ETSN10: Network Architecture and Performance - Reference Sheet

Karl Hallsby

Last Edited: January 28, 2020

Contents

List of Theorems

1 Networking Review

This course assumes that you already know about basic networking terms and models. However, this section is meant as a refresher for people who have already taken that particular set of courses, or as a quick introduction for those who have not taken those courses yet.

1.1 Networking Stack

Defn 1 (Networking Stack)**.** The *networking stack* is a set of layers that are built on top of another that are used to implement inter-computer communication. There are 5 layers:

1. Physical Layer
2. Data-Link Layer
3. Network Layer
4. Transport Layer
5. Application Layer

Each of these fulfill different roles, allowing for different implementations to be used interchangeably. Additionally, not all points in the network graph require the entire networking stack. For example, a router or switch does not require the Transport Layer or the Application Layer, since the job of this hardware is to just move packets around.

Each layer of the networking stack add a header and possibly a footer. Information in headers and footers includes source and destination addresses, checksums, packet size, and protocol identifiers.

Defn 2 (Encapsulation)**.** The process of adding headers and footers is called *encapsulation*.

Defn 3 (Decapsulation)**.** The process of removing the headers and footers of a Packet at the other end is called *decapsulation*.

1.2 Physical Layer

Defn 4 (Physical Layer)**.** The *physical layer* in the Networking Stack is the lowest layer in the stack. It consists of the physical connection that is made between computers and the modulation and coding to represent data as a signal on that connection. For obvious reasons, all devices **must** implement this layer.

Some examples of this layer are:

- Copper twisted-pair cables
- Fiber optic wires
- Radio transmission

Remark 4.1. The design of different Physical Layers is more a hardware design question than that of a software implementation issue. So, it is not discussed in this course, beyond the use of cellular networks.

1.3 Data-Link Layer

Defn 5 (Data-Link Layer)**.** The *data-link layer* is the next layer in the Networking Stack. All devices must implement this layer. The data-link layer is responsible for getting data between two devices that have a Physical Layer connection between them. It is also known as the *Medium Access Control (MAC) layer*. It controls when each device should use the link, and may also include error correction and retransmission.

Some examples of these are:

- Ethernet
- Point-to-Point Protocol (PPP)
- 802.11 (WiFi)
- Bluetooth

1.4 Network Layer

Defn 6 (Network Layer)**.** The *network layer* is the third layer in the Networking Stack. This is the last layer that every device must have. It is responsible for end-to-end data delivery between two devices (hosts) anywhere on the network. It is responsible for routing and forwarding.

Some examples of this layer are:

- Internet Protocol (IP)
- Internet Control Message Protocol (ICMP)

- Routing Protocols

Defn 7 (Routing)**.** *Routing* is the process of determining and selecting the best end-to-end path through a network.

There are algorithms designed to discover the other participants in the network by analyzing the grpah that is created when each participant is considered a node in this graph.

Defn 8 (Forwarding)**.** *Forwarding* is the process of selecting the next hop for the given packet.

Defn 9 (Circuit Switching)**.** In *circuit switching*, dedicated resources are allocated end-to-end for a particular flow of data. No other flows may use those resources while the connection is maintained. Meaning, that during a connection, there is a single dedicated circuit between the host and the receiver.

Defn 10 (Packet Switching)**.** In *packet switching*, no resources are allocated to a flow, and each flow's data is broken up into discrete packets. Each Packet is routed and forwarded independently.

Defn 11 (Packet)**.** A connection from one device to another is a stream. However, to effectively implement many things in the Networking Stack, the stream must be broken into several discrete *packets*.

1.4.1 Discovering Routing Information

1.4.1.1 Dijkstra's Algorithm Dijkstra's Algorithm starts at the source and builds up the shortest path step-by-step.

There are 2 main problems with Dijkstra's Algorithm:

1. The weights along the edges of the graph cannot be negative.
   - If this were to occur, then we would have an infinite cycle going through the negative edge.
2. We **must** know the entire network's topology before beginning the algorithm.
   - Typically, this is not possible on vast networks, like the Internet.

Algorithm 1.1: Dijkstra's Algorithm

Input : A graph with weights on each edge representing "distance" between the nodes.
Output: The path through the graph with the least weight from some starting node.

```
Assign tentative distance estimate to each node: 0 for the initial node, ∞ for all other nodes.
Mark all nodes except the initial node as unvisited.
Set the initial node as current node.
for The current node's unvisited neighbours do
    Calculate their distance.
    Compare this new distance against the current distance.
    Take whichever distance is smaller and use that as the new value.
    Mark the current node as visited.
    if Reached target node OR No more unvisited nodes with distance < ∞ then
        Stop and terminate the algorithm.
    Set the unvisited node with the smallest distance as the new current node.
```

1.4.1.2 Bellman-Ford Algorithm The Bellman-Ford algorithm works with graphs that have negative weight values on edges. It also **does not** require us to know the entire network's topology before beginning. However, it is much slower than

Dijkstra's Algorithm.

Algorithm 1.2: Bellman-Ford Algorithm

Input : A graph with weights on each edge representing "distance" between the nodes.
Output: The path through the graph with the least weight from some starting node.

```
Set the distance for the source node to 0 and for all other nodes to ∞.
Set the predecessor of all nodes to null.
for n ∈ N where N is the number of nodes AND n ≤ N − 1 do
    We repeat N − 1 times, because N − 1 is the maximum length of a non-cyclic path.
    for Each edge (u, v) do
        if distance to u, plus the edge weight < current distance to v then
            We have discovered a shorter path to v.
            Set the distance to v to this new value and the predecessor of v to u.
for each edge (u, v) do
    if distance to u plus the edge weight < the distance to v then
        The graph contains a negative-weight cycle.
        Terminate.
    We have found the shortest paths to each node from the source.
    Terminate.
```

1.4.2 Routing Information

Defn 12 (Routing Protocol)**.** A *routing protocol* is used to construct a Routing Table in each node.

Defn 13 (Routing Table)**.** The *routing table* is a table that contains entries on the shortest distances through the current network graph. This is used to help determine where to next forward packets onto.

Each node has a routing table that contains the cost to the destination and the next hop for each destination. For a valid routing table, we need to have:

1. Which destination the packets are going to?
2. What is the next hop after the destination?
3. The "metric" or weight of that particular path through the network graph?
    - We need this term so we can update the routing table as we discover new routes.
    - This way if a new route with a lower cost presents itself, we can replace the higher-cost one with the lower one.

There are 2 ways to collect the routing information necessary to properly route packets through a network. These 2 routing tables are:

1. Static Routing Tables
2. Dynamic Routing Tables

1.4.2.1 Static Routing Tables

Defn 14 (Static Routing Table)**.** *Static routing table*s are manually configured by a user or a program before the system begins routing the information.

This means that there is no overhead to attempt to figure out the network's topology. The information needed has already been given to the device's Routing Table, so that step is completed.

1.4.2.2 Dynamic Routing Tables

Defn 15 (Dynamic Routing Table)**.** A *dynamic routing table* is a Routing Table that is built automatically by the device. To do this, a Routing Protocol is used to build the table while the device is running. This is done using either:

1. Distance Vector Routing Protocol
2. Link State Routing Protocol

Defn 16 (Distance Vector Routing Protocol)**.** The *distance vector routing algorithm* is a Routing Protocol that uses Bellman-Ford Algorithm to construct a Routing Table.

Each node only begins with knowledge of their immediate neighbours and the costs to reach them.

- Nodes then send this information (the routing table) to their neighbours.

- If a neighbour sends us a route that is shorter than one we already have, update our table to reflect this.
- After updating, send the new table to our neighbours.
- If a node goes down, discard any lines in the routing table that have it as the next hop and follow the above to find a new route.

Defn 17 (Link State Routing Protocol)**.** The *link state routing protocol* is a Routing Protocol that uses Dijkstra's Algorithm to construct a Routing Table. Thus, we need to know what the whole network looks like.

- Each node floods the network with the list of nodes it can connect to and the costs to them.
- Every node builds up a picture of the entire network, then can use Dijkstra's Algorithm to determine the shortest path to each destination.
- The Routing Table is then constructed based on the computed shortest paths.

However, the problem with both Distance Vector Routing Protocol and Link State Routing Protocol is that they do not scale well. However, to help with this, both protocols are contained within a single Autonomous System.

Defn 18 (Autonomous System)**.** An *autonomous system* is a smaller network, like a business or home, that performs a Routing Protocol upon itself. It is done this way because both the Distance Vector Routing Protocol and Link State Routing Protocol do not scale to Internet-sized networks well.

To combat this, each of the autonomous systems has a Speaker Node that speaks to the outside world. This speaker node is used in the Path Vector Routing Protocol.

Defn 19 (Speaker Node)**.** A *speaker node* is a specially designated node within a single Autonomous System that communicates **only with other speaker nodes**. Then, the Path Vector Routing Protocol is performed upon all the speaker nodes to construct a wider network graph consisting only of the Autonomous Systems that the speaker nodes belong to.

Defn 20 (Path Vector Routing Protocol)**.** Between Autonomous Systems, we use a variant of Distance Vector Routing Protocol called *path vector routing protocol*. Each Autonomous System has its own Speaker Node. Only the Speaker Nodes can communicate across the Autonomous System boundary, and exchange information about which destinations they can reach and the paths to them.

1.4.3 IP Addresses

Defn 21 (IP Address)**.** *IP Address*es are the unique end-point routing identifiers used on Packets to deliver their data. A natural analogy is that of apartment numbers on apartment buildings.

IPv4 uses 32 bits, split up into 4 8-bit chunks. Each chunk is read as its decimal equivalent, for humans. IPv6 uses 128 bits, where every 16 bits are interpreted as a hexadecimal number.

Defn 22 (Subnet)**.** All hosts that are in the subnet will have IP Addresses that match the number of bits that are present in the subnet. Keeping with the apartment analogy, the subnet is like the address of the apartment building.

For example, in `192.168.2.0/24`, the `24` means the first 24 bits of the subnet are 1's (`255.255.255.0`). So, the first 24 bits, or 3 IP Address blocks (`192.168.2`), will remain the same for every client in that subnet.

Defn 23 (Classless Inter-Domain Routing)**.** *Classless Inter-Domain Routing* or *CIDR* (pronounced "cider") is a hierarchical way to organize IP Addresses.

This allowed:

- Subnets to be of any length.
- Destinations in a router's routing and forwarding tables may be full IP Addresses or Subnets.
  - A destination IP Address will then be matched to the most specific destination in the table when making forwarding decisions.

1.5 Transport Layer

Defn 24 (Transport Layer)**.** The *transport layer* is built where the Network Layer has delivered all the data to the end host.

This means that only the source and destination hosts have this layer. So, routers and switches do not have this, but your phone, laptop, and the server you're connecting to do.

This layer may be responsible for:

- Connection-Oriented Communication
- Multiplexing Data Flows
- Reliable data delivery
- Data flow control
- Data congestion control

Some implementation of this layer do not handle all of these functions, but all **must** handle the multiplexing of different data flows.

There are 2 main transport layers in-use today:

1. Transmission Control Protocol
2. User Datagram Protocol

Defn 25 (Connection-Oriented Communication)**.** A *connection-oriented communication* system means that a connection must be established before any data is transferred. In addition, this connection must be maintained throughout the transmission of data.

Remark 25.1 (Connectionless Communication)*.* In a *connectionless communication* system, hosts can send data at any time without prior connection being made.

1.5.1 Transport Protocols

Defn 26 (User Datagram Protocol)**.** *User Datagraph Protocol (UDP)* is the simplest transport protocol available. It performs multiplexing and error checking, **but nothing else**.

When a host wants to send data to another host, it just sends it, making UDP a Connectionless Communication protocol. If the data gets lost, too bad.

Defn 27 (Transmission Control Protocol)**.** *Transmission Control Protocol (TCP)* is a Connection-Oriented Communication protocol. It performs:

- Multiplexing
- Reliable data delivery
- Error detection
- Ordered data delivery
- Flow control
- Congestion control

Transmission Control Protocol uses a 3-way hand shake to establish a connection.

Defn 28 (Three-Way Handshake)**.** Transmission Control Protocol uses a *Three-Way Handshake* to establish a connection.

The client sends a `SYN` message to the recipient. The receiver sends back a `SYN_ACK` message to acknowledge the receipt of the original `SYN` message. The client then sends another `ACK` to the receiver to acknowledge the receipt of the `SYN_ACK` message.

Because this handshake is asymmetrical, there is a difference between servers and clients. A server must have a Port open, and be listening for client connections.

Defn 29 (Four-Way Handshake)**.** Transmission Control Protocol uses a *four-way handshake* to close a connection. Each host can close its side of the connection independently.

1.5.2 Multiplexing Data Flows

Defn 30 (Port)**.** A *port* is a single instance of a data flow. There are many different flows of data. These may be from different applications or different instances of the same application. In both Transmission Control Protocol and User Datagram Protocol, flows are given unique *port numbers*.

Some of these are standard for particular applications, e.g. port 80 for HTTP (web), port 25 for SMTP (email). The transport protocol uses the port number to deliver data to the correct application.

1.5.3 Data Delivery

1.5.3.1 Data Delivery in Transmission Control Protocol

Defn 31 (Sequence Number)**.** Transmission Control Protocol uses *sequence number*s to make sure data is complete and in order when it is delivered. It also includes error detection, and segments with errors are retransmitted.

1.5.4 Flow Control

Defn 32 (Flow Control)**.** *Flow control* refers to signalling between the sender and receiver to ensure the sender does not send data faster than the receiver can process.

A receiving host may have limited buffer space for incoming messages and takes time to process each message.

1.5.4.1 Flow Control in Transmission Control Protocol

Defn 33 (Sliding Window)**.** Flow Control in Transmission Control Protocol uses a *sliding window* mechanism.

1.5.5 Congestion Control

Defn 34 (Congestion Control)**.** *Congestion control* refers to mechanisms for detecting and reducing Congestion.

Defn 35 (Congestion)**.** *Congestion* in a network occurs when there is too much data being sent and the network is unable to deliver it all. This can result in data loss or long delays.

1.5.5.1 Congestion Control in Transmission Control Protocol

Defn 36 (Congestion Window)**.** In Transmission Control Protocol, lost data is considered a sign of Congestion and the sender should reduce its rate. The sender has a *congestion window*, which refers to the maximum number of unacknowledged segments that may be in transit at a time.

A Transmission Control Protocol connection begins in **slow start**, where the Congestion Window is doubled every round trip. When a threshold is reached, it changes to *congestion avoidance*, where the congestion window increases by 1 maximum segment size each time. When packet loss is detected, the congestion window is halved.

$$\mathrm{BW}_{Max} = \frac{\mathrm{MSS} \times \sqrt{\frac{3}{2}}}{\mathrm{RTT} \times \sqrt{p}} \tag{1.1}$$

- BW_{Max}: Maximum bandwidth.
- MSS: Maximum Transmission Control Protocol Packet size.
- RTT: Round trip time.
- p: Packet loss probability.

1.6 Application Layer

Defn 37 (Application Layer)**.** The *application layer* is the last and highest layer in the Networking Stack. This layer is created by the actual applications that interface with the user, their email client, web browser, games, etc. This is the layer where data is actually generated and consumed by any application that communicates over the Internet.

Some examples of item in this layer are:

- Hypertext Transfer Protocol (HTTP)
- Simple Mail Transfer Protocol (SMTP)
- Extensible Messaging and Presence Protocol (XMPP)
- Skype

2 Probability Review

This section is meant to quick review and introduce the equations that will be used throughout this course. It is not meant to be comprehensive and/or in-depth. For more information about the topic of probability and statistics, refer to the Math 374 - Probability and Statistics document.

2.1 Axioms of Probability

Defn 38 (Sample Space)**.** The *sample space* is the set of all possible outcomes in a random experiment. It is denoted with the capital Greek omega.

$$\Omega \tag{2.1}$$

Defn 39 (Event)**.** An *event* is a subset of the Sample Space that we are interested in. These are generally denoted with capital letters.

$$A \subseteq \Omega \tag{2.2}$$

Defn 40 (Mutually Exclusive)**.** Any two Events are *mutually exclusive* if the equation below holds.

$$\mathrm{P}(A \cup B) = \mathrm{P}(A) + \mathrm{P}(B) \tag{2.3}$$

Laws that follow from the above definitions (Definitions 38 to 40).

1. The conjugate of the Event occurring, i.e. the Event **not** occurring is:

$$\mathrm{P}(\bar{A}) = 1 - \mathrm{P}(A) \tag{2.4}$$

2. The probability of the union of 2 Events is:

$$\mathrm{P}(A \cap B) = \mathrm{P}(A) + \mathrm{P}(B) - \mathrm{P}(A \cup B) \tag{2.5}$$

   - If A and B are Mutually Exclusive, then $\mathrm{P}(A \cup B) = 0$.

2.2 Conditional Probability

Defn 41 (Conditional Probability)**.** *Conditional probability* is the probability of an Event occurring when it is known that another Event occurred.

$$\mathrm{P}(A \,|\, B) = \frac{\mathrm{P}(A \cap B)}{\mathrm{P}(B)} \tag{2.6}$$

Defn 42 (Independent)**.** Events are *independent* if the probability of the events' intersection is the same as their probabilities multipled together.

$$\mathrm{P}(A \cap B) = \mathrm{P}(A)\,\mathrm{P}(B) \tag{2.7}$$

Remark 42.1 (Conditional Probability and Independent Events)*.* If A and B are Events and are Independent, then

$$\begin{aligned} \mathrm{P}(A \,|\, B) &= \mathrm{P}(A) \\ \mathrm{P}(B \,|\, A) &= \mathrm{P}(B) \end{aligned} \tag{2.8}$$

2.3 Random Variables

Defn 43 (Random Variable)**.** A *random variable* is a mapping from an Event's outcome to a real number.

There are 2 types of random variables, based on what the mapping ends up with:

1. Discrete Random Variables are mapped to integers, $\mathbb{Z}$.
2. Continuous Random Variables are mapped to the real numbers, $\mathbb{R}$.

2.3.1 Discrete Random Variables

Defn 44 (Discrete Random Variable)**.** A *Discrete Random Variable* is one whose values are mapped from an Event's outcome to the integer numbers ($\mathbb{Z}$). These Random Variables are drawn from outcomes that are finite (sides on a die) or countably infinite.

The probability of a single value of the discrete random variable is denoted differently here than in the course material. The subscript refers to which discrete random variable we are working with (in this case X) and the variable in parentheses is the value we are calculating for (in this case $x \in X$).

$$p_X(x) \tag{2.9}$$

The sum of all probabilities for values that the discrete random variable can take **must** sum to 1.

$$\sum_{x \in X} p_X(x) = 1 \tag{2.10}$$

The mean or expected value of a discrete random variables is shown below:

$$\begin{aligned} \mu &= \sum_{x \in X} x p_X(x) \\ \mathbb{E}[X] &= \sum_{x \in X} x p_X(x) \end{aligned} \tag{2.11}$$

The variance of a discrete random variable is how "off" a value from the random variable is from the mean/expected value.

$$\begin{aligned} \sigma^2 &= \sum_{x \in X} (x - \mu)^2 p_X(x) \\ \mathrm{VAR}[X] &= \sum_{x \in X} \big(x - \mathbb{E}[X]\big)^2 p_X(x) \end{aligned} \tag{2.12}$$

The standard deviation is the square root of the variance.

$$\begin{aligned}\sigma = \sqrt{\sigma^2} &= \sqrt{\sum_{x\in X}(x-\mu)^2 p_X(x)} \\ \mathrm{STD}[X] = \sqrt{\mathrm{VAR}[X]} &= \sqrt{\sum_{x\in X}\big(x-\mathbb{E}[X]\big)^2 p_X(x)}\end{aligned} \tag{2.13}$$

There are 5 different Discrete Random Variable distributions that we will be heavily utilizing in this course.

2.3.1.1 Uniform Random Variable

Defn 45 (Uniform Random Variable)**.** The *uniform random variable* is a Discrete Random Variable whose probabilities for each outcome is equal.

For a Discrete Random Variable X, which has $|X|$ possible values,

$$p_X(x) = \frac{1}{|X|} \tag{2.14}$$

Example 2.1: Uniform Random Variable. Lecture 1

For example, the roll of a die is typically modelled as a uniform random variable. Find the probability distribution function, the expected value, and the variance.

Let's assume this is a 6-sided die. And let's map each side's number to a value in the range of $X \in [1, 6]$.
Using Equation (2.14), we can find the probability distribution easily.

$$p_X(x) = \begin{cases} \frac{1}{6} & x = 1 \\ \frac{1}{6} & x = 2 \\ \frac{1}{6} & x = 3 \\ \frac{1}{6} & x = 4 \\ \frac{1}{6} & x = 5 \\ \frac{1}{6} & x = 6 \end{cases}$$

Using Equation (2.11), we can find the the expected value/mean.

$$\begin{aligned}\mu = \mathbb{E}[X] &= \sum_{x=1}^{6} x p_X(x) \\ &= \frac{1}{6} + \frac{2}{6} + \frac{3}{6} + \frac{4}{6} + \frac{5}{6} + \frac{6}{6} \\ &= \frac{1+2+3+4+5+6}{6} \\ &= \frac{21}{6} = 3.5\end{aligned}$$

Using Equation (2.12), we can find the variance.

$$\begin{aligned}\sigma^2 = \mathrm{VAR}[X] &= \sum_{x=1}^{6}\big(x-\mathbb{E}[X]\big)^2 p_X(x) \\ &= \sum_{x=1}^{6}(x-3.5)^2\left(\frac{1}{6}\right) \\ &= 2.91667\end{aligned}$$

Using Equation (2.13), we can find the standard deviation.

$$\begin{aligned}\sigma = \text{STD}[X] &= \sqrt{\sum_{x=1}^{6} \big(x - \mathbb{E}[X]\big)^2 p_X(x)} \\ &= \sqrt{\sum_{x=1}^{6} (x - 3.5)^2 \left(\frac{1}{6}\right)} \\ &= \sqrt{2.91667} \\ &= 1.70783\end{aligned}$$

2.3.1.2 Bernoulli Random Variable

Defn 46 (Bernoulli Random Variable)**.** The *Bernoulli random variable* is one where **only one** test occurs, and there are only 2 outcomes.

The probability of success is denoted

$$p_X(\text{success}) = p \tag{2.15}$$

The probability of failure is denoted

$$p_X(\text{failure}) = 1 - p \tag{2.16}$$

The mean/expected value is:

$$\mu = \mathbb{E}[X] = p \tag{2.17}$$

The variance is:

$$\sigma^2 = \text{VAR}[X] = (1 - p)p \tag{2.18}$$

2.3.1.3 Binomial Random Variable

Defn 47 (Binomial Random Variable)**.** The *binomial random variable* is one where n trials are run with no stops for a success, where the Random Variable in each run is a Bernoulli Random Variable.

The probability of k successes with n trials is

$$\binom{n}{k} p^k (1 - p)^{n-k} = \frac{n!}{k!(n-k)!} (1 - p)^{n-k} \tag{2.19}$$

The mean/expected value after n trials is

$$\mu = \mathbb{E}[X] = np \tag{2.20}$$

The variance after n trials is

$$\sigma^2 = \text{VAR}[X] = np(1 - p) \tag{2.21}$$

2.3.1.4 Geometric Random Variable

Defn 48 (Geometric Random Variable)**.** The *geometric random variable* is one where n trials are run, where the nth trial is a success, meaning there are $n - 1$ previous failures. The Random Variable in each run is a Bernoulli Random Variable.

This means **each trial** has a probability of success of

$$p_X(\text{success}) = p \tag{2.22}$$

And **each trial** has a probability of failure of

$$p_X(\text{failure}) = 1 - p \tag{2.23}$$

The mean/expected value is

$$\mu = \mathbb{E}[X] = \frac{1}{p} \tag{2.24}$$

The variance is

$$\sigma^2 = \text{VAR}[X] = \frac{1 - p}{p^2} \tag{2.25}$$

2.3.1.5 Poisson Random Variable

Defn 49 (Poisson Random Variable)**.** The *Poisson random variable* is used to model the number of independent events that occur over a given period of time.

The Poisson random variable has one parameter,

$$\lambda \tag{2.26}$$

λ is the average number of events per unit of time.

The probability function for the value of $x \in X$ of this random variable is

$$p_X(x) = e^{-\lambda}\left(\frac{\lambda^x}{k!}\right) \tag{2.27}$$

The mean/expected value is

$$\mu = \mathbb{E}[X] = \lambda \tag{2.28}$$

The variance is

$$\sigma^2 = \text{VAR}[X] = \lambda \tag{2.29}$$

2.3.2 Continuous Random Variables

Defn 50 (Continuous Random Variable)**.** A *Continuous Random Variable* is one whose values are mapped from an Event's outcome to the real numbers ($\mathbb{R}$).

Continuous random variables cannot be calculated at single points, but must be integrated over a range. This is called the Cumulative Distribution Function (CDF). The subscript refers to the random variable we are using, and x is the value being calculated for.

$$F_X(x) = \text{P}(X \leq x) \tag{2.30}$$

Continuous random variables have a Probability Density Function (PDF), which is the derivative of the CDF.

$$f_X(x) = \frac{d}{dx}F_X(x) \tag{2.31}$$

This PDF must integrate to 1

$$\int\limits_{x\in X} f_X(x) = 1 \tag{2.32}$$

The expected value/mean is

$$\begin{aligned} \mu &= \int\limits_{x\in X} x f_X(x)\, dx \\ \mathbb{E}[X] &= \int\limits_{x\in X} x f_X(x)\, dx \end{aligned} \tag{2.33}$$

The variance is

$$\begin{aligned} \sigma^2 &= \int\limits_{x\in X} (x-\mu)^2 f_X(x)\, dx \\ \text{VAR}[X] &= \int\limits_{x\in X} \big(x-\mathbb{E}[X]\big)^2 f_X(x)\, dx \end{aligned} \tag{2.34}$$

2.3.2.1 Negative Exponential Random Variable

Defn 51 (Negative Exponential Random Variable)**.** The *negative exponential random variable*, sometimes shortened to exponential random variable, is a Continuous Random Variable that has 1 parameter μ, the rate of decay.

The negative exponential random variable has a PDF of

$$f_X(x) = \mu e^{-\mu x} \tag{2.35}$$

The negative exponential random variable has a CDF of

$$F_X(x) = 1 - e^{-\mu x} \tag{2.36}$$

Its mean/expected value is

$$\begin{aligned} \mu &= \frac{1}{\mu} \\ \mathbb{E}[X] &= \frac{1}{\mu} \end{aligned} \tag{2.37}$$

Its variance is

$$\begin{aligned} \sigma^2 &= \left(\frac{1}{\mu}\right)^2 \\ \mathrm{VAR}[X] &= \left(\frac{1}{\mu}\right)^2 \end{aligned} \tag{2.38}$$

Its standard deviation is

$$\begin{aligned} \sigma &= \sqrt{\sigma^2} = \frac{1}{\mu} \\ \mathrm{STD}[X] &= \sqrt{\mathrm{VAR}[X]} = \frac{1}{\mu} \end{aligned} \tag{2.39}$$

2.3.2.2 Gaussian Random Variable

Defn 52 (Gaussian Random Variable)**.** The *Gaussian random variable*, sometimes called the normal random variable, has 2 parameters, μ and σ.

The Probability Density Function (PDF) is

$$f_X(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x-\mu^2}{2\sigma^2}} \tag{2.40}$$

The mean/expected value is

$$\begin{aligned} \mu &= \mu \\ \mathbb{E}[X] &= \mu \end{aligned} \tag{2.41}$$

The variance is

$$\begin{aligned} \sigma^2 &= \sigma^2 \\ \mathrm{VAR}[X] &= \sigma^2 \end{aligned} \tag{2.42}$$

The standard deviation is

$$\begin{aligned} \sigma &= \sqrt{\sigma^2} = \sigma \\ \mathrm{STD}[X] &= \sqrt{\mathrm{VAR}[X]} = \sigma \end{aligned} \tag{2.43}$$

2.4 Multiple Random Variables

Defn 53 (Joint Probability Function)**.** If X, Y are Discrete Random Variables, the probability of the joint event occurring is

$$p_{X,Y}(x, y) \tag{2.44}$$

where the subscripted X, Y refers to the Discrete Random Variables in use and x, y refers to the values being calculated for.

$$\mathrm{P}(x, y) = \mathrm{P}(X = x, Y = y) \tag{2.45}$$

Defn 54 (Joint Cumulative Distribution Function)**.** If X, Y are Continuous Random Variables, then their *joint cumulative distribution function* is

$$F_{X,Y}(x, y) = \mathrm{P}(X \leq x, Y \leq y) \tag{2.46}$$

Remark 54.1 (Independent Multiple Random Variables)*.* X, Y are independent if and only if

$$F_{X,Y}(x, y) = F_X(x) F_Y(y) \tag{2.47}$$

Defn 55 (Joint Probability Density Function)**.** If X, Y are Continuous Random Variables, then their *joint probability density function* is

$$f_{X,Y}(x, y) = \frac{\partial^2}{\partial x \partial y} F_{X,Y}(x, y) \tag{2.48}$$

2.5 Properties of Expected Value

(i) Expected Value obeys the laws of linearity.

$$\mathbb{E}[aX + b] = a\,\mathbb{E}[X] + b \tag{2.49}$$

(ii) For any Random Variables X, Y

$$\mathbb{E}[X + Y] = \mathbb{E}[X] + \mathbb{E}[Y] \tag{2.50}$$

(iii) For any Independent Multiple Random Variables X, Y,

$$\mathbb{E}[XY] = \mathbb{E}[X]\,\mathbb{E}[Y] \tag{2.51}$$

2.6 Properties of Variance

(i) Variance does not obey the laws of linearity.

$$\mathrm{VAR}[aX + b] = a^2\,\mathrm{VAR}[X] \tag{2.52}$$

(ii) For any Independent Multiple Random Variables X, Y,

$$\mathrm{VAR}[X + Y] = \mathrm{VAR}[X] + \mathrm{VAR}[Y] \tag{2.53}$$

2.7 Covariance

Defn 56 (Covariance)**.** The *covariance* of two Random Variables is defined as

$$\begin{aligned}\mathrm{Cov}[X, Y] &= \mathbb{E}\Big[\big(X - \mathbb{E}[X]\big)\big(Y - \mathbb{E}[Y]\big)\Big] \\ &= \mathbb{E}[XY] - \mathbb{E}[X]\,\mathbb{E}[Y]\end{aligned} \tag{2.54}$$

Remark 56.1 (Relationship Between Covariance and Variance)*.* Variance is actually a case of Covariance of a Random Variable with itself.

$$\mathrm{VAR}[X] = \mathrm{Cov}[X, X] \tag{2.55}$$

The Covariance of two Random Variables can have 3 possible values:

1. Positive: If one Random Variable increases, the other does too.
2. Negative: If one Random Variable increases, the other decreases.
3. Zero: If one Random Variable increases, the other does nothing.

2.8 Correlation Coefficient

Defn 57 (Correlation)**.** The *correlation of X and Y* is defined as the $1, 1$ moment.

$$\mathbb{E}\left[X^1 Y^1\right] \tag{2.56}$$

Defn 58 (Correlation Coefficient)**.** The *correlation coefficient of X and Y* is a measure of the **LINEAR relationship** between X and Y. It does not say anything about nonlinear dependence. It is defined as

$$\begin{aligned}\rho_{X,Y} &= \frac{\mathrm{Cov}[X, Y]}{\sigma_X \sigma_Y} \\ &= \frac{\mathrm{Cov}[X, Y]}{\mathrm{STD}[X]\,\mathrm{STD}[Y]}\end{aligned} \tag{2.57}$$

Remark 58.1. $\rho_{X,Y}$ only ranges from $-1 \leq \rho_{X,Y} \leq 1$.

Remark 58.2. The closer $\rho_{X,Y}$ are to $+1$, the closer X and Y are to having a positive linear relationship. The closer $\rho_{X,Y}$ are to -1, the closer X and Y are to having a negative linear relationship. The closer $\rho_{X,Y}$ are to 0, the closer X and Y are to being *uncorrelated.*

If $\rho_{X,Y} = 0$, then

- If X, Y are Independent Multiple Random Variables, then they are uncorrelated.
- If X, Y are uncorrelated ($\rho_{X,Y} = 0$), then they may still **not** be independent.

2.9 Stochastic Processes

Defn 59 (Stochastic Process)**.** A *stochastic process* or random process $\mathbf{x}(t)$ has 2 meanings:

1. For every time instant, $x(t)$ is a Random Variable.
2. For every point (sample) in an outcome space Ω, $x(t)$ is a real-valued function of time.

There are 4 different possible types:

1. Discrete-Time and discrete-value
2. Discrete-Time and continuous-value
3. Continuous-Time and discrete-value
   - Packet arrival to destination over time.
4. Continuous-Time and continuous-value
   - End-to-end delay in a network.

Defn 60 (Stationary Process)**.** A Stochastic Process $x(t)$ is called (weakly) stationary if:

- $\mathbb{E}[x(t)]$ is a constant, it is independent of t.
- The Correlation or Covariance of the process at 2 points in time $x(t_1)$ and $x(t_2)$, is a function of the difference $t_2 - t_1$ **only**.

2.10 The Poisson Process

Defn 61 (The Poisson Process)**.** This is a continuous-time, discrete-value Stochastic Process. It is also a Stationary Process. This process in **memoryless**, the previous time instant's values do not affect this time instant's value.

This is very commonly used to describe the arrivals into a queueing system or network. There is a parameter λ that is the average rate (packets per second, in this case).

There are 3 different definitions for this process:

1. Behaviour for a very small interval of time.
   - Approximately a Bernoulli Random Variable.
   - The time interval is small enough such that only 1 event occurs with probability $\lambda\Delta t + o(\Delta t)$, which becomes $\lambda\Delta t$ for small Δt.
   - 0 events occur with probability $1 - \lambda\Delta t + o(\Delta t)$, which becomes $1 - \lambda\Delta t$ for small Δt.
   - Probabilities between non-overlapping intervals are independent
2. Behaviour over a longer period of time.
   - Receive multiple events, k.
   - Probability of k events is $p_X(k) = e^{-\lambda}\frac{\lambda^k}{k!}$
3. Behaviour between events.
   - Approximately a Negative Exponential Random Variable.
   - Time t between events, $p(t) = \lambda e^{-\lambda t}$

Poisson Process Definition 1/2 Equivalence. Find the probability of k arrivals in time t. Divide the period t into N small periods of Δt each such that as $N \to \infty$, $\Delta t \to 0$.

Approximate the sum of N independent Bernoulli Random Variables with probability $\frac{\lambda t}{N}$ each.

The probability that k out of N will be 1 and the rest 0 is:

$$\frac{N!}{k!(N-k)!}\left(\lambda\frac{t}{N}\right)^k\left(1-\lambda\frac{t}{N}\right)^{N-k}$$

When N is large, 3 things happen.

1.
$$\frac{N!}{(N-k)!} \approx N^k \tag{2.58}$$

2.
$$\left(1-\lambda\frac{t}{N}\right)^N \approx e^{-\lambda t} \tag{2.59}$$

3.
$$\left(1-\lambda\frac{t}{N}\right)^{-k} \approx 1 \tag{2.60}$$

So, when $N \to \infty$, we obtain

$$\mathrm{P}(k \text{ arrivals in time } t) = e^{-\lambda t}\frac{(\lambda t)^k}{k!}$$

■

Poisson Process Definition 2/3 Equivalence. Since

$$\mathrm{P}(k \text{ arrivals in time } t) = e^{-\lambda t}\frac{(\lambda t)^k}{k!}$$

The probability that time between arrivals is more than t is the same as the probability of 0 arrivals during time $t = e^{-\lambda t}$. So, this is exponentially distributed as required. ■

Poisson Process Definition 3/1 Equivalence. Suppose that the last arrival was time t ago.

A priori probability that the next arrival comes after more than t from the previous arrival is $e^{-\lambda t}$.
A priori probability this next arrival is after more than $t + \Delta t = e^{-\lambda(t+\Delta t)}$.
A priori probability that this next arrival happens within the next $\Delta t = e^{-\lambda t} - e^{-\lambda(t+\Delta t)}$.

Conditioning this on knowing that there were no other arrivals in the last t time, and using a Taylor expansion:

$$\frac{e^{-\lambda t} - e^{-\lambda(t+\Delta t)}}{e^{-\lambda t}} = 1 - e^{-\lambda \Delta t} \qquad \approx \lambda\Delta t - \frac{(\lambda\Delta t)^2}{2} + \frac{(\lambda\Delta t)^3}{6} - \dots$$

Since this result is independent of t, this satisfies the requirement of the independence from previously arrived messages. ■

2.10.1 Sums of the Poisson Processes

The superposition of several different Poisson processes.

- There are m independent Poisson Processes, with rates λ_i for $i = 1, 2, \dots, m$
- Sum is also a Poisson Process: $\lambda = \sum_{i=1}^{m} \lambda_i$

All of this comes together for the equation below. It models the number of things k that have arrived over a certain period of time t.

$$\mathrm{P}(\mathbf{x}(t) = k) = e^{-\lambda t}\frac{(\lambda t)^k}{k!} \tag{2.61}$$

3 Performance Evaluation

We do this to:

- Evaluate existing systems
- Design new network systems
- Predict system behaviors under different conditions

3.1 Performance Measures

How do we measure the performance of a large complex network?

- Data transfer speed
- Reliability:
  - Guaranteed throughput
  - Guarantee of any other performance measurement
  - Integrity of data
  - Predictability of errors
  - Uptime/Downtime/Availability
- Security
- User satisfaction
- Sustainability
- Maintainability
- Throughput/Goodput
- Delay/Latency
- Energy Efficiency
- Jitter (Delay variance)
- Packet Loss

3.2 Performance Evaluation

How can we evaluate the performance of a large complex network?

- Analysis: Mathematical modeling, calculations.
- Simulation: Software implementation of system model.
- Real-World Experimentation: Testing the actual system.

Analysis	Simulation	Experimentation
— Requires detailed understanding of system properties	+ Only requires modeling the environment with a straightforward implementation	++ No modeling or understanding of how the system required
— Usually requires approximations and simplifying assumptions.	+ Possible to implement complex details of system without approximation	++ Captures complete behavior of system and environment without approximation.
++ Allows for deep insight for a broad range of scenarios.	+ Allows insight to broad range of scenarios.	— Requires deployment of every scenarios tested and may be difficult to reproduce.
+ Rare events and boundary cases are included.	+ Study of rare events is tricky, but possible.	— Rare events may be impossible to study.

Table 3.1: Performance Evaluation Pros and Cons

3.3 Statistical Data Analysis

Only analysis produces exact results. Simulation and experimentation produce samples from some underlying random distribution. This means we need to perform statistical analysis of these results.

3.3.1 Sample Mean

We assume a random variable Z with an unknown probability distribution, but we can assume a distribution to start with. We estimate the key distribution metrics:

- Mean (1st moment)
- Variance (2nd moment)
- Variance of the variance (3rd moment)

We obtain n **independent** samples, $z_1, z_2, \ldots, z_n$. To estimate the mean/expected value, we use the equation below.

$$\bar{z} = \frac{1}{n} \sum_{i=1}^{n} z_i \tag{3.1}$$

Where $\bar{z}$ is also a Random Variable. So, we can perform an expected value calculation on $\bar{z}$.

$$\mathbb{E}[\bar{z}] = \mathbb{E}\left[\frac{1}{n} \sum_{i=1}^{n} z_i\right] = \frac{1}{n} \sum_{i=1}^{n} \mathbb{E}[z_i] \tag{3.2}$$

So, as $n \to \infty$, $\mathbb{E}[\bar{z}] \to \mu$.

3.3.2 Sample Variance

Now that we have found the sample mean (Equation (3.1)), we can attempt to find the variance.

We start by estimating the variance:

$$\begin{aligned}
\mathrm{VAR}\big[\mathbb{E}[\bar{z}]\big] &= \mathbb{E}\left[(\bar{z}-\mu)^2\right] \\
(\bar{z}-\mu)^2 &= \frac{1}{n^2}\left(\sum_{i=1}^{n}(z_i-\mu)\right)^2 \\
&= \frac{1}{n^2}\sum_{i=1}^{n}(z_i-\mu)^2 + \sum_{i=1}^{n}\sum_{j\neq i}(z_i-\mu)(z_j-\mu) \\
\mathbb{E}\left[(\bar{z}-\mu)^2\right] &= \frac{1}{n^2}\sum_{i=1}^{n}\mathbb{E}\left[(z_i-\mu)^2\right] + \sum_{i=1}^{n}\sum_{j\neq i}\mathbb{E}[(z_i-\mu)]\,\mathbb{E}[(z_j-\mu)] \\
&= \frac{1}{n^2}\left(n\sigma^2+0\right) \\
&= \frac{\sigma^2}{n}
\end{aligned}$$

The estimated variance is:

$$\mathrm{VAR}[\bar{z}] = \frac{\sigma^2}{n}$$

meaning that the variance of the sample mean $\bar{z}$ around the population mean μ decreases as the number of trials n increases.

The sample variance is the variance of the sample data around its mean.

$$\widehat{V} = \frac{1}{n}\sum_{i=1}^{n}(z_i-\bar{z})^2 \tag{3.3}$$

The expected value of the sample variance, $\mathbb{E}[\widehat{V}]$ is

$$\mathbb{E}[\widehat{V}] = \frac{n-1}{n}\sigma^2$$

This means that the expected value of the variance is actually slightly biased by the $n-1$ numerator. To correct for this, we use Bessel's Correction.

Defn 62 (Bessel's Correction)**.** The division of $n-1$ instead of n in Equation (3.4) is called *Bessel's Correction.*

Defn 63 (Unbiased Sample Variance)**.** By using Definition 62, instead of the normal n value, we find ourselves with the *unbiased sample variance.*

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(z_i-\bar{z})^2 \tag{3.4}$$

3.3.3 Confidence Intervals

Defn 64 (Confidence Interval)**.** *Confidence interval*s are a tool to describe how much we can trust the set of results that we gather. Essentially, how confident we are in the results that we gathered. More mathematically, they describe how sure we are (95%-confident, 99%-confident, etc.) we are that the data point we gathered is the same as the underlying distribution's value.

The confidence interval is derived from the Gaussian Random Variable. Thus, it is only useful if the sample size is large, because of the Central Limit Theorem. It is derived from the below equation:

$$\mathrm{P}\left(|\bar{z}-\mu| \leq \alpha\frac{\sigma}{\sqrt{n}}\right) \tag{3.5}$$

Some common values of this are:

- $= 0.9$ for $\alpha = 1.645$
- $= 0.95$ for $\alpha = 1.96$
- $= 0.99$ for $\alpha = 2.576$

The actual interval is calculated with this equation:

$$\left[\bar{z}-\alpha\frac{\sigma}{\sqrt{n}}, \bar{z}+\alpha\frac{\sigma}{\sqrt{n}}\right] \tag{3.6}$$

However, because the standard deviation σ is not known, we estimate using the square root of the Unbiased Sample Variance $\sqrt{s^2}$ instead. So, the actual confidence interval is:

$$\left[\bar{z} - \alpha\frac{\sqrt{s^2}}{\sqrt{n}}, \bar{z} + \alpha\frac{\sqrt{s^2}}{\sqrt{n}}\right] \tag{3.7}$$

4 Queuing Theory

In queuing theory, we view a network as a collection as first-in-first-out (FIFO) data structures. Queuing theory provides a probabilistic analysis of these queues.

Defn 65 (Queuing System)**.** In a *queuing system* model, there is a FIFO queue with a task arrival rate of λ. The server processes these with a service time of μ.

4.1 Little's Law

Defn 66 (Little's Law)**.** *Little's Law* or *Little's Formula* models the mean number of tasks in a queue-based system. This applies to **any system in equilibrium**, as long as the system does not create or destroy tasks itself.

$$r = \lambda T_r \tag{4.1}$$

- r — The mean number of tasks in a queuing system.
- λ — The average arrival rate of tasks to the system.
- T_r — The mean time for which the task sits in the queue waiting.

Example 4.1: Application of Little's Law. Lecture 3

If 40 customers visit a pub per hour, and these customers spend an average of 15 minutes in the pub, what is the average number of customers in the pub at any given time?

Using Equation (4.1),

$$\lambda = 40$$
$$T_r = \frac{15}{60} = \frac{1}{4} = 0.25$$

Multiplying these together,

$$r = 40 * \frac{1}{4} = 10$$

4.2 Kendall's Notation

Defn 67 (Kendall's Notation)**.** *Kendall's Notation* is a shorthand to specify the characteristics of a Queuing System. There are 6 symbols, where the first 2 are distributions, the next 3 are numbers, and the last is the discipline the server uses to process tasks.

$$x_1/x_2/x_3/x_4/x_5/x_6 \tag{4.2}$$

1. x_1: The arrival distribution (See Section 4.2.1).
2. x_2: The server's service distribution (See Section 4.2.1).
3. x_3: The number of servers.
4. x_4: The total capacity of the system (Assumed to be ∞ if not specified).
    - The assumption of ∞ is not a bad assumption usually.
    - Packets are usually only a couple of kibibytes, whereas main memory is gibibytes.
5. x_5: The population size (the total possible tasks, assumed to be ∞ unless specified).
6. x_6: Service Discipline (FIFO, LIFO, etc.) (FIFO is assumed, unless specified).

Remark 67.1 (Shortened Kendall's Notation). Typically, only the first 3 terms in Kendall's Notation are used. The last 3 have assumed conditions if they are not specified.

4.2.1 Distributions in Kendall's Notation

- M — Exponential, Memoryless, Markovian, Poissonian.
- D — Deterministic, Fixed arrival rate.
- E_k — Erlang with a parameter k.
- H_k — Hyperexponential with a parameter k.
- G — General (The distribution can be anything).

Some examples are:

Example 4.2: Kendall's Notation 1. Lecture 3

What does $M/M/1$ mean in Kendall's Notation?

- Really $M/M/1/\infty/\infty/FIFO$, since last 3 not specified.
- Exponential arrival distribution
- Exponential service distribution
- 1 server
- Infinite capacity
- Infinite population
- FCFS (FIFO)

Example 4.3: Kendall's Notation 2. Lecture 3

What does $M/M/n$ mean in Kendall's Notation?

- Really $M/M/n/\infty/\infty/FIFO$, since last 3 notation specified.
- Exponential arrival distribution
- Exponential service
- n servers
- Infinite capacity
- Infinite population
- FCFS (FIFO)

Example 4.4: Kendall's Notation 3. Lecture 3

What does $G/G/3/20/1500/SPF$ mean in Kendall's Notation?

- General arrival distribution
- General service distribution
- 3 servers
- 17 queue slots $(20 - 3)$
- 1500 total jobs
- Shortest Packet First

4.3 A General $M/M/1$ Queue

If we have a general $M/M/1$ queue, then:

- We may assume messages arrive at the channel according to The Poisson Process at a rate λ messages/sec.
- We may assume that the message lengths have a Negative Exponential Random Variable with a mean of $\frac{1}{v}$ bits/message.
- The channel transmits messages from its buffer at a constant rate c bits/sec.
- The buffer associated with the channel can be considered to be effectively of infinite length.

Defn 68 (Message Transmission Rate)**.** The *message tramisssion rate*, c, is the rate at which tasks are moved from the queue/buffer to the server.

Defn 69 (Message Arrival Rate)**.** The *message arrival rate*, denoted μ is defined to be the number of messages that are received per unit time.

$$\mu = cv \tag{4.3}$$

- c: Message Transmission Rate
- v: Message length

Defn 70 (Occupancy)**.** The *occupancy* of a system, denoted ρ, is the factor to which the rate of message arrivals is greater than the message transmissions after processing.

$$\rho = \frac{\lambda}{\mu} \tag{4.4}$$

- λ: The message arrival rate.
- μ: The message tramisssion rate after processing.

Remark 70.1 (Steady-State Solution). The only steady-state solution for infinite-buffer systems occurs when $\rho < 1$. If $\rho \geq 1$, then packets are arriving at the same or greater rate as they are pushed out after processing. Meaning, eventually the queue will fill up.

If we want to find the average number of messages in the system, we need to find $\mathbb{E}[R]$.

$$\mathbb{E}[R] = \frac{\rho}{1-\rho} \tag{4.5}$$

The average number of packets in **just** the queue is given by $\mathbb{E}[T_W]$.

$$\mathbb{E}[T_W] = \frac{1}{\mu}\frac{\rho}{1-\rho} \tag{4.6}$$

The mean delay of a packet due to the system is given by $\mathbb{E}[T_R]$.

$$\mathbb{E}[T_R] = \frac{1}{\mu}\frac{1}{1-\rho} \tag{4.7}$$

Defn 71 (Probability of Delay)**.** The *probability of delay for an $M/M/1$ queuing system* for a given amount of time is given by

$$p_{T_R}(t) = \mu(1-\rho)e^{-\mu(1-\rho)t} \tag{4.8}$$

The probability the delay is within a given range is given by the equation below.

$$\mathrm{P}(T_R \leq t) = 1 - e^{-\mu(1-\rho)t} \tag{4.9}$$

4.4 State-Dependent Queues

Defn 72 (State-Dependent Queue)**.** A *state-dependent queue* is a queue that behaves in certain ways depending on the state that it is in. It can cycle endlessly **between** states, but can never cycle on itself.

There are 2 variables needed for each state.

1. λ_i: The arrival rate of tasks when the system is in state i.
2. μ_i: The servers' service rate when the system is in state i.

In each state, there needs to be a balance between the packets going to the next stage and leaving the current one and vice versa.

$$\lambda_{i-1}p_{i-1} = \mu_i p_i, \text{ for } i = 1, 2, \ldots \tag{4.10}$$

The general solution to Equation (4.10) is:

$$p_i = \frac{\lambda_0\lambda_1\cdots\lambda_{i-1}}{\mu_1\mu_2\cdots\mu_i}p_0, \text{ for } i = 1, 2, \ldots \tag{4.11}$$

4.4.1 A General $M/M/2$ Queue

A general $M/M/2$ Queuing System can be represented as a State-Dependent Queue.

$$\lambda_i = \lambda, \text{ for } i = 1, 2, \ldots$$
$$\mu_i = \begin{cases} \mu & i = 1 \\ 2\mu & i = 2, 3, \ldots \end{cases}$$

where i is the number of packets in the system, which represents a different state of the system.

Now, let's compare a $M/M/2$ Queuing System against a $M/M/1$ Queuing System.

$$\begin{aligned}
\mathbb{E}[T_R]_{1 \text{ server, } 2\mu \text{ rate}} &= \frac{1}{2\mu}\frac{1}{1-\rho} \\
\mathbb{E}[T_R]_{2 \text{ servers}} &= \frac{1}{\mu}\frac{1}{(1+\rho)(1-\rho)} \\
\frac{\mathbb{E}[T_R]_{1 \text{ server, } 2\mu \text{ rate}}}{\mathbb{E}[T_R]_{2 \text{ servers}}} &= \frac{1+\rho}{2} \leq 2
\end{aligned}$$

The conclusion here is concrete.

A single faster server is always more efficient.

If we extend our findings about State-Dependent Queues to $M/M/\infty$ systems, the general solution becomes a Poisson Random Variable's distribution.

$$\begin{aligned}
p_i &= \frac{\lambda_i}{\mu^i i!} p_0 \\
&= e^{-\frac{\lambda}{\mu}} \frac{\left(\frac{\lambda}{\mu}\right)^i}{i!} \\
&= e^{-A}\frac{A^i}{i!}
\end{aligned} \tag{4.12}$$

4.5 The Finite Buffer $M/M/1/N$ Queue

The $M/M/1/n$ is a queue with a finite-sized buffer. The real question with this is what is the probability the buffer will be full and the next packet will have to be blocked.

Defn 73 (Blocking Probability)**.** The *blocking probability* is the probability that the next packet that arrives to the system will be blocked because the queue is full. Instead of making the queue longer to keep track of the packets, the system just denies the packet access to the queue.

$$P_B = \text{P[Message is blocked]} = p_n = \frac{1-\rho}{1-\rho^{n+1}}\rho^n \tag{4.13}$$

Example 4.5: Length of Queue for No Packets Blocked. Lecture 4

Assume $\rho = 0.5$, and that we want a system that blocks packets with probability $P_B \leq 10^{-3}$. Find n, the minimum length of the queue to satisfy these requirements?

Using Equation (4.13),

$$\begin{aligned}
\frac{1-\rho}{1-\rho^{n+1}}\rho^n &= \frac{1-0.5}{1-0.5^{n+1}}0.5^n \\
&= \frac{0.5^{n+1}}{1-0.5^{n+1}} \leq 10^{-3} \\
0.5^{n+1} &\leq \frac{10^{-3}}{1+10^{-3}} \\
n+1 &\geq 9.96 \\
n &\geq 8.96
\end{aligned}$$

So, n should be 9.

Remark. Note that the solution found in Example 4.5 is actually the general solution. Meaning, that solution for n is valid for all ρ. This is because the finite buffer copes with overloading by blocking any additional messages from entering the system.

4.5.1 Applying Little's Law

To even use Little's Law, we must account for the blocking of messages by only counting the messages that make it into the system.

$$\gamma = \lambda(1 - P_B) \tag{4.14}$$

This means our equations change a little bit.

$$\rho = \frac{\lambda}{\mu}, \text{ can be } \geq 1 \text{ r} < 1 \tag{4.15}$$

$$\begin{aligned} \mathbb{E}[R] &= \gamma\, \mathbb{E}[T_R] \\ &= \lambda(1 - P_B)\, \mathbb{E}[T_R] \end{aligned} \tag{4.16a}$$

$$\begin{aligned} \mathbb{E}[R] &= \sum_{i=0}^{n} i p_i \\ &= \frac{1-\rho}{1-\rho^{n+1}} \sum_{i=0}^{n} i\rho_i \end{aligned} \tag{4.16b}$$

4.6 Modeling Circuit Switching

This is easiest to visualize with the old-school telephone communications with human operators physically connecting calls to make a single large continuous circuit.

- The rate of call initiation can be modeled as a Poisson Random Variable, with an arrival rate of λ.
- The length of the calls can be modeled as a Negative Exponential Random Variable, denoted as h, $\mu = \frac{1}{h}$.
- The number of circuits is fixed at n.
- If all circuits are in use, there is no queue to wait in until one is available.

This makes Circuit Switching a $M/M/n/n$ Queuing System in Kendall's Notation.

Defn 74 (Offered Traffic)**.** *Offered traffic* is the amount of traffic that the end-users are offering the network.

$$A = \lambda h \tag{4.17}$$

Defn 75 (Carried Traffic)**.** *Carried traffic* is the amount of traffic that the network is currently handling.

$$A_C = A(1 - P_B) \tag{4.18}$$

Defn 76 (Lost Traffic)**.** *Lost traffic* is traffic that is lost because all the resources available have been allocated and no more are available. This cannot be regained later because there is no queue available to handle the things that arrive to the system after all resources have been allocated.

$$L = A - A_C \tag{4.19}$$

Remark 76.1 (Validity). Equation (4.19) is valid for all non-negative exponential distributions. However, it is invalidated if repeats occur at all, or if they are allowed to occur, occur to soon.

Let the Random Variable R represent the number of customers currently in the system, meaning R is between 0 and n. The service rate for this is now

$$i\mu, \text{ when } R = i$$

Defn 77 (Blocking Probability)**.** The *blocking probability* is the probability that the next thing that comes to the system will be blocked because all resources have already been allocated and there is no queue.

$$\begin{aligned} P_B &= p_n \\ &= \frac{\frac{A^n}{n!}}{\sum_{i=0}^{n} \frac{A^j}{j!}} \end{aligned} \tag{4.20}$$

Defn 78 (Time Congestion)**.** *Time congestion* represents the proportion of time that all the circuits are busy. This is only viewed from the system-side, so the end-user will never know about time congestion.

To find the time congestion, it is simply the expected value of the Offered Traffic, when in state n.

$$\begin{aligned}\mathbb{E}_n[A] &= P_B \\ &= p_n \\ &= \frac{\frac{A^N}{N!}}{\sum_{j=0}^{n} \frac{A^j}{j!}}\end{aligned} \tag{4.21}$$

Defn 79 (Call Congestion)**.** *Call congestion* is the proportion of calls that find the system busy. This is viewed from the end-user side.

Remark. For arrivals that follow a Poisson Random Variable's distribution, Time Congestion = Call Congestion. This follows according to the PASTA Theorem.

Theorem 4.1 (PASTA Theorem)**.**

4.7 Queuing Networks

Defn 80 (Queuing Network)**.** A *queuing network* is a network of nodes in which each node is a queue. The output of one queue is connected to the input of another node's queue.

There are 2 types of queuing networks:

1. Closed Queuing Network
2. Open Queuing Network

Remark 80.1 (Limiting Analysis). We will only consider $M/M/n$ queues here. Analysis of more general queueing networks gets complicated fast.

Defn 81 (Closed Queuing Network)**.** A *closed queuing network* is one in which packets (customers, tasks, etc.) can never enter or leave the Queuing Network.

Defn 82 (Open Queuing Network)**.** An *open queuing network* is a Queuing Network that is not closed. This means that an open queuing network is one in which packets (customers, tasks, etc.) may enter a node's queue from outside the Queuing Network, and after processing, may leave the Queuing Network entirely.

Theorem 4.2 (Burke's Theorem)**.** *The interdeparture times from a Markovian ($M/M/n$) queue are exponentially distributed with the same parameter as the interarrival times. Meaning*

$$\lambda_{\text{Out}} = \lambda_{\text{In}} \tag{4.22}$$

Remark 82.1. This allows us to analyze each queue/node independently. We do not need to consider all the nodes at the same time.

4.7.1 Jackson Networks

Defn 83 (Jackson Network)**.** A *Jackson network* is an Open Queuing Network of $M/M/n$ queues. Each node can receive traffic from other nodes and from outside the network. Traffic from each node can go to other nodes, or leave the network entirely.

- N: The total number of nodes in a Jackson Network.
- λ_i: Total incoming average traffic rate to node i.
- $gamma_i$: Average rate of traffic entering node i from outside the network
- r_{ij}: Probability a packet leaving node i will then go to node j
  - Note r need not be 0 – a packet may immediately return to the node it just left
- So the probability that a packet leaves the network after leaving node i is given by

$$1 - \sum_{j=1}^{N} r_{ij} \tag{4.23}$$

- You can find the total incoming traffic to node i by summing all incoming streams together.

$$\lambda_i = \gamma_i + \sum_{j=1}^{N} \lambda_i r_{ji}, \text{ for } i = 1, 2, \ldots, N \tag{4.24}$$

- In general, the whole network will not behave like a The Poisson Process.
- Jackson, the namesake of the Jackson Network, showed that each individual node behaves as though it were.
- The state variable for the entire system of N nodes consists of the vector

$$\langle k_1, k_2, \ldots, k_N \rangle \tag{4.25}$$

  where k_i is the number of packets (including those currently in service) at the ith node.
- Let the equilibrium probability associated with a given state be denoted

$$\mathrm{P}\big[\langle k_1, k_2, \ldots, k_N \rangle\big] \tag{4.26}$$

  and the probability of finding k_i customers in the ith node be

$$\mathrm{P}_i[k_i]$$

- The joint probability distribution for all nodes is the product of the distributions for the individual nodes

$$\mathrm{P}\big[\langle k_1, k_2, \ldots, k_N \rangle\big] = \mathrm{P}_1[k_1]\,\mathrm{P}_2[k_2] \cdots \mathrm{P}_N[k_N]$$

  and each $\mathrm{P}_i[k_i]$ is given by the solution to the classical $M/M/n$ system.
- So we can treat each node as an $M/M/n$ queue, even though the input is not necessarily Markovian.

4.8 Queuing Disciplines

Defn 84 (Queuing Discipline)**.** A *queuing discipline* is a way to order the packets to send them in a particular order. Depending on how we order, namely how we schedule the packets, we achieve different things. However, the overarching requirement is to minimize delay, improve throughput, increase busy time, and decrease idle time.

There are 2 types of queuing disciplines:

1. Work Conserving
2. Non-Work Conserving

Defn 85 (Work Conserving)**.** A *work conserving* Queuing Discipline always handles packets right away. Meaning, if there is a packet waiting in the queue and the server is sitting idle, it will always serve the packet.

This is in contrast to a Non-Work Conserving Queuing Discipline.

Defn 86 (Non-Work Conserving)**.** A *non-work conserving* Queuing Discipline may not handle packets right away. Meaning, if there is a packet waiting in the queue and the server is sitting idle, the server might not server the packet right away. Packets are switched only at **the right time towards the right destination**.

While this type of Queuing Discipline may not make intuitive sense, it has some advantages.

- Reduces jitter
- It *shapes* the traffic, making it predictable by holding traffic back until a certain time.

This is contrast to a Work Conserving Queuing Discipline.

Traditionally, First-In First-Out was used. But, there are some other ones possible that improve the Queuing System in various ways.

4.8.1 First-In First-Out

Defn 87 (First-In First-Out)**.** In a *first-in first-out* Queuing Discipline, the first packet that enters the queue is the first one served. This is a Work Conserving Queuing Discipline. This follows Kleinrock's Conservation Law.

$$C = \sum_{n=1}^{N} \rho_n q_n \tag{4.27}$$

- C: A constant value of throughput for the system.
- ρ: The Occupancy for each flow.
- q: The mean scheduler delay.

However, there are some problems:

- Small packets are held up by large packets ahead of them in the queue. This leads to:
  - Larger average delays for smaller packets.
  - Greater throughput for larger packets.
  - Flows of greedy packets get better service.
- Greedy TCP connections crowd out more altruistic connections.
  - If one connection does not back off, others may back off more.
  - This is a feature of the TCP protocol, because this manages network congestion.

4.8.2 Fair Queuing (FQ)

Defn 88 (Fair Queuing)**.** The *fair queuing* Queuing Discipline is a Work Conserving one. There are multiple queues for each port, one for each source or flow of data to that port. The various queues are serviced in a round-robin fashion, where one packet from a queue is handled in a single cycle, before moving onto the next queue. This achieves load balancing among the different flows, meaning there is no advantage for a particular flow to be greedy.

However, the drawback with this is that short packets are penalized as only one packet from one queue is sent per round.

4.8.3 Priority Queuing (PQ)

Defn 89 (Priority Queuing)**.** *Priority queuing* is a Work Conserving Queuing Discipline. There are K queues that lead to the server. The kth queue is constrained by $1 \leq k \leq K$. The $k+1$th queue has a higher priority than the kth queue. The higher priority queue is served first.

This has a simple implementation with low processing overhead. There needs to be an extra step where the packets are sorted into the correct priority queue, but that is quite efficient now.

However, there is no fairness among the packets. The lower priority queues can be starved of service time if the higher queues are always full.

4.8.4 Processor Sharing (PS)

Defn 90 (Processor Sharing)**.** A *processor sharing* Queuing Discipline is Work Conserving. However, it is not practical, because it is modeled on sending individual bits rather than whole packets. It is more of a meta-Queuing Discipline, because it forms the basis for Bit-Round Fair Queuing.

In this discipline, there are multiple queues, just like in Fair Queuing. One bit from each queue is sent per round of the round-robin. This prevents longer packets from getting an advantage.

The system needs to figure out the virtual start and finish times for a given packet, i, of the α queue.

$$F_i^\alpha = S_i^\alpha + P_i^\alpha \tag{4.28}$$

where

- F_i^α: The αth queue's ith packet's finish time.
- S_i^α: The αth queue's ith packet's start time, which is defined in Equation (4.29).
- P_i^α: The αth queue's ith packet's time to send.

$$S_i^\alpha = \max\left(F_{i-1}^\alpha, A_i^\alpha\right) \tag{4.29}$$

4.8.5 Bit-Round Fair Queuing (BRFQ)

Defn 91 (Bit-Round Fair Queuing)**.** The *bit-round fair queuing* Queuing Discipline is based off the Processor Sharing discipline and inherits its Work Conserving property. However, instead of sending individual bits, we send whole packets, but choose which to send based on the same calculations as Processor Sharing Each data flow gets approximately $\frac{1}{N}$ of the overall bandwidth in the system, given there are N flows.

The virtual start and finish times are still computed. However, unlike the Fair Queuing that Processor Sharing is inspired by, the next packet that is sent is the one with the **earliest finish time**.

The same equations are used as in Processor Sharing (Equations (4.28) to (4.29)).

4.8.6 Generalized Processor Sharing (GPS)

Defn 92 (Generalized Processor Sharing)**.** *Generalized processor sharing* is a Work Conserving meta-Queuing Discipline. Like in Processor Sharing, we send some number of bits per round, specified by the weight w_α. If a queue has a weighting of 5, then $w_\alpha = 5$ means 5 bits are sent from the αth queue each round. This solves the problem in Bit-Round Fair Queuing of not being able to handle queues that output different amounts of information during each round. This is used to form the basis of Weighted Fair Queuing.

There are slight generalizations that need to be made to Equations (4.28) to (4.29). They are shown below.

$$F_i^\alpha = S_i^\alpha + \frac{P_i^\alpha}{w_\alpha} \tag{4.30}$$

where

- F_i^α: The αth queue's ith packet's finish time.
- S_i^α: The αth queue's ith packet's start time, defined in Equation (4.31).

- P_i^α: The αth queue's ith packet's time to send.
- w_α: The weight (number of bits sent per round) of the αth queue.

$$S_i^\alpha = \max\left(F_{i-1}^\alpha, A_i^\alpha\right) \tag{4.31}$$

Remark 92.1 (Service Differentiation). The ability to attach different weights to different queues means we now have a way to respond to different service levels. This leads to the concept of *service differentiation*.

4.8.7 Weighted Fair Queuing (WFQ)

Defn 93 (Weighted Fair Queuing)**.** *Weighted fair queuing* is a Work Conserving implementation of Generalized Processor Sharing. It also uses the same packet management strategy as Bit-Round Fair Queuing, of sending the packet with the **earliest virtual finish time**.

By attaching a weight for the number of possible bits to send, we can guarantee a bound on delay. The maximum buffer/queue size needed is proportional to this delay.

Example 4.6: WFQ vs. BRFQ. Lecture 5

Given the packet reception table below, which arrive at the same time on the same link, find the packet transmission order?

Packet	Size	Flow
1	100	1
2	100	1
3	60	2
4	120	2
5	60	2

You may assume that the link can send one unit of each packet per time instance.

4.8.8 Class-Based Queuing (CBQ)

Defn 94 (Class-Based Queuing)**.** *Class-based queuing* is a meta-Queuing Discipline, that is neither Work Conserving or Non-Work Conserving. It is a method of assigning fractions of overall bandwidth to a set of nodes within a class. Nodes that are within the same class may borrow bandwidth if the allocated node is not using it.

4.9 Actual IP Network Behavior

Actual IP traffic doesn't actually follow a Poisson Random Variable's distribution very well. It has:

- Self-similarity
- Lots of dependency (Current traffic depends on the previous traffic).
- Infinite variance.

Defn 95 (Autocorrelation)**.** *Autocorrelation* is a function that measures the similarity of a function with itself, typically over a period of time.

$$R_{x,x}(t_1, t_2) = \mathbb{E}[X(t_1), X(t_2)] \tag{4.32}$$

Defn 96 (Long-Range Dependence)**.** A process that has *long-range dependence* has an autocorrelation that decays slowly as the difference between t_1 and t_2 tends towards ∞. Mathematically, these processes are characterized by an atocorrelation function that decays hyperbolically as k increases. This is represented with Equation (4.33), the *Non-summability of correlation*.

$$\sum_k r(k) = \infty \tag{4.33}$$

Remark 96.1 (Uncorrelated). If the result from Equation (4.32) is 0 when $t_2 = t_1 + \Delta t$, i.e. when the time difference is very small, then the function $X(t)$ is no correlation.

Remark 96.2 (Short-Range Correlation). if the result from Equation (4.32) decays to 0 very quickly, then the function $X(t)$ has *short-range correlation*.

4.9.1 The Hurst Parameter

Defn 97 (Hurst Parameter)**.** The *Hurst parameter*, denoted H, is a measure of the "burstiness" of a system. It is also considered a measure of self-similarity.

H always lies within the range $0.5 < H < 1.0$.

The expected value of the Hurst parameter is

$$\mathbb{E}[x(t)] = \frac{\mathbb{E}[x(at)]}{a^H} \tag{4.34}$$

The variance of the Hurst parameter is

$$\text{VAR}[x(t)] = \frac{\text{VAR}[x(at)]}{a^{2H}} \tag{4.35}$$

The autocorrelation of the Hurst parameter is

$$R_{x,x}(t,s) = \frac{R_{x,x}(at,as)}{a^{2H}} \tag{4.36}$$

4.9.2 Self-Similarity and Autocorrelation

For this section, let $X(t)$ be a Stochastic Process with

- Constant mean μ
- Constant variance σ^2
- An autocorrelation function depending only on k.

$$r(k) = \frac{\mathbb{E}[(X(t)-\mu)(X(t+k)-\mu)]}{\mathbb{E}\left[\left(X(t)-\mu\right)^2\right]}$$

Now let $X^{(m)}$ be a new **aggregate** time series formed by averages of X over non-overlapping blocks in time of size m. This has some intersting implications.

- X is exactly self-similar if
  - The aggregated processes have the same autocorrelation structure as X.
  - $r^{(m)}(k) = r(k), k \geq 0$ for all $m = 1, 2, \ldots$
- X is asymptotically self-similar if the above holds when $r^{(m)}(k) \sim r(k), m \to \infty$
- The most striking feature of self-similarity is correlation structures of the aggregated process do not degenerate as $m \to \infty$
- This is in contrast to traditional models
  - Correlation structures of their aggregated processes degenerate, i.e. $r^{(m)}(k) \to 0$ as $m \to \infty, k = 1, 2, 3, \ldots$

4.9.3 Squared Coefficient of Variation

This is a critical factor in determining the type of Queuing System to use. It is a measure of variability of the system.

$$c^2 = \frac{\sigma^2}{\mu^2} \tag{4.37}$$

The squared coefficient of variation for some common Queuing Systems are shown below.

- $M/M/1 = 1$
- $M/D/1 = 0$
  - Namely, for any system with a deterministic distribution, the squared coefficient of variation will be 0.

A Complex Numbers

Complex numbers are numbers that have both a real part and an imaginary part.

$$z = a \pm bi \tag{A.1}$$

where

$$i = \sqrt{-1} \tag{A.2}$$

Remark (i vs. j for Imaginary Numbers). Complex numbers are generally denoted with either i or j. Since this is an appendix section, I will denote complex numbers with i, to make it more general. However, electrical engineering regularly makes use of j as the imaginary value. This is because alternating current i is already taken, so j is used as the imaginary value instad.

$$Ae^{-ix} = A\left[\cos\left(x\right) + i\sin\left(x\right)\right] \tag{A.3}$$

A.1 Complex Conjugates

If we have a complex number as shown below,

$$z = a \pm bi$$

then, the conjugate is denoted and calculated as shown below.

$$\overline{z} = a \mp bi \tag{A.4}$$

Defn A.1.1 (Complex Conjugate)**.** The conjugate of a complex number is called its *complex conjugate*. The complex conjugate of a complex number is the number with an equal real part and an imaginary part equal in magnitude but opposite in sign.

The complex conjugate can also be denoted with an asterisk ($*$). This is generally done for complex functions, rather than single variables.

$$z^* = \overline{z} \tag{A.5}$$

A.1.1 Complex Conjugates of Exponentials

$$\overline{e^z} = e^{\overline{z}} \tag{A.6}$$

$$\overline{\log(z)} = \log(\overline{z}) \tag{A.7}$$

A.1.2 Complex Conjugates of Sinusoids

Since sinusoids can be represented by complex exponentials, as shown in Appendix B.2, we could calculate their complex conjugate.

$$\begin{aligned} \overline{\cos(x)} &= \cos(x) \\ &= \frac{1}{2}\left(e^{ix} + e^{-ix}\right) \end{aligned} \tag{A.8}$$

$$\begin{aligned} \overline{\sin(x)} &= \sin(x) \\ &= \frac{1}{2i}\left(e^{ix} - e^{-ix}\right) \end{aligned} \tag{A.9}$$

B Trigonometry

B.1 Trigonometric Formulas

$$\sin(\alpha) + \sin(\beta) = 2\sin\left(\frac{\alpha+\beta}{2}\right)\cos\left(\frac{\alpha-\beta}{2}\right) \tag{B.1}$$

$$\cos(\theta)\sin(\theta) = \frac{1}{2}\sin(2\theta) \tag{B.2}$$

B.2 Euler Equivalents of Trigonometric Functions

$$e^{\pm j\alpha} = \cos(\alpha) \pm j\sin(\alpha) \tag{B.3}$$

$$\cos(x) = \frac{e^{jx} + e^{-jx}}{2} \tag{B.4}$$

$$\sin(x) = \frac{e^{jx} - e^{-jx}}{2j} \tag{B.5}$$

$$\sinh(x) = \frac{e^{x} - e^{-x}}{2} \tag{B.6}$$

$$\cosh(x) = \frac{e^{x} + e^{-x}}{2} \tag{B.7}$$

B.3 Angle Sum and Difference Identities

$$\sin(\alpha \pm \beta) = \sin(\alpha)\cos(\beta) \pm \cos(\alpha)\sin(\beta) \tag{B.8}$$

$$\cos(\alpha \pm \beta) = \cos(\alpha)\cos(\beta) \mp \sin(\alpha)\sin(\beta) \tag{B.9}$$

B.4 Double-Angle Formulae

$$\sin(2\alpha) = 2\sin(\alpha)\cos(\alpha) \tag{B.10}$$

$$\cos(2\alpha) = \cos^2(\alpha) - \sin^2(\alpha) \tag{B.11}$$

B.5 Half-Angle Formulae

$$\sin\left(\frac{\alpha}{2}\right) = \sqrt{\frac{1-\cos(\alpha)}{2}} \tag{B.12}$$

$$\cos\left(\frac{\alpha}{2}\right) = \sqrt{\frac{1+\cos(\alpha)}{2}} \tag{B.13}$$

B.6 Exponent Reduction Formulae

$$\sin^2(\alpha) = \frac{1-\cos(2\alpha)}{2} \tag{B.14}$$

$$\cos^2(\alpha) = \frac{1+\cos(2\alpha)}{2} \tag{B.15}$$

B.7 Product-to-Sum Identities

$$2\cos(\alpha)\cos(\beta) = \cos(\alpha-\beta) + \cos(\alpha+\beta) \tag{B.16}$$

$$2\sin(\alpha)\sin(\beta) = \cos(\alpha-\beta) - \cos(\alpha+\beta) \tag{B.17}$$

$$2\sin(\alpha)\cos(\beta) = \sin(\alpha+\beta) + \sin(\alpha-\beta) \tag{B.18}$$

$$2\cos(\alpha)\sin(\beta) = \sin(\alpha+\beta) - \sin(\alpha-\beta) \tag{B.19}$$

B.8 Sum-to-Product Identities

$$\sin(\alpha) \pm \sin(\beta) = 2\sin\left(\frac{\alpha \pm \beta}{2}\right)\cos\left(\frac{\alpha \mp \beta}{2}\right) \tag{B.20}$$

$$\cos(\alpha) + \cos(\beta) = 2\cos\left(\frac{\alpha + \beta}{2}\right)\cos\left(\frac{\alpha - \beta}{2}\right) \tag{B.21}$$

$$\cos(\alpha) - \cos(\beta) = -2\sin\left(\frac{\alpha + \beta}{2}\right)\sin\left(\frac{\alpha - \beta}{2}\right) \tag{B.22}$$

B.9 Pythagorean Theorem for Trig

$$\cos^2(\alpha) + \sin^2(\alpha) = 1^2 \tag{B.23}$$

B.10 Rectangular to Polar

$$a + jb = \sqrt{a^2 + b^2}e^{j\theta} = re^{j\theta} \tag{B.24}$$

$$\theta = \begin{cases} \arctan\left(\frac{b}{a}\right) & a > 0 \\ \pi - \arctan\left(\frac{b}{a}\right) & a < 0 \end{cases} \tag{B.25}$$

B.11 Polar to Rectangular

$$re^{j\theta} = r\cos(\theta) + jr\sin(\theta) \tag{B.26}$$

C Calculus

C.1 Fundamental Theorems of Calculus

Defn C.1.1 (First Fundamental Theorem of Calculus)**.** The *first fundamental theorem of calculus* states that, if f is continuous on the closed interval $[a, b]$ and F is the indefinite integral of f on $[a, b]$, then

$$\int_a^b f(x)\,dx = F(b) - F(a) \tag{C.1}$$

Defn C.1.2 (Second Fundamental Theorem of Calculus)**.** The *second fundamental theorem of calculus* holds for f a continuous function on an open interval I and a any point in I, and states that if F is defined by

$$F(x) = \int_a^x f(t)\,dt,$$

then

$$\begin{aligned} \frac{d}{dx}\int_a^x f(t)\,dt &= f(x) \\ F'(x) &= f(x) \end{aligned} \tag{C.2}$$

Defn C.1.3 (argmax)**.** The arguments to the *argmax* function are to be maximized by using their derivatives. You must take the derivative of the function, find critical points, then determine if that critical point is a global maxima. This is denoted as

$$\underset{x}{\operatorname{argmax}}$$

C.2 Rules of Calculus

C.2.1 Chain Rule

Defn C.2.1 (Chain Rule)**.** The *chain rule* is a way to differentiate a function that has 2 functions multiplied together.

If

$$f(x) = g(x) \cdot h(x)$$

then,

$$\begin{aligned} f'(x) &= g'(x) \cdot h(x) + g(x) \cdot h'(x) \\ \frac{df(x)}{dx} &= \frac{dg(x)}{dx} \cdot g(x) + g(x) \cdot \frac{dh(x)}{dx} \end{aligned} \tag{C.3}$$

D Laplace Transform

Defn D.0.1 (Laplace Transform)**.** The *Laplace transformation* operation is denoted as $\mathcal{L}\{x(t)\}$ and is defined as

$$X(s) = \int_{-\infty}^{\infty} x(t)e^{-st}dt \tag{D.1}$$