400 LEVEL CYBERSECURITY

DEPARTMENT OF COMPUTER SCIENCE, BINGHAM UNIVERSITY, KARU

CMP411 – SYSTEM MODELLING AND SIMULATION

LECTURE I – INTRODUTION

Course Content

Basic Definitions and Uses

Simulation Process

Basic Statistics Distribution Theories

Model and Simulation

Modelling Methods

Queues:

Basic Components

Kendal Notation

Queuing Rules

Little's Law

Queuing Networks

Special Types of Queues

Stochastic Processes

Discrete State and Continuous State Processes

Markov Processes

Birth-Death Processes

Poisson Process

Random Numbers

Types of random numbers

Basic Definitions - Model, Modelling and Simulations

A **Model** is a human construct that helps us understand the real world systems. It is usually made up of input, processing and output sub-systems

A **model** in general is a pattern, plan, miniaturized representation, or description designed to show the main object or workings of an object, system, or concept.

In science, Models are often theoretical constructs that represent any particular thing with a set of variables and a set of logical and or quantitative relationships between them. Models in this sense are constructed to enable reasoning within an idealized logical framework about these processes and are an important component of scientific theories

Basic Definitions – Model, Modelling and Simulations cont'd

Modelling is the process of generating abstract, conceptual, graphical and/or mathematical models. There are several methods, techniques and theories for all kinds of specialized scientific modelling.

Modelling also means to find relations between systems and models. That is the abstraction of real or imaginary worlds that are created by humans for the purpose of performing experiments, making projections, etc.

Simulation –is the manipulation of a model in such a way that it operates on time or space to compress it, thus enabling one to perceive the interactions that would not otherwise be apparent because of their separation in time or space.

Why Modelling and Simulations

When there is significant uncertainty regarding the outcome or consequences of a particular alternative under consideration. It allows you to deal with uncertainty and imprecision in a quantifiable way.

When the system under consideration involves complex interactions and requires input from multiple disciplines. In this case, it is difficult for any one person to easily understand the system. A simulation of the model can in such situations act as the framework to integrate the various components in order to better understand their interactions. As such, it becomes a management tool that keeps you focused on the "big picture" without getting lost in unimportant details.

Why Modelling and Simulations

when the consequences of a proposed action, plan or design cannot be directly and immediately observed (i.e., the consequences are delayed in time and/or dispersed in space) and/or it is simply impractical or prohibitively expensive to test the alternatives directly

What is Simulation?

- A Simulation of a system is the operation of a model, which is a representation of that system.
- The model is amenable to manipulation which would be impossible, too expensive, or too impractical to perform on the system which it portrays.
- The operation of the model can be studied, and, from this, properties concerning the behavior of the actual system can be inferred.

Computer simulation

- A computer simulation or a computer model is a computer program that attempts to simulate an abstract model of a particular system.
- Computer simulations have become a useful part of mathematical modelling of many natural systems in physics, chemistry and biology, human systems in economics, psychology, and social science and in the process of engineering new technology, to gain insight into the operation of those systems.

Computer simulation Cont'd

- Traditionally, the formal modeling of systems has been via a mathematical model, which attempts to find analytical solutions to problems which enables the prediction of the behaviour of the system from a set of parameters and initial conditions.
- Computer simulations build on, and are a useful adjunct to purely mathematical models in science, technology and entertainment.
- The reliability and the trust people put in computer simulations depends on the validity of the simulation model.

Types of Computer Simulations

- Two types of computer simulation are often distinguished:
  - *equation-based simulations* and
  - *agent-based* (or *individual-based*) *simulations*.
- Computer Simulations of both types are used for three different general sorts of purposes:
  - prediction (both pointwise and global/qualitative),
  - understanding, and
  - exploratory or heuristic purposes.

Equation-based Simulations

- Equation-based simulations are most commonly used in the physical sciences and other sciences where there is governing theory that can guide the construction of mathematical models based on differential equations.
- The term “equation based” here refers to simulations based on the kinds of global equations associated with physical theories—as opposed to “rules of evolution”.
- Equation based simulations can either be particle-based, where there are n many discrete bodies and a set of differential equations governing their interaction, or they can be field-based, where there is a set of equations governing the time evolution of a continuous medium or field.

Equation-based Simulations Cont'd

- An example of the former is a simulation of galaxy formation, in which the gravitational interaction between a finite collection of discrete bodies is discretized in time and space.
- An example of the latter is the simulation of a fluid, such as a meteorological system like a severe storm.
- Here the system is treated as a continuous medium—a fluid—and a field representing its distribution of the relevant variables in space is discritized in space and then updated in discrete intervals of time.

Agent-based simulations

- Agent-based simulations are most common in the social and behavioral sciences, though we also find them in such disciplines as artificial life, epidemiology, ecology, and any discipline in which the networked interaction of many individuals is being studied.
- Agent-based simulations are similar to particle-based simulations in that they represent the behavior of n-many discrete individuals.
- But unlike equation-particle-based simulations, there are no global differential equations that govern the motions of the individuals.
- Rather, in agent-based simulations, the behavior of the individuals is dictated by their own local rules

Agent-based simulations cont'd

- To give one example: a famous and groundbreaking agent-based simulation was Thomas Schelling's (1971) model of "segregation."
- The agents in his simulation were individuals who "lived" on a chessboard.
- The individuals were divided into two groups in the society (e.g. two different races, boys and girls, smokers and non-smokers, etc.)
- Each square on the board represented a house, with at most one person per house.
- An individual is happy if he/she has a certain percent of neighbors of his/her own group.
- Happy agents stay where they are, unhappy agents move to free locations.
- Schelling found that the board quickly evolved into a strongly segregated location pattern if the agents' "happiness rules" were specified so that segregation was heavily favored.
- Surprisingly, however, he also found that initially integrated boards tipped into full segregation even if the agents' happiness rules expressed only a mild preference for having neighbors of their own type.

Multiscale Simulations

- Some simulation models are hybrids of different kinds of modeling methods.
- Multiscale simulation models, in particular, couple together modeling elements from different scales of description.
- A good example of this would be a model that simulates the dynamics of bulk matter by treating the material as a field undergoing stress and strain at a relatively coarse level of description, but which zooms into particular regions of the material where important small scale effects are taking place, and models those smaller regions with relatively more fine-grained modeling methods.
- Such methods might rely on molecular dynamics, or quantum mechanics, or both—each of which is a more fine-grained description of matter than is offered by treating the material as a field.
- Multiscale simulation methods can be further broken down into:
  - serial multiscale and
  - parallel multiscale methods.

Multiscale Simulations - serial multiscale

- The more traditional method is serial multi-scale modeling

- The idea here is to choose a region, simulate it at the lower level of description, summarize the results into a set of parameters digestible by the higher level model, and pass them up to into the part of the algorithm calculating at the higher level.

- Serial multiscale methods are not effective when the different scales are strongly coupled together.

Multiscale Simulations - parallel multiscale modeling

- When the different scales interact strongly to produce the observed behavior, what is required is an approach that simulates each region simultaneously.
- This is called parallel multiscale modeling.
- Parallel multiscale modeling is the foundation of a nearly ubiquitous simulation method: so called “sub-grid” modeling.

Multiscale Simulations: parallel multiscale modeling - Sub-grid modeling

- Parallel multiscale modeling is the foundation of a nearly ubiquitous simulation method: so called “sub-grid” modeling.
- Sub-grid modeling refers to the representation of important small-scale physical processes that occur at length-scales that cannot be adequately resolved on the grid size of a particular simulation.
- (Remember that many simulations discretize continuous equations, so they have a relatively arbitrary finite “grid size.”)
- In the study of turbulence in fluids, for example, a common practical strategy for calculation is to account for the missing small-scale *vortices* (or *eddies*) that fall inside the grid cells.
- This is done by adding to the large-scale motion an *eddy viscosity* that characterizes the transport and dissipation of energy in the smaller-scale flow—or any such feature that occurs at too small a scale to be captured by the grid.

Multiscale Simulations: parallel multiscale modeling - Sub-grid modeling Cont'd

- In climate science and kindred disciplines, sub-grid modeling is called "parameterization."
- This, again, refers to the method of replacing processes—ones that are too small-scale or complex to be physically represented in the model—by a more simple mathematical description.
- This is as opposed to other processes—e.g., large-scale flow of the atmosphere—that are calculated at the grid level in accordance with the basic theory. It is called "parameterization" because various non-physical *parameters* are needed to drive the highly approximative algorithms that compute the sub-grid values.

Multiscale Simulations: parallel multiscale modeling - Sub-grid modeling Cont'd

- Examples of parameterization in climate simulations include the descent rate of raindrops, the rate of atmospheric radiative transfer, and the rate of cloud formation.
- For example, the average cloudiness over a 100 km^2 grid box is not cleanly related to the average humidity over the box.
- Nonetheless, as the average humidity increases, average cloudiness will also increase—hence there could be a parameter linking average cloudiness to average humidity inside a grid box.
- Even though modern-day parameterizations of cloud formation are more sophisticated than this, the basic idea is well illustrated by the example.
- Winsberg (2001) has argued that the use of sub-grid modeling methods in simulation has important consequences for understanding the structure of the epistemology of simulation.

Monte Carlo Simulations

- Monte Carlo simulations, is a powerful way to understand complex systems and their uncertainties
- Imagine playing a game with dice – we roll the dice many times to predict the outcome.
- Monte Carlo simulations work in a similar way, but with mathematical models.
- Instead of dice, we use random numbers to represent uncertain factors in a system.
- By running these simulations repeatedly, we can explore different 'what-if' scenarios, helping us see how a system behaves under various conditions.
- This approach is incredibly useful across various fields, like finance, engineering, and science, where we need to make informed decisions and deal with uncertain situations.

Monte Carlo Simulations

- In the scientific literature, there is another large class of computer simulations called Monte Carlo (MC) Simulations.
- MC simulations are computer algorithms that use randomness to calculate the properties of a mathematical model and where the randomness of the algorithm is not a feature of the target model.
- An example is the use of a random algorithm to calculate the value of π.
- If you draw a unit square on a piece of paper and inscribe a circle in it, and then randomly drop a collection of objects inside the square, the proportion of objects that land in the circle would be roughly equal to π/4.
- A computer simulation that simulated a procedure like that would be called a MC simulation for calculating π.

Types of Data/Information Needed to Develop a Simulation Model

- The overall process flow and its associated resources
- What is being produced, served, or acted upon by the process (entities)
- Frequency at which the entities arrive in the process
- How long do individual steps in the process take
- Probability distributions that characterize real life uncertainties and variations in the process

The modeling Process

Modeling: A simplified formal description of a suitable extract from the item of interest, which will then serve as the basis for the subsequent computations. Involves **examining** the real world, **extraction** of essential features from the real world and construction of the model using the extracted essential features.

Computation* or *Simulation: The model will be preprocessed (e.g., discretized) so that it is compatible with a computer platform. The solution of this preprocessed model requires the identification of efficient algorithms.

Implementation: The computational algorithms previously determined must be implemented efficiently (with respect to computational time and storage complexities, parallelization issues, etc.) on the target architecture or architectures. This is essentially translating the algorithms at the computation/simulation level into a software system.

The modeling Process cont'd

visualisation (***data exploration***): The data resulting from a simulation run must be interpreted. In some cases—e.g., for scalar quantities such as the drag coefficient in aero dynamics—this will be easy, in others—e.g., for high dimensional data sets—extracting the relevant information from the flood of numbers is a science of its own.

The *validation*: How reliable are the results? Sources for errors lurk in the model, in the algorithm, in the code or in the interpretation of the results. Therefore, it is important to compare different models, different algorithms, and different codes, as well as simulation results with manual experiments. Depending on the source of the error, the process has to be restarted at the respective step and the pipeline has to be traversed once more starting from this point.

Embedding: Simulations take place in a context—e.g., a development or production process—and should be integrated accordingly. This requires the definition of interfaces, a reasonable software engineering, simple testing environments, etc

Applications of Modelling

Modelling finds applications in many industries – some are mathematical formula heavy and some more descriptive

Geophysics, researchers want to understand processes which eventually lead to earthquakes

Drug design is concerned with the systematic design of agents having an exactly specified functionality.

medicine increasingly utilizes models, e.g., in the research of aneurysms or the optimization of implants.

climate research, which are driven by models, have had a high audience appeal—ranging from global warming, holes in the ozone layer or the future of the gulf stream. On a shorter term, is ***weather forecasting***, which relies on a mix of computations and measurements.

automobile industry - Whether one considers crash tests (structural mechanics), deep drawing (structure optimization), aerodynamics or air conditioning (fluid dynamics), sound emission (aeroacoustics), fuel injection (combustion), vehicle dynamics (optimal control) or sensor and actuator technologies (coupled systems, micro-electro-mechanical systems) involve models

semiconductor industry depends completely on models and simulations—examples are given by device simulation (transistors etc.), process simulation (production of highly purified crystals), circuit simulation as well as questions regarding the optimization of chip layout. Other application areas are: economics, banking and insurance models, traffic technology, population

SYSTEMS AND COMPONENTS

- Systems and components refer to the fundamental building blocks of complex systems that are studied and analysed.
- Systems are overarching entities composed of interconnected components or elements, each of which plays a distinct role in achieving the system's objectives.
- Understanding systems and components is crucial in the context of modelling and simulation as it involves identifying, defining, and characterizing the relationships, behaviours, and interactions among these elements.
- This knowledge serves as the foundation for creating accurate mathematical models and simulations, allowing for the exploration, analysis, and optimization of real-world systems in various domains, from engineering and economics to healthcare and logistics.

EXAMPLES OF SYSTEMS AND COMPONENTS

System	Entities	Attributes	Activities	Events	State Variables
Banking	Customers	Checking account balance	Making deposits	Arrival; Departure	# of busy tellers; # of customers waiting

Note: State Variables may change continuously (continuous sys.) over time or they may change only at a discrete set of points (discrete sys.) in time.

Types of Models - Discrete vs. Continuous Models

Discrete Models

Discrete Values: Variables take on discrete and separate values representing integers and countable sets. i.e they exploit discrete or combinatorial descriptions (binary or integer quantities, state transitions in graphs or automata). They deal with data that is countable with finite values that have nothing in-between. E.g count of items, number of students in a class.

Time steps: Discrete models advance in time through discrete time steps. Events occur at specific time points, and the system's state changes instantaneously at these time steps. This is often represented by a sequence of events or a timeline.

Applications: Discrete models are used computer science (e.g., discrete-event simulation), discrete mathematics, and combinatorics. They are well-suited for systems with distinct states, events, or decisions, such as queuing systems, network protocols, and digital circuits.

Types of Models – Discrete vs. Continuous Models – cont'd

Continuous Models

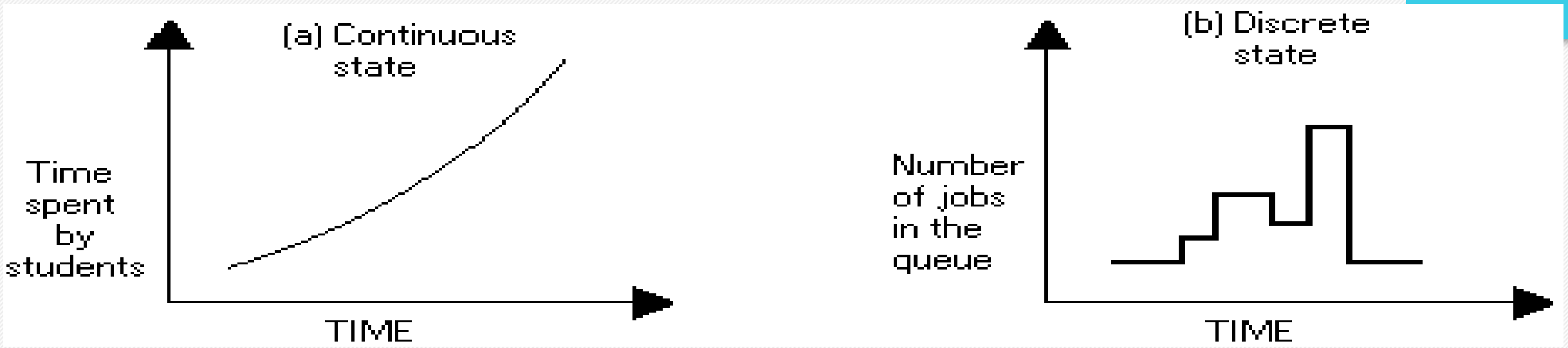
Continuous Values: are based on real-valued or continuous descriptions (real numbers, physical quantities, algebraic equations, differential equations). Deals with data that is measurable containing fractions with decimal values. i.e. variables can take on a wide range of values within a continuous range. These values are often represented by real numbers, e.g. temperature, distance, and time measured in fractions of a second.

Continuous Time: Continuous models describe systems where time is continuous and flows smoothly, rather than progressing in discrete steps. Changes in the system's state happen smoothly and continuously over time.

Applications: Continuous models are used in physics (e.g., modeling motion and fluid dynamics), engineering (e.g., control systems and structural analysis), and mathematical calculus.

Graphical Examples of Discrete vs continuous models

CPU scheduling model vs. number of students attending the class.



Types of Models - Deterministic vs. Stochastic Models

Deterministic Models

- Deterministic models assume that all input parameters and initial conditions are known with certainty without the involvement of randomness.
- They are characterized by fixed, well-defined relationships between variables, where the outcome is entirely determined by the inputs and the model structure.
- Typically represented by equations, and they produce a single, specific output for a given set of inputs.

An example of a deterministic model is a simple linear regression model that predicts a person's weight based on their height. In this model, each height value corresponds to a single, specific weight prediction, assuming no measurement errors or variability.

Types of Models - Deterministic vs. Stochastic Models

Stochastic Model

- Incorporate randomness and uncertainty explicitly into the modeling process.
- They assume that certain aspects of the system are inherently random, and they account for this randomness using probability distributions.
- the same set of parameter values and initial conditions will lead to an ensemble of different outputs, (i.e. generate a range of possible outcomes, each with a probability associated with it.)

An example of the application of stochastic model is a stock price model that incorporates random fluctuations to account for market volatility. In this model, the future stock price is not determined with certainty but follows a probability distribution, reflecting the inherent uncertainty in financial markets.

Random Numbers

Random numbers are numbers that show no consistent pattern with each number in a series and are neither affected in any way by the preceding number, nor predictable from it. i.e they are generated unpredictably or by chance, without any specific pattern or order.

Application areas of random numbers include: statistics, cryptography, simulations, gaming, etc.

Random Number Generators – (RNG) hardware device or software algorithm that generates a number that is taken from a limited or unlimited distribution and outputs it.

Random Numbers cont’d

Types of Random Numbers and Random Number Generators (RNGs)

Pseudorandom Numbers

- These numbers are generated by algorithms and are not truly random but appear random.
- They are deterministic and based on an initial value called a seed.
- If you use the same seed, you will get the same sequence of numbers.
- Pseudorandom number generators are widely used in computer programs and applications because they are computationally efficient and sufficient for many purposes.

True Random Numbers

- These numbers are generated from a source of true randomness,
- such as atmospheric noise or radioactive decay.
- True random numbers are not predictable and are not generated by algorithms.
- They are used in applications that require a high degree of unpredictability and security, such as cryptographic systems.

Properties of Good Random Number Generators

i. **Uniform Distribution:** The **RNG** should produce numbers that are uniformly distributed across the entire range of possible values. This means that every number within the range has an equal probability of being generated.

ii. **Independence:** Each generated number should be statistically independent of the previous numbers. In other words, the outcome of one random number should not provide any information about the next number.

iii. **Periodicity:** A good **RNG** should have a long period, meaning it takes a very long time for the sequence of numbers to repeat. Short periods can lead to predictable and non-random behaviour.

iv. **Reproducibility:** While random numbers should be unpredictable, it should be possible to reproduce the same sequence of random numbers when needed by using the same seed value. This is crucial for debugging and reproducibility in simulations.

v. **Statistically Unpredictable**: The generated numbers should pass various statistical tests for randomness, such as the Chi-squared test, Kolmogorov-Smirnov test, or spectral test. Passing these tests indicates that the RNG produces numbers that behave like truly random values.

Properties of Good Random Number Generators – cont'd

vi. **Low Bias:** The RNG should not exhibit any noticeable bias, where certain numbers are generated more frequently than others. Biased RNGs can lead to incorrect results in statistical analysis or simulations.

vii. **Speed and Efficiency:** RNGs should generate random numbers quickly, especially in applications where a large number of random values are needed, such as Monte Carlo simulations.

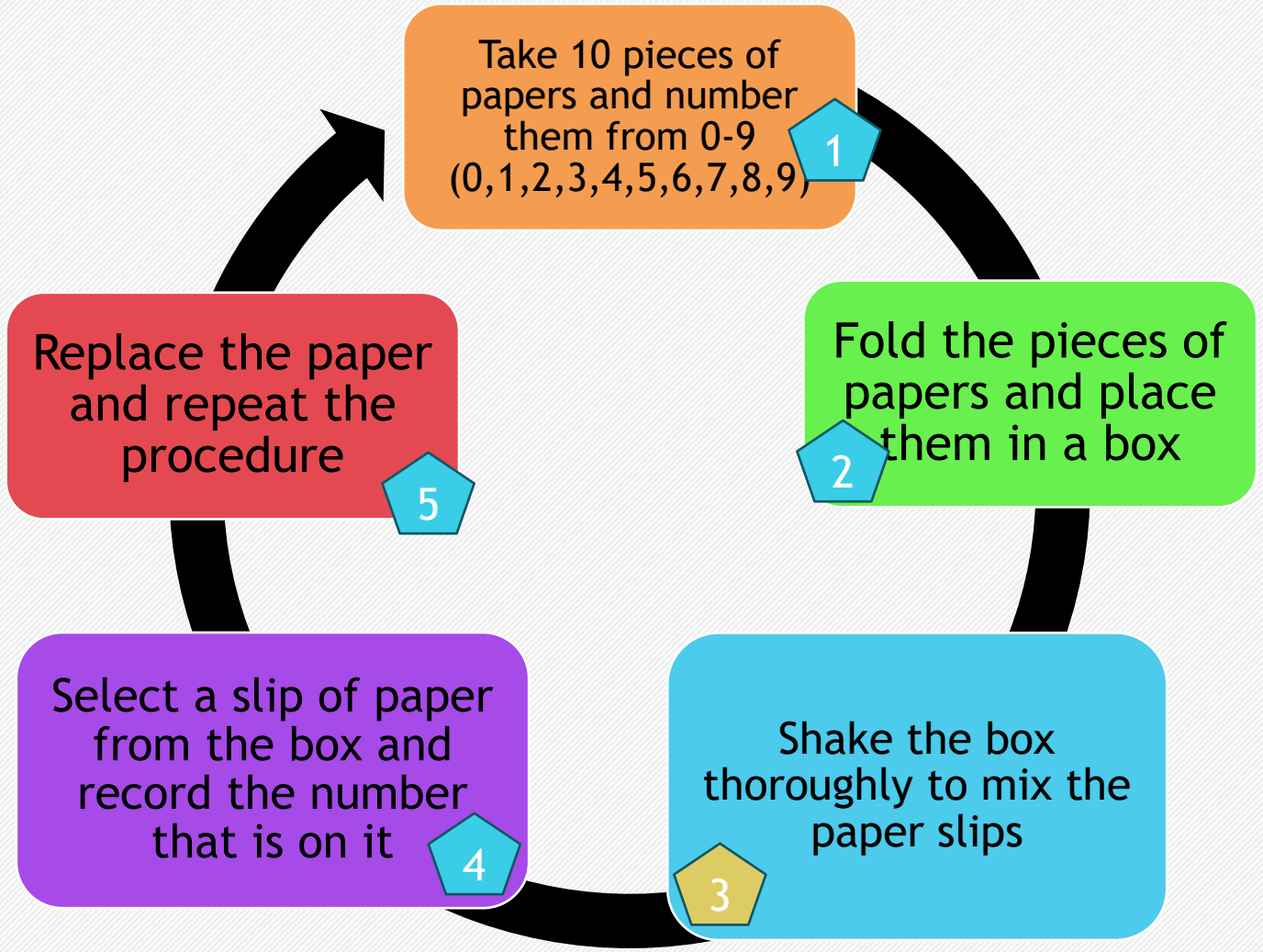
viii. **Seed Initialization:** The RNG should provide a way to initialise the random sequence by specifying a seed value. This allows for reproducibility and controlled experimentation.

ix. **Resistance to Predictive Attacks:** In cryptographic applications, good RNGs should be resistant to predictive attacks, meaning that it should be computationally infeasible to predict future random values even if an attacker knows a portion of the sequence.

x. **Periodic Checks:** For long-running simulations or applications, the RNG should be periodically reseeded to ensure that the generated sequence remains unpredictable and statistically sound.

Random Numbers - Pseudorandom Number Generation

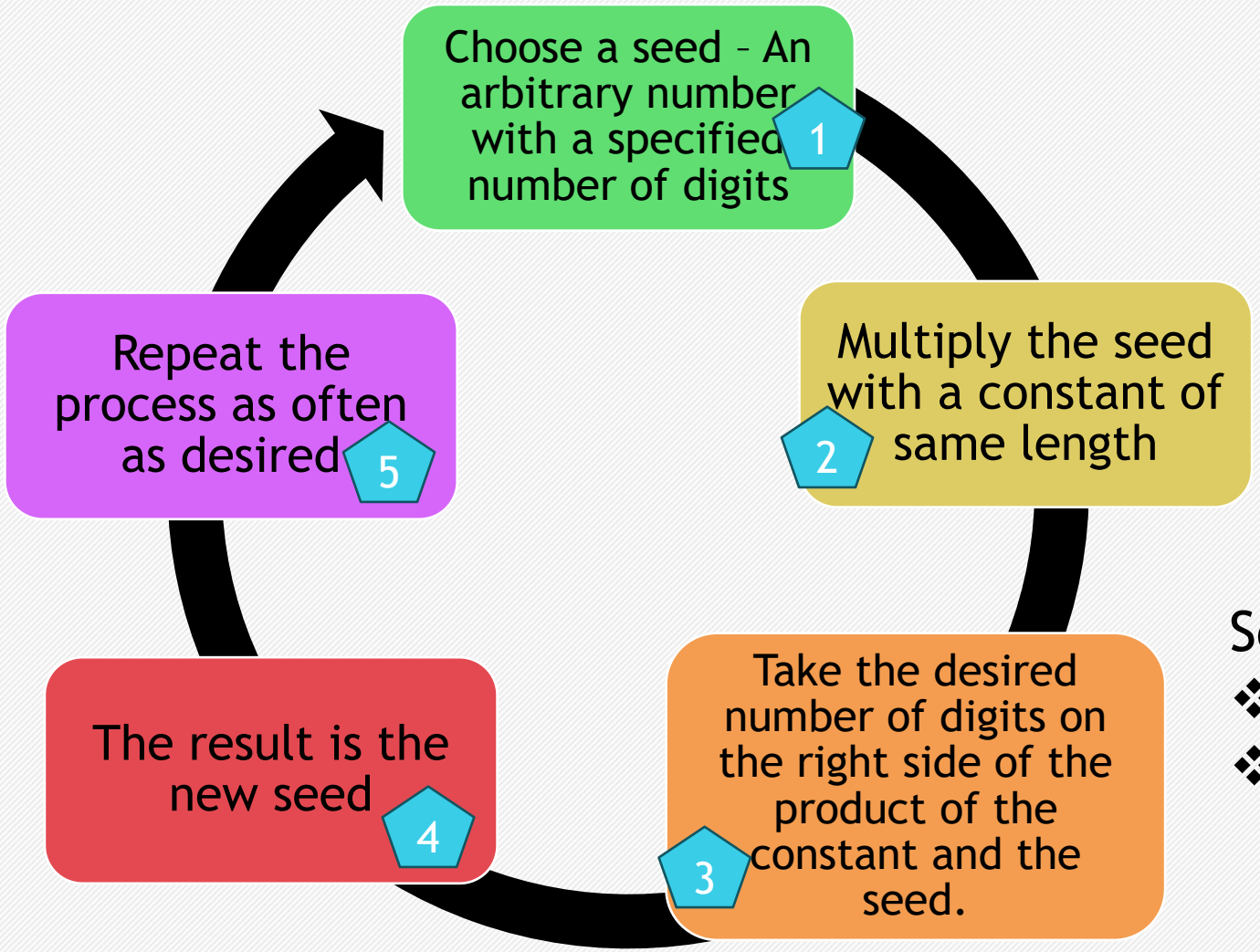
Random number generated by humans for use by computers or in simulations



Supposing your recorded random numbers are recorded below: **3580834292613567….** and you intend to generate random numbers with four digits, your results will be as follows:

3580
8342
9261
3567
.
.
Nth number

Random Numbers – Congruential Generator



SN	Seed	Constant	Product
1	5274	8132	42888168
2	8168		66422176
3	2176		17695232
4	5232		42546624
5	6624		53866368
6	6368		

Seed numbers can be generated from:

- Computer system clock
- Interval between keystrokes in milliseconds

Random Numbers - Random Numbers - Congruential Generator

The Congruential Method

The Congruential Method is a widely used algorithm for generating pseudo-random numbers. Pseudo-random numbers are not truly random but are generated using a deterministic process that appears random for most practical purposes. The method is named for the congruential relation it uses to generate the numbers.

The basic idea of the Congruential Method involves iterating through a sequence of numbers, each of which depends on the previous number. The algorithm uses the following recurrence relation:

$X_{n+1}=(aX_n+c) \bmod m$

Where:

- Xn is the current pseudo-random number in the sequence.
- X_{n+1} is the next pseudo-random number in the sequence.
- a, c, and m are constants chosen to control the properties of the generated sequence.

How the method works:

Initialization: You start with an initial seed value, X_0. This seed is the starting point for generating the sequence.

Iteration: To generate subsequent pseudo-random numbers, you apply the recurrence relation above.

Each new number depends on the previous one, and the operation **mod m** ensures that the result stays within the range [0, m−1].

Repeat: You continue this process, generating a sequence of numbers as long as needed. The sequence appears random but is entirely determined by the choice of constants and the initial seed value.

Random Numbers - Random Numbers - Congruential Generator

Key considerations when using the Congruential Method

Choice of Constants:

a: The multiplier, which should be a large prime number or a number with good properties in terms of randomness.

c: The increment, which is typically a small constant.

m: The modulus, which determines the range of the generated numbers. It should be a large prime number or a power of two for efficient computation.

Seed Value

The quality of the pseudo-random sequence depends on the choice of the initial seed value. Different seed values will result in different sequences. A poor choice of the seed can lead to patterns in the generated numbers.

Period Length

The period of the sequence is the number of values it generates before it starts repeating. The choice of constants and the seed value determine the period length. Ideally, you want a long period to avoid quickly repeating sequences.

Statistical Properties

It's essential to test the generated sequence for statistical properties, such as uniformity and independence, to ensure it is suitable for the intended application.

Random Numbers - RANECU

RANECU (short for "**RA**ndom **N**umber **ECU**niform") is a type of random number generator algorithm. Specifically, it's a pseudorandom number generator (PRNG) that produces a sequence of seemingly random numbers.
The RANECU algorithm is a particular PRNG that was developed by James Gentle and Jean-Hugues Réty. It's known for its simplicity and ease of implementation, making it suitable for various applications. The algorithm typically involves a series of mathematical operations on the current seed to generate a new pseudorandom number. RANECU is classified as a Lehmer-style generator, which is a type of linear congruential generator (LCG).

The key features of RANECU may include:

Initialization: It requires an initial seed value to start generating the random sequence. If the same seed is used, the same sequence of numbers will be generated, making it useful for reproducible results in simulations and testing.

Periodicity: Like other PRNGs, RANECU has a finite period. This means that after a certain number of iterations, the sequence will repeat itself. The length of the period is an important characteristic of a PRNG.

Uniformity: The generated numbers should have good statistical properties and be uniformly distributed, meaning each number in the range of possible outcomes is equally likely.

Efficiency: RANECU is often designed to be computationally efficient, making it suitable for use in resource-constrained environments.

It is important to note that RANECU is just one of many PRNG algorithms, and the choice of PRNG depends on the specific requirements of the application, including considerations of randomness, speed, and security.

Random Numbers - The Quadratic congruential method

The Quadratic congruential method

The Quadratic Congruential Method (QCM) is a type of pseudorandom number generator (PRNG) that is used to generate sequences of seemingly random numbers. It is a variation of the more common linear congruential generator (LCG) method, which generates pseudorandom numbers using a linear recurrence equation. The key difference is that QCM uses a quadratic recurrence equation. The general form of the QCM equation is as follows:

$X_{n+1}=(aX_n^2+bX_n+c) \bmod m$

Where:

Xn is the current random number.

Xn+1 is the next random number in the sequence.

a, b, c are coefficients that determine the behavior of the QCM.

m is the modulus, which defines the range of possible values.

Just like in linear congruential generators, the values of a, b, c, and m are crucial to the quality and statistical properties of the random numbers generated by the QCM. Properly chosen values are required to ensure a long period (the number of values before the sequence repeats) and good randomness properties.

Random Numbers - The Quadratic congruential method

The QCM has some advantages over LCGs, such as potentially longer periods and better statistical properties for certain parameter choices. However, it can also be more complex to set up due to the quadratic equation, and not all parameter combinations will produce good random sequences. Careful testing and analysis of the generated numbers are essential when using QCM.

The choice of which PRNG to use depends on the specific requirements of the application. Some applications may require higher-quality random numbers and might prefer more advanced PRNGs or even true random number generators (TRNGs) for cryptographic or security-critical purposes. Others may use simpler PRNGs like QCM for non-cryptographic tasks where the statistical properties of the generated numbers are less critical.

Queue Modelling and Simulation

- **Queues** are found where there exist one or more shared resources.
- A **queuing system** is any system where the customer requests a service from a **finite - capacity resource**
- During the execution, a waiting line is formed in the system because the arrival time of each customer is not predictable, and the service time often exceeds customer inter - arrival times
- A significant number of arrivals makes each customer wait in line longer than usual.

Queue Modelling and Simulation

Queuing models

- Constructed to analyze the performance of a dynamic system where waiting can occur.

The goals of a queuing model are:

- To minimize the average number of waiting customers in a queue

- Predict the estimated number of facilities in a queuing system.

- The performance results of queuing model simulation are produced at the end of a simulation in the form of aggregate statistics.

Performance and Analysis of Queuing Systems

Two approaches are available: Analytical Modeling and **Simulation**

- An analytical model is the abstraction of a system based on probability theory.

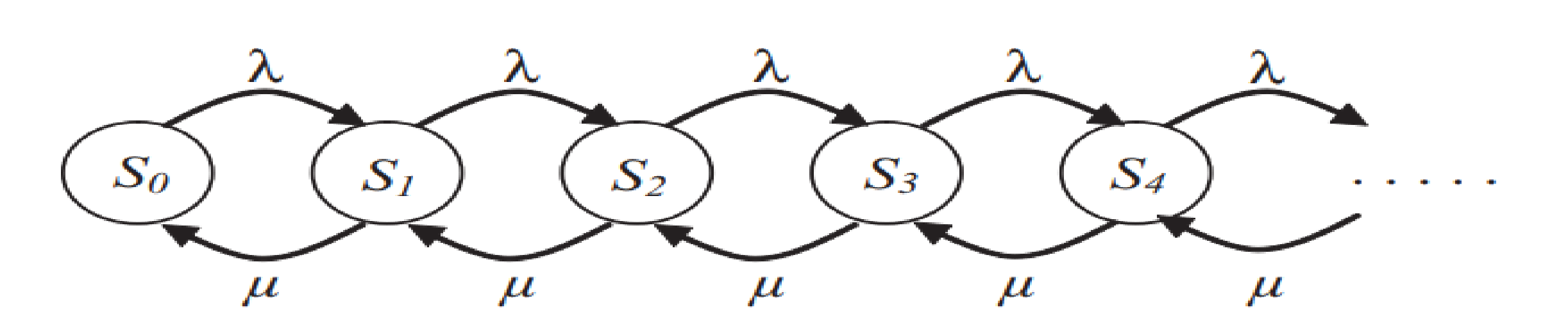
- Represents the description of a formal system consisting of equations used to estimate the performance of the system.

Analytical model Solution

Supposing we have a single - server queuing model,

Assuming that the queue discipline is first - in first - out (FIFO), the customer who comes in first is served first,

A queuing system can be modeled based on a Markov chain, which consists of discrete state spaces with the property that the next state depends only on the current state, and is independent of the previous state



The first five states of a Markov chain representing the states of a single - server queue are presented in diagram above.

Analytical model Solution

Each state is defined by the number of customers in the system.

The number of customers in the system increases based on the arrival rate (λ) and decreases based on the service rate (μ).

Si denotes the *i*th state of a Markov chain, and the number *i* is the number of customers in the system.

Traffic intensity or utilization (ρ) is defined by:

$$\rho = \frac{\lambda}{\mu}$$

where λ and μ refer to the arrival and service rates respectively

Queuing Models - Attributes

A queuing model is described by its attributes:

- **Call Population-** can be finite or infinite, it is the pool of customers who possibly can request the service in the near future.
- **Arrival and Service Pattern** - are the two most important factors determining behaviours of queuing models. A queuing model may be **deterministic** or **stochastic**. If stochastic, new arrivals occur in a random pattern and their service time is obtained by probability distribution.

- ✓ **Arrival rate** is the mean number of customers per unit time, and the **service rate** is defined by the capacity of the server in the queuing model.
- ✓ If the service rate is less than the arrival rate, the size of the queue will grow infinitely.
- ✓ The arrival rate must be less than the service rate in order to maintain a stable queuing system
- ✓ The randomness of arrival and service patterns cause the length of waiting lines in the queue to vary

Queuing Models - Attributes

- **Queue Discipline-** Is the strategy for selecting the next customers from queue. Queue discipline is a scheduling algorithm to select the next customer from the queue. The common algorithms of queue discipline are:

- First - In First - Out (FIFO),
- Last - In First - Out (LIFO),
- Service In Random Order (SIRO),
- And Priority Queue

- ✓ In real world settings, the earlier arrived customer is usually selected from a queue in the real world, thus the most common queue discipline is FIFO.
- ✓ In a **priority** queue discipline, each arrival has its priority. The arrival that has the highest priority is enters the server from queue among waiting customers. Priority could be preemptive or nonpreemptive

Queuing Models - Attributes

- Queue Capacity - The limits of the system as per the number of customers in line.
- Number Of Servers - Refers to the total number of servers or service points attending to the customers on the queue

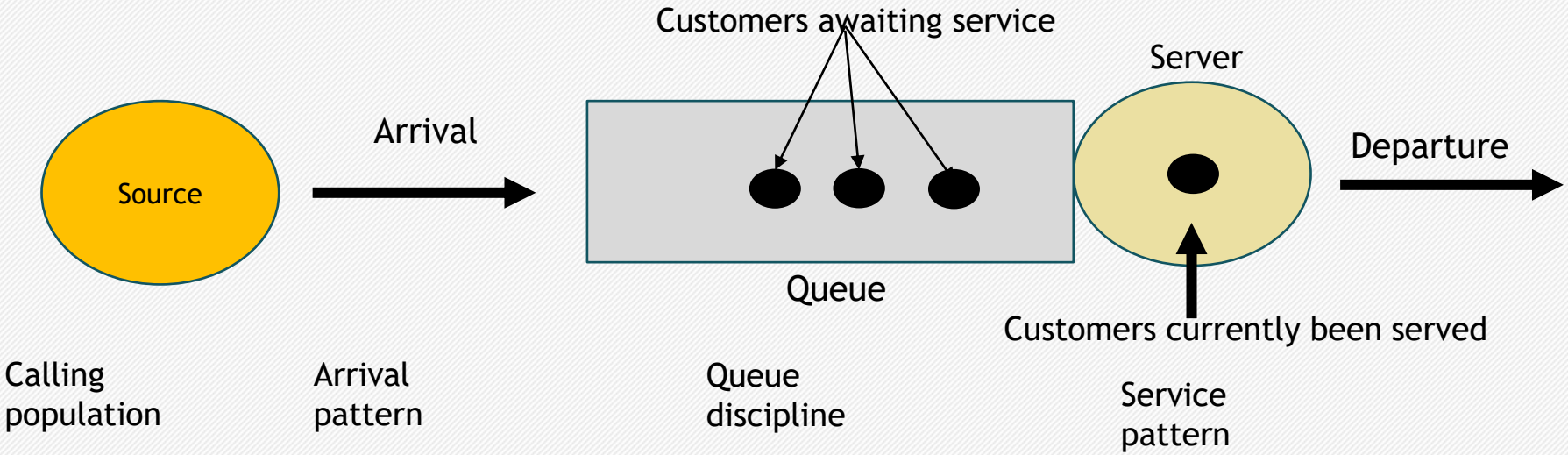
Queuing Models - customer decisions on a queue.

Customers in the real world may take any of the following decisions when they arrive a queue:

- **Balking** - when an arriving customer does not enter the queue due to the limited queue capacity.
- **Reneging** - customer leaves the queue after waiting in a queue upon arrival. This is without been served
- **Jockeying** - customer decides to switch the queue for earlier service. Especially in a parallel queue – multi-server queuing model

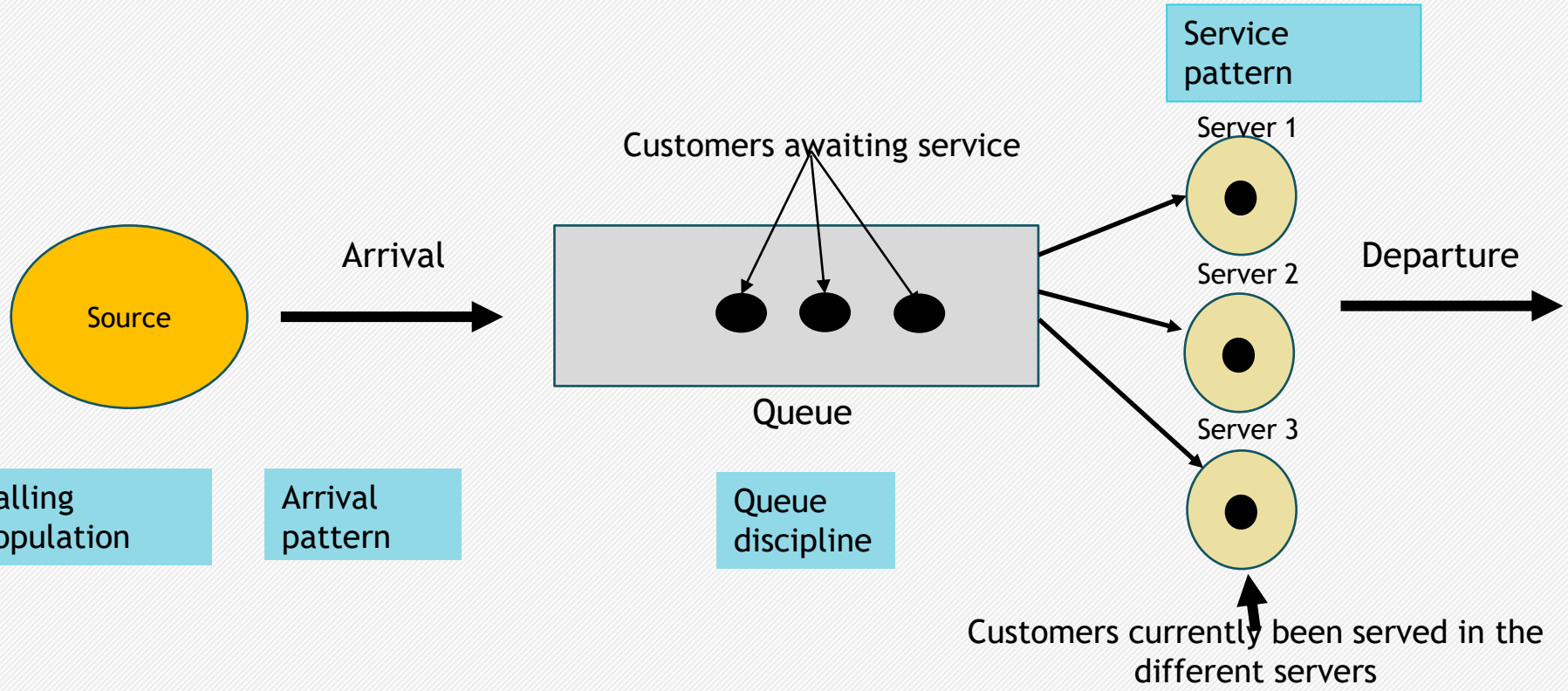
The decision of balking is deterministic, whereas those of reneging and jockeying are considered as probabilistic

Queuing Models – Single server Queuing Model



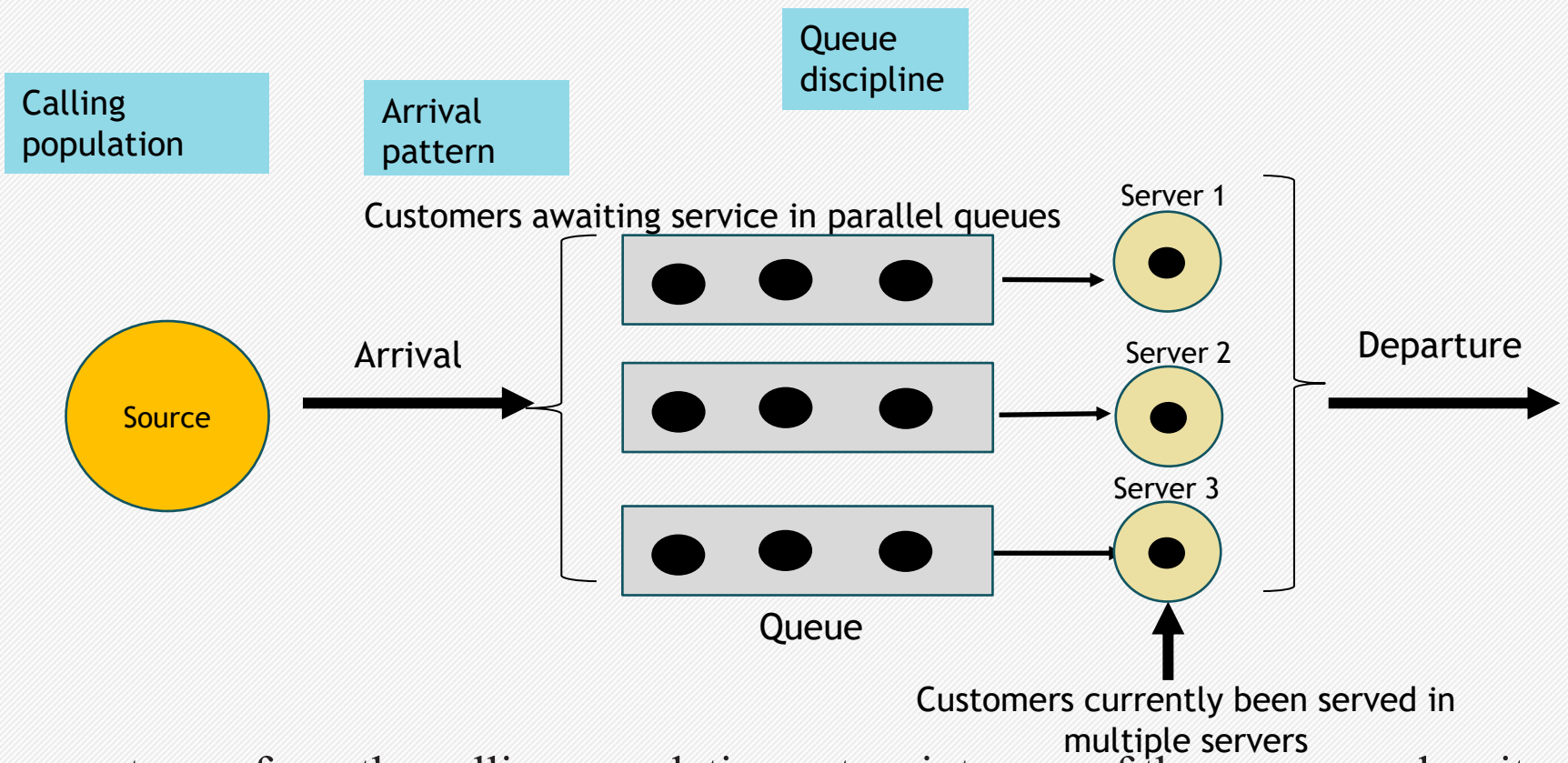
- A new customer from the calling population enters into the queuing model and waits for service in the **queue**.
- If the **queue** is empty and the **server** is idle, the new customer is immediately sent to the **server** for service,
- Otherwise, the customer remains in the **queue** joining the waiting line until the queue is empty and the **server** becomes **idle**.
- When a customer enters into the **server**, the status of the server becomes **busy**, not allowing any more arrivals to gain access to the server.
- After being served, a customer exits the system

Queuing Models - Single queue – multi-server model



- A new customer from the calling population enters into the queuing model and waits for service in the **queue**.
- If the **queue** is empty and any of the **servers** is idle, the new customer is immediately sent to a **server** for service
- Otherwise, the customer remains in the **queue** joining the waiting line until the queue is empty and the **server** becomes **idle**.
- When a customer enters into any of the **server**, that server's status becomes **busy**, not allowing any more arrivals to gain access to the server.
- After being served, a customer exits the system

Queuing Models - parallel queues – multi-server model



- A new customer from the calling population enters into any of the queues and waits for service in the **queue**.
- If the **queue** is empty and any of the **servers** is idle, the new customer is immediately sent to a **server** for service.
- Otherwise, the customer remains in the **queue** joining the waiting line until the queue is empty and the **server** becomes **idle**.
- When a customer enters into the **server**, that server's status becomes **busy**, not allowing any more arrivals to gain access to the server.
- After being served, a customer exits the system. NB: Customers can Jokey in this model

Notations – Kendal's Notation

A/B/c/N/K, is used to concisely define a queue and its parameters:

- A and B represent the inter - arrival and service distribution, respectively;
- c represents the number of servers.
- N represents the queue capacity;
- K represents the size of the calling population.

- D (deterministic),
- M (Poisson),
- G (general),
- and Ek (Erlang) are used to represent A and B .
- Usually the A/B/c notation is used when N and K are infinite.
- For example, M/M/1 represents a single server queuing model, and the inter - arrival and service time are exponentially distributed.
- The *queue discipline* is often added to describe the system.

Notations –Notations for Queuing Model Statistics

Notation	Description
ar_i	Arrival time for customer i
a_i	Inter-arrival time for customer i
$\bar{a}$	Average inter-arrival time
Λ	Arrival rate
T	Total simulation time
n	Number of arrived customers
s_i	Service time of ith customer
μ	Service rate
ss_i	Service start time of ith customer
d_i	Departure time of ith customer
$\overline{de}$	Mean delay time
$\bar{w}$	Mean residence time
ρ	Utilization
L	Number of customers in the system
B	System busy time
I	System idle time

Notations –Equations for key Queuing Model Statistics

#	Name	Equation	Description
1	Inter - arrival time	$a_i = ar_i - ar_{i-1}$	Interval between two consecutive arrivals
2	Mean inter - arrival time	$\lambda = \sum a_{i/n}$	Average inter - arrival time
3	Arrival rate	$\lambda = n/T$ or $\lambda = 1/\bar{a}$	The number of arrivals at unit time
4	Mean service time	$\bar{S} = \sum S_i/n$	Average time for each customer to be served
5	Service rate	$\mu = 1/n$	Capability of server at unit time

Notations –Equations for key Queuing Model Statistics

#	Name	Equation	Description
6	Mean delay time	$\bar{d}e = (\sum SS_i - ar_1)/n$	Average time for each customer to spend in a queue
7	Mean residence time	$\bar{w} = \sum (d_i - ar_1)/n$	Average time each customer stays in the system
8	System busy time	$B = \sum S_i$	Total service time of server
9	System idle time	$I = T - B$	Total idle time of server
10	System utilization	$\rho = B/T$	The proportion of the time in which the server is busy

Queuing Rules –Little's Law

- The average number of customers (L) is equal to the arrival rate (λ) multiplied by the average time (w) the customer spends in the system:
- $L = \lambda w$.
- Little ' s law is meaningful in that the law holds regardless of any kind of the arrival and service distribution.
- Thus, Little ' s law does not require restricted assumptions for the types of arrival and service patterns

Queuing Networks

What is a Queuing Network?

Queuing networks are classified into types: *open* and *closed*.

- A *token* denotes any type of customer that requests service at the service facility.
- In an open queuing network, each token arrives at the system, based on the arrival rate, and leaves the system after being served.
- In the closed queuing network, the finite number of tokens is assigned, and each token moves between queues without leaving the system.

Queuing Networks – cont'd

- The main difference between these two types of networks is that the open queuing network has new arrivals during simulation, whereas the closed queuing network does not have new arrivals.
- The number of tokens in the open queuing network at an instant of time is always different due to the arrival and departure rates, but the number of tokens in the system is always constant during the simulation of a closed queuing network.