Computer Networks

Hard- and software components that form the foundation of distributed computer systems

Leon Muscat

Contents

0 Introduction

The course is partly based on the computer systems course. The purpose is to understand how networks of processors work. Questions such as…

- How do we encode information to send it over a cable or through the air?
- How does the information know where it should go?
- How does the receiving processor know what to do with the information?
- How do we make these flows more reliable, secure, and less energy- consuming?

… will be answered.

0.1 Learning Goals

You can explain in detail and to kindergarten kids how the Internet and the Web work.

You are able to assign typical concerns in computer networking to the different layers of the networking stack and explain the implications of this separation of concerns.

You understand and can explain the functioning of central computer networking protocols (IP, TCP, QUIC, HTTP). You understand and can evaluate new protocols from a technical perspective.

You have fundamental knowledge in the areas of wireless communication, resource-constrained networking, and network security.

You can evaluate current and future developments in computer networking regarding technical aspects in their (political, business) context.

0.2 Exam

There are 6 assignments, which count for 40% of the final grade. The assignments must be completed in teams of two and are mostly carried out on RPi 400e hardware.

The final exam counts for 60% of the final grade and is similar to the computer systems exam. The points correspond to the estimated number of minutes needed. The questions are generally open and involve applying knowledge from the lectures.

0.3 Literature

It is advised to take a look at the textbook “Computer Systems: A Programmer’s Perspective (3rd edition)” by Bryant & O’Hallaron.

1 Networking Basics

Understanding the basics of how information from a computer makes its way through networks to other computers.

1.1 Entropy

In computer networking, entropy describes the **level of randomness** and the **amount of information** encoded. Therefore, entropy specifies how much information is needed to fully specify a state.

Entropy can be calculated through

$$H(X) = -\sum_{x \in X} p(x) \log p(x).$$

In this course only binary entropy is relevant, which can be calculated through

$$H_b(p) = -p * \log_2(p) - (1-p) * \log_2(1-p).$$

1.1.1 Capacity of a Channel

A communicating with B means that some physical acts of A have induced a desired physical state in B. The communication was successful if B and A agree about what was sent.

In a **noiseless binary channel**, any transmitted bit is received without error. Therefore, the **capacity** of the channel is **1 bit**.

In a **binary symmetric channel**, any transmitted bit has a probability of p to be received correctly, and a probability of $1-p$ of being flipped. The capacity of this channel is $C = 1 - H_b(p)$ **bits**.

1.2 From Circuit Switching to Packet Switching

1.2.1 Circuit switching

Traditional phone calls are based on circuit switching (See figure 1). There is a **dedicated channel**, which connects the sender to the receiver. This means that the **bandwidth** between the parties is **constant** in both directions.

For computers, this system is very inefficient, which has two reasons. First, the channel is **underutilized** most of the time because computers send data **non-continuously and bursty**. Second, when computers communicate, they need **immediate access** and send a lot of data at once.

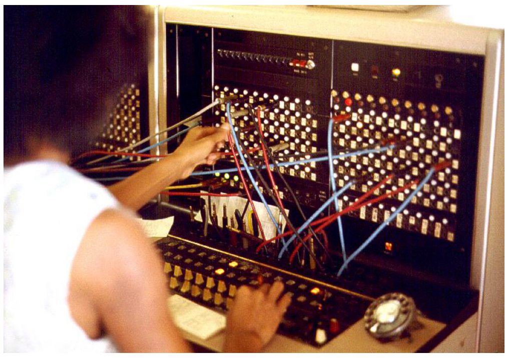

Figure 1: Traditional phone circuit switching

1.2.2 Packet Switching

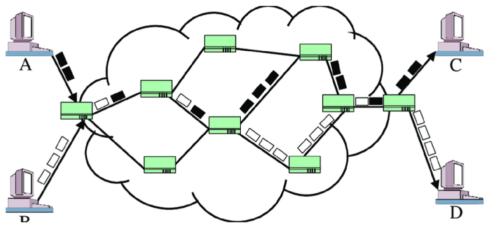

Figure 2: Packet switching illustration

Packet switching is similar to the postal service. Each message is transported using different means from postal office to postal office. Similarly, data sent by computers is split into chunks called **packets** and each packet is routed from machine to machine **independently** (See figure 2).

This offers a more efficient use of bandwidth and is more robust against disruption. For a connection to fail, all links between sender and receiver have to be broken.

1.3 The Internet

The Internet is a global system of interconnected computer networks that use the **Internet protocol suite (TCP/IP)** and packet switching. IP is the **Internet Protocol** and responsible for addressing and routing. TCP is the **Transmission Control Protocol** and is responsible for reliable data delivery.

1.4 The World Wide Web

The WWW is built on the Internet and allows for Web resources sharing. All resources are uniqely identified by **URIs** (Uniform Resource Identifiers) and can be interlinked by **hyperlinks**. It is built on a suite of standards, including **URL** (Uniform Resource Locators), **HTTP** (Hypertext Transfer Protocol), and **HTML** (Hypertext Markup Language).

Originally, In 1990, Tim Berners-Lee defined the Web as a web of documents, which are identified by Universal Document Identifiers (UDI) and link to other documents. Therefore, the user does not need to worry about protocols, except when something goes wrong (for example, document not found) or if to trust the Web site.

Later this idea is expanded to the web of things, where each “thing” is identified by a URI (Universal Ressource Identifier). This allows the user to not worry about documents anymore except the source and quality of the information.

The size of the Web can not be precisely stated, but the Google Search index contains hundreds of billions of webpages. The Web is therefore an Internet-scale system.

1.5 Representational State Transfer (REST)

REST is a software architectural style that guides the design and interaction of components within distributed systems, especially those that communicate over the World Wide Web. In REST, components (User Agent, Intermediaries, Servers) communicate through request-response interactions. REST provides a uniform interface to the Web, which allows for heterogeneous implementations of components to interact in a uniform manner.

1.5.1 Key Concepts

1. **Resources**: Fundamental concepts in REST. They can be any identifiable entity like a user, an article, or a photo. The identifier is a 2-part representation: type/subtype (like text/html, application/json, …). Ressources can’t be accessed directly. Instead, components exchange representations of ressources.
2. **URLs (Uniform Resource Locators)**: Each resource is identified by a specific URL. They provide a global addressing space for resource and service discovery. Each URL (https://www.w3.org/TR/webarch/#def-resource) is made up of a scheme (https), an authority (www.w3.org), a path (/TR/webarch) and a fragment (#def-resource)
3. **HTTP Methods**: REST uses standard HTTP methods like GET, POST, PUT, DELETE, etc., to perform operations on resources.
4. **Stateless**: Each request from a client to a server must contain all the information necessary (metadata) to understand and process the request. The server should not store anything about the latest request the client made.
5. **Representation**: Resources are decoupled from their representation so that their content can be accessed in a variety of formats, like HTML, XML, plain text, PDF, JPEG, JSON, and others.

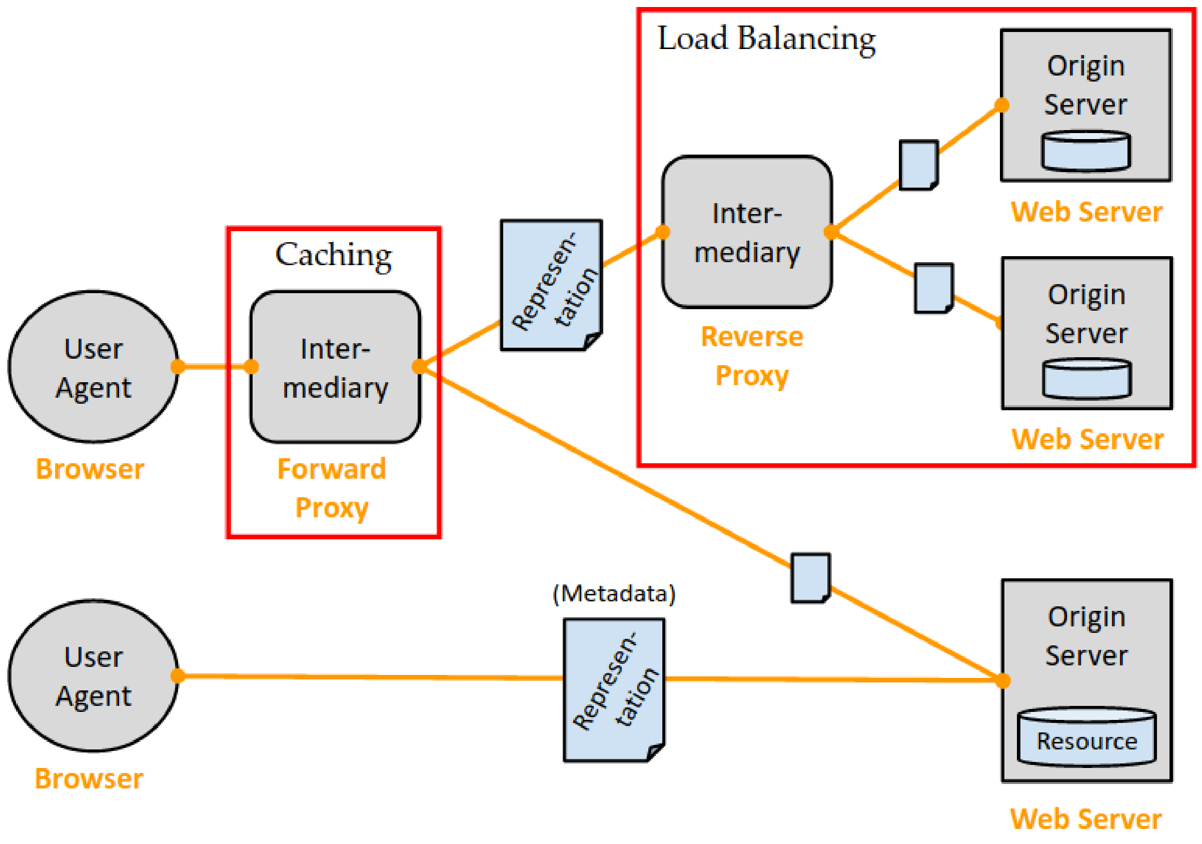

Figure 3: REST based Web

1.5.2 Relationship Between REST and Web Architecture

REST was defined by Roy Fielding, one of the principal authors of the HTTP specification, in his Ph.D. dissertation. It is a way to design networked applications that takes advantage of the protocols and conventions of the web. Essentially, REST is a set of principles that can be used to design systems that scale, are reliable and performant, by leveraging the existing infrastructure of the web.

1.6 HTTP

HTTP is a text-based protocol to transmit text message. Messages can contain resource representations encoded in a variety of formats, such as text/html, image/png, application/json. Most HTTP messages are therefore human-readable, which makes HTTP easy to debug, but not very efficient.

1.6.1 HTTP 1.1

HTTP 1.1 is a revision of the original HTTP (HTTP 1.0). Introduced in 1997, it added a few significant improvements and capabilities over its predecessor.

1.6.1.1 Key Features

1. **Persistence**: Unlike HTTP 1.0, where every request needed a new TCP connection, HTTP 1.1 introduced persistent connections. This feature allows multiple requests and responses to be sent over the same TCP connection, reducing the overhead of establishing a new connection for each request.
2. **Chunked Responses**: HTTP 1.1 allows servers to send responses in chunks, enabling the client to start processing partial data while the server prepares the rest of the response.
3. **Additional Cache Control Mechanisms**: HTTP 1.1 introduced more granular cache controls, improving performance and efficiency in the HTTP communications.
4. **Content Negotiation**: Clients can use HTTP 1.1 to tell the server what type of data they can understand (like HTML, XML, etc.) and the server can then send the best matching type back to the client.

1.6.2 HTTP 2

HTTP/2 is the second major version of the HTTP protocol. It was developed by the HTTP Working Group of the Internet Engineering Task Force (IETF) and was published in 2015.

1.6.2.1 Key Features

1. **Multiplexing**: This feature allows multiple requests and responses to be in flight at the same time on a single TCP connection. This is a significant improvement over HTTP 1.1, where only one response could be delivered at a time (head-of-line blocking).
2. **Pipelining**: This allows clients and servers to send requests without waiting for answer.
3. **Header Compression**: HTTP/2 uses HPACK compression, which reduces overhead and improves performance by compressing request and response header metadata.
4. **Server Push**: This feature allows a server to send resources proactively to the client's cache for future use, thus potentially reducing the latency of subsequent requests.

1.6.3 Comparison Between HTTP 1.1 and HTTP 2

HTTP 2 introduces significant improvements over HTTP 1.1 in terms of performance, efficiency, and security. It was developed to overcome the limitations and inefficiencies of HTTP 1.1, especially in the context of the modern, complex web applications. Despite this, HTTP 1.1 is still widely used and remains a relevant and important part of the web's architecture.

1.6.4 HTTP 3

HTTP/3 integrates with QUIC and makes the switch to UDP.

1.7 APIs

According to RedHat an application programming interface (API) is a set of tools, definitions, and protocols for integrating application software and services. It's the stuff that lets your products and services communicate with other products and services without having to constantly build new connectivity infrastructure.

The basis for APIs is object-oriented programming and data hiding (the client does not care how the server works internally).

2 Physical Layer

The physical layer of a communication system is responsible for transmitting bits from the sender to the receiver. This can be achieved through wired or wireless carriers.

2.1 Wired Carriers

Wired communication is often considered the most reliable and fast form of communication.

Fiber-optic communication is the fastest type of wired communication, capable of transmitting information through pulses of lasers at speeds of 30 to 40 Gbit/s.

However, the cost of fiber optics makes it impractical for connecting devices within a network. As an alternative, IEEE 802.3 Ethernet is used, which carries a voltage of up to 13V on the wire. Data rates for this typically range from 100 Mbit/s to 1 Gbit/s.

For sensors that require less bandwidth, the 1-wire protocol is used. It carries a voltage of either 3.3V or 5V and has a bandwidth of 16 kbit/s.

2.2 Wireless Carriers

Wireless carriers use electromagnetic waves to transmit information without wires.

NEC Infrared, for example, uses infrared light that has a wavelength of 870nm and 930-950nm, which is used in TV remotes.

For faster wireless communication, wireless radio is used. Bluetooth Low Energy operates in the 2.4-2.4835 GHz ISM band and uses 40 channels of 2 MHz width each. IEEE 802.11 (Wifi) is more complex and uses channels in various bands including the 2.4GHz ISM (Industrial, Scientific, Medical) band, 3.65GHz, 4.9 GHz, 5.0 GHz, 5.9 GHz, 60 GHz, 900 MHz, and 54-790 MHz bands.

The higher the frequency, the faster the data can be transferred. However, shorter wavelengths make the signal easier to block. This is why 5G signals have a shorter range.

3 Data Link Layer

3.1 Medium Access Control (MAC)

The issue with communication thorugh a shared channel (= collision domain) is that data from different clients might overlap. To make sure the data ends up at the correct clients, every device gets assigned a unique address.

Medium Access Control (MAC) addresses are unique identifiers assigned to device network interface controllers (NICs). This implies that a device can (and typically does) have multiple MAC addresses.

MAC addresses are assigned by device manufacturers as Universally Administered Addresses (UAA). However, MAC addresses can typically be changed. In this case, they are flagged as Locally Administered Address (LAA) by setting the second-least significant bit of the first octet to 1

On the Data Link Layer the correct routing to all NICs within a broadcast domain (= a link-layer network segment) is handeled with MAC addresses. All computers require MAC addresses for addressing within broadcast domains.

The issue with unique identifiers is that they can be tracked. Therefore, all modern devices randomize their MAC addresses during network scanning. (However, this is not enough to avoid tracking / fingerprinting.)

3.2 Address Resolution Protocol (ARP)

ARP is a protocol to discover link-layer addresses from network- layer addresses. Since it establishes the connection between the link layer and the network layer, it is a critical protocol of the Internet.

The addresses are saved (for a specific amount time) in a table, called the ARP table.

4 Network Layer

The network layer is responsible for managing the routing and forwarding of data packets between and within networks. The primary goal of the network layer is to enable communication between devices that are not necessarily on the same local network.

4.1 Internet Protocol (IP)

The Internet Protocol (IP) is a principal communication protocol used for relaying datagrams (packets) across network boundaries and to address devices. It enables internetworking, forming the backbone of the Internet. IP has two primary versions: IPv4 and IPv6.

IPv4:

- Addressing: 32-bit address space, providing around 4.3 billion unique addresses
- Header: 20-60 bytes in length, with a variable header length
- The checksum of an IPv4 packet has to be recalculated every time it's routed because the Time To Live (TTL) is reset. The checksum is the 16-bit one's complement sum of all 16-bit words in the header

IPv6:

- Addressing: 128-bit address space, providing a vast number of unique addresses
- Header: Fixed length of 40 bytes

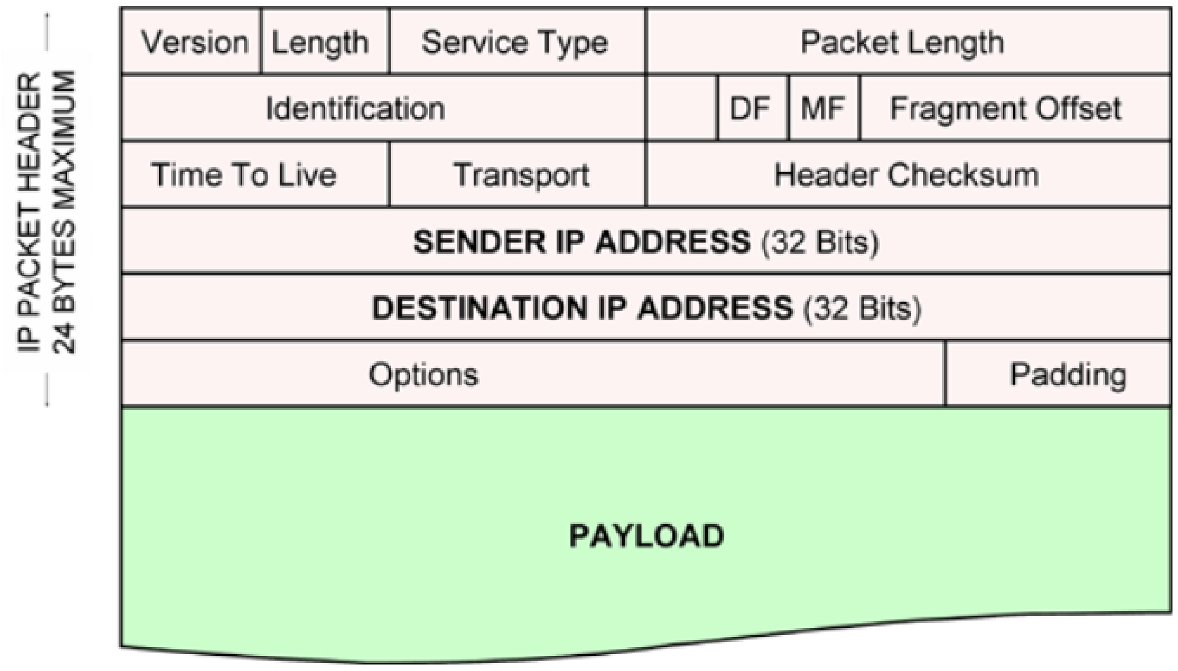

Figure 4: IPv4 datagram (Identification: Identify multiple fragments of one IP datagram, DF: Don't Fragment, MF: More Fragments)

- Addressing Modes: Unicast (the address identifies an individual recipient), anycast (the address identifies a group of recipients where any of these should receive the message), and multicast (the address identifies a group of recipients where each recipient should receive the message)

IP is so essential for networking that making changes is very difficult. This is demonstrated by the decade-long transition from IPv4 to IPv6.

4.1.1 Subnet mask

Part of the 32 bits identify the network while the other part identifies the host interface (i.e., the computer's network interface). The amount of bits used to identify the network is flexible and defined by the subnet mask. The subnet mask is 32 bits long and all bits that are used for network identification are marked by a 1. The more bits that are used by network identification, the less bits are available to device identifaction, therefore less devices can connect to that network.

The device IP with all 0 is not allowed and the device IP with all 1 is the broadcast IP address.

A typical network is given by this: 172.16.0.0/24

The first part, 172.16.0.0 is the network ID. The /24 means the subnet mask has 24 bits allocated to network identification. So the subnet mask is 255.255.255.0 or 11111111.11111111.11111111.00000000. The amount of clients that can connect is therefore 2^8 - 2 = 254.

5 Transport Layer

The transport layer is the fourth layer in the OSI (Open Systems Interconnection) model and is responsible for facilitating reliable, efficient, and orderly communication between devices on a network. It ensures that data is transmitted correctly and in sequence, while also providing error checking and flow control mechanisms.

5.1 Key Responsibilities

1. **Connection Establishment & Termination**: The transport layer establishes and terminates connections between devices, ensuring that data transmission begins and ends appropriately.
2. **Data Segmentation & Reassembly**: It breaks down large data chunks into smaller packets for transmission and reassembles them at the destination.
3. **Flow Control**: The transport layer manages the rate at which data is sent and received to prevent data overflow, ensuring smooth and efficient communication.

4. **Error Checking**: It checks for errors in the transmitted data, such as lost or duplicated packets, and ensures that they are corrected or retransmitted.
5. **Reliability**: The transport layer provides end-to-end communication, offering optional reliability features to guarantee that data arrives at its destination intact and in the correct order.

5.2 Protocols

5.2.1 TCP (Transmission Control Protocol)

TCP is a widely used transport layer protocol, known for its reliability and connection-oriented nature. It ensures that data is transmitted and received in the correct order, without errors. Here's an in-depth look at its main features:

1. **Connection-Oriented**: TCP establishes a virtual connection between the sender and receiver before data transmission begins. It uses a three-way handshake process (SYN, SYN-ACK, ACK) to set up the connection and a four-way handshake process (FIN, ACK, FIN, ACK) to close it.
2. **Reliable Data Transfer**: TCP uses sequence numbers and acknowledgments to ensure that data is received correctly and in the correct order. If any data is lost or corrupted, the receiver requests retransmission, ensuring that the data arrives intact.
3. **Flow Control**: To prevent data overflow, TCP uses flow control mechanisms like sliding windows. The receiver specifies the window size, which indicates the number of bytes it can accept without being overwhelmed. The sender then adjusts its data transmission rate accordingly.
4. **Error Detection**: TCP uses checksums to detect errors in the transmitted data. If an error is detected, the receiver discards the corrupted data and requests retransmission.
5. **Congestion Control**: To avoid network congestion, TCP employs congestion control algorithms like slow start, congestion avoidance, and fast retransmit. These mechanisms help regulate the data transmission rate and ensure efficient use of network resources.

5.2.1.1 Handshake TCP uses a three-way handshake to establish a connection between two hosts. The three steps are:

1. **SYN (Synchronize sequence numbers):** The client sends a SYN packet to the server to initiate a connection. This packet includes an initial sequence number (ISN) for the client.
2. **SYN-ACK (Synchronize-Acknowledgement):** In response, the server sends back a SYN-ACK packet. This packet includes the server's own ISN, as well as an acknowledgement number, which is the client's ISN plus one.

3. **ACK (Acknowledgement):** Finally, the client sends an ACK packet back to the server. The acknowledgement number in this packet is the server's ISN plus one.

The handshake looks like this:

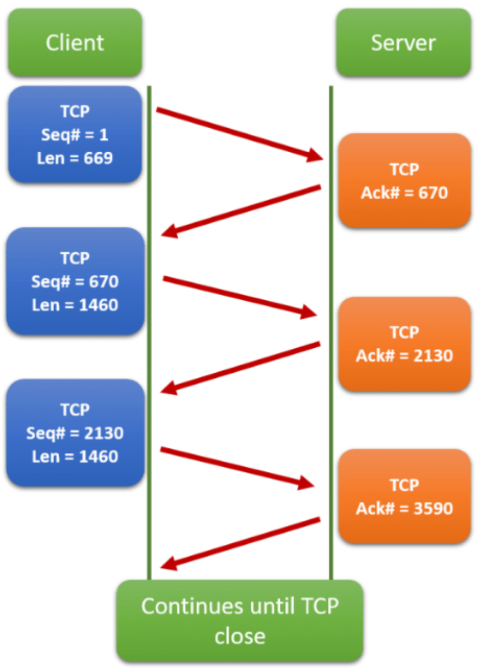

Figure 5: TCP Handshake

5.2.2 UDP (User Datagram Protocol)

UDP is a simpler and faster transport layer protocol compared to TCP. It is connectionless and does not guarantee reliability, making it suitable for applications that prioritize speed over error-free delivery. Here's a closer look at its main features:

1. **Connectionless**: UDP does not establish a connection before data transmission. Instead, it sends datagrams directly to the recipient, without any prior communication.
2. **Fast Data Transfer**: Since UDP does not use connection establishment, flow control, or error detection mechanisms, it can transmit data faster than TCP. However, this comes at the cost of potentially losing, duplicating, or reordering data.
3. **Low Overhead**: UDP has a simpler header structure than TCP, resulting in lower overhead and reduced resource consumption. This makes UDP more suitable for applications with limited bandwidth or computational power.

4. **No Guarantee of Delivery**: Unlike TCP, UDP does not guarantee that data will be delivered to the destination, nor does it guarantee the order of delivery. Applications using UDP must be able to handle data loss, duplication, or reordering.
5. **Multicast & Broadcast Support**: UDP supports multicast (one-to-many) and broadcast (one-to-all) communication, making it suitable for applications that require data to be sent to multiple recipients simultaneously, such as video conferencing or online gaming.

5.2.3 QUIC (Quick UDP Internet Connections)

QUIC is a relatively new transport layer protocol developed by Google to improve the performance of web applications by reducing latency and enhancing security. It combines the best features of TCP and UDP while addressing some of their limitations. Here's a closer look at QUIC's main features:

1. **Built on UDP**: QUIC is built on top of UDP, inheriting its connectionless nature, low overhead, and multicast support. This enables QUIC to provide faster data transfer compared to TCP.
2. **Connection Establishment**: QUIC significantly reduces connection establishment time by combining the transport layer and the cryptographic handshake in a single step. This feature, known as "0-RTT (Zero Round Trip Time) connection establishment," allows data to be transmitted immediately, reducing latency in web applications.
3. **Multiplexed Streams**: QUIC supports multiplexed streams within a single connection, allowing multiple streams of data to be transmitted simultaneously without head-of-line blocking. This feature helps overcome one of TCP's limitations, where a single lost packet can stall the transmission of all streams.
4. **Forward Error Correction**: QUIC uses forward error correction (FEC) to improve data transmission reliability. FEC allows the receiver to recover lost or corrupted data without the need for retransmission, reducing latency and improving performance.
5. **Congestion Control**: QUIC incorporates congestion control mechanisms similar to those used in TCP, such as slow start and congestion avoidance, to manage network congestion and ensure efficient data transmission.
6. **Built-in Encryption**: QUIC incorporates built-in encryption using the TLS (Transport Layer Security) protocol, making it more secure than both TCP and UDP. This feature enhances security and privacy, particularly for web applications.

6 Application Layer

The application layer is the seventh and topmost layer of the OSI (Open Systems Interconnection) model, which is responsible for providing user interfaces and facilitating communication between applications and lower-level network services. It acts as the interface between users and the underlying network, enabling data exchange between software applications and networked devices.

6.1 Key Functions

1. **User Interface**: The application layer provides the interface for users to interact with software applications, enabling data input and output, as well as presentation and visualization of data.
2. **Data Exchange**: It facilitates communication between applications on different devices, ensuring that data is correctly formatted, transmitted, and received.
3. **Protocol Implementation**: The application layer contains protocols and services that support various application-specific tasks, such as file transfer, email, and web browsing.
4. **Resource Sharing**: It enables resource sharing among multiple applications, allowing users to access and manage shared resources like printers, files, and databases across a network.
5. **Application Services**: The application layer offers a range of services, such as authentication, data synchronization, and data compression, to enhance application performance and user experience.

6.2 Common Protocols

Several protocols operate at the application layer, catering to various application-specific requirements. Some of the most widely used protocols include:

1. **HTTP (Hypertext Transfer Protocol)**: HTTP is the foundation of the World Wide Web, enabling communication between web browsers and web servers. It facilitates the transfer of web pages, images, and other resources over the internet.
2. **HTTPS (Hypertext Transfer Protocol Secure)**: HTTPS is a secure version of HTTP that uses encryption (SSL/TLS) to protect data exchanged between web browsers and web servers, ensuring privacy and integrity.
3. **FTP (File Transfer Protocol)**: FTP is a protocol used to transfer files between client and server devices over a network, allowing users to upload, download, and manage files.

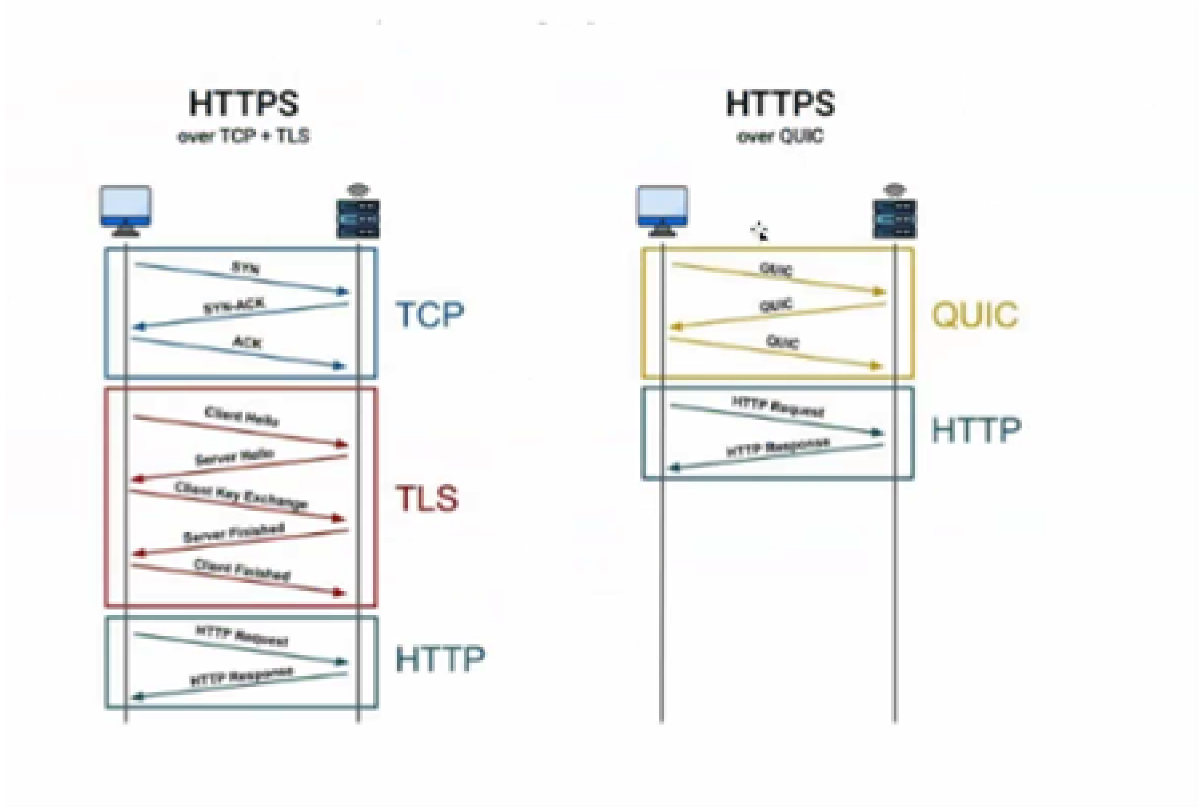

Figure 6: HTTPS Handshake over TCP/TLS and QUIC

4. **SMTP (Simple Mail Transfer Protocol)**: SMTP is an email transmission protocol that enables the sending of email messages from a client to a server and between email servers.
5. **IMAP (Internet Message Access Protocol)**: IMAP is an email retrieval protocol that allows users to access and manage their email messages stored on a remote mail server.
6. **DNS (Domain Name System)**: DNS is a hierarchical naming system that translates human-readable domain names (e.g., www.example.com) into IP addresses, enabling devices to locate and connect with each other on the internet.

7 Network Security

To make a system secure, the following points have to be considered:

- **Design objective**: What is your system intended to do? (ex.: Only Alica has access to her mailbox)
- **Threat model**: Assumptions about the adversary (ex.: An attacker can try to guess passwords)
- **Policy**: Rules that ensure your system will achieve the design objective (ex.: Require a password)
- **Mechanism**: Software/hardware that your system uses to enforc the policy (ex.: Use user accounts and password with 8 or more characters containing a mix of letters, numbers and symbols)

In case of a network, the points are:

- **Design objective**: Machines exchange packets without eavesdropping or tampering (confidentiality and integrity)
- **Threat model**: An attacker can observe packets, send arbitrary packets, modify packets in transit, etc.
- **Policy**: Require end-to-end encryption
- **Mechanism**: Use a secure channel over TLS

7.1 Caesar's Cipher

To encrypt text using Caesar's Cipher, shift each letter by x in the alphabet. For example: HELLO WORLD -> EBIIL TLOIA (shift of 3).

Breaking this encryption is pretty easy, as there are only 26 letters in the alphabet, and therefore only 25 possible shifts.

Another method is frequency analyzing. The frequency of letters doesn't change after applying the encryption. So by analyzing the shift in letter frequency, we can calculate the shift used by the encryption.

It is very clear, that Caeser's Cipher is not a good encryption method because there aren't many possibilities and the encrypted version leaks a lot of information. Ideally, you want to the exact opposite.

7.2 Symmetric Key Cryptography

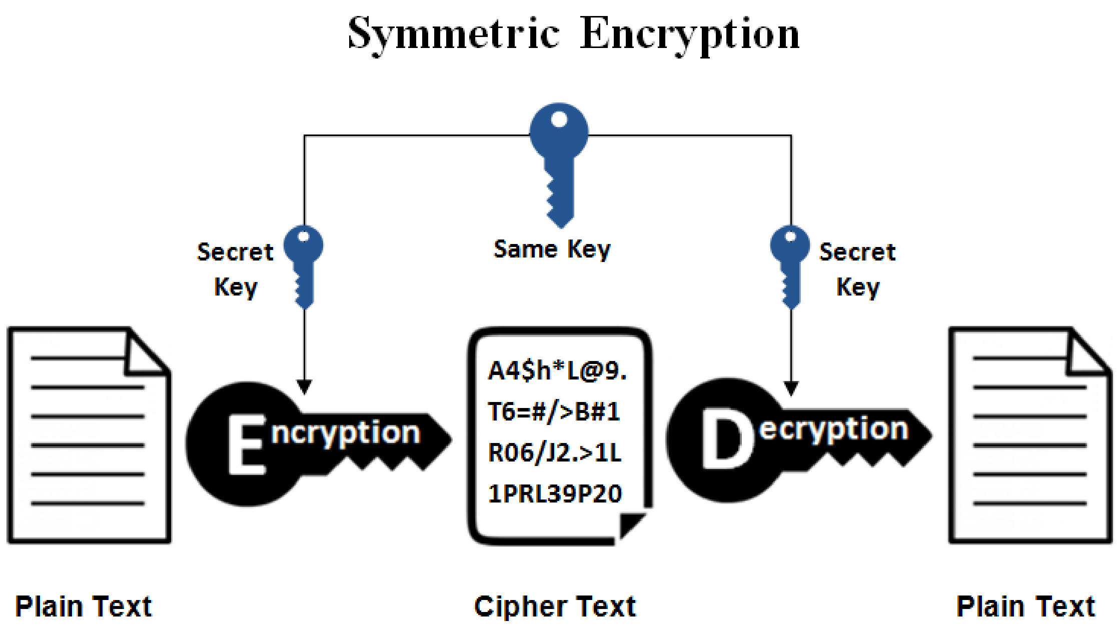

Figure 7: Symmetric Key Cryptography

Symmetric Key Cryptography, also known as secret key cryptography, involves the use of the same key for both the encryption and decryption of data. This key is known only to the sender and the receiver and it must be kept secret to ensure the confidentiality of the data.

7.2.1 Advantages

- Faster than asymmetric encryption.
- Requires less computational resources.

7.2.2 Disadvantages

- Key distribution: Securely distributing the key to both parties can be challenging.
- Scalability: In a network of n users, each pair of users needs a unique key, leading to $n(n-1)/2$ keys.

7.2.3 AES (Advanced Encryption Standard)

AES is a widely used symmetric encryption algorithm. It was established by the U.S. National Institute of Standards and Technology (NIST) in 2001.

- It supports key sizes of 128, 192, and 256 bits.
- It uses a series of transformations such as substitution, permutation, mixing, and shifting.
- It is secure, fast, and efficient, and is used widely in many types of data encryption.

7.3 Diffie-Hellman Key Exchange

Diffie-Hellman (DH) Key Exchange is a method of securely exchanging cryptographic keys over a public network. The protocol allows two users to each generate a public-private key pair and then share their public keys. Each user can then combine their own private key with the other user's public key to create a shared secret.

7.3.1 Advantages

- Key distribution: The key exchange protocol provides a solution to the problem of distributing keys in symmetric cryptography.
- Perfect forward secrecy: New keys are generated for each session, so the compromise of a single key does not affect past or future sessions.

7.3.2 Disadvantages

- Susceptible to man-in-the-middle attacks: Without a way to verify the authenticity of the public key, an attacker could intercept and replace the public key with their own.

7.3.3 Diffie-Hellman Key Exchange Process

1. **Public Information**: Both users agree on a large prime number p and a base g (also known as a generator).
2. **Private Computation**: Each user chooses a secret value (a and b) and calculates a public value (A and B) using the formula $A = g^a \mod p$ and $B = g^b \mod p$.
3. **Exchange of Public Values**: The users exchange their public values A and B.
4. **Shared Secret Calculation**: Each user calculates the shared secret using the formula $s = B^a \mod p$ or $s = A^b \mod p$. Both formulas will result in the same value s.

The security of Diffie-Hellman relies on the difficulty of the Discrete Logarithm Problem.

7.4 Assymmetric (Public) Key Cryptography

Asymmetric Key Cryptography, also known as public key cryptography, involves the use of a pair of keys: a public key, which may be known by anyone, and a private key that is known only to the user. The public key is used to encrypt data, while the private key is used to decrypt it.

7.4.1 Advantages

- Key distribution: Since the public key can be distributed openly, the challenge of secure key distribution present in symmetric cryptography is solved.
- Digital signatures: Asymmetric cryptography can be used to create digital signatures, which provide non-repudiation and integrity checks.

7.4.2 Disadvantages

- Performance: Asymmetric encryption is computationally intensive and slower than symmetric encryption.
- Key size: Asymmetric algorithms require longer key lengths for the same level of security.

7.4.3 RSA (Rivest-Shamir-Adleman)

RSA is one of the first practical public-key cryptosystems and is widely used for secure data transmission. It is based on the fact that factoring large numbers is computationally difficult.

- Data is encrypted with the public key, but can only be decrypted using the private key
- Data signed with a secret key can be verified with the public key
- RSA is a block cipher. The block size is equal to the key size (1024, 2048 or 4096 bits)

7.4.4 RSA Process

See discrete math lecture.

The security of RSA relies on the difficulty of factoring large prime numbers. However, advancements in quantum computing could potentially break RSA in the future.

7.5 TLS

7.5.1 TLS with Diffie-Hellman Key Exchange.

There are 3 message types:

- **Hello messages**: used to agree on parameters for the TLS connection (session identifier, supported ciphers, etc.)
- **Key Exchange messages**: used to establish a shared key K with variations of the Diffie-Hellman algorithm
- **Finished messages** (encrypted with K): once these messages are received and validated, start sending application data. The key K is refreshed for each session to ensure forward secrecy (= a future adversary cannot decrypt old messages).

However, this doesn't solve the **man-in-the-middle attack**:

- an adversary intercepts the communication and responds to the client
- the adversary can now receive the client's Diffie-Hellman parameters and can establish a connection to the server as well
- the adversary can then continue to relay - and tamper with - messages between the client and the server

Therefore, we introduce **Certificate Authorities** (CA), which issue a certificate for the server, to bind the server's public key to the server's name. The certificate issued by the CA is signed with the CA's secret key (`SK_CA`), and can be verified with the CA's public key. Certificates are issued, managed, used, revoked through a **Public Key Infrastructure** (a hierarchical chain of trust). The TLS handshake now looks like this:

7.5.2 TLS with RSA

There are again 3 message types:

- Hello messages: used to agree on parameters for the TLS connection (session identifier, supported ciphers, etc.)
- Client Key Exchange message: sends a 384-bit Secret that is encrypted with PK and used to establish symmetric key K
- Finished messages (encrypted with K): once these messages are received and validated, start sending application data

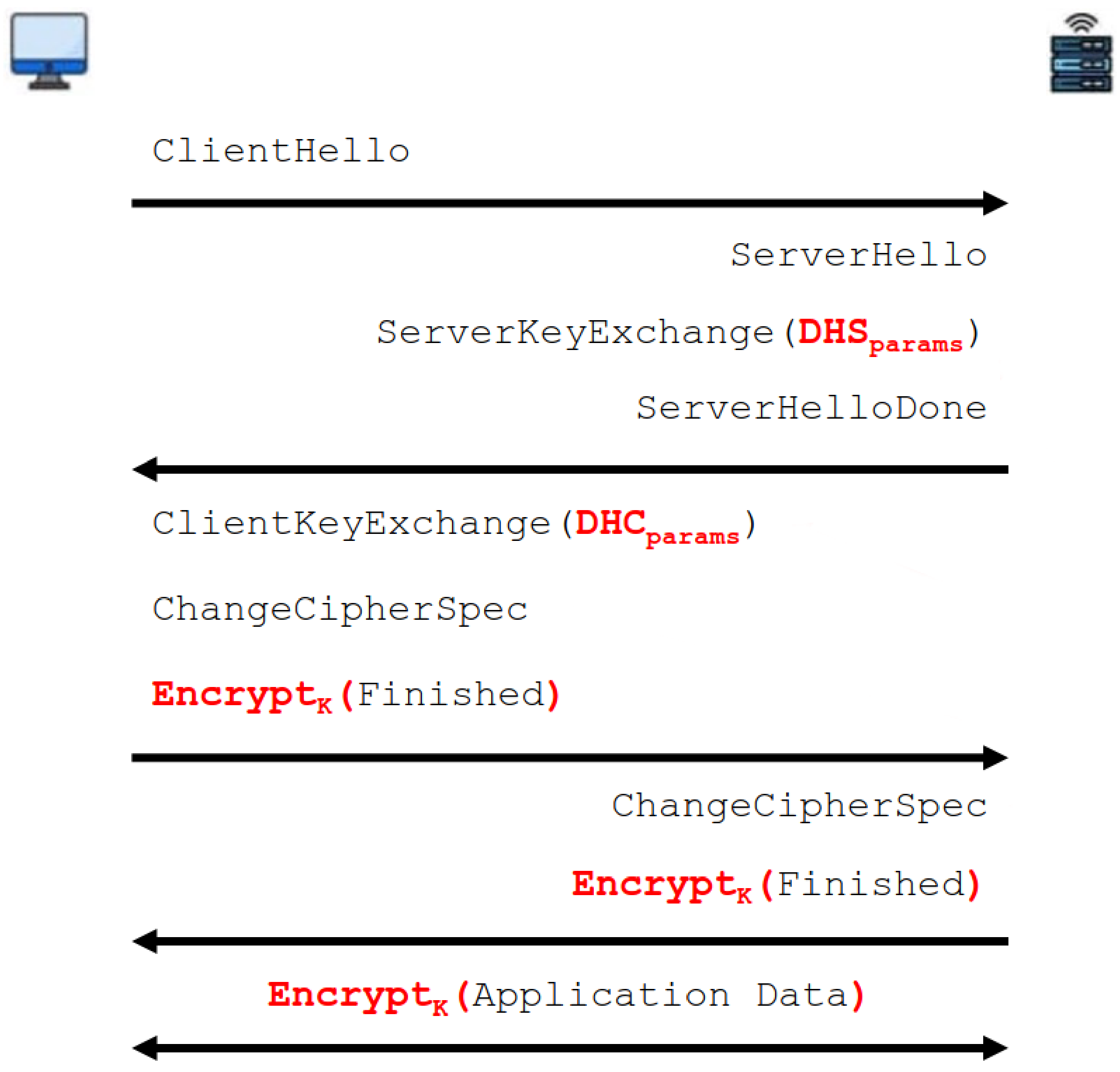

Figure 8: TLS handshake using Diffie-Hellman Key Exchange

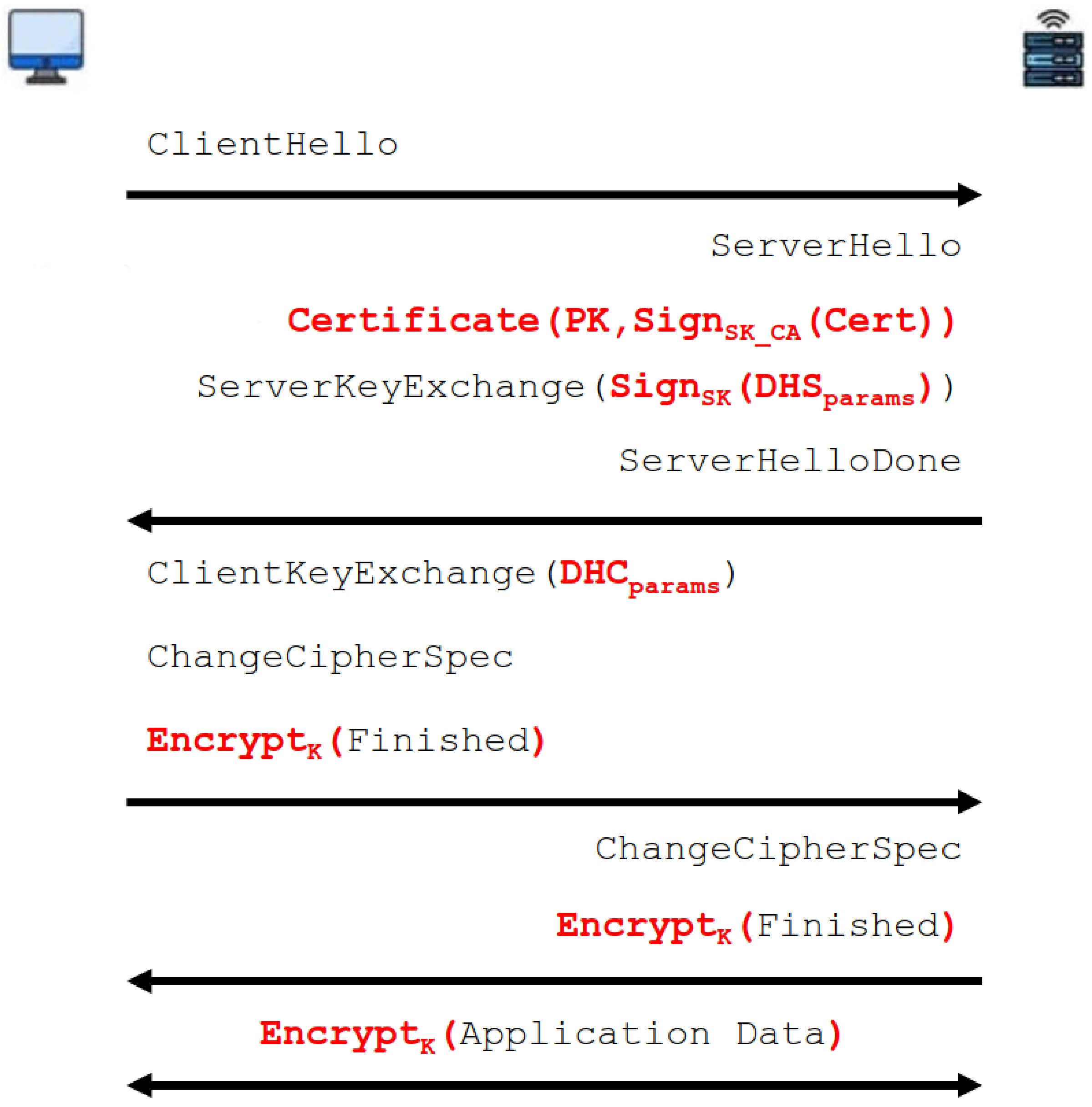
ClientHello
ServerHello
Certificate(PK,Sign_SK_CA(Cert))
ServerKeyExchange(Sign_SK(DHS_params))
ServerHelloDone
ClientKeyExchange(DHC_params)
ChangeCipherSpec
Encrypt_K(Finished)
ChangeCipherSpec
Encrypt_K(Finished)
Encrypt_K(Application Data)

Figure 9: TLS handshake using Diffie-Hellman Key Exchange and Certificates

8 Constrained Networking

8.1 Constrained-Nodes Classes

The IETF terminiology for constrained-node networks provides 3 class to differentiate the capabilities of devices.

These devices usually operate under the IEEE 802.15.4 standard, which is an important standard for LoWPANs (for example in home networking, industrial control, etc.).

These devices usually use the Thread stack, which uses 802.15.4 for the lower layers, incorperates IPv6 with 6LoWPAN, uses IP routing for the network layer, UDP for the transport layer (to reduce overhead), and DTLS for security.

There are many different options for the MAC layers, including

- Classic CSMA: Always-on radio for high-bandwith
- 802.15.4 + ContikiMAC: Asynchronous Low-Power Listening (LPL)
- 802.15.4 + TSCH: Time Slooted Chanel Hopping (TSCH)

8.1.1 Class 0

Class 0 devices are too small to securly run on the internet. They usually run a proprietary and minimalistic protocol to connect to the Internet. They also require an application-layer gateway for protocol translation, neighbor discovery, security, etc.

- Data: 10 KiB
- Code: 48 KiB

8.1.2 Class 1

Class 1 devices are capable of connecting to the Internet with on-board security. However, they are still to constrained to run HTTP with TLS.

- Data: 10 KiB
- Code: 100 KiB

8.1.3 Class 2

Class 2 devices are close to full-fledged Internet nodes, like smartphones. However, they can still benifit from lightweight protocols to free up resources and reduce operational costs.

- Data: 50 KiB
- Code: 250 KiB

8.2 6LoWPAN

8.2.1 Challenges

There are many challenges for IP-based low-power and lossy wireless networks:

- Duty cycling: IP assumes that devices are always connected, therefore, they have to active for a percentage of time
- Reliability: Standard Internet protocols are not optimized for low-power and lossy wireless networks
- Low data rate and small frame size: Data rate is usually between 20-250kbps, and the frame size is between 40-200 bytes (802.15.4 has a frame size of 127 bytes)
- No native multicast support: 802.15.4 does not support multicast
- Mesh technologies: multi-hop mesh networks are sometimes required for coverage and efficiency, however, IP routing is not easily applicable to such networks

8.2.2 Architecture

A LoWPAN is a stub network, meaning a collection of LoWPAN nodes share a common IPv6 prefix. There are three types of LoWPANs:

- **Simple LoWPAN:** A single edge router, which connects the LoWPAN to the Internet.
- **Extended LoWPAN:** Multiple edge routers with a common backbone link
- **Ad-hoc LoWPAN:** No edge router at all and therefore no route outside the LoWPAN

8.2.2.1 Header Compression A LoWPAN is made up of a single IPv6 subnet with a unique MAC address. Inside the LoWPAN, a compressed version of IPv6 is used. This IPv6 header compression is lossless and applied/reversed at the edge.

The compression allows UDP/IPv6 headers (48 bytes) down to 6 bytes in the best case.

8.2.2.2 Fragmentation IPv6 requires transmissions of at least 1280 Bytes. That's a lot for constrained devices, therefore the IPv6 payload gets fragmented into smaller chunks. This allow sthe IPv6 datagrams to fit small frame sizes (127 bytes for 802.15.4)

8.2.2.3 Neighbor Discovery IPv6 uses different kinds of messages to discover neigbors (neighbor soliciation, neighbor advertisement, router advertisement, router solicitation) and they all rely on multicast. This is not an option for LoWPAN. So how does it work?

In a LoWPAN network, the task of neighbor discovery is handeled by the edge routers because they are less constrained. The edge routers maintain a list of nodes in the LoWPAN, called a whiteboard (in extended LoWPANs, the edge routers synchronize their whiteboards over the backbone link).

There are two types of messages:

- Node Registration (NR)
- Node Confirmation

Those messages allow neighbor discovery over a multi-hop network (each node inside the LoWPAN forwards the messages).

8.2.2.4 Mobility LoWPANs allow for nodes to change their LoWPAN network (IPv6 changes) and move within the same network (for example connecting to another edge router within the same extended LoWPAN).

The edge routers can also change their point of attachment to the Internet.

8.2.2.5 Routing 6LoWPAN uses the Routing Protocol for Low-power and lossy network (RPL). RPL is specialized for LoWPANs and routes packets based on a **directed acyclic graph (DAG)** oriented towards the root:

- The **rank** characterizes the distance and is used to avoid loops
- Leaf nodes and intermediary nodes can have **multiple parents**
- Each node has one **preferred parent**
- The DAG can be distributed across nodes (**storing mode**) or can be stored only in the root (**non-storing mode**)

Redundancies in the rooted DAG help to cope with dynamic topologies and lossy links.

DAG is optimized for the most common packet flow: Packets flowing towards the root (edge router), and from the root towards leaf nodes.

8.2.3 Application Layer: CoAP

the 6LoWPAN stack allows for any application layer, such as Dotdot over Thread. The Web of Things (WoT) fosters interoperability and lowers the entry-barrier for developers. However, we can’t use HTTP

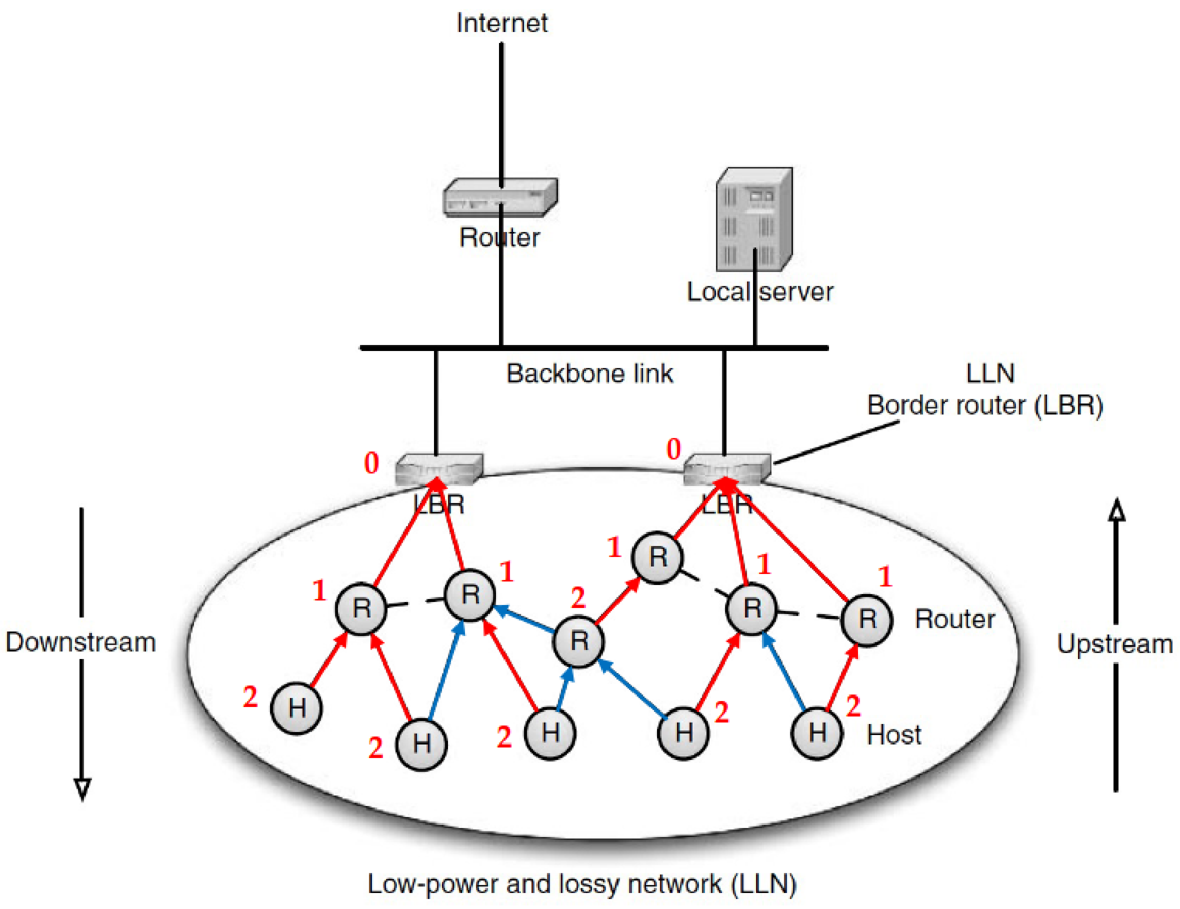

Figure 10: 6LoWPAN Routing

because of it needs a reliable connection. Therefore, a new standard, the Constrained Application Protocol (CoAP), was created. CoAP targets constrained nodes and networks:

- Optimized for bandwith and processing efficiency
- Better for near real time: No three-way handshake
- No connection management overhead
- Smaller headers

CoAP properties:

- Efficient REST uniform interace on top of UDP
- Lightweioght security to prevent eavesdropping, tampering, forgery, …
- Other IoT-specific features: Multicast & Observe
- Easy to proxy from/into HTTP (but don't forget that CoAP is a complete redesign of HTTP)

8.2.3.1 Observing CoAP extends the REST uniform interface to permit observing resources. Clients may retrieve a representation of a resource and keep this representation updated by the server (sensor) over a period of time (contrary to repeated HTTP polling).

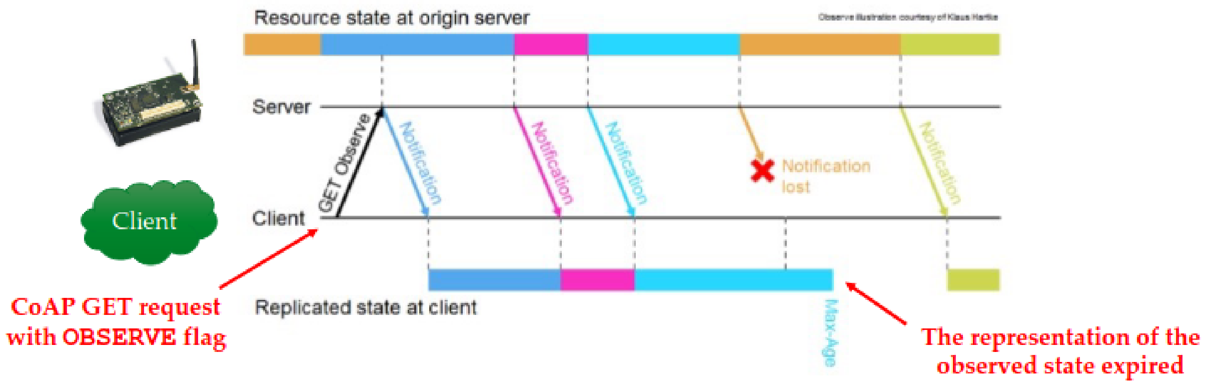
Resource state at origin server
Observe illustration courtesy of Klaus Hartke
Server
Client
Client
GET Observe
Notification
Notification
Notification
Notification lost
Notification
Replicated state at client
Max-Age
CoAP GET request with OBSERVE flag
The representation of the observed state expired

Figure 11: CoAP observe

HTTP polling creates more traffic and if the client is not polling the server often enough, the client might miss events.

8.2.3.2 Reliability CoAP has four message types:

- **Confirmable (CON):**: Reliable message with retransmissions
- **Non-Confirmable (NON)**: Best effort transmission
- **Acknowledgement (ACK)**: Confirms a CON message (compulsory)

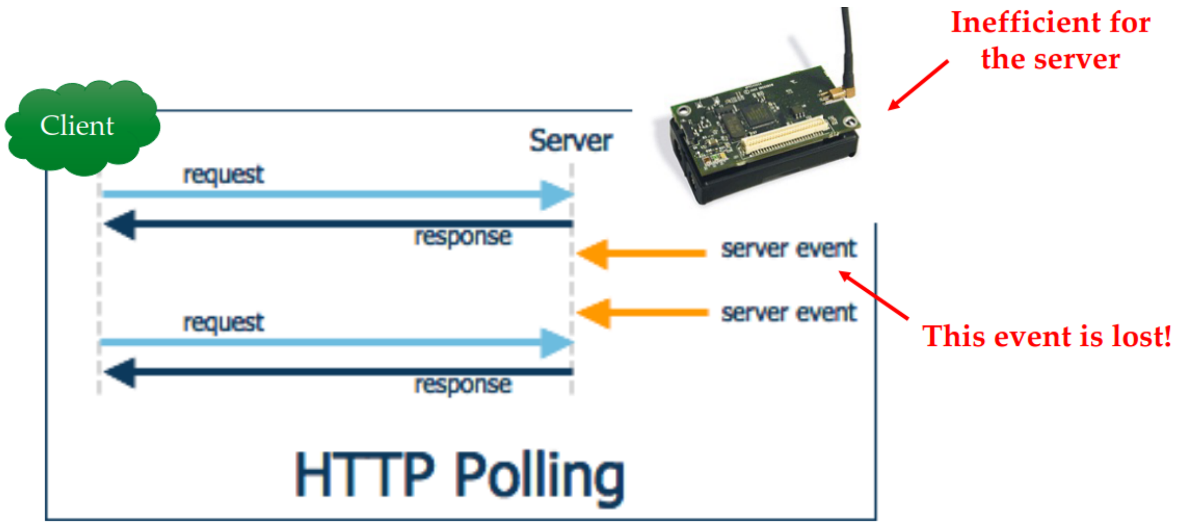

Figure 12: HTTP polling caveats

- **Reset (RST)**: Used when CON (or NON) cannot be processed

8.2.4 IETF Constrained RESTful Environments (CoRE)

The Internet Engineering Task Force (IETF) CoRE group is responsible for defining and unifying standard for constrained environments.

The CoRE architecture is the main framework used for constrained devices. Its key components are:

- Constrained Application Protocol (CoAP)
- Resource Directory (RD): The RD provides a way for devices to register their resources and services, and for other devices to discover these services. The RD serves as a dynamic library for available APIs, which is crucial for constrained networks because they change often.
- Link Format: A CoAP server can use the CORE Link Format to advertise the ressources it exposes to other devices in the network. The resources are advertised with multicast and stored in the RD. This means, CoAP, like HTML, hypermedia-driven discovery.

Example CoRE Link Format:

```
GET /.well-known/core
    </config/groups>;rt="core.gp";ct=50,
    </sensors/temp>;rt="ucum.Cel";ct="0 50";obs,
    </large>;rt="block";sz=1280,
    </device>;title="Device management"
```

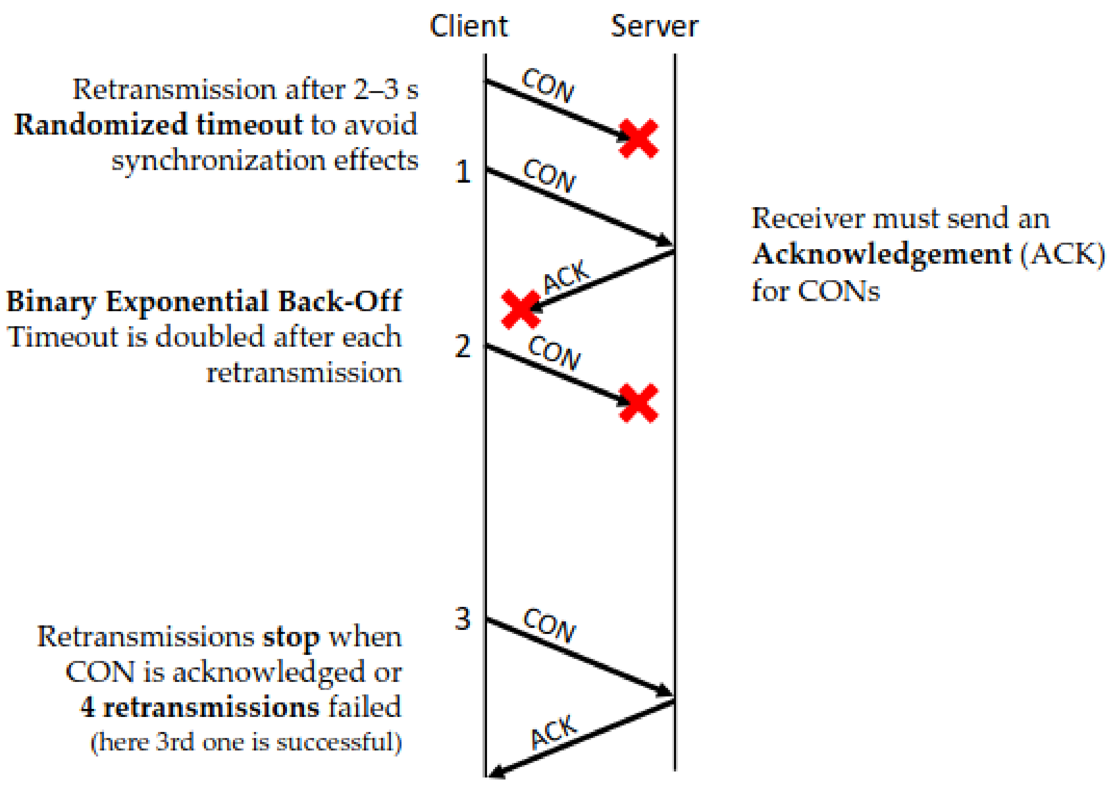
Client
Server
CON
CON
ACK
CON
CON
ACK
1
2
3
Retransmission after 2–3 s
Randomized timeout to avoid synchronization effects
Binary Exponential Back-Off
Timeout is doubled after each retransmission
Retransmissions **stop** when CON is acknowledged or **4 retransmissions** failed
(here 3rd one is successful)
Receiver must send an **Acknowledgement** (ACK) for CONs

Figure 13: CoAP reliability