**Chapter 13: Introduction to the Transport Layer**

Table of Contents

[13.1 Transport Layer Services 2](#_Toc86834063)

[Port Numbers 2](#_Toc86834064)

[Encapsulation and Decapsulation 3](#_Toc86834065)

[Multiplexing and Demultiplexing 3](#_Toc86834066)

[Flow Control, Error Control and Congestion Control 3](#_Toc86834067)

[Connectionless and Connection-Oriented Service 4](#_Toc86834068)

[13.2 Transport-Layer Protocols 7](#_Toc86834069)

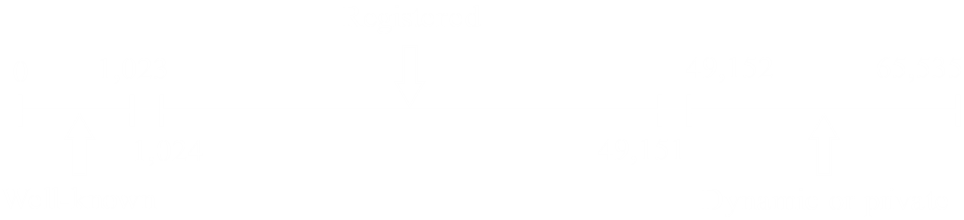
## 13.1 Transport Layer Services

### Port Numbers

The transport layer deals with **process-to-process communication**, meaning a process on the source device communicates with a process on the destination device using the transport layer. Since there can be **several processes** running on each device, we need some form of **addressing mechanism** to identify specific processes. The addressing mechanism of the transport layer deals with **port numbers**.

The process on the source device is identified using a specific port number, as is the process on the destination device. More specifically, we are dealing with **socket addressing**, which is a combination of **IP Address :** **Port Address**, e.g. .

Port numbers are assigned by the **Internet Corporation for Assigned Names and Numbers** (ICANN), an international organization. They have divided port numbers into three ranges, Well-Known, Registered and Dynamic or Private.



**Well-Known** port numbers are used by **servers**, since they run for long time periods. **Clients** on the other hand, use port numbers from the **Dynamic** range. The **Registered** range exists simply so organizations can register port numbers in order to **avoid duplication**.

### Encapsulation and Decapsulation

On the sender side, the data along with some header information is **encapsulated** into a **packet**. On the receiver side, this is **decapsulated**.

### Multiplexing and Demultiplexing

A sender device could be sending out data from **multiple processes**, which means the transport layer is receiving data from **multiple sources**. This is called **multiplexing**.

A receiver device could be receiving data that is supposed to be delivered to **multiple destinations**. This is called **demultiplexing**.

Note that when we say **source** or **destination** in this context, we mean a specific **process**, not the device itself.

### Flow Control, Error Control and Congestion Control

In the transport layer, we mainly use the **TCP protocol**. Under this protocol, we have error control, flow control and congestion control.

For **Error Control**, the main mechanism used is **detection and retransmission**.

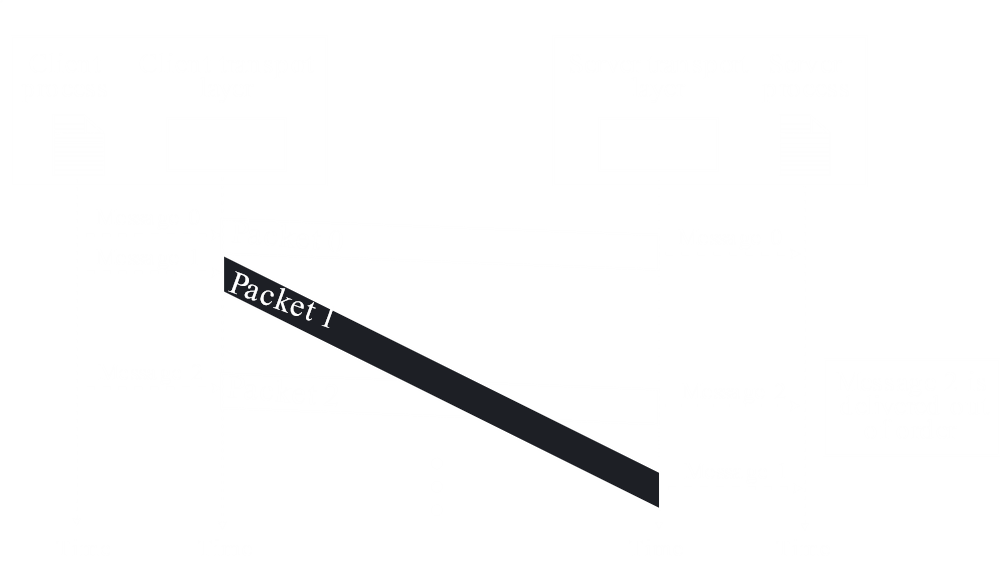
For **Flow Control**, we mainly use the **sliding window** techniques, such as the Go-Back-N ARQ, Selective Repeat ARQ and also Stop-and-Wait, even though that one is not a sliding window technique. Note that flow control in the transport layer is concerned with just the **sender** and the **receiver**, as opposed to the flow control in Layer 2, the **Data Link Layer**, which is concerned with **each hop**.

**Congestion Control** is similar to flow control, except that it can affect each hop. It is possible that the **queue** in a router has become full, which causes **congestion**. Congestion control is concerned with preventing this.

All of these topics will be discussed in depth when we study the TCP protocol later on.

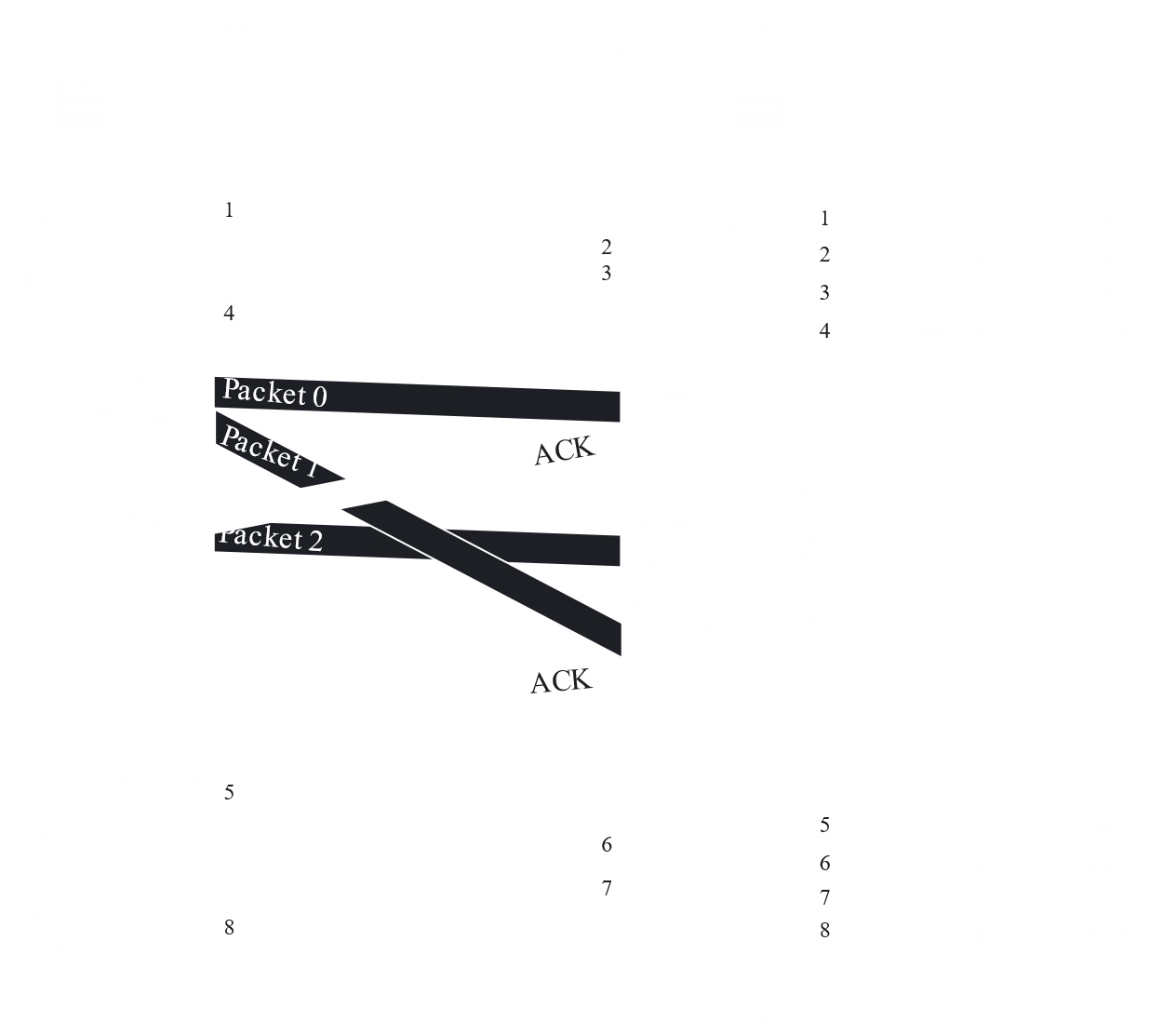
### Connectionless and Connection-Oriented Service

Imagine that the client is sending packets one after another without waiting for any **acknowledgements**. This can cause a variety of issues, such as **undelivered packets** and **out-of-order** packets.

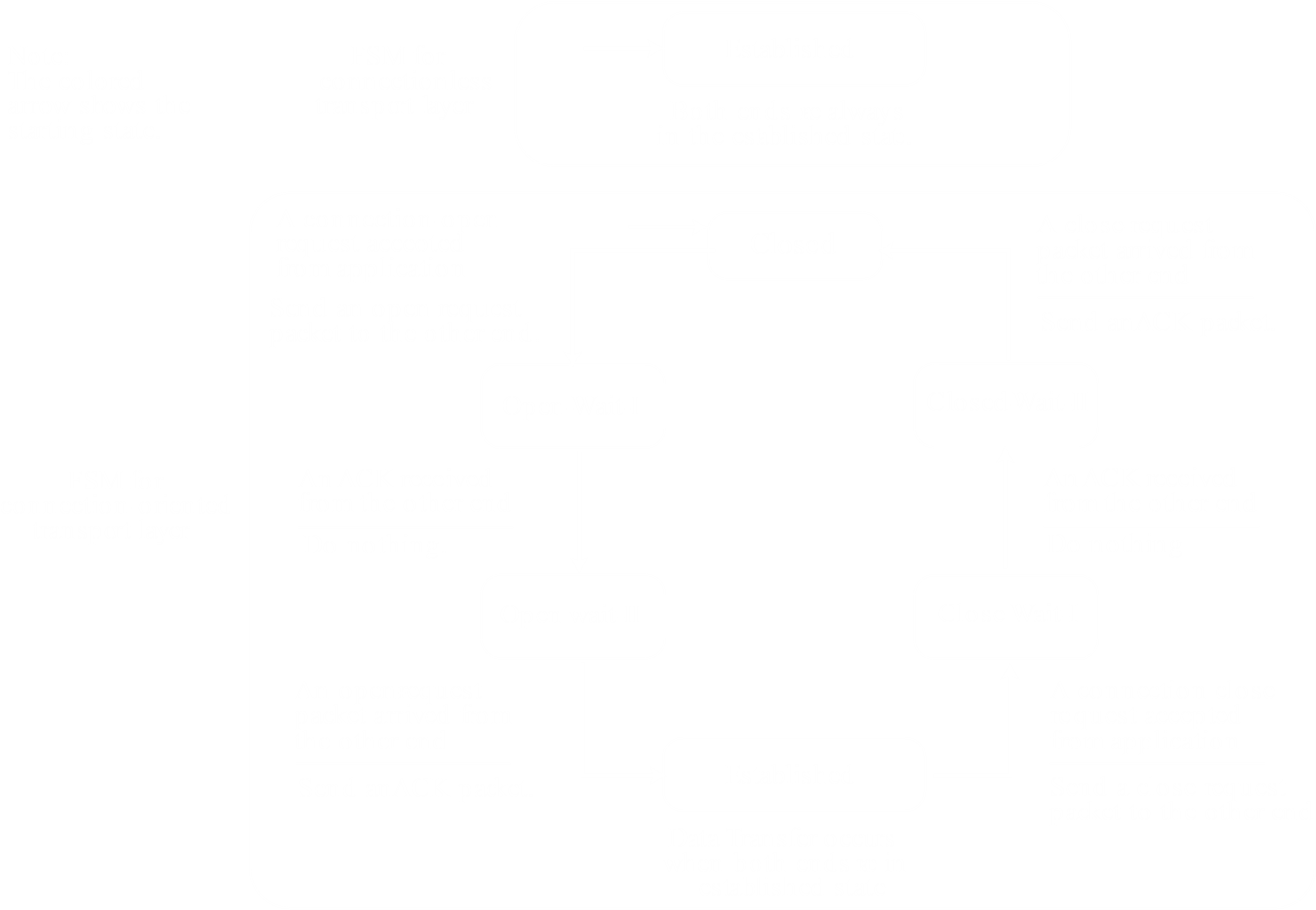


Such a service is called a **connectionless service**. This is the service used by the **UDP** protocol, which makes it **unreliable**. However, it also makes it faster.

**TCP** on the other hand, is **connection-oriented**. This means that some information is exchanged before the actual data is sent out, which avoids the problems mentioned above.



The process of opening and closing a connection can be described with the help of a **finite state machine**, such as the one below.



## 13.2 Transport-Layer Protocols

This section discusses different protocols used by the transport layer to communicate between the sender and receiver. These protocols are:

* Simple Protocol
* Stop-and-Wait Protocol
* Go-Back-N Protocol
* Selective Repeat Protocol
* Piggybacking

These protocols, except that last one, have been discussed previously. Thus, those parts are being skipped.

**Piggybacking** is simply the process of sending an acknowledgement of a previously received packet with a new packet that is being sent out. This improves **efficiency**. It can be used along with the other protocols.