WWU MÜNSTER

ERCIS

The Internet

(Usage hints for this presentation)

Summer Term 2023

Dr. Jens Lechtenbörger (License Information)

Data Science: Machine Learning and Data Engineering (Prof. Gieseke)
Dept. of Information Systems
WWU Münster, Germany

Speaker notes

- To toggle these notes, press `v`
  - If a slide contains audio, notes might show transcript
- Press `?` for key bindings (in particular, `o`, `n`, `p`, `Ctrl-Shift-f`)
- Presentations support two different PDF formats, see usage notes
  - Concise PDF format (replace `.html` and whatever follows in address bar with `.pdf`)
  - Print browser view to PDF (add `?print-pdf` after `.html`, then print to PDF; suggested settings)
- If you find the amount of outgoing links to be distracting, see usage notes
  - Add `?hidelinks` (maybe with a number) after `.html`
- See usage notes for other non-obvious features

Agenda

- 1. Introduction
- 2. Basics
- 3. Layering and Protocols
- 4. Internet and OSI Models
- 5. Internet Communication
- 6. End-to-End Argument
- 7. Conclusions

1. Introduction

1.1. Today's Core Questions

- What is the **Internet**?
- How to provide **global** connectivity in view of **heterogeneous** network technologies, diverse devices, and novel (and forthcoming) applications?
- How to cope with **complexity**?

1.2. Learning Objectives

- Explain and contrast Internet and OSI architectures
- Explain layers in Internet architecture
  - Roles and interplay for communication
  - Basic properties of IP, UDP, TCP
- Explain forwarding of Internet messages based on (IP and MAC) addresses and demux keys
  - Use Wireshark for basic network diagnosis
    - What DNS server is in use? Does it reply? Do response messages for TCP/IP arrive? What next-hop router is in use?
- Explain end-to-end argument

1.3. Previously on CACS …

1.3.1. Communication and Collaboration

- Communication frequently takes place via the **Internet**
  - Telephony
  - Instant messaging
  - E-Mail
  - Social networks
- Collaboration frequently supported by tools using **Internet** technologies
  - All of the above means for communication
  - ERP, CRM, e-learning systems
  - File sharing: Sciebo, etherpad, etc.
  - Programming (which subsumes file sharing): Git, subversion, etc.
- All of the above are instances of **DSs**

1.3.2. Ubiquity of ~~DSs~~ Internet

- ~~DSs are~~ The Internet is everywhere
  - Decentralized, heterogeneous, evolving
  - Variety of applications
  - Variety of physical networks and devices
    - Cloud computing, browser as access device
- IT permeates our life
  - Internet of Things (IoT)
  - From smart devices to smart cities
- How does that really work?
  - Complexity? Functionality?
  - Security? Privacy?

"Internet of Things" by Wilgengebroed on Flickr under CC BY 2.0; from Wikimedia Commons

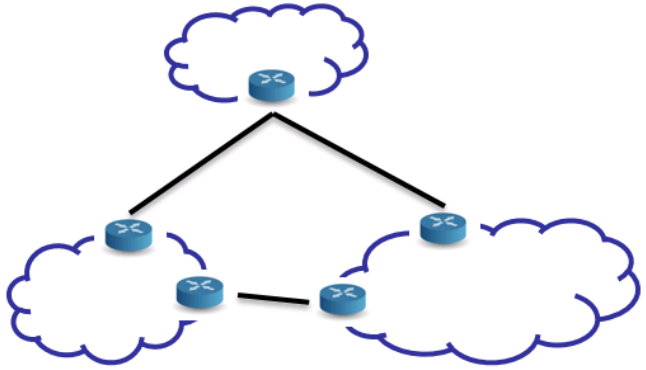
2. Basics

2.1. (Computer) Networks

[PD11]: A **network** can be defined recursively as

- two or more nodes/devices/hosts connected by a link
  - (e.g., copper, fiber, nothing)

- or two or more networks connected by one or more nodes (with necessary links)
  - (e.g., gateway, router)



Speaker notes

Networks can be defined recursively as shown here. An elementary network on the left is an example of a so-called *Local Area Network* (LAN), although LAN is not a term with a well-defined meaning. For example, your LAN at home (if you've got one) may intuitively be understood as the network consisting of your household's locally connected devices and may contain *several* links.

A network of networks as shown on the right is also called *internetwork*. The Internet (spelled with a capital "I") is a special internetwork.

2.1.1. On Routers

- Previous slide mentions **routers** as nodes that connect networks
  - One example: router at home that connects home network to ISP's network
  - Other example: router that connects "large" networks at backbone of Internet
    - Independently managed networks are called autonomous systems 🚀
      - E.g., networks of the University of Münster are part of an autonomous system run by Deutsches Forschungsnetz 🚀
    - Routers exchange information about reachable networks with protocols such as BGP 🚀
      - Usually, multiple paths between networks (and nodes) exist (alternatives may allow to "route around" link and router failures)
      - Routers choose paths based on local policies (e.g., distance, cost)

2.2. Internet vs Web

- The Internet is a **network** of networks
  - **Connectivity** for heterogeneous devices
  - Various **protocols**, some details on later slide ▶
    - IPv4 and IPv6 to send messages between devices on the Internet
    - TCP and UDP to send messages between processes on Internet devices
      - (E.g., process of Web browser talks with remote process of Web server)
      - TCP: Reliable full-duplex byte streams
      - UDP: Unreliable message transfer
- The Web is an **application** using the Internet
  - Clients and servers talking **HTTP** over TCP/IP
    - E.g., `GET` requests asking for **HTML** pages (separate presentation)
    - Web servers provide resources to Web clients (browsers, apps)
- Internet and Web **are** and **contain** DSs

2.3. Heterogeneity

- Internet is network of networks
- Potentially each network with
  - independent administrative control
  - different applications and protocols
  - different performance and security requirements
  - different technologies (fiber, copper, wired, wireless)
  - different hardware and operating systems
- How to overcome heterogeneity?

3. Layering and Protocols

3.1. Layering

General technique in Software Engineering and Information Systems

- Use **abstractions** to **hide complexity**
  - Abstractions naturally lead to **layering**
  - **Alternative** abstractions at each layer
    - Abstractions specified by **standards/protocols/APIs**
- Thus, problem at hand is decomposed into manageable components
  - Design becomes (more) modular

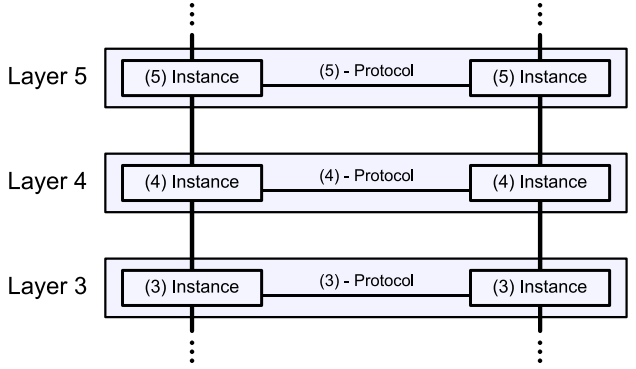
3.2. Network Models/Architectures

- **Models** frequently have different **layers** of abstraction
  - Goal of layering: Reduce complexity
    - Each layer offers **services** to higher layers
      - Semantics: What does the layer do?
    - Layer **interface** defines how to access its services from higher layers
      - Parameters and results
      - Implementation details are hidden
      - (Think of class with interface describing method signatures while code is hidden)
- **Peer entities**, located at same layer on different machines, communicate with each other
  - **Protocols** describe rules and conventions of communication
    - E.g., message formats, sequencing of events
- **Network architecture** = set of layers and protocols

(Based on: [Tan02])

3.3. Protocol Layers

- Each protocol instance talks virtually to its **peer**
  - E.g., HTTP `GET` request from Web browser to Web server
- Each layer communicates only by using the one below
  - E.g., Web browser asks lower layer to transmit `GET` request to Web server
  - Lower layer **service** accessed by an **interface**
- At bottom, messages are carried by the medium



"Layered Communication in OSI Model" by Runtux under Public domain; from Wikimedia Commons

(Based on: [Tan02])

3.4. Famous Models/Architectures

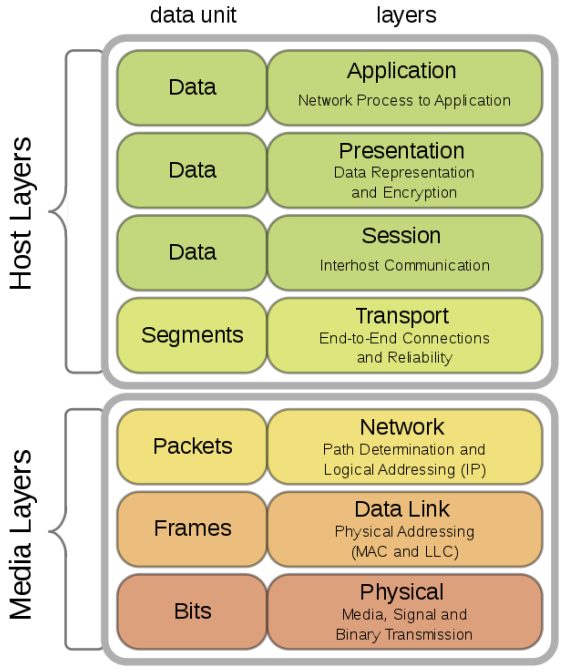
- ISO OSI Reference Model
  - Mostly a model, describes what each layer should do
    - But no specification of services and protocols (thus, no real architecture)
  - Predates real systems/networks
- TCP/IP Reference Model
  - Originally, no clear distinction between services, interfaces, and protocols
    - Instead, focus on protocols
  - Model a la OSI as afterthought

(Based on: [Tan02])

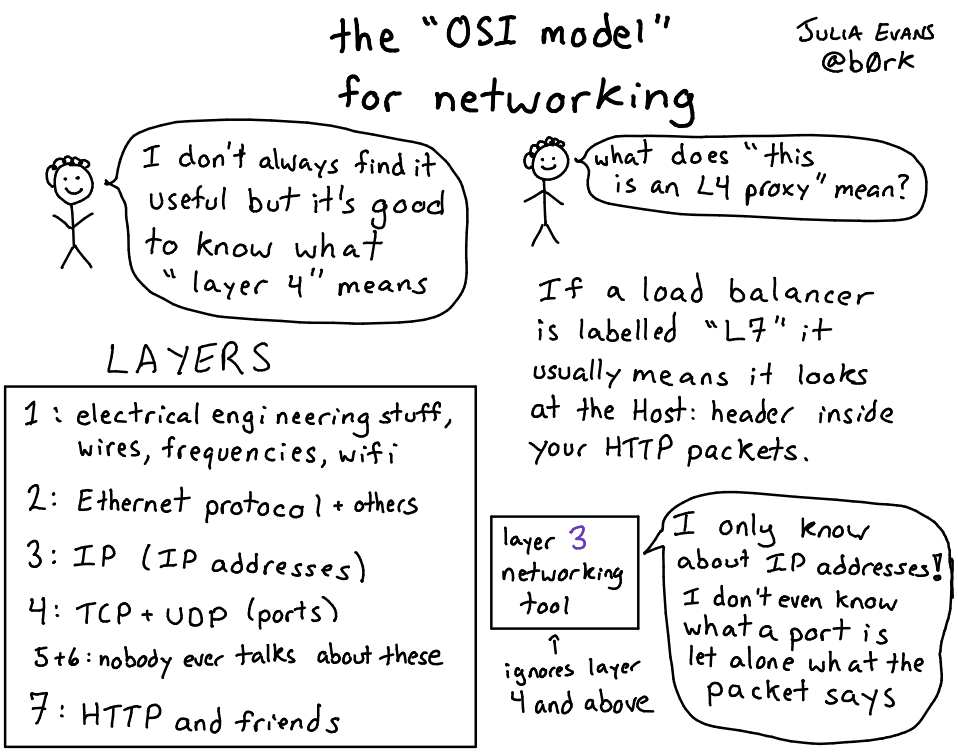
4. Internet and OSI Models

4.1. OSI Reference Model

- International standard
  - Seven layer model to connect different systems
    - Media Layers
      1. Sends bits as signals
      2. Sends frames of information
      3. Sends packets from source host over multiple links to destination host
    - Host layers
      4. Provides end-to-end delivery
      5. Manages task dialogues
      6. Converts different representations
      7. Provides functions needed by users/applications



4.1.1. Drawing for OSI Model



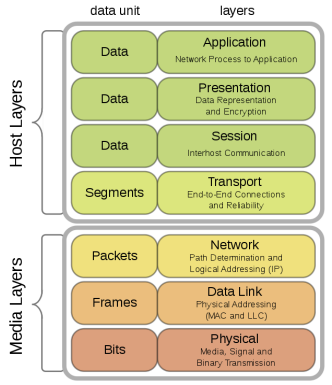
Networking layers

4.1.2. Where are Top and Bottom?

- In layered architectures, lower layers represent more technical details while higher layers abstract away details
  - E.g., in the OSI model the top layer (7) is the application layer, which does not care about technical communication details
- The ◂ previous drawing does not follow that convention when showing layers, but implicitly assumes it anyways (layer 3 “ignores layers 4 and above”)

4.2. OSI Model on Internet

- Internet architecture involves following subset of OSI layers
  - Application layer
    - E.g., Web (HTTP), e-mail (SMTP), naming (DNS)
    - (Presentation and session omitted; part of application protocols)
  - Transport layer
    - E.g., TCP, UDP (also QUIC 🚀 to replace TLS over TCP ▶)
  - Network layer
    - Unifying standard: Internet Protocol (IP; v4, v6)
    - Everything over IP, IP over everything
  - Data link layer
    - E.g., Ethernet, WiFi, cellular phone network, satellite link

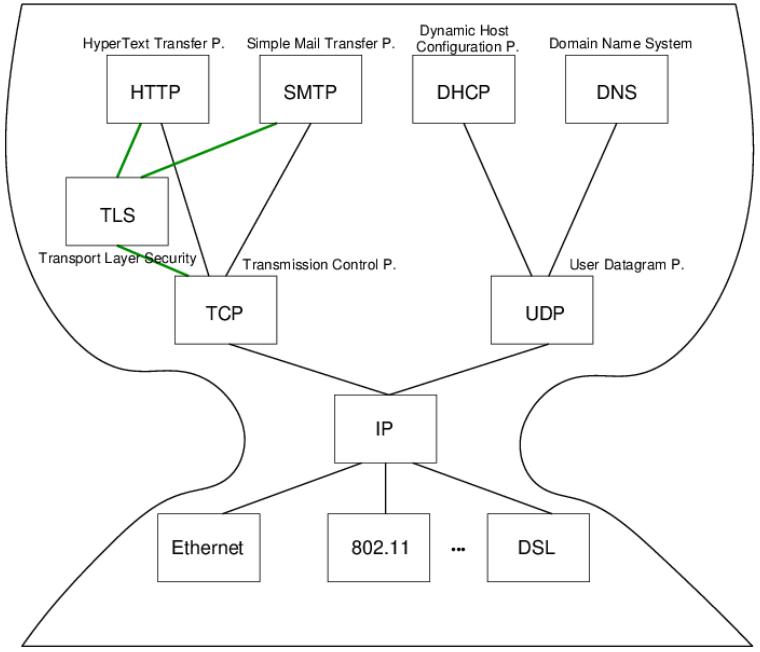


4.3. Internet Standards

- Defined by Internet Engineering Task Force (IETF)
  - Current list
- Each standard specified by set of RFCs (Requests For Comments)
  - But not every RFC is a standard, e.g., April fool's day
  - Statuses: Informational, Experimental, Best Current Practice, Standards Track, Historic
  - Community process
    - Everyone may submit Internet Draft; typically, produced by IETF working groups
    - Afterwards peer reviewing; eventually, publication as RFC
    - David Clark: "We reject kings, presidents and voting. We believe in rough consensus and running code."

4.3.1. Internet Architecture

- “Hourglass design”
- IP is focal point
  - “Narrow waist”
  - Application independent!
    - Everything over IP
  - Network independent!
    - IP over everything
  - No security
    - “IP datagrams are like postcards, written with erasable pencils”
      - Usual security protocol is TLS ↔ for encryption and integrity protection, located between application and TCP



4.3.2. IP, UDP, and TCP

- IP (Internet protocol)
  - Offers best-effort host-to-host connectivity
    - **Best effort**: Try once, no effort to recover from transmission errors
    - Connection-less delivery of datagrams
- Transport layer alternatives
  - UDP (User Datagram Protocol)
    - Extends IP towards **best-effort application-to-application** connectivity
      - Ports identify applications/processes (e.g., 53 for DNS)
      - Connection-less
  - TCP (Transmission Control Protocol)
    - Offers **reliable application-to-application** connectivity
      - Ports identify applications/processes (e.g., 80/443 for Web servers)
      - Full-duplex byte stream
      - Three-way handshake to establish **connection**
      - Acknowledgements and timeouts for **retransmissions**

While this introduction does not aim to present individual protocols in detail, you should be able to explain the following:

IP is an acronym for “Internet Protocol” (and not for IP address). Currently, two IP versions are in use, namely IPv4 and IPv6. Every device that is connected to the Internet needs to have at least one IP address. Actually, IP addresses are bound to networking hardware of devices. For example, your smartphone’s WiFi hardware might currently be configured with one IP address, while its GSM/UTMS/LTE modem might be configured with a different IP address. Obviously, those IP addresses are reconfigured when your devices moves between different networks: Your WiFi network at home probably uses a different range of IP addresses than the one at the university.

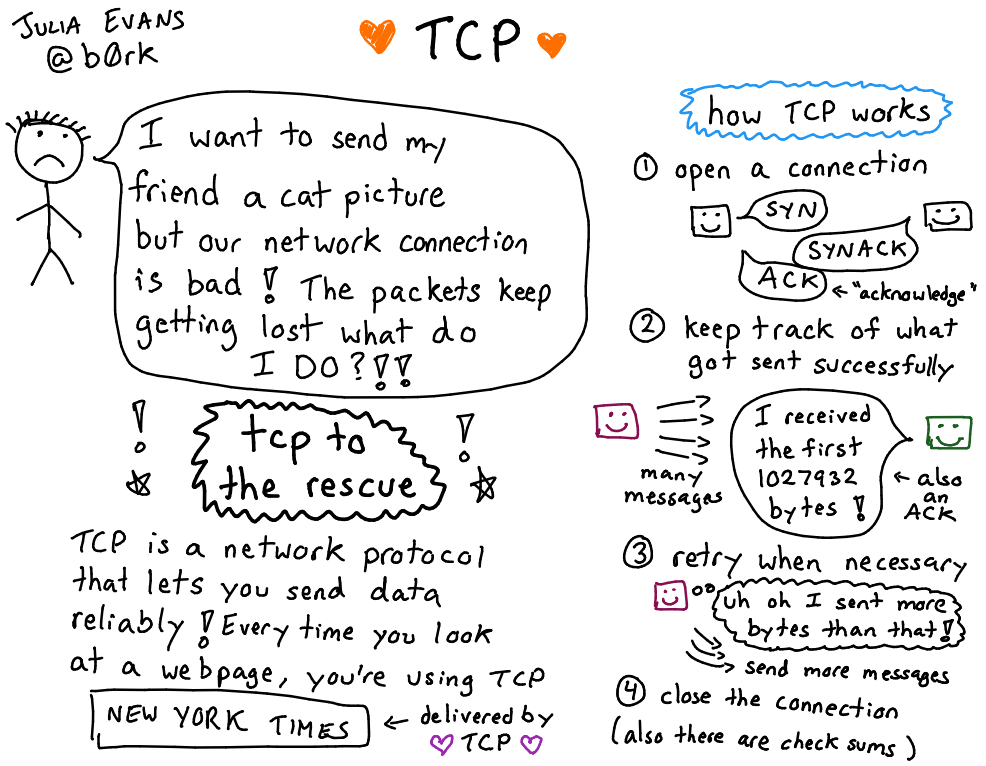
Messages transmitted via IP are called *datagrams*, and each datagram carries a source and a destination IP address. Ideally, a datagram sent from one host anywhere in the universe reaches its destination host based on forwarding and routing functionality coming with the Internet architecture.

IP is called *best-effort* protocol as participating devices give their best in one attempt to deliver messages. However, they do not make attempts to recover from transmission errors. You may find this interpretation of “best effort” surprising.

Also, note that IP works *without* a notion of connection, which means that individual datagrams sent between two devices may travel on different routes through the Internet.

Typically, multiple *applications* may run on each host. With modern OSs, those applications are managed as processes, and transport layer protocols such as UDP and TCP allow those end points to communicate. Both, UDP and TCP, add source and destination *ports* to IP, which are just integer numbers to be used by the OS to identify the processes as end points of the communication. Briefly, UDP is again a best-effort protocol. TCP adds functionality for *reliable* connections of byte streams from and to which processes can read and write arbitrary data. Such a connection is established by a so-called three-way handshake (see SYN, SYN/ACK, and ACK in the subsequent drawing), and reliability is guaranteed with retransmission mechanisms based on acknowledgements and timeouts.

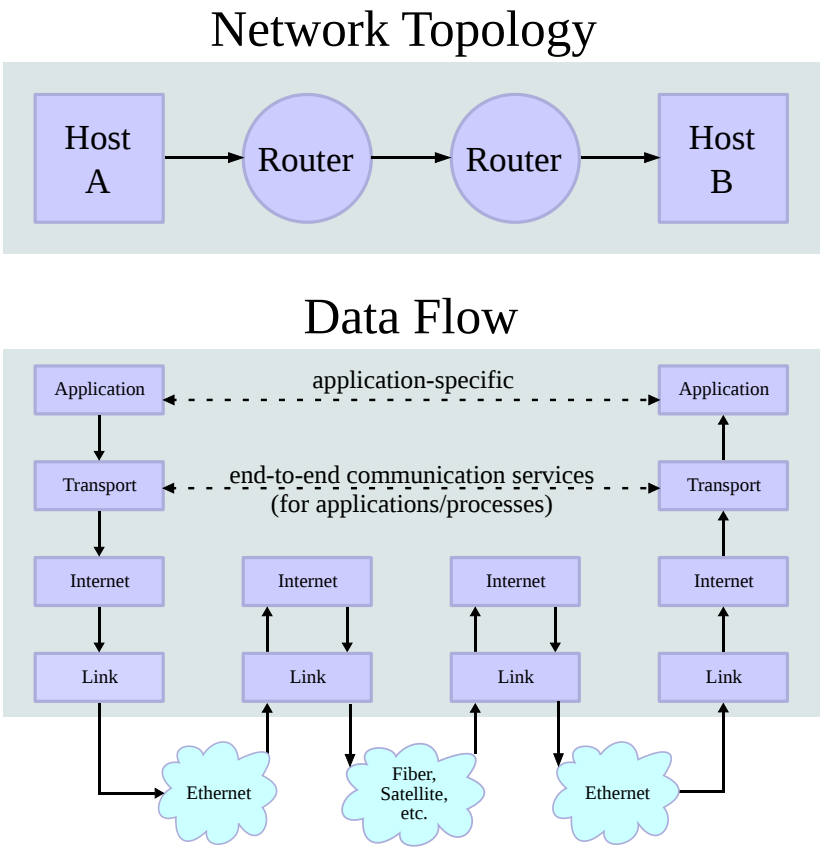
4.3.3. Drawing on TCP



TCP basics!

5. Internet Communication

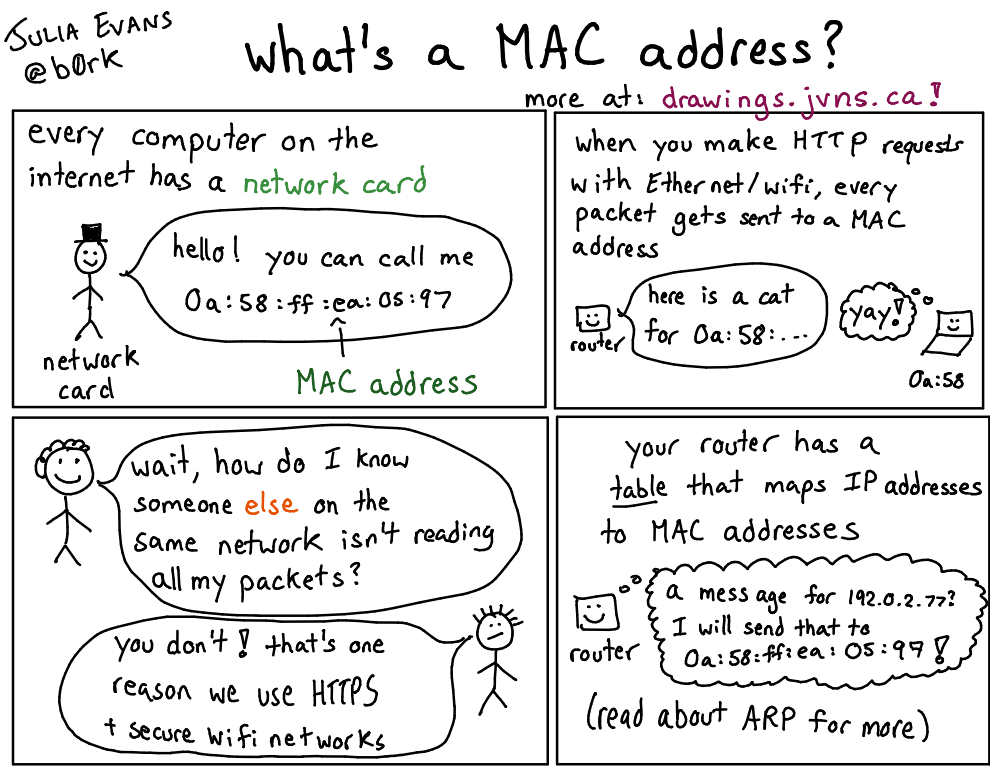
5.1. IP Stack Connections



Speaker notes

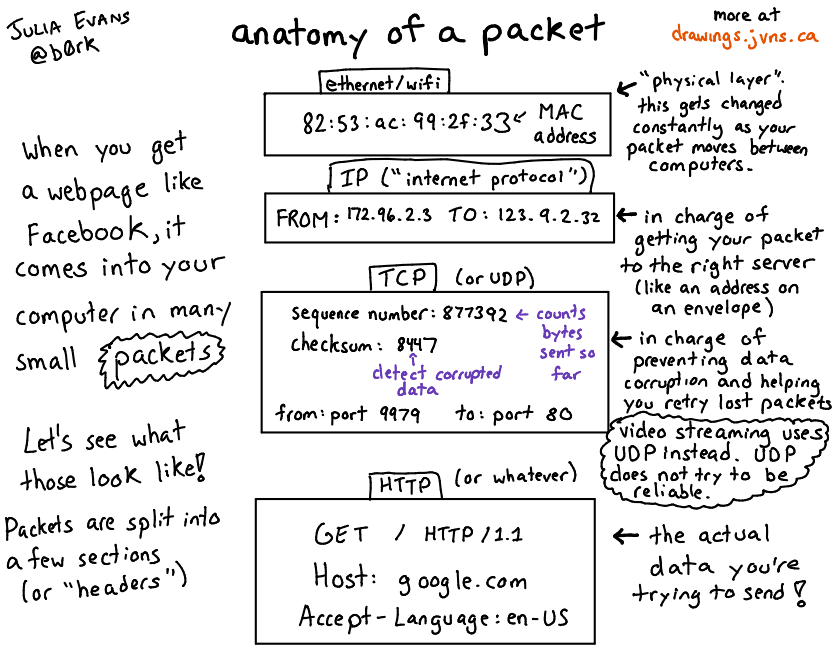
Consider communication between two applications on Internet hosts A and B. As shown at the top, messages between A and B flow through two routers, connecting different underlying networks. The use of different protocol layers on hosts and routers is shown at the bottom. Hosts and routers use the Internet layer to forward each datagram via next hops based on the destination IP address. Transport and application layer information is only relevant at source and target hosts but not at intermediate hops. Transport layer protocols such as UDP and TCP provide end-to-end communication services for applications, while the application layer accommodates protocols such as HTTP, SMTP, and DHCP, each of which relies on transport layer protocol services for end-to-end communication.

5.1.1. Drawing on MAC Addresses



What's a MAC address?

5.1.2. Drawing of Packet



Anatomy of a packet

◂ Again, this figure does not show the order of layers. Instead, it shows the order of bits and bytes in a message. The first bits encode MAC addresses as part of LAN headers, while the final bits belong to the application message. Verify that yourself with Wireshark.

5.1.3. Typical Communication Steps (0/2)

- Prerequisites
  - Internet communication requires numeric IP addresses
    - **Lookup** of IP addresses for human readable names via **DNS**
      - DNS is request-reply protocol
      - DNS client (e.g., the browser) asks DNS server for IP address of name, e.g., query for www.wwu.de may result in 128.176.6.250
      - (And more)
  - LAN communication requires MAC addresses
    - MAC (media access control) address: Hardware address of network card, e.g., for Ethernet, WiFi
      - Typical format with hexadecimal digits: 02:42:fa:5c:4a:4a
    - **Lookup** of MAC addresses for IP addresses via **ARP** (Address Resolution P.)
      - Send ARP request into local network: “If you have IP addresses x, what is your MAC address?
      - ARP request is a **broadcast**: Sent to every device in LAN
      - Device that has IP address x replies with its MAC address

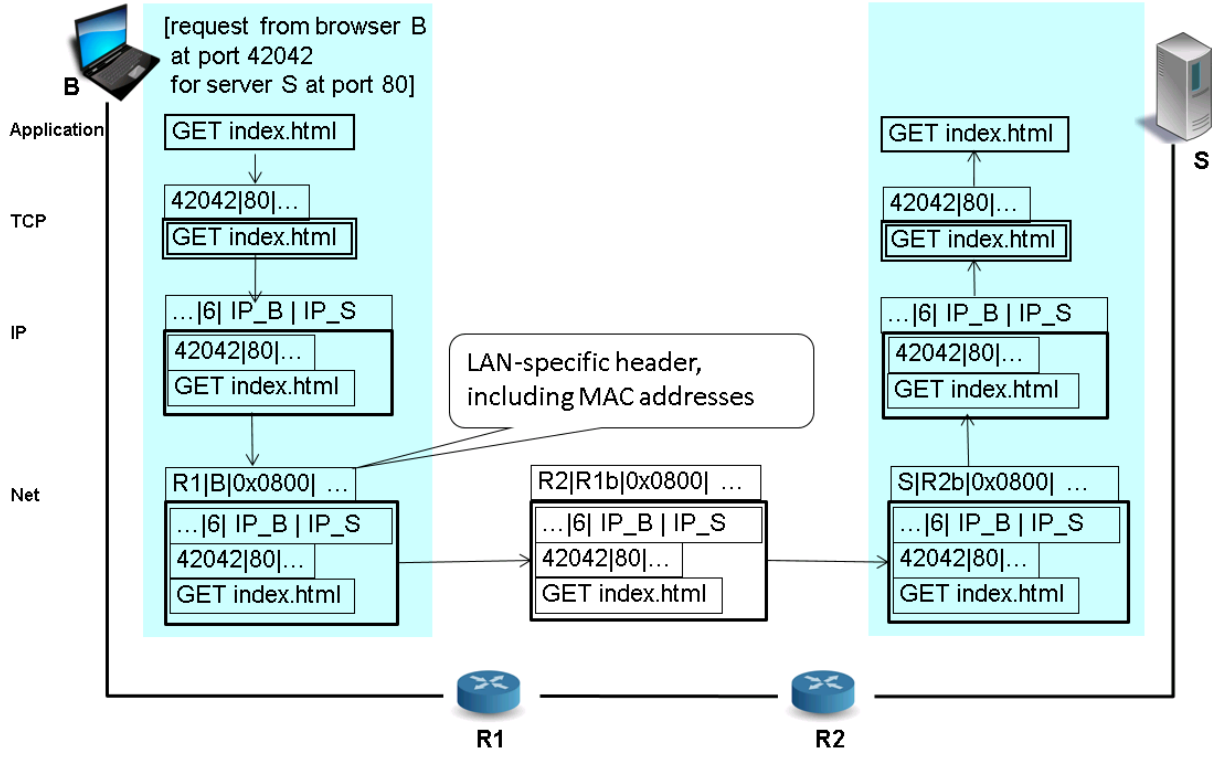
5.1.4. Typical Communication Steps (1/2)

- Ex.: Send HTTP message M to host www.wwu.de
  1. Perform **DNS** lookup for www.wwu.de
     - Returns IP address 128.176.6.250
  2. Encapsulate ▶ M by adding TCP header
     - Source and destination TCP **ports**: Numbers that identify processes
       - Typically, destination port 80 for Web servers with HTTP (443 for HTTPS)
       - Random source ports for Web browsers
  3. Encapsulate TCP segment by adding IP header
     - Source and destination IP addresses
     - Demux key ▶ to indicate that TCP segment is contained

5.1.5. Typical Communication Steps (2/2)

- Ex.: Send HTTP message M to host www.wwu.de
  1. Perform DNS lookup for www.wwu.de
  2. Encapsulate with TCP header
  3. Encapsulate with IP header
  4. **Routing** decision to determine IP address of **next hop** router
     - Returns IP address IP_R within sender's network
     - E.g., 128.176.158.1 at my work, 192.168.178.1 at home
  5. **ARP** lookup to determine MAC address for IP_R
     - E.g., 0:0:c:7:ac:0
  6. Encapsulate IP datagram with LAN-specific header with MAC address, send via LAN to router
- Routers repeat steps (4) - (6) to forward M to final destination

5.2. Encapsulation



Speaker notes

You should be able to explain this figure based on information of surrounding slides, roughly as follows.

At the top, you see a client sending an HTTP GET request as application message M to Web server S. The client does not care about any details of underlying layers beyond the fact that TCP is used for transmission, and it asks its Operating System (OS, for short) to transmit M to server S on port 80. The OS chooses an unassigned source port, here 42042, and *encapsulates* M in a TCP header, forming a TCP segment, called T.

As shown on the previous slide, a DNS lookup for S may be necessary, resulting in an IP address for S. Here, the IP addresses for browser and server are represented with IP_B and IP_S, resp. The OS encapsulates TCP segment T in an IP header to form an IP datagram D.

The OS is configured with IP addresses for a set of connected routers or uses protocols such as DHCP to learn the IP address of at least one default router. It uses the target IP address IP_S to perform a routing decision that determines the next hop router among its known routers, here R1.

With an ARP request, the OS determines the MAC address for R1's known IP address and builds a LAN header, say for WiFi transmission. Here, R1 denotes the MAC address for R1 and B the one of the sending machine's WiFi networking card. The resulting WiFi frame is broadcast into the neighborhood, where it may be seen by lots of devices. Receiving network cards check the target MAC address against their own; some may take a closer look even if the frame is *not* addressed to them, which is why it would have been a good idea to encrypt M before sending it down the protocol stack. Anyways, R1 receives the frame as well, traversing the first link.

As explained on the next slide, each header contains a demux key to indicate what type of data is encapsulated. When receiving data, the demux key allows to determine what next higher level should process the data. The demux key hex 800 in the WiFi header tells R1 that an IP datagram is contained, so its unwraps the frame header to obtain IP datagram D and sees that the datagram's IP target is IP_S, not itself. If R1 is curious or malicious, it may still take a look at the contained message M, just as any other router along the way. Again, encryption of M is necessary to prevent this.

Anyways, R1 performs a routing decision to bring datagram D closer to IP_S, identifying R2 as next hop. It encapsulates D in another LAN header with its own source MAC address and a target MAC address for R2 (potentially again using ARP) and injects the resulting frame into the LAN shared with R2. Such router-to-router communication may be repeated several times before the final router injects a frame into the target LAN, where S receives it and unwraps headers layer by layer until M reaches the Web server process.

I suggest to use the free software Wireshark to inspect DNS and ARP requests, frames, headers, and messages of your devices, see my Wireshark demo.

5.3. Encapsulation and Demux Keys

- **Encapsulation**
  - Protocol specific header added for each layer
    - Starting from “pure” application message
    - Headers prepended when moving down the protocol stack
  - Headers “unwrapped” when moving up again
- **Demux** key
  - Identifies recipient protocol at next higher layer
  - Different protocols use different forms of demux keys (see previous slide)
    - Ethernet header contains **type** field (IPv4 = 0x0800, ARP = 0x0806)
    - IP header contains **protocol** field (TCP = 6, UDP = 17)
    - TCP header contains **port** (application id) as demux key

5.4. Review Questions

Let's see what you remember about the Internet...

1. Select correct statements about the Internet Protocol
   - [ ] Every device that is part of the Internet implements the Internet Protocol
   - [ ] 128.176.6.250 is a valid address with IPv4
   - [ ] Some ranges of IP addresses are private in the sense that they can be reused in different networks (not part of presentation)
   - [ ] IP is a best-effort protocol, which means that the loss of messages is countered with retry mechanisms

2. Select correct statements about IP addresses
   - [ ] When forwarding a datagram, the destination IP address changes on a hop-by-hop basis
   - [ ] When forwarding a datagram, the source IP address (usually) remains unchanged
   - [ ] ARP aims to obtain IP addresses from human-readable names
   - [ ] DNS aims to obtain IP addresses from human-readable names
   - [ ] A single device may have multiple IP addresses

3. Select all correct statements about MAC addresses
   - [ ] IP datagrams do not contain MAC addresses (usually)
   - [ ] ARP aims to obtain MAC addresses for known IP addresses
   - [ ] ARP aims to obtain IP addresses for known MAC addresses
   - [ ] An ARP request uses a special broadcast address as target MAC address, which means that every device on the Internet receives that request

4. Select all correct statements about demux keys
   - [ ] A demux key is used to encrypt data
   - [ ] Demux keys are added to headers when going down the protocol stack
   - [ ] When going down the protocol stack, the demux key indicates the protocol to be used on the next lower layer
   - [ ] When protocol B encapsulates a message from protocol A, protocol B adds a demux key identifying A

5.5. Wireshark Demo

- Wireshark is a network protocol analyzer
  - For live or recorded traffic
  - Wireshark Demo (including 8-minute video) to get you started
- Use of Wireshark improves understanding of networking and device communication

6. End-to-End Argument

6.1. Network: Core, Edge, Endpoint

- Network **core**: Devices **implementing** the network
  - Routers, switches
- Network **edge**: Devices **using** the network
  - Computers, “smart” devices, IoT devices
- **Endpoints** of communication: Distributed **applications**
  - Processes that send and receive messages
    - E.g., your e-mail client, your Web browser, your messenger
    - Beware: Who is the other end for your browser? Who for your mail client and messenger?

Here you see major terminology related to computer networks: The network *core* is implemented by devices such routers, while it is used by *endpoint* applications on devices at the network *edge*.

Note that answers to the questions on this slide are not obvious, but require knowledge of the specific application protocol. For example, you will see in the upcoming session that your Web browser talks directly with the Web server in an end-to-end fashion.

In contrast, your e-mails and messages do not directly reach the final recipient. Instead, your client transfers the message (possibly encrypted) to some intermediary. Thus, from an Internet perspective, the end-to-end communication here is between sender and intermediary. Note that the intermediary may potentially be the first in a sequence of intermediaries forwarding your message. Ultimately, the recipient picks up the message from the final intermediary. Thus, the communication between sender and recipient is not end-to-end but hop-by-hop.

6.2. Overarching Question

- What functionality to implement in the network core, what within communication endpoints?
  - Observations
    - If functionality is available as Internet standard, every application can immediately use it. No need to reinvent wheels.
    - Simplicity and generality of protocols increase potential for re-use, e.g., IP allows to connect “everything.”
  - Answer to question given in [SRC84]: End-to-end argument
  - Intuition
    - Some functionality needs application knowledge
    - Such functionality cannot be implemented inside the net
    - In general, application functionality should not be implemented in the net

6.3. End-to-End Definition

- Quotes from [SRC84]
  - “The principle, called the end-to-end argument, suggests that functions placed at low levels of a system may be redundant or of little value when compared with the cost of providing them at that low level.”
  - “The function in question can completely and correctly be implemented only with the knowledge and help of the application standing at the end points of the communication system. Therefore, providing that questioned function as a feature of the communication system itself is not possible. (Sometimes an incomplete version of the function provided by the communication system may be useful as a performance enhancement.)”

Speaker notes

The end-to-end argument deals with the question of “good” protocol design: Which functions should one implement in a specific protocol at a specific layer of some architecture, which ones in higher or lower layers?

Clearly, if functionality is provided by a low layer, say L0, all layers above L0 can just *use* L0’s functions, without any need for redundant implementations. Thus, having re-usable functionality in low layers is a Good Thing. However, higher-level requirements may be too specific to provide completely at L0. The next slide shows error-free file transfer as an example (end-to-end security goals are similar). Here, L0 might provide an incomplete version of the required functionality. In this case, L0 gets more complex, and higher layers still need to implement their own specific functions.

The end-to-end argument suggests (a) not to implement incomplete versions at L0 if they are redundant given the need for more complete implementations at higher layers and (b) to also consider cost of implementing incomplete functions at L0.

6.4. End-to-End Example

- Careful file transfer
  - Read file from disk, transfer over Internet, write to disk at remote end
  - Possible errors, leading to corrupted data
    - Disk error
    - Software errors in file system, file transfer, network protocol, buffering or copying
    - Hardware errors (e.g., processor or memory failures)
    - Network failures/attacks (messages lost or bits changed)
    - Crash in the middle of the transfer
  - Possible solutions
    - Lots of “small” tests
    - One end-to-end checksum check, with retry in case of errors

- How many “small” tests will be necessary?
  - Notice: A test regarding network transfer does not help much since all other types of errors can still corrupt data
    - Hence, an **end-to-end check** will be **necessary** anyways
  - However, from a **performance** perspective, a single end-to-end check may be costly
    - Consider transfer of some GB, which may take a long time
      - The end-to-end check detects individual errors only after full transfer
      - In contrast, intermediate checks may identify individual bit errors early, allowing partial retries

6.5. End-to-End Security

- Above observations also apply to security goals of confidentiality and integrity
  - Confidentiality and integrity are end-to-end security goals
  - If you want them, you must neither rely on link level nor hop-by-hop assurances
    - As offered by, e.g., WPA variants 🚀, IPsec, VPN, De-Mail
- You must protect your data inside your applications (end-to-end)
  - Recall e-mail and messaging mentioned above

First, I'd like to point out that not everybody understands what end-to-end security means. For example, until 2020 Zoom advertised its videoconferencing service with the claim of supporting end-to-end encryption, which was false and was one reason for an investor to sue the company.

Absence of end-to-end encryption is no small issue. If you are not aware of the story of Ladar Levison who shut down his company Lavabit for secure e-mail communication instead of betraying his customers, I encourage you to read it. He gave this piece of advice: "This experience has taught me one very important lesson: without congressional action or a strong judicial precedent, I would *strongly* recommend against anyone trusting their private data to a company with physical ties to the United States."

You may want to think about the implications of his experience on your use of US-based services, but that is not our topic here.

To illustrate the end-to-end argument in a security context, consider WiFi networking. As you know, WiFi protocols are located at the lowest layer of the Internet architecture, and the questions considered here are (a) whether WiFi encryption protocols such as WPA are redundant and (b) whether it pays off to implement encryption in wireless network protocols.

With WiFi protection, your data is encrypted between your device and the WiFi router. Suppose your application sends out plaintext data, which is like a postcard written with an erasable pencil: Everybody along the way is free to read and modify the data.

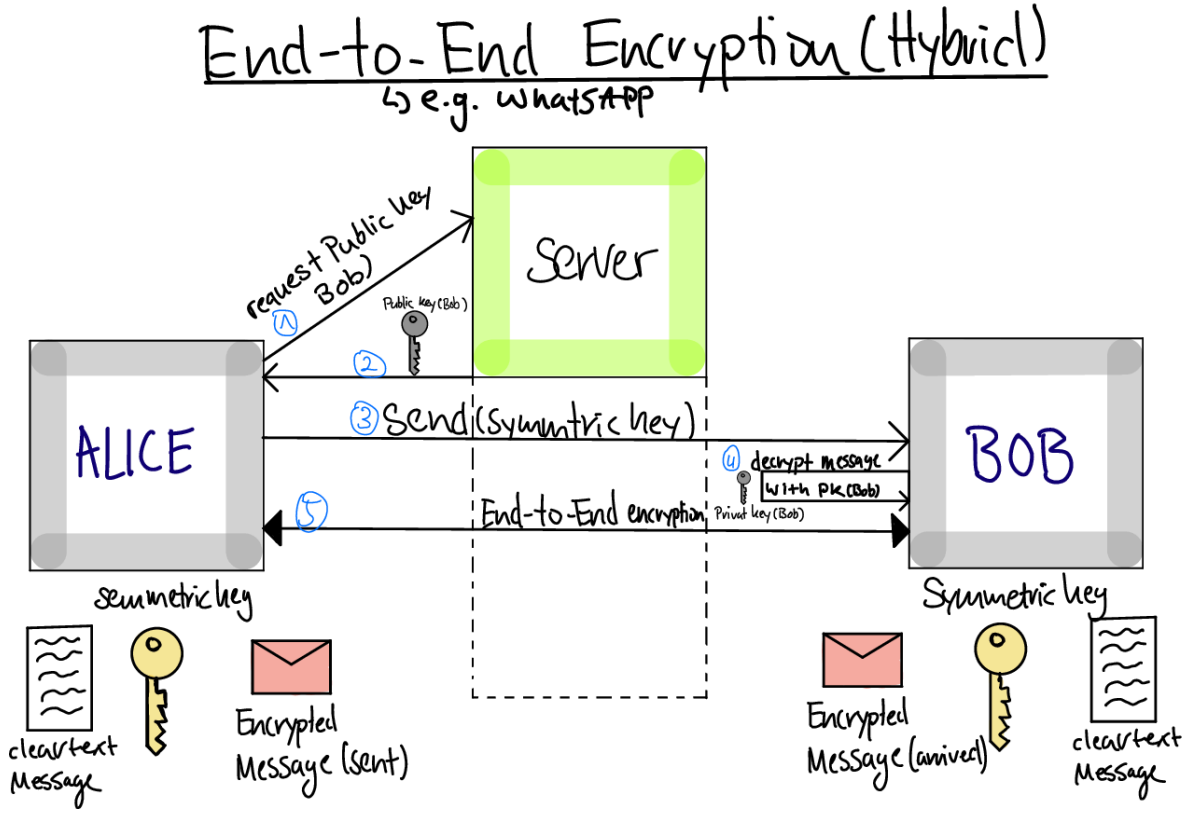
With WiFi encryption, that data is protected between you and your router, but forwarded in its original plaintext from your router into the Internet, where it can again be read and modified by lots of parties (all ISPs along the way, everybody with control over any router along the way, such as intelligence agencies and ordinary criminals). Clearly, if you care about confidentiality or integrity of your data, you do not transmit it as plaintext in the first place but secure it with end-to-end cryptography, which applies not only in your own network but along the entire way to the final destination. Then, however, WiFi encryption is redundant.

Also, security issues from WEP over WPA up to its most recent version 3 suggest that the cost of providing those functions at such a low layer may be too high: If you depend on those functions, you may need to replace your devices with each new standard.

Note however, that WPA is not only about encryption but also about access control. If you connect “smart” (or dumb) devices in your home network, you may want to check whether they and their services apply strong cryptography to prevent unauthorized access. If they do (and you do not mind others sharing your bandwidth), access control functionality in WiFi networking is of little value.

In view of the end-to-end argument, WiFi protection seems to be a good candidate for elimination. I happily use open WiFi networks on the road, and I run a Freifunk router at home to support the vision of open WiFi networking as commodity.

6.5.1. Hybrid End-to-End Encryption



This figure was created by students in the context of the course Communication and Collaboration Systems (CACS) in 2020, although details for its understanding are not part of CACS but can be found elsewhere.

To understand the figure, you need to know the following:

- **Encryption** transforms a bit string, e.g., a message M, into another bit string that looks like random data, i.e., the encrypted data does not reveal any information about the original data M. Thus, encryption protects the security goal of **confidentiality** where M should not be accessible to third parties. Some form of keys (which are other bit strings) are necessary to encrypt M and to decrypt the encrypted data to obtain M again.
- **Symmetric** encryption requires the use of a secret key that is shared by the communicating parties, here Alice and Bob; a symmetric key used for encryption is then also required for decryption.
- **Asymmetric** encryption relies on **key pairs**, consisting of public key and private key, where the public key is published and can be used to send encrypted messages to its owner who needs the private key for decryption.
- **Hybrid** encryption relies on asymmetric setup phases to exchange shared symmetric keys which are then used for ordinary messages. There are several reasons to use hybrid protocols, in particular performance (symmetric operations are much faster than asymmetric ones) and security (e.g., asymmetric operations for proofs of identity and “secure” derivation of symmetric keys).

In the simplistic and non-realistic protocol shown here, Alice obtains Bob’s public key and uses this to send him a new symmetric key, encrypted with his public key, which he can then obtain with his private key. Afterwards, Alice and Bob share a symmetric key to protect their communication.

6.6. Then vs Now

- [BC01]: Rethinking the Design of the Internet
  - Challenges since 1980s
    - Untrustworthy world, e.g., attacks, spam, DDoS
      - Need more mechanism in the core to enforce “good” behavior?
    - More demanding applications, e.g., video streaming
      - Best effort model may not be good enough, need intermediate storage sites for streaming?
    - ISP service differentiation
      - Different pieces of content provided with different QoS guarantees?
    - Rise of third-party involvement
      - Officials of organizations or governments interpose themselves
    - Less sophisticated users
      - From initial experts to Joe Sixpack, who may be overwhelmed by complexity in endpoints
  - RFC 3724 🚀, 2004: End-to-end is still relevant, though
    - End-to-end manages state at the edges, not the core
      - Failures in core do not affect application state
    - Protection of innovation, reliability, trust

6.7. Review Questions

- Think of your favorite messenger application. Do you know how messages are transferred? Is communication ◀ hop-by-hop or end-to-end? Does it implement ◀ end-to-end security (details are not important for your response here—maybe provide a pointer to verifiable source)? Does security of communication benefit from WPA?

6.8. Concluding Questions

- What did you find difficult or confusing about the **contents** of the presentation? Please be as specific as possible. For example, you could describe your current understanding (which might allow us to identify misunderstandings), ask questions in a Learnweb forum that allow us to help you, or suggest improvements (maybe on GitLab). Most questions turn out to be of general interest; please do not hesitate to ask and answer in the forum. If you created additional original content that might help others (e.g., a new exercise, an experiment, explanations concerning relationships with different courses, …), please share.

7. Conclusions

7.1. Summary

- Computer networks are general purpose networks
  - The Internet forms the backbone for modern communication and collaboration
- Complexity reduced via layered architecture
  - Modular design
  - Internet vs OSI architecture
  - Encapsulation and demux keys

Bibliography

[BC01] Blumenthal & Clark, Communications Policy in Transition, 2001. https://dl.acm.org/citation.cfm?id=566696.566700

[PD11] Peterson & Davie, Computer Networks, Fifth Edition: A Systems Approach, Morgan Kaufmann Publishers Inc., 2011. https://booksite.elsevier.com/9780123850591/

[SRC84] Saltzer, Reed & Clark, End-to-end Arguments in System Design, ACM Trans. Comput. Syst. 2(4), 277-288 (1984). http://web.mit.edu/Saltzer/www/publications/endtoend/endtoend.pdf

[Tan02] Tanenbaum, Computer Networks, Prentice-Hall, Inc., 2002. https://www.pearson.com/us/higher-education/product/Tanenbaum-Computer-Networks-4th-Edition/9780130661029.html

License Information

This document is part of an OER collection to teach basics of distributed systems. Source code and source files are available on GitLab under free licenses.

> *No warranties are given. The license may not give you all of the permissions necessary for your intended use.*