**DSC-204A: Scalable Data Systems, Winter 2024**

# Network - 2

*Lecturer: Hao Zhang* *Scribe: Hongbei Liu, Shuying Li, Sihan Wang*

## Recap

### NetWork Hardware

**Network (interface) card/controller (NIC):** Before starting communication, the OS and NIC assign ports to specific processes on a computer, and these ports are then associated with IP addresses. Afterwards, connections are established between the NICs located on two distinct computers, enabling the initiation of communication between them.

### Packet Switch

**Packet Forwarding:** Essentially, when communication begins, packet forwarding takes place. Different applications share links, with each application sending information in packets across these links. This process involves a sort of sharing on the link.

## New Content

## 1 Layers

### 1.1 The problem

Although everything seems to work fine at an abstract level, in reality, two main issues arise.

1. **On the upper layer**: there are numerous applications each with distinct characteristics. For instance, email latency may not be a concern, but online gaming requires low latency with immediate delivery. Similarly, online video streaming demands no frame loss. Essentially, various applications sharing the links have unique properties and specific requirements that need to be met.

2. **On the lower layer**: a wide range of network styles and technologies exists, including wireless and wired connections, with options like optical fiber.

Therefore, it's crucial to figure out how to organize such mess to ensure everything works cohesively.

### 1.2 Solution: Intermediate Layers

The problem with implementing a specific protocol (networking software) for each pair of applications and transmission media is that this approach is clearly non-scalable because it requires re-implementing every application for every technology.

The solution is to introduce **intermediate layers** with a set of functionalities, ensuring that each layer can communicate with others through unified protocols or interfaces. This concept parallels how software development, particularly object-oriented programming, functions. For any application and hardware transmission media, there's provision to connect to the intermediate layer, facilitating communication and ensuring packet delivery. For instance, as shown in fig1, in addition to existing media like Coaxial cable and Fiber optic, if we wish to introduce a new transmission media called Packet radio, it can simply be integrated into the intermediate layer.

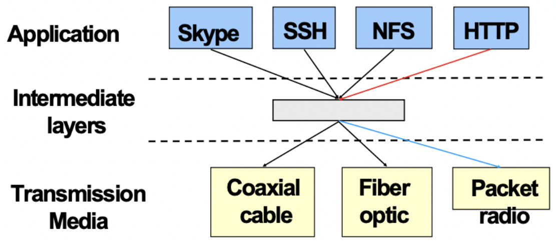

Figure 1: Add Packet radio

### 1.3 Goal of This Intermediate Layer

- **Delivery:** deliver packets between to any host in the Internet
  - E.g., deliver a packet from a host in UCSD to a host in Tokyo.
- **Reliability:** tolerate packet losses
  - E.g., how do you ensure all bits of a file are delivered in the presence of packet loses?
  - The hardware setups are often unstable, as there is a multitude of different routers and links situated in various locations under diverse conditions. These can occasionally drop packets; therefore, the capability to tolerate packet loss is important.
- **Flow control:** avoid overflowing the receiver buffer
  - E.g., how do you ensure that a server that can send at 10Gbps doesn't overwhelm a LTE phone?
  - Your phone might be equipped with 4G technology, capping its maximum bandwidth to receive data from the internet at a few Mbps. In contrast, a cloud data center's network might have the capacity to send data at rates as high as 10Gbps. In such scenarios, it's crucial to prevent the server from sending excessive amounts of data that could overwhelm your phone. This not only leads to increased power consumption but could also potentially cause damage to the device.
- **Congestion control:** avoid overflowing the buffer of a router along the path
  - What happens if we don't do it?

  - Routers have limited memory, so it's important to avoid sending too many packets to a single router. As the router's buffer becomes full, it has no room to store incoming packets. When new packets arrive and there is no space in the buffer, the router will drop or discard packets. Besides, it can also lead to increased latency, congestion spreading, unpredictable behavior for applications, and potential network outages. We need to ensure there is load balancing to distribute network traffic evenly across routers.

## 1.4 Building software for networks

Layering is just like building the software system for the network with a lot of abstractions, and each layer takes some functionalities. Layering is just like building a software system for the network with a lot of abstractions. Each layer takes some functionalities. Upper layer will call the layer below it, and try to implement some higher level functionalities.

- Pros: many functionalities implemented in other layers can be reused.
- Cons: excessive abstractions can impact performance and complicate the debugging process. For instance, when dealing with lost packets, navigating through multiple layers becomes necessary to identify the source of the problem.

## 1.5 Network's Software: Layers

The implementation of layering in software resembles the operating system found in network devices like NICs and routers. However, these implementations are distributed across each NIC.

## 1.6 Layers == Microsevices architecture s.t. constraints

The entire internet functions as a microservice, with its layers providing services. However, it differs from traditional microservices in that it adheres to specific constraints. In this context, the lower layers exclusively offer services to the higher layers, never providing services to those below them. For instance, a layer at the bottom only serves layers situated above it, without extending services to layers beneath it.

## 1.7 Physical Layer

- **Service:** what a layer does
- **Service interface:** how to access the service
  - It's important for layers to communicate with each other in a unified manner, which can be achieved through a defined interface. This setup allows for the easy addition of new features by incorporating them into the interface.
- **Protocol:** how peers communicate to achieve the service
  - Protocols establish the rules for how applications on different computers communicate with each other. If you're creating programs that need to communicate between two machines, you can use any programming language. The key is to ensure that when communication begins, your messages are formatted to follow these protocols.

## 1.8 Datalink Layer

- **Service:** The datalink layer enables end hosts to frames or atomic messages over a shared physical or wireless link. Frames represent a more advanced and organized form of data communication compared to the raw bit streams handled at the physical layer.
  - **Arbitrate access:** It manages the way multiple devices share the same communication medium
  - **Reliable transmission:** It ensures that data frames are transferred reliably across the network.
  - **flow control:** It moderates the rate of data transmission between two devices.
- **Interface:** It manages the dispatch and receipt of data frames between devices on the same network.
- **Protocols:**
  - The addressing used to identify devices at the datalink Layer is known as the **Media Access Control (MAC) address**. MAC addresses are unique identifiers for network interfaces and have 48-bit, assigned by the device manufacturer.
  - **Local Area Networks (LANs):** groups of devives or hosts in same geographical area or using same medium.
  - **Broadcasting and Switching:** Within LANs, data can be broadcast to all devices, but this isn't efficient or secure. Switches improve efficiency by directing data frames only to the intended recipient, reducing network congestion and enhancing security.

The datalink Layer is concerned only with the local network presence and doesn't handle broader, abstract network addressing—that's the role of higher layers. Figure 2 shows the structure of datalink layer.

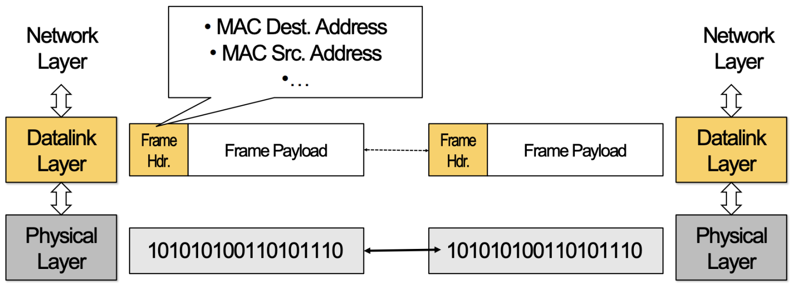

Figure 2: Datalink Structure

### 1.8.1 MAC Protocols

- Problems:
  - How do hosts access a broadcast media?
  - How do they avoid collisions?
- Solutions:
  - **Channel partitioning protocols:**

    - ∗ These protocols allocate an equal division of bandwidth to each host on a network (1/N of the total bandwidth). They are efficient and fair when the network load is high, meaning many hosts are sending data.
    - ∗ However, they become inefficient under low load conditions since the bandwidth allocation remains the same even if fewer hosts are active, leading to underutilization of available resources.
    - ∗ Frequency Division Multiple Access (FDMA) and optical networks are examples of channel partitioning protocols.
  - **"Taking turns" Protocols:**
    - ∗ These protocols allow users to take turns in sharing the channel, which is facilitated by passing a token among users.
    - ∗ A user with the token can use up the entire bandwidth of the link, but this introduces complexity.
    - ∗ Problems include token loss, which can disrupt communication or lead to unfairness, and additional overhead from the need to send tokens along with the packets.
  - **Random Access:**
    - ∗ These protocols allow users to send information randomly.
    - ∗ This method is efficient and low-cost as long as the network is not too busy, since the probability of message collisions is low under light traffic. However, when the network is busy, the probability of collisions increases, leading to potential problems.
    - ∗ Incorporate Collision Detection (CD) to identify overlapping data transmissions.
    - ∗ Utilize Carrier Sense (CS) to check for pre-existing signals before transmitting.
    - ∗ Employ randomness in transmission timing to decrease the likelihood of collisions.

## 2 OSI Layering Model

The OSI model, which has 7 layers, is seen as complicated. As a result, a simpler version called the Internet Protocol (IP) model was created, featuring just 5 layers.

### 2.1 Physical Layer(1)

**Purpose:** providing the means for transmitting raw bits over a physical data link connecting network nodes.

### 2.2 Network Layer(3)

#### 2.2.1 Wide Area Network

- Wide Area Network (WAN): network that covers a broad area (e.g., city, state, country, entire world), it connects local network together
  - E.g., Internet is a WAN
- WAN connects multiple datalink layer networks (LANs)
- Datalink layer networks are connected by **routers**
  - Different LANs can use different communication technologies (e.g., wireless, cellular, optics)

### 2.2.2 Routers

- **Forward** each packet received on an **incoming link** to an **outgoing link** based on packet's destination IP address (towards its destination)
- **Store and forward:** Routers have limited capacity, so packets are buffered before being forwarded;
  - If the packets are way too much, the router will be start dropping packets. There are also techniques (upper layer) will detect whether the packet is dropped, if so, they will resend the packet.
- **Forwarding table:** mapping between IP address and the output link, find the right router

### 2.2.3 Packet Forwarding

- Upon receiving a packet, a router
  - read the **IP destination address** of the packet
  - consults its forwarding table → output port
  - **forwards** packet to corresponding output port
  - the packet will be transferred between different LANs and finally reach the destination

### 2.2.4 The Internet Protocol(IP)

The IP is the internet network layer(defined in the third layer) and it will **try its best** to deliver the packet, but the packet will also be **lost, corrupted, or delivered out of order.**

## 2.3 Transport Layer(4)

- **Service:**
  - Provide end-to-end communication between processes
  - Demultiplexing of communication between hosts
  - Possible other services: Reliability in the presence of errors, Timing properties(packet sent in order), Rate adaption (flow-control, congestion control)
- **Interface:** Send message to specific process at given destination, local process receives messages sent to it.
- **Protocol:** port numbers, perhaps implement reliability, flow control, packetization of large messages, framing (increase the reliability of the errors)
- **Examples:** TCP and UDP

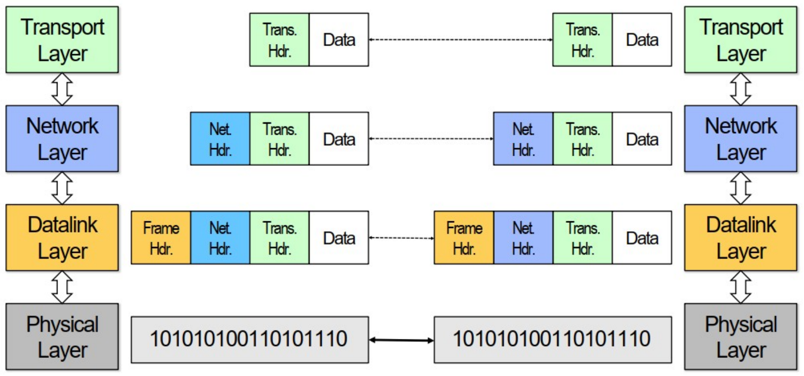

Figure 3: Data Transport Process

#### 2.3.1 Port Numbers

#### 2.3.2 Internet Transport Protocols

**UDP:** Straightforward extension of the IP protocol. UDP will attach a port on top of the IP packet and do the transmission. UDP will provide no guarantee.

**TCP:** It will provide guarantee that the packet will be delivered. It will also provide flow control and congestion control.

### 2.4 Application Layer(7 not 5)

- Service: any service provided to the end user
- Interface: depends on the application
- Protocol: depends on the application

The layer 5 and 6 are part of the OSI architecture but not in the Internet architecture, as they are absorbed in the application layer. The application layer will **define which data to transmit.**

### 2.5 Summary

- Lower three layers **implemented everywhere**.
- Top two layers implemented only at **hosts**.
- Logically, layers interacts with peer's corresponding layer. Actually they will go through many layers to pass the information.

## 3 Hourglass

The IP layer is pretty **thin** and thus the internet is **scalable**.

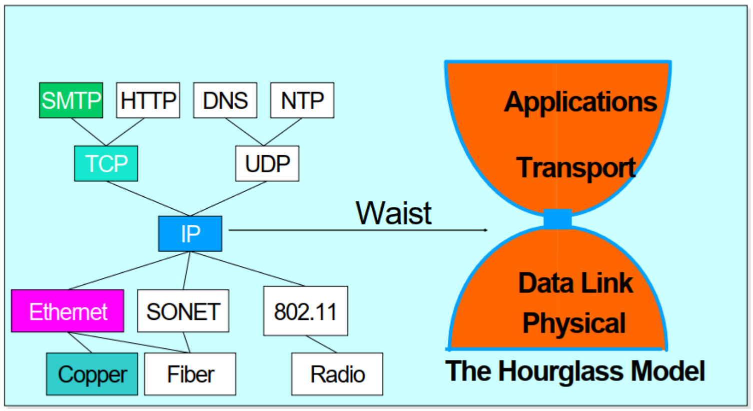

Figure 4: Hourglass Model

- Single Internet-layer module (IP):
  - Allows arbitrary networks to interoperate; Any network technology that supports IP can exchange packets
  - Allows applications to function on all networks: Applications that can run on IP can use any network
  - Supports simultaneous innovations above and below IP. But changing IP itself, i.e., IPv6 is very **complicated and slow**