannot be broken down into fractions or decimals. The most common discrete probability distributions include binomial, Poisson, Bernoulli, and multinomial. One example where discrete distribution can be valuable for businesses is in inventory management. Studying the frequency of inventory sold in conjunction with a finite amount of inventory available can provide a business with a probability distribution that leads to guidance on the proper allocation of inventory to best utilize square footage.

Discrete distributions can also arise in the Monte Carlo simulation. Monte Carlo simulation is a modeling technique that identifies the probabilities of different outcomes through programmed technology. It is primarily used to help forecast scenarios and identify risks. In Monte Carlo simulation, outcomes with discrete values will produce discrete distributions for analysis. These distributions are used in determining risk and trade-offs among different items being considered.

A discrete distribution describes the probability of occurrence of each value of a discrete random variable. A discrete random variable is a random variable that has countable values, such as a list of non-negative integers.

With a discrete probability distribution, each possible value of the discrete random variable can be associated with a non-zero probability. Thus, a discrete probability distribution is often presented in tabular form.

The discrete distributions we will discuss include: Bernoulli distribution, Binomial distribution, Geometric distribution and Poisson distribution

Poisson Distribution

The **Poisson Process** is the model we use for describing randomly occurring events and by itself, isn't that useful. We need the **Poisson Distribution** to do interesting things like finding the probability of a number of events in a time period or finding the probability of waiting some time until the next event. The Poisson process is one of the most widely-used counting processes. It is usually used in scenarios where we are counting the occurrences of certain events that appear to happen at a certain rate, but completely at random (without a certain structure). For example, suppose that from historical data, we know that earthquakes occur in a certain area with a rate of 2 per month. Other than this information, the timings of earthquakes seem to be completely random. Thus, we conclude that the Poisson process might be a good model for earthquakes. In practice, the Poisson process or its extensions have been used to model

− the number of car accidents at a site or in an area;

− the location of users in a wireless network;

− the requests for individual documents on a web server;

− the outbreak of wars;

− photons landing on a photodiode.

A Poisson Process meets the following criteria (in reality many phenomena modeled as Poisson processes don't meet these exactly):

1. Events are **independent** of each other. The occurrence of one event does not affect the probability another event will occur.
2. The average rate (events per time period) is constant.
3. Two events cannot occur at the same time.

The Poisson Distribution probability mass function gives the probability of observing **k** events in a time period given the length of the period and the average events per time:

Common examples of Poisson processes are customers calling a help center, visitors to a website, radioactive decay in atoms, photons arriving at a space telescope, and movements in a stock price. Poisson processes are generally associated with time, but they do not have to be. In the stock case, we might know the average movements per day (events per time), but we could also have a Poisson process for the number of trees in an acre (events per area).

(One instance frequently given for a Poisson Process is bus arrivals (or trains or now Ubers). However, this is not a true Poisson process because the arrivals are not independent of one another. Even for bus systems that do not run on time, whether or not one bus is late affects the arrival time of the next bus. Jake VanderPlas has a great article on applying a Poisson process to bus arrival times which works better with made-up data than real-world data.)

Input Source (Calling Population)

The figure below shows the queuing system under consideration. Customers requiring some service are the small circles and servers are the numbered rectangles. Customers arrive to the system from some input source. If some server is not busy, the customer immediately begins to be served. Otherwise, the customer must wait in a queue until a server is available. Some time is required for service, after which the customer departs.

The input source, also called the calling population, is the collection of customers providing inputs to the queuing system. We assume for this model that the calling population is infinite, that is, the rate of arrivals into the system is unaffected by the number already there.A queue discipline defines the rules by which customers are selected for service. A common discipline, assumed here, is first-come-first-served. Service is provided by one or more servers (or channels) operating in parallel. The servers may or not be identical.

The queuing system is the combination of the queue and the service channels. For this model, we assume that the total number of customers that can be present in the system is limited to some maximum number. That is, the size of the queue is finite and limited by the maximum number in the system less the number of servers. When customers arrive and find the queue to be full, the customer does not enter the system and does not receive service.

The state/event model is illustrated by the network below. The states are the nodes and the number in each circle is the number of customers in the system. The event of an arrival indicated by a, adds 1 to the number in the system. The number of servers is s. When the state number is less than or equal to s, all customers are in service and there is no queue. When the number in the system exceeds s, a queue forms.The service event, indicated by d, reduces the number in the system.

The queuing model is really a birth-death model with prescribed birth and death rates that depend on the number in the system.

Service Mechanism

The service mechanism is the way that customers receive service once they are selected from the front of a queue. A service mechanism is also called a server (in fact, this is the more common terminology). The amount of time which a customer takes to be serviced by the server is called the *service time*. A statistical distribution is used to model the service time of a server. Some queueing models assume a single server, some multiple servers. For most general analysis, most queueing models assume that the system has either a single server or allow the number of servers to become a variable

Queuing Discipline

How the customers will be served form a queuing discipline. Queuing discipline is closely related to the notion of fairness and efficiency. The order of the queuing discipline is usually first in first out (first come first serve) because it maintains fairness among customers.

However, waiting line queue can also be formed using other form of queuing discipline such as

> **At once** : after class, after church service a bunch of people go out of a room at once. In a conference lunch, all the participants want to eat at the same hour. Vehicles in an intersection moves together after the red light turns into green. Passengers of an airplane departs at the same time. The actual order on which customers to be served is not that important in this case, we only know that all customers come at once and waiting to be served.
>
> **First in first out (FIFO)** : first come first serve. This is the most popular queuing discipline it maintains fairness among customers.
>
> **Last in first out (LIFO)** : the last customers will be served first. Goods inside a delivery truck usually arranged such that the first item enters the truck will be delivered last. Stack of pancakes are eaten from the last item on the top.
>
> **Loop** : children in playground join the same queue after being served. Machines in a factories form a loop to be maintained.
>
> **Priority** : certain type of preferred customers will be served first. Business class passengers will enter the airplane first before the economic class. Patient with severe cases will be served first in the hospital ahead of ordinary sickness.

There are many detail human behaviors such as changing to the shortest queue and leaving the queue when it is too long will also affect the queuing system.

❑ **Relationships between L, W, L_q, and W_q**

- ✓ Assume that λ_n is a constant for all n.
- ✓ In a steady-state queueing process, $L = \lambda W$ (Little's formula) and $L_q = \lambda W_q$.
- ✓ If the λ_n are not equal, then λ can be replaced in these equation by $\overline{\lambda}$, the average arrival rate over the long time.
- ✓ Assume that the mean service time ($1/\mu$) is a constant. Thus, $W = W_q + \frac{1}{\mu}$.
- ✓ These four fundamental quantities (L, W, L_q, and W_q) could be immediately determined as soon as one is found analytically.

What is Little's Law?

Little’s Law is a theorem that determines the average number of items in a stationary queuing system based on the average waiting time of an item within a system and the average number of items arriving at the system per unit of time.

The law provides a simple and intuitive approach for the assessment of the efficiency of queuing systems. The concept is hugely significant for business operations because it states that the number of items in the queuing systems primarily depends on two key variables, and it is not affected by other factors such as the distribution of the service or service order.

Little’s Law can only be used in queuing systems. Almost any queuing system and even any sub-system (think about the single teller in a supermarket) can be assessed using the law. In addition, the theorem can be applied in different fields, from running a small coffee shop to the maintenance of the operations of a military airbase.

Origin of Little’s Law

Massachusetts Institute of Technology (MIT) professor John Little developed Little’s Law in 1954. The initial publication of the law did not contain any proofs of the theorem. However, in 1961, Little published proof that there is no such situation when the described relationship does not hold. Little later received recognition for his works in operations research.

Formula for Little’s Law

Mathematically, Little’s Law is expressed through the following equation:

$$L = \lambda \times W$$

Where: L – the average number of items in a queuing system
λ – the average number of items arriving at the system per unit of time
W – the average waiting time an item spends in a queuing system

Example of Little’s Law

John owns a small coffee shop. He wants to know the average number of customers queuing in his coffee shop to decide whether he needs to add more space to accommodate more clients. Currently, his queuing area can accommodate no more than eight customers.

John measured that on average, 40 customers arrive at his coffee shop every hour. He also determined that on average, a customer spends around 6 minutes in a store (or 0.1 hours). Given the inputs, John can find the average number of the customers queuing in his coffee shop by applying the Little's Law:

$$L = 40 \times 0.1 = 4 \text{ customers}$$

The Little's Law shows that on average, there are only four customers queuing in John's coffee shop. Therefore, he does not require to create more space in the store to accommodate more queuing customers.

KENDALL'S NOTATION

Kendall's Notation is a system of notation according to which the various characteristics of a queuing model are identified.Kendall (Kendall, 1951) has introduced a set of notations which have become standard in the literature of queuing models. A general queuing system is denoted by (a/b/c): (d/e) where

a	=	probability distribution of the interarrival time.
b	=	probability distribution of the service time.
c	=	number of servers in the system.
d	=	maximum number of customers allowed in the system.
e	=	queue discipline

In addition, the size of the population is important for certain types of queuing problem although not explicitly mentioned in the Kendall's notation. Traditionally, the exponential distribution ...

Birth and Death Processes

The **birth–death process** (or **birth-and-death process**) is a special case of continuous-time Markov process where the state transitions are of only two types: "births", which increase the state variable by one and "deaths", which decrease the state by one. The model's name comes from a common application, the use of such models to represent the current size of a population where the transitions are literal births and deaths. Birth–death processes have many applications in demography, queueing theory, performance

engineering, epidemiology, biology and other areas. They may be used, for example, to study the evolution of bacteria, the number of people with a disease within a population, or the number of customers in line at the supermarket.
When a birth occurs, the process goes from state *n* to *n* + 1. When a death occurs, the process

goes from state *n* to state *n* − 1. The process is specified by birth rates and death

rates .

M/M/c queue

In queueing theory, a discipline within the mathematical theory of probability, the **M/M/c queue** (or **Erlang–C model**) is a multi-server queueing model.[2] In Kendall's notation it describes a system where arrivals form a single queue and are governed by a Poisson process, there are *c* servers and job service times are exponentially distributed. It is a generalisation of the M/M/1 queue which considers only a single server. The model with infinitely many servers is the M/M/∞ queue.

Model definition

An M/M/c queue is a stochastic process whose state space is the set {0, 1, 2, 3, ...} where the value corresponds to the number of customers in the system, including any currently in service.

- Arrivals occur at rate λ according to a Poisson process and move the process from state *i* to *i*+1.
- Service times have an exponential distribution with parameter μ. If there are fewer than *c* jobs, some of the servers will be idle. If there are more than *c* jobs, the jobs queue in a buffer.
- The buffer is of infinite size, so there is no limit on the number of customers it can contain.

The model can be described as a continuous time Markov chain with transition rate matrix

6.2 Queueing Notation

Recognizing the diversity of queueing systems, Kendall [1953] proposed a notational system for parallel server systems which has been widely adopted. An abridged version of this convention is based on the format $A/B/c/N/K$. These letters represent the following system characteristics:

A represents the interarrival-time distribution.

B represents the service-time distribution.

c represents the number of parallel servers.

N represents the system capacity.

K represents the size of the calling population.

Common symbols for A and B include M (exponential or Markov), D (constant or deterministic), E_k (Erlang of order k), PH (phase-type), H (hyperexponential), G (arbitrary or general), and GI (general independent).

For example, $M/M/1/\infty/\infty$ indicates a single-server system that has unlimited queue capacity and an infinite population of potential arrivals. The interarrival times and service times are exponentially distributed. When N and K are infinite, they may be dropped from the notation. For example, $M/M/1/\infty/\infty$ is often shortened to $M/M/1$. The nurse attending 5 hospital patients might be represented by $M/M/1/5/5$.

Additional notation used throughout the remainder of this chapter for parallel server systems is listed in Table 6.2. The meanings may vary slightly from system to system. All systems will be assumed to have a FIFO queue discipline.

Table 6.2 Queueing Notation for Parallel Server Systems

P_n	Steady-state probability of having n customers in system
$P_n(t)$	Probability of n customers in system at time t
λ	Arrival rate
λ_e	Effective arrival rate
μ	Service rate of one server
ρ	Server utilization
A_n	Interarrival time between customers $n-1$ and n
S_n	Service time of the nth arriving customer
W_n	Total time spent in system by the nth arriving customer
W_n^Q	Total time spent waiting in queue by customer n
$L(t)$	The number of customers in system at time t
$L_Q(t)$	The number of customers in queue at time t
L	Long-run time-average number of customers in system
L_Q	Long-run time-average number of customers in queue
w	Long-run average time spent in system per customer
w_Q	Long-run average time spent in queue per customer

6.3.4 Server Utilization

Server utilization is defined as the proportion of time that a server is busy. Observed server utilization, denoted by $\hat{\rho}$, is defined over a specified time interval $[0, T]$. Long-run server utilization is denoted by ρ. For systems that exhibit long-run stability,

$$\hat{\rho} \longrightarrow \rho \text{ as } T \longrightarrow \infty$$

Server utilization in $G/G/1/\infty/\infty$ queues

Consider any single-server queueing system with average arrival rate λ customers per time unit, average service time $E(S) = 1/\mu$ time units, and infinite queue capacity and calling population. Notice that $E(S) = 1/\mu$ implies that, when busy, the server is working at the rate μ customers per time unit, on the average; μ is called the *service rate*. The server alone is a subsystem that can be considered as a queueing system in itself; hence, the conservation Equation (6.9), $L = \lambda w$, can be applied to the server. For stable systems, the average arrival rate to the server, say λ_s, must be identical to the average arrival rate to the system, λ (certainly $\lambda_s \leq \lambda$—customers cannot be served faster than they arrive—but, if $\lambda_s < \lambda$, then the waiting line would tend to grow in length at an average rate of $\lambda - \lambda_s$ customers per time unit, and so we would have an unstable system). For the server subsystem, the average system time is $w = E(S) = \mu^{-1}$. The actual number of customers in the server subsystem is either 0 or 1, as shown in Figure 6.9 for the system represented by Figure 6.6. Hence, the average number in the server subsystem, $\hat{L}_s$, is given by

$$\hat{L}_s = \frac{1}{T}\int_0^T (L(t) - L_Q(t))\,dt = \frac{T - T_0}{T}$$

In this case, $\hat{L}_s = 17/20 = \hat{\rho}$. In general, for a single-server queue, the average number of customers being served at an arbitrary point in time is equal to server utilization. As $T \longrightarrow \infty$, $\hat{L}_s = \hat{\rho} \longrightarrow L_s = \rho$. Combining these results into $L = \lambda w$ for the server subsystem yields

$$\rho = \lambda E(S) = \frac{\lambda}{\mu} \tag{6.12}$$

Server utilization in $G/G/c/\infty/\infty$ queues

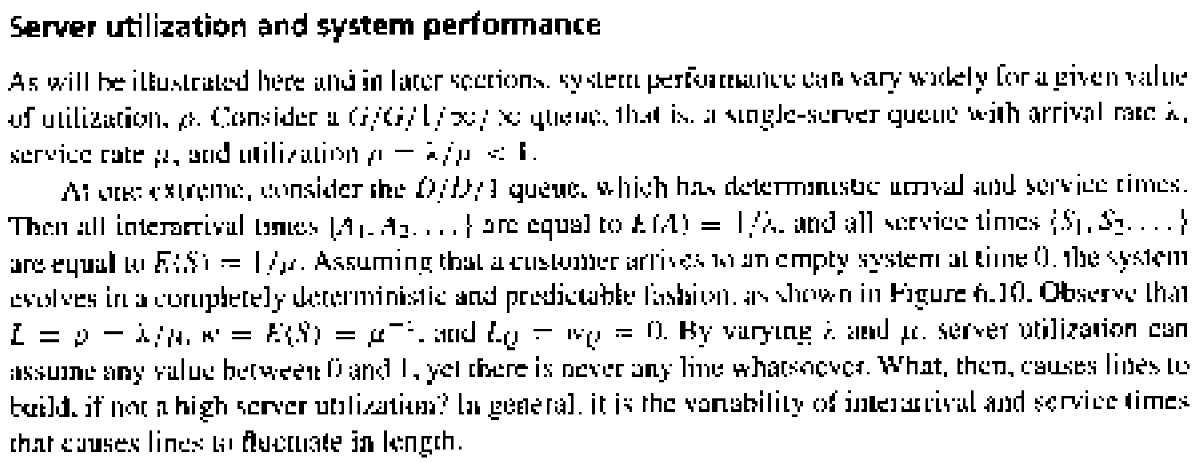
Consider a queueing system with c identical servers in parallel. If an arriving customer finds more than one server idle, the customer chooses a server without favoring any particular server. (For example, the choice of server might be made at random.) Arrivals occur at rate λ from an infinite calling population, and each server works at rate μ customers per time unit. From Equation (6.9), $L = \lambda w$, applied to the server subsystem alone, an argument similar to the one given for a single server leads to the result that, for systems in statistical equilibrium, the average number of busy servers, say L_s, is given by

$$L_s = \lambda E(S) = \frac{\lambda}{\mu} \tag{6.13}$$

Clearly, $0 \leq L_s \leq c$. The long-run average server utilization is defined by

$$\rho = \frac{L_s}{c} = \frac{\lambda}{c\mu} \tag{6.14}$$

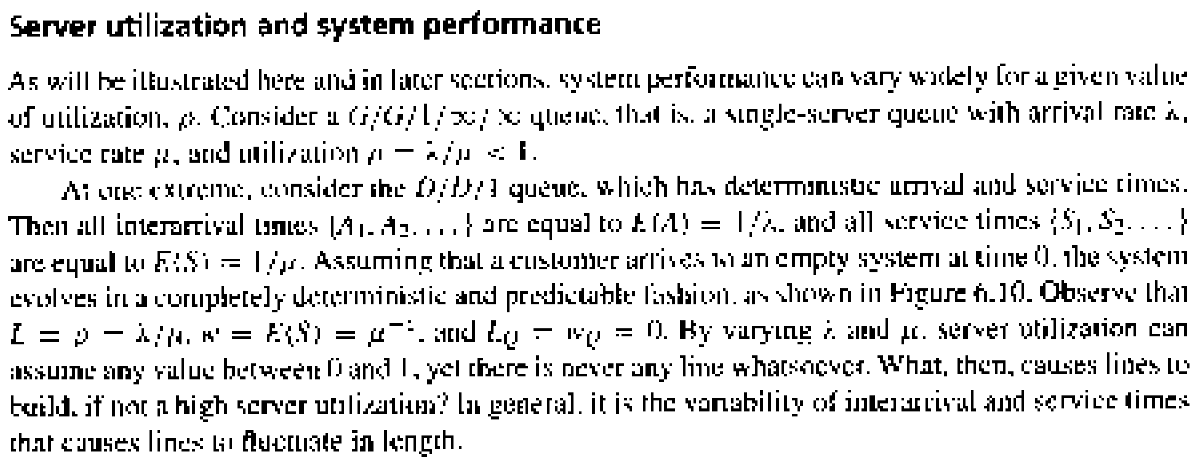
and so $0 \leq \rho \leq 1$. The utilization ρ can be interpreted as the proportion of time an arbitrary server is busy in the long run.



Server utilization and system performance

As will be illustrated here and in later sections, system performance can vary widely for a given value of utilization, ρ. Consider a $G/G/1/\infty/\infty$ queue, that is, a single-server queue with arrival rate λ, service rate μ, and utilization $\rho = \lambda/\mu < 1$.

At one extreme, consider the $D/D/1$ queue, which has deterministic arrival and service times. Then all interarrival times $\{A_1, A_2, \ldots\}$ are equal to $E(A) = 1/\lambda$, and all service times $\{S_1, S_2, \ldots\}$ are equal to $E(S) = 1/\mu$. Assuming that a customer arrives to an empty system at time 0, the system evolves in a completely deterministic and predictable fashion, as shown in Figure 6.10. Observe that $L = \rho = \lambda/\mu$, $w = E(S) = \mu^{-1}$, and $L_Q = w_Q = 0$. By varying λ and μ, server utilization can assume any value between 0 and 1, yet there is never any line whatsoever. What, then, causes lines to build, if not a high server utilization? In general, it is the variability of interarrival and service times that causes lines to fluctuate in length.



Server utilization and system performance

As will be illustrated here and in later sections, system performance can vary widely for a given value of utilization, ρ. Consider a $G/G/1/\infty/\infty$ queue, that is, a single-server queue with arrival rate λ, service rate μ, and utilization $\rho = \lambda/\mu < 1$.

At one extreme, consider the $D/D/1$ queue, which has deterministic arrival and service times. Then all interarrival times $\{A_1, A_2, \ldots\}$ are equal to $E(A) = 1/\lambda$, and all service times $\{S_1, S_2, \ldots\}$ are equal to $E(S) = 1/\mu$. Assuming that a customer arrives to an empty system at time 0, the system evolves in a completely deterministic and predictable fashion, as shown in Figure 6.10. Observe that $L = \rho = \lambda/\mu$, $w = E(S) = \mu^{-1}$, and $L_Q = w_Q = 0$. By varying λ and μ, server utilization can assume any value between 0 and 1, yet there is never any line whatsoever. What, then, causes lines to build, if not a high server utilization? In general, it is the variability of interarrival and service times that causes lines to fluctuate in length.

QUEUEING NETWORKS
A network consisting of several interconnected queues
• Network of queues
Examples
• Customers go form one queue to another in post office, bank, supermarket etc
• Data packets traverse a network moving from a queue in a router to the queue in another router
Systems modeled by queueing networks can roughly be grouped into four categories
- Open networks
- Closed networks
- Networks with population constraints (Loss Networks)
- Mixed network

Open Networks

♦ Customers arrive from outside the system are served
and then depart.
♦ Example: Packet switched data network.

Closed Networks
♦ Fixed number of customers (*K*) are trapped in the
system and circulate among the queues.
♦ Example: CPU job scheduling problem
Loss Networks with Population Constraints
♦ Customers arrive from outside the system if there is
room in the system. They enter, served and then depart.
♦ Example: queues sharing a common buffer pool –
customers are lost when arriving to full system

Mixed Network
♦ Any combination of previous types.
♦ Example: simple model of

Properties in Queueing Networks
Queueing Networks exhibit behavior not seen in single queue
scenarios
♦ **Jockeying**: Customers moving among parallel queues.
♦ **Blocking** � Customer waiting depart a server and join next queue is
unable to due to limited waiting space, and therefore stays in server
(blocking it.)
♦ **Routing** - Customer leaving a queue may have options as to where
to go next
♦ **Forking** Customer leaving a queue clones into multiple customers
7
� possibly going along different routes.
♦ **Joining** � Multiple customers are combined into a single customer
� Forking and joining are used in models of parallel processing systems, packet
fragmentation and reassembly.

Arrival theorem

In queueing theory, a discipline within the mathematical theory of probability, the **arrival theorem** (also referred to as the **random observer property**, **ROP** or **job observer property**) states that "upon arrival at a station, a job observes the system as if in steady state at an arbitrary instant for the system without that job."

The arrival theorem always holds in open product-form networks with unbounded queues at each node, but it also holds in more general networks. A necessary and sufficient condition for the arrival theorem to be satisfied in product-form networks is given in terms of Palm probabilities in Boucherie & Dijk, 1997 A similar result also holds in some closed networks.

Examples of product-form networks where the arrival theorem does not hold include reversible Kingman networks and networks with a delay protocol.

Mitrani offers the intuition that "The state of node *i* as seen by an incoming job has a different distribution from the state seen by a random observer. For instance, an incoming job can never see all '*k* jobs present at node *i*, because it itself cannot be among the jobs already present

Jackson network

In queueing theory, a discipline within the mathematical theory of probability, a **Jackson network** (sometimes **Jacksonian network**[1]) is a class of queueing network where the equilibrium distribution is particularly simple to compute as the network has a product-form solution.

James Jackson (UCLA Math professor) did the basic work on queueing networks
♦ Jackson Networks – special class of open queueing networks
 Network of *M* queues
 There is only one class of customers in the network
 A job can leave the network from any node
 All service times are exponentially distributed with rate *i* at queue i
 The service discipline at all nodes is FCFS.
 All external customer arrival processes are Poisson processes with rate *at* queue *i*

It was the first significant development in the theory of networks of queues, and generalising and applying the ideas of the theorem to search for similar product-form solutions in other networks has been the subject of much research,[2] including ideas used in the development of the Internet.[3] The networks were first identified by James R. Jackson[4][5] and his paper was re-printed in the journal *Management Science's* 'Ten Most Influential Titles of Management Sciences First Fifty Years.'[6]

Jackson was inspired by the work of Burke and Reich,[7] though Jean Walrand notes "product-form results … [are] a much less immediate result of the output theorem than Jackson himself appeared to believe in his fundamental paper".[8]

An earlier product-form solution was found by R. R. P. Jackson for **tandem queues** (a finite chain of queues where each customer must visit each queue in order) and cyclic networks (a loop of queues where each customer must visit each queue in order).[9]

A Jackson network consists of a number of nodes, where each node represents a queue in which the service rate can be both node-dependent (different nodes have different service rates) and state-dependent (service rates change depending on queue lengths). Jobs travel among the nodes following a fixed routing matrix. All jobs at each node belong to a single "class" and jobs follow the same service-time distribution and the same routing mechanism. Consequently, there is no notion of priority in serving the jobs: all jobs at each node are served on a first-come, first-served basis.

Jackson networks where a finite population of jobs travel around a closed network also has a product-form solution described by the Gordon–Newell theorem

Necessary conditions for a Jackson network

A network of *m* interconnected queues is known as a **Jackson network** or **Jacksonian network** if it meets the following conditions:

1. if the network is open, any external arrivals to node *i* form a Poisson process,
2. all service times are exponentially distributed and the service discipline at all queues is first-come, first-served,
3. a customer completing service at queue *i* will either move to some new queue *j* with probability P_{ij} or leave the system with probability $1 - \sum_{j=1}^{m} P_{ij}$, which, for an open network, is non-zero for some subset of the queues,
4. the utilization of all of the queues is less than one.