Digital Modulation, Synthesis of Digital Signals, and PAM

I. Introduction to Digital Modulation

- **Core Task:** The digital modulator transforms a final stream of bits into an analog signal suitable for transmission over a physical channel.
- **Bits and Symbols:**
  - The modulator groups the incoming bitstream into blocks of k **bits**.
  - Each unique k-bit block is called a **symbol**.
  - The total number of possible symbols is $M = 2^k$.
- **Bit Rate vs. Symbol Rate:**
  - **Symbol Rate (R_s)**: The number of symbols transmitted per second, given by $R_s = 1/T_s$, where T_s is the symbol duration.
  - **Bit Rate (R)**: The number of bits transmitted per second, given by $R = k \cdot R_s$.

II. Synthesis of Digital Signals

- **Signal Space Concept:** Any set of M signal waveforms, $s_m(t)$, can be represented as a linear combination of N orthonormal basis functions, $\phi_n(t)$.
- **General Formula:**
  - $s_m(t) = \sum_{n=1}^{N} s_{mn}\phi_n(t)$, for $m = 1, \ldots, M$.
- **Vector Representation:**
  - This allows each complex waveform to be represented as a simple N-dimensional vector of coordinates: $s_m = [s_{m1}, \ldots, s_{mN}]^T$.
  - The set of all M of these vectors forms the **constellation diagram**, which is a discrete representation of the signal space.

III. Pulse Amplitude Modulation (PAM)

- **Definition:** PAM is a modulation scheme that encodes information by varying the **amplitude** of the signal.
- **Waveform Formula:**
  - $s_m(t) = A_m g(t) \cos(2\pi f_0 t)$.
  - Information is carried in the discrete amplitude levels, $A_m = 2m - 1 - M$.
  - $g(t)$ is the pulse-shaping signal, and f_0 is the carrier frequency.
- **Vector Representation and Constellation:**
  - PAM is a **1-dimensional** modulation scheme ($N = 1$).
  - Its single basis function is $\phi_1(t) = \sqrt{\frac{2}{E_g}} \cos(2\pi f_0 t)$.
  - The constellation points, $s_m = A_m \sqrt{E_g/2}$, lie along a single line.
- **Energy and Performance:**
  - The energy per symbol, $E_m = A_m^2 E_g/2$, depends on the amplitude, meaning different symbols have different energies.
  - The average energy, $E_{avg} = \frac{(M^2-1)E_g}{6}$, grows rapidly with M, making PAM **power-inefficient** for large constellations.

MAC Protocols: Aloha and CSMA

I. The Need for MAC Protocols

- **Problem:** On a shared **broadcast channel**, a mechanism is needed to decide who gets to use the channel when there is competition.
- **The MAC Sublayer:** The protocols used for this purpose belong to the Medium Access Control (MAC) sublayer of the data link layer.
- **Inefficiency of Static Allocation:** Statically dividing a channel (e.g., via FDM) is inefficient for bursty traffic, leading to wasted resources and a mean delay N times worse than dynamic access.

II. ALOHA Protocols

- **Core Idea:** Users transmit whenever they have data, without sensing the channel first. Collisions occur if transmissions overlap.
- **Pure ALOHA:**
  - **Mechanism:** If a collision occurs, each sender waits a **random amount of time** before retransmitting.
  - **Vulnerable Period:** A frame can collide with another that starts one frame time before or during its transmission, making the vulnerable period **two frame times** long.
  - **Throughput:** $S = Ge^{-2G}$, with a maximum of about **18.4%**.
- **Slotted ALOHA:**
  - **Mechanism:** Time is divided into discrete slots, and stations can only begin transmitting at the start of a slot.
  - **Vulnerable Period:** This rule halves the vulnerable period to **one frame time**, as collisions only occur if two stations transmit in the same slot.
  - **Throughput:** $S = Ge^{-G}$, which doubles the maximum throughput to about **36.8%**.

III. Carrier Sense Multiple Access (CSMA) Protocols

- **Core Idea:** "Listen before you talk." Stations first sense the channel to check for ongoing transmissions before sending. This significantly outperforms ALOHA.
- **Protocol Variants:**
  - **1-persistent CSMA:** If the channel is busy, the station waits and **continuously senses** until it becomes idle, then transmits immediately. This is greedy and can lead to guaranteed collisions if multiple stations are waiting.
  - **nonpersistent CSMA:** If the channel is busy, the station **waits for a random period of time** before sensing again. This is less greedy and performs better under heavy load.
  - **p-persistent CSMA:** A compromise for slotted channels. If the channel is idle, it transmits with probability p or defers to the next slot with probability $q = 1 - p$.
- **CSMA/CD (Collision Detection):**
  - An enhancement where stations listen *while* transmitting.
  - If a collision is detected (by comparing transmitted and received signals), the station **immediately aborts** its transmission to save bandwidth and waits a random time before trying again. This is the basis for Classic Ethernet.

Routing Algorithms: Link State and Distance Vector

I. Introduction to Routing

- **Goal:** A routing algorithm's primary responsibility is to decide which output line an incoming packet should be sent on.
- **Graph Model:** The network is modeled as a graph where nodes are routers and edges are communication links. The goal is to find the "shortest path" based on a metric like hops, delay, or bandwidth.

II. Distance Vector Routing

- **Core Idea:** Each router maintains a **distance vector** (a table) listing the best known distance and the next hop to every other router. Routers only have knowledge of their direct neighbors.
- **Mechanism:**
  1. **Information Exchange:** Periodically, every router sends its entire distance vector to its direct neighbors.
  2. **Route Calculation:** When a router receives a vector from a neighbor (say, X), it calculates a new potential path cost to every destination (i) by adding its own cost to reach X (m) to X's reported cost to i (X_i).
  3. **Table Update:** If this new path ($X_i + m$) is shorter than the current best path, the router updates its table with the new shorter distance and sets the next hop to be X.
- **Key Characteristic:** Relies on information passed from neighbors; "routing by rumor."

III. Link State Routing

- **Core Idea:** Each router builds a **complete map (graph)** of the entire network topology. Once the map is built, each router independently calculates the shortest path to all destinations using an algorithm like Dijkstra's.
- **Mechanism (5 Steps):**
  1. **Discover Neighbors:** Routers learn the addresses of their directly connected neighbors (e.g., using HELLO packets).
  2. **Set Link Costs:** The cost (metric) for the link to each neighbor is determined, often inversely proportional to bandwidth.
  3. **Construct Link State Packet (LSP):** Each router creates a packet containing its identity and the list of its neighbors with their link costs.
  4. **Distribute LSPs:** The LSP is distributed to **all other routers** in the network using flooding. Sequence numbers and an "age" field are used to manage this process.
  5. **Compute Routes:** With a full set of LSPs, each router has the complete network map and runs a shortest path algorithm (like Dijkstra's) locally to build its forwarding table.
- **Key Characteristic:** Each router has full topology information, leading to faster convergence.

TCP Protocol: Congestion Control

I. The Congestion Problem

- **Definition:** Congestion occurs when "too many sources are sending too much data too fast for the network to handle."
- **Congestion vs. Flow Control:**
  - **Flow Control:** An end-to-end mechanism to prevent a fast sender from overwhelming a *slow receiver*.

    - **Congestion Control:** A network-wide mechanism to prevent a sender from overwhelming the *network itself*.
- **Consequences of Congestion:** Lost packets due to router buffer overflow and long delays due to queuing.

II. TCP's Approach to Congestion Control

- **Core Strategy:** TCP probes for usable bandwidth by adjusting a **Congestion Window (Congwin)**, which limits the amount of unacknowledged data the sender can have in flight.
    - The sender gradually **increases** Congwin to find the available capacity.
    - Upon detecting packet loss (a sign of congestion), it **decreases** Congwin.
- **Key Variables:**
    - Congwin: The congestion window size.
    - **Threshold (ssthresh):** A variable that determines the boundary between the Slow Start and Congestion Avoidance phases.

III. The Congestion Control Algorithm

- **Phase 1: Slow Start**
    - **When:** Used at the beginning of a connection or after a timeout.
    - **Mechanism:** Congwin starts at 1 MSS and **doubles** with each acknowledged segment, resulting in **exponential growth** per RTT.
    - **End Condition:** Continues until Congwin reaches the Threshold value or a loss event occurs.
- **Phase 2: Congestion Avoidance**
    - **When:** Begins after Congwin surpasses the Threshold.
    - **Mechanism:** Probes for bandwidth more cautiously, increasing Congwin by approximately 1 MSS per RTT (**linear growth**).
- **Reacting to a Loss Event:**
    - The Threshold is updated to half of the current Congwin value ($\text{Threshold} = \text{Congwin}/2$).
    - Congwin is reset to 1 MSS.
    - The protocol re-enters the **Slow Start** phase.
- **TCP Tahoe vs. TCP Reno:**
    - **TCP Tahoe:** Always resets Congwin to 1 and enters Slow Start after any loss event.
    - **TCP Reno:** If loss is detected by 3 duplicate ACKs (Fast Retransmit), it halves Congwin and enters Fast Recovery (a form of Congestion Avoidance), avoiding a full slow start. It only reverts to Slow Start after a timeout.

Channel Coding Theorem

I. The Goal: Reliable Communication

- The **Channel Coding Theorem**, also known as Shannon's second theorem, establishes the theoretical limits for achieving reliable communication over a noisy channel. It answers the question: what is the maximum rate at which we can send data with zero errors?

II. Statement of the Theorem

- **The Promise:** For any communication channel, there exists a maximum rate, known as the **channel capacity** C, at which information can be transmitted. It is possible to transmit data at any rate $R < C$ with an **arbitrarily small probability of error** by using sufficiently sophisticated error-correcting codes.
- **The Limit:** If the transmission rate $R > C$, it is **impossible** to achieve an arbitrarily low error probability. Errors will occur with a non-negligible probability regardless of the coding scheme used.

III. The Shannon Capacity Formula

- For an **Additive White Gaussian Noise (AWGN)** channel, the capacity C is defined by the formula:

$$C = B \log_2(1 + \mathrm{SNR})$$

- **Components of the Formula:**
  - C **(Channel Capacity):** The theoretical maximum reliable data rate, measured in bits per second (bps).
  - B **(Bandwidth):** The bandwidth of the channel, measured in Hertz (Hz).
  - SNR **(Signal-to-Noise Ratio):** A dimensionless ratio of the average received signal power to the average noise power. A higher SNR indicates a cleaner channel.
  - $\log_2(1 + \mathrm{SNR})$**:** This term quantifies how efficiently the bandwidth can be used, with the base-2 logarithm indicating the result is in bits.

IV. Practical Implications

- **Theoretical Upper Bound:** The Shannon Capacity C is a fundamental upper limit; practical systems will always achieve rates somewhat below it.
- **Design Guidance:** The formula shows that to increase capacity, one must either **increase the bandwidth (**B**)** or **improve the** SNR (by increasing signal power or reducing noise).
- **Role of Coding:** The theorem proves that codes *exist* to achieve near-capacity rates, but it does not specify how to construct them. Achieving rates close to C requires very complex and long error-correcting codes, like Turbo codes or LDPC codes.

The Digital Communication System: From Source to Destination

I. Overview

- A digital communication system encompasses the entire process of moving information from a source, across a physical medium, to a destination. The process involves encoding, modulation, transmission, demodulation, and decoding.

II. The Transmitter Path

1. **Source:** The origin of the message, which can be an analog signal (like voice) or digital data.
2. **Transducer:** Converts the original information into an electrical signal (e.g., a microphone converting sound waves). This step is skipped if the source is already a digital device.
3. **Source Encoder:** Performs **data compression** to remove redundancy and represent the message with the minimum possible number of bits. This can be lossless (like ZIP) or lossy (like JPEG).

4. **Channel Encoder:** Adds **redundant bits** to the data stream to make the communication robust against channel noise. This enables **error detection and correction** at the receiver.
    - **Note the Trade-off:** Source encoding *removes* redundancy for efficiency, while channel encoding *adds* redundancy for robustness.
5. **Digital Modulator:** Maps the final digital bitstream into an **analog signal** (e.g., an electromagnetic wave) that is suitable for transmission over the physical channel.

III. The Channel

- This is the physical transmission medium (e.g., fiber optic cable, air) where the signal travels. It is where noise and interference can corrupt the signal.

IV. The Receiver Path

7. **Digital Demodulator:** Receives the noisy analog signal from the channel and converts it back into a digital signal (a stream of bits).
8. **Channel Decoder:** Uses the redundant bits added by the channel encoder to **detect and/or correct errors** that occurred during transmission.
9. **Source Decoder: Decompresses** the data, reversing the source encoding process to restore the original message.
10. **Transducer:** Performs the reverse action of the first transducer, converting the electrical signal back into a form usable by the destination (e.g., a speaker converting the signal to sound).
11. **Destination:** The final recipient of the information.

Framing Techniques in the Data Link Layer

I. The Purpose of Framing

- The Data Link Layer receives packets from the Network Layer and encapsulates them into units called **frames**.
- **Function:** Framing breaks the raw bit stream from the physical layer into discrete blocks, which is necessary for error detection.
- A good framing method must make it easy for a receiver to find the start of a new frame, even after losing synchronization, while using minimal overhead.

II. Framing Methods

- **1. Byte Count:**
    - **Mechanism:** The frame header contains a field specifying the number of bytes in the frame. The receiver reads this count and then reads that many bytes.
    - **Major Drawback:** This method is **not robust**. If the count field is corrupted by a transmission error, the receiver loses synchronization and will likely interpret all subsequent frames incorrectly.
- **2. Flag Bytes with Byte Stuffing:**
    - **Mechanism:** Each frame starts and ends with a special **flag byte** (e.g., FLAG). If the receiver loses sync, it can scan for the next FLAG byte to resynchronize.
    - **The Problem:** The FLAG byte pattern might appear in the actual data.
    - **The Solution: Byte Stuffing.**

        - The sender inserts a special **escape byte (ESC)** into the data just before any accidental FLAG byte.
        - If an ESC byte itself appears in the data, the sender "stuffs" another ESC byte before it.
        - The receiver performs the reverse process of "de-stuffing" to restore the original data.
- **3. Flag Bits with Bit Stuffing:**
    - **Mechanism:** This method is more flexible as it is not tied to byte boundaries. Each frame begins and ends with a special bit pattern: **01111110**.
    - **The Problem:** The flag's bit pattern could appear in the data.
    - **The Solution: Bit Stuffing.**
        - **Sender:** Whenever the sender encounters **five consecutive 1s** in the data, it automatically "stuffs" a **0 bit** into the outgoing stream.
        - **Receiver:** When the receiver sees five consecutive 1s, it inspects the sixth bit. If it's a 0, it is removed (de-stuffed). If it's a 1, it checks the seventh bit to see if it's a valid flag (01111110) or an error.

Error Control: Error Detection vs. Error Correction

I. The Need for Error Control

- Errors are a common occurrence during data transmission, and robust protocols must be able to handle them. There are two fundamental approaches to managing these errors.

II. Two Fundamental Strategies

- **1. Error-Correcting Codes (Forward Error Correction - FEC):**
    - **Goal:** To allow the receiver to **deduce and fix** what the transmitted data must have been, even with errors present.
    - **Mechanism:** Includes **enough redundant information** in the message to enable correction.
    - **Use Case:** Often used for real-time services like video streaming, where waiting for a retransmission (latency) is unacceptable.
- **2. Error-Detecting Codes (with ARQ):**
    - **Goal:** To allow the receiver to know **that an error has occurred**, but not to fix it.
    - **Mechanism:** Includes **only enough redundancy for detection**.
    - **Use Case:** When an error is detected, the receiver requests a retransmission from the sender, a process known as Automatic Repeat Request (ARQ). This is common where reliability is paramount and some delay is acceptable.

III. The Role of Hamming Distance

- **Definition:** The **Hamming distance** between two codewords is the number of bit positions in which they differ. It requires d single-bit errors to convert one codeword into another if their Hamming distance is d.
- **Error Control Capability:** The error-handling properties of a code depend directly on its minimum Hamming distance.
    - **To Detect d Errors:** The code must have a minimum Hamming distance of at least $d + 1$. This ensures that d errors cannot change one valid codeword into another valid codeword; the result will be an invalid codeword that is detected as an error.

- **To Correct d Errors:** The code must have a minimum Hamming distance of at least $2d + 1$. This ensures that even with d errors, the corrupted word is still mathematically closer to the original correct codeword than to any other valid codeword, allowing the receiver to choose the closest one for correction.

IV. How Correction Works

- Error correction is possible because out of 2^n possible n-bit words, only a small fraction (2^m) are valid codewords.
- When a non-legal codeword is received, the receiver assumes the original was the **closest valid codeword** in terms of Hamming distance.

IP Addressing: Subnetting and Classless Inter-Domain Routing (CIDR)

I. Fundamentals of IP Addressing

- **Structure:** An IPv4 address is 32 bits long and is divided into a **network portion (prefix)** and a **host portion**. All devices on the same network share the same network prefix.
- **Notation:** Addresses are written in dotted decimal notation (e.g., 192.168.1.1) for readability. The prefix length is specified with a slash (e.g., /24), known as CIDR notation.

II. The Problem with Classful Addressing

- The original system (Class A, B, C) divided the address space into fixed-size blocks.
- This was highly **inefficient**, often assigning organizations blocks that were either too large (wasting millions of addresses) or too small for their needs. This inefficiency drove the need for a new system.

III. Subnetting: Dividing a Network

- **Concept:** Subnetting allows a single, larger block of IP addresses assigned to an organization to be **divided into several smaller, independent networks (subnets)**.
- **Purpose:**
  - Provides more flexible address allocation for internal use.
  - Allows for better organization and improved security through isolation.
  - Enables more efficient use of the assigned IP address space.
- **Mechanism:** An organization can take its assigned prefix (e.g., a /16) and create multiple subnets with longer prefixes (e.g., /17, /18, /19) to match the needs of different departments or functions.

IV. CIDR: Aggregating Networks

- **Concept:** Classless Inter-Domain Routing (CIDR) replaced the classful system to slow the exhaustion of IPv4 addresses and manage the size of global routing tables.
- **Core Idea: Route Aggregation (Supernetting).** CIDR allows multiple smaller, contiguous IP prefixes to be combined and advertised to the rest of the world as a **single, larger prefix**. For example, sixteen /24 networks can be advertised as a single /20 route.
- **Benefits:**
  - **Flexible Prefixes:** Network prefixes can be of any length (e.g., /19, /22), allowing allocations to closely match an organization's needs.

    - **Reduced Routing Table Size:** Route aggregation means Internet backbone routers need to store fewer individual routes, which improves routing efficiency.
- **Longest Matching Prefix Rule:** When a router has multiple overlapping routes in its table, it forwards a packet based on the **most specific match** (the route with the longest prefix).

Transport Layer Services: A Comparison of TCP and UDP

I. The Role of the Transport Layer

- **Function:** While the network layer provides host-to-host delivery, the transport layer extends this to **process-to-process delivery**.
- **Operation:** Transport protocols like TCP and UDP run only in the **end systems**, not in intermediate routers. They take application data and create **segments**, which are then encapsulated in network-layer packets.

II. Transmission Control Protocol (TCP)

- **Service Model:** Provides **reliable, in-order delivery** of a stream of bytes.
- **Connection-Oriented:** A logical connection must be established between sender and receiver via a **three-way handshake** before data transfer can begin.
- **Key Features:**
    - **Reliable Delivery:** Guarantees data will arrive correctly and without loss, using sequence numbers, acknowledgments (ACKs), and retransmissions.
    - **In-Order Delivery:** Ensures data segments are delivered to the application in the same order they were sent.
    - **Flow Control:** Prevents a fast sender from overwhelming a slow receiver by using a receiver-advertised window ($\mathrm{RcvWindow}$).
    - **Congestion Control:** Adjusts the sending rate to avoid overloading the network, helping to ensure fairness.
- **Use Cases:** Used where reliability is critical, such as the World Wide Web (HTTP), email (SMTP), and file transfer (FTP).

III. User Datagram Protocol (UDP)

- **Service Model:** Provides a minimalistic, **connectionless** "best-effort" service.
- **No Handshake:** There is no initial connection setup, which saves time. Each UDP segment is handled independently.
- **Key Features:**
    - **Unreliable:** Provides no guarantees on delivery, order, or error-free transmission. If reliability is needed, it must be implemented by the application.
    - **Simplicity:** No connection state is maintained by the sender or receiver.
    - **Low Overhead:** UDP headers are very small (8 bytes).
    - **No Congestion Control:** Does not automatically throttle its sending rate, which can be good for some real-time applications but can also contribute to network congestion.
- **Use Cases:** Used for applications that are delay-sensitive and can tolerate some data loss, such as streaming multimedia, DNS, and SNMP.

Principles of Reliable Data Transfer (RDT)

I. The Goal of RDT

- **Objective:** To provide a reliable data transfer service to the application layer, built on top of an underlying network layer that is inherently **unreliable** (e.g., IP can lose, reorder, or corrupt packets).
- The complexity of an RDT protocol depends on the types of errors the underlying channel can introduce, such as bit errors and packet loss.

II. Building Blocks of RDT

- **1. Handling Bit Errors:**
  - **Error Detection:** The first step is to detect errors, typically by including a **checksum** with the data.
  - **Feedback:** The receiver must provide feedback to the sender.
    - **Acknowledgments (ACKs):** A positive message indicating a packet was received correctly.
    - **Negative Acknowledgments (NAKs):** An explicit message indicating a packet was received with errors, prompting a retransmission.
- **2. Handling Duplicates from Corrupted Feedback:**
  - **The Problem:** If an ACK/NAK is corrupted or lost, the sender may retransmit a packet that was already successfully received, creating a duplicate.
  - **The Solution: Sequence Numbers.**
    - The sender adds a **sequence number** to each data packet to uniquely identify it.
    - The receiver checks the sequence number of an incoming packet. If it has already processed a packet with that number, it recognizes it as a **duplicate and discards it**.
- **3. Handling Packet Loss:**
  - **The Problem:** If a data packet or its ACK is lost entirely, the sender will wait forever without feedback.
  - **The Solution: Timeouts.**
    - The sender starts a **countdown timer** for each packet it sends.
    - If the timer expires before a corresponding ACK is received, the sender **assumes the packet was lost** and retransmits it.
    - Sequence numbers are again crucial to handle cases where a delayed packet (not a lost one) causes a premature timeout and a duplicate retransmission.

III. The Stop-and-Wait Protocol

- A simple RDT protocol that combines these principles: the sender transmits a single packet and then **stops and waits** for an ACK before sending the next one. It uses timeouts to recover from lost packets.

Pipelined Protocols: Go-Back-N vs. Selective Repeat

I. The Need for Pipelining

- **Problem: Stop-and-Wait** protocols are simple but highly inefficient, as they only allow one packet to be "in-flight" at a time, leading to poor channel utilization.

- **Solution: Pipelining.** Pipelining allows the sender to transmit **multiple packets** without waiting for an acknowledgment for each one, dramatically improving efficiency.
- **Requirements:** This requires a larger range of sequence numbers and buffering at the sender and/or receiver.

II. Go-Back-N (GBN)

- **Sender Behavior:**
  - Maintains a send window of up to N consecutive unacknowledged packets.
  - Uses a single timer, typically for the oldest unacknowledged packet.
  - **Timeout Action:** If the timer expires (e.g., for packet n), the sender **retransmits packet n and all subsequent higher-numbered packets** that were already in the window. This is the "go back" step.
- **Receiver Behavior:**
  - **Acknowledgments:** Uses **cumulative acknowledgments**. An $\mathrm{ACK}(n)$ confirms that all packets up to and including n have been correctly received.
  - **Out-of-Order Packets:** The receiver is simple; it **discards any packet that arrives out of order**. It does not buffer them.
  - After discarding an out-of-order packet, the receiver re-sends an ACK for the highest in-order packet it has received so far.

III. Selective Repeat (SR)

- **Goal:** To reduce the unnecessary retransmissions of GBN by having the sender retransmit **only those packets that were actually lost**.
- **Sender Behavior:**
  - Maintains a send window of up to N unacknowledged packets.
  - Maintains a **separate timer for each unacknowledged packet**.
  - **Timeout Action:** If the timer for a specific packet (n) expires, the sender **retransmits only packet n**.
- **Receiver Behavior:**
  - **Acknowledgments:** Acknowledges each correctly received packet **individually** using selective ACKs.
  - **Out-of-Order Packets:** The receiver **buffers correctly received packets that arrive out of order**.
  - Once a missing packet (e.g., n) arrives, the receiver can deliver it and any consecutively numbered packets that were already in its buffer to the application, advancing its receive window.

IV. Comparison

- **Retransmissions:** GBN can be wasteful as a single packet loss may cause many packets to be retransmitted. SR is more efficient as it only retransmits what is necessary.
- **Receiver Complexity:** The GBN receiver is very simple (no buffering of out-of-order packets). The SR receiver is more complex as it must maintain a buffer and manage out-of-order data.