The Internet

Jens Lechtenbörger

Summer Term 2018

Contents

1 Introduction

1.1 Learning Objectives

- Explain and contrast Internet and OSI architectures
- Explain forwarding of Internet messages based on (IP and MAC) addresses and demux keys
  - Use Wireshark to inspect network traffic
- Explain end-to-end argument

1.2 Previously on CACS ...

1.2.1 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**

1.2.2 General Importance of ~~DSs~~ Internet

- ~~**DSs are**~~ The Internet is everywhere
  - Decentralized, heterogeneous, evolving

Figure 1: "Internet of Things" by Wilgengebroed on Flickr under CC BY 2.0; from Wikimedia

  - 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?

1.3 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?

2 Basics

2.1 (Computer) Networks

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

- two or more nodes connected by a link
  - (e.g., copper, fibre, nothing)

Figure 2: Figure under CC0

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

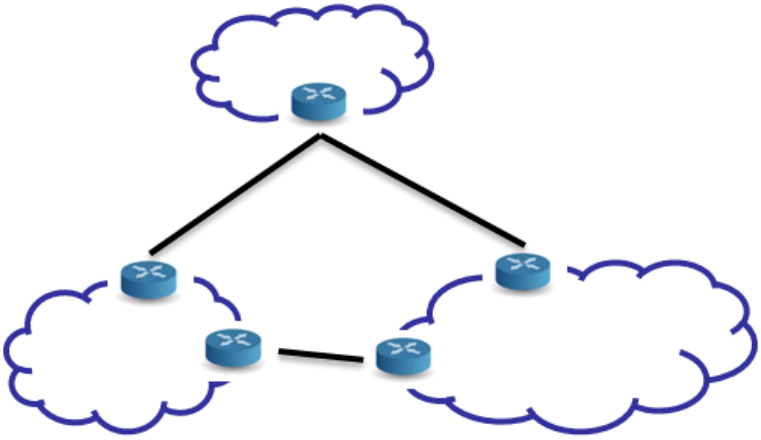

Figure 3: Figure under CC0

2.2 Internet vs Web

- The Internet is a **network** of networks
  - **Connectivity** for heterogeneous devices
  - Various **protocols**
    - IPv4 and IPv6 for **host-to-host** connectivity
  - TCP and UDP for **process-to-process** connectivity (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
    - 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**
- Each layer communicates only by using the one below
- Lower layer **service** accessed by an **interface**
- At bottom, messages are carried by the medium

(Based on: [Tan02])

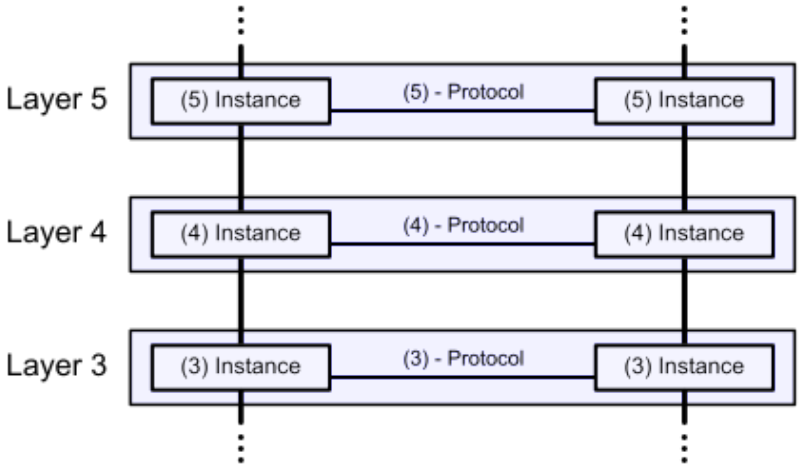

Figure 4: "Layered Communication in OSI Model" by Runtux under Public domain; from Wikimedia

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 Drawing for OSI Model

Warning! External figure **not** included: "Networking layers" under © 2016 Julia Evans, all rights reserved; from julia's drawings
(See HTML presentation instead.)

4.2 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 over multiple links
  - Host layers
    4. Provides end-to-end delivery
    5. Manages task dialogs
    6. Converts different representations
    7. Provides functions needed by users/applications

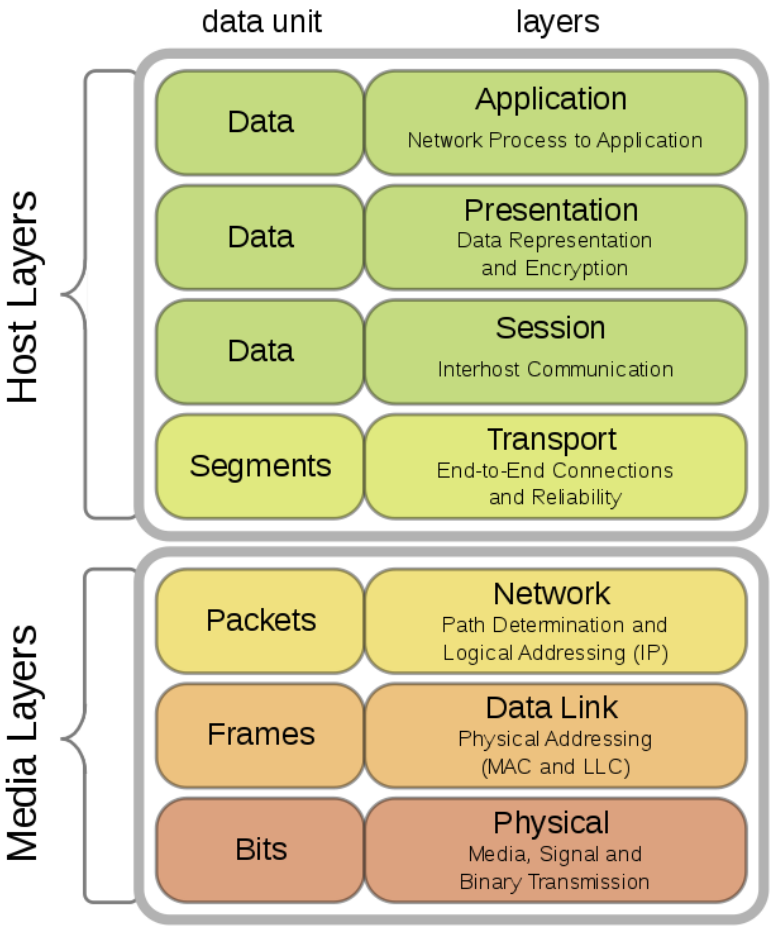

Figure 5: "OSI Model" by Offnfopt under CC0; from Wikimedia

4.3 OSI Model on Internet

- OSI vs Internet

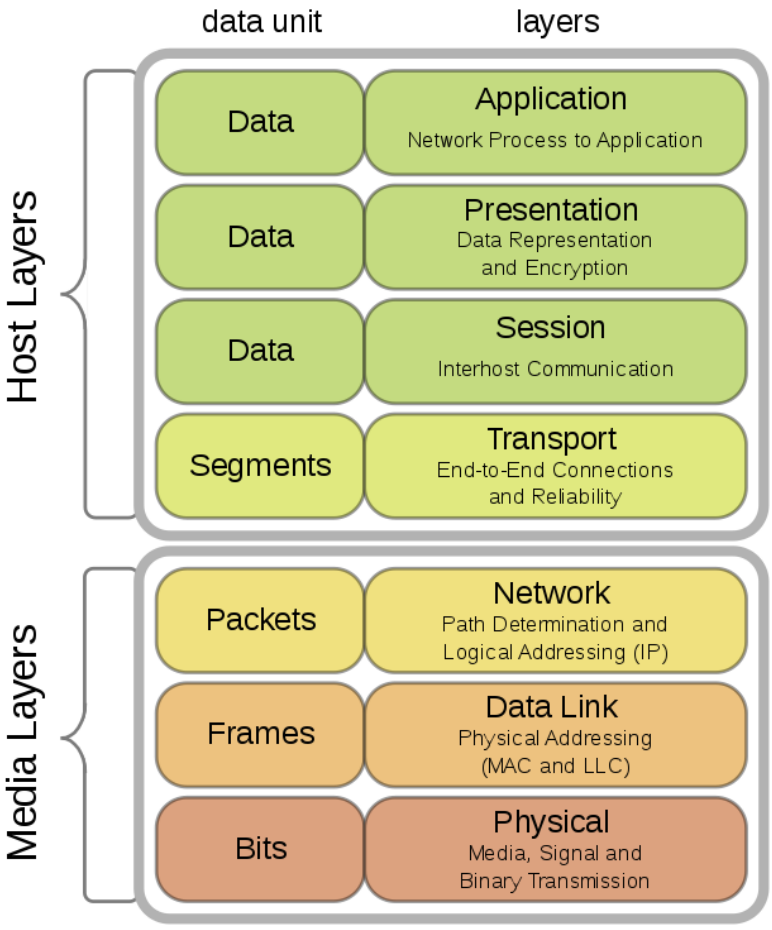

Figure 6: "OSI Model" by Offnfopt under CC0; from Wikimedia

  - Application layer
    - E.g., Web (HTTP), e-mail (SMTP), naming (DNS)
  - (Presentation and session omitted)
  - Transport layer
    - E.g., TCP, UDP
  - Network layer
    - Unifying standard: Internet Protocol (IP; v4, v6)
    - Everything over IP, IP over everything
  - Data link layer
    - E.g., Ethernet, WiFi

4.4 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
  - Stati: 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."

4.4.1 Internet Architecture

- "Hourglass design"
- IP is focal point
  - "Narrow waist"
  - Application independent!
    - Everything over IP
  - Network independent!
    - IP over everything

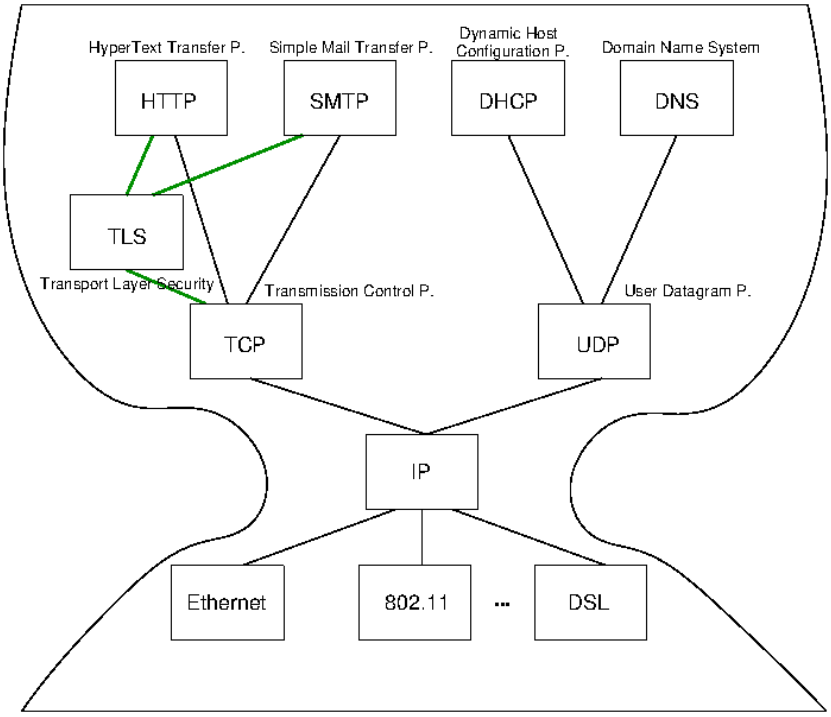

Figure 7: Internet Architecture with narrow waist

4.4.2 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 process-to-process** connectivity
      - Ports identify processes (e.g., 53 for DNS)
      - Connection-less
  - TCP (Transmission Control Protocol)
    - Offers **reliable process-to-process** connectivity
      - Ports identify processes (e.g., 80 for web servers)
      - Full-duplex byte stream
      - Three-way handshake to establish connection
      - Acknowledgements and timeouts for retransmissions

4.4.3 Drawing on TCP

Warning! External figure **not** included: "TCP basics!" under © 2016 Julia Evans, all rights reserved; from julia's drawings
(See HTML presentation instead.)

5 Internet Communication

5.1 IP Stack Connections

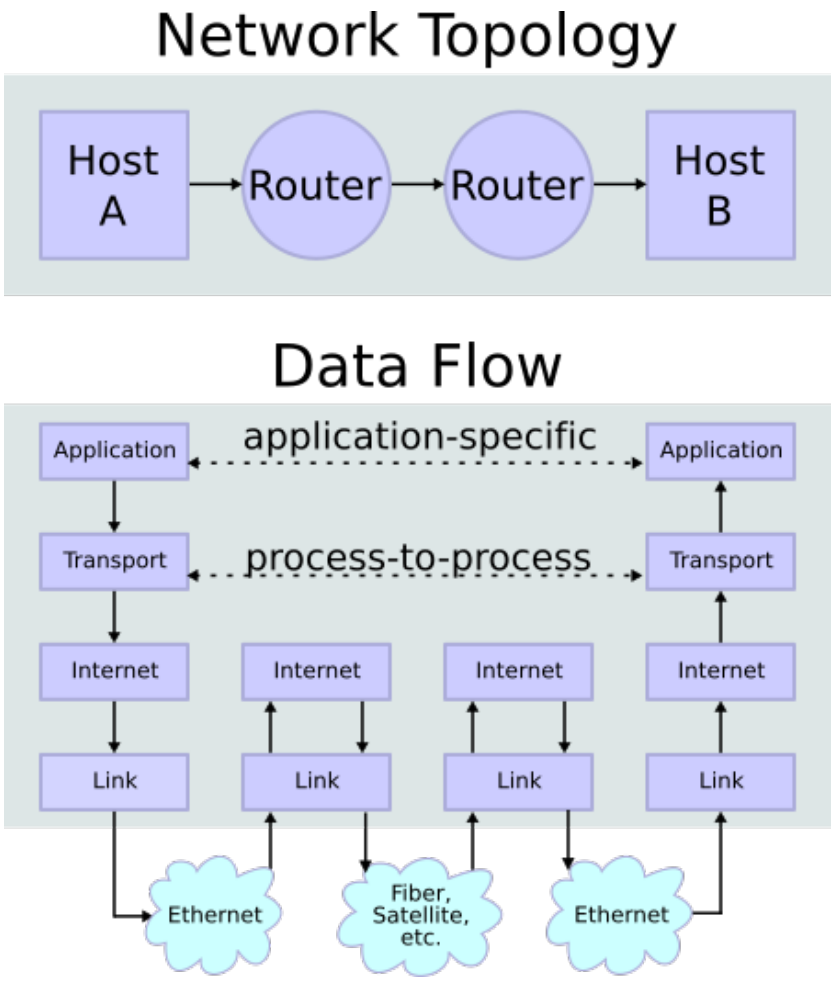

Figure 8: "IP stack connections (based on work by en:User:Kbrose and en:User:Cburnett)" by Jens Lechtenbörger under CC BY-SA 4.0; from GitLab

5.1.1 Drawing on MAC Addresses

Warning! External figure **not** included: "What's a MAC address?" under © 2016 Julia Evans, all rights reserved; from julia's drawings

(See HTML presentation instead.)

5.1.2 Drawing of Packet

Warning! External figure **not** included: "Anatomy of a packet" under © 2016 Julia Evans, all rights reserved; from julia's drawings
(See HTML presentation instead.)

5.1.3 Typical Communication Steps (0/2)

- Prerequisites
  - Internet communication requires numeric IP addresses
    - **Bindings** of human readable names and IP addresses via **DNS**
      - DNS is request-reply protocol
      - DNS client asks DNS server for IP address of name such as www.wwu.de
      - (And more)
  - LAN communication requires MAC (media access control, hardware) addresses
    - MAC address: Hardware address of network card (e.g., Ethernet, wifi)
      - E.g., 02:42:fa:5c:4a:4a
    - **Bindings** of IP addresses and MAC addresses via **ARP** (Address Resolution P.)
      - Send ARP request (broadcast) into local network, asking for MAC address of given IP addresses

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 128.176.6.250
2. Encapsulate M by adding TCP header
   - TCP **port**: Number that identifies process
     - Typically, 80 for web servers with HTTP (443 for HTTPS)
     - Random number 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 PC
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
7. Routers repeat steps (4) - (6) to forward M to final destination

5.2 Encapsulation

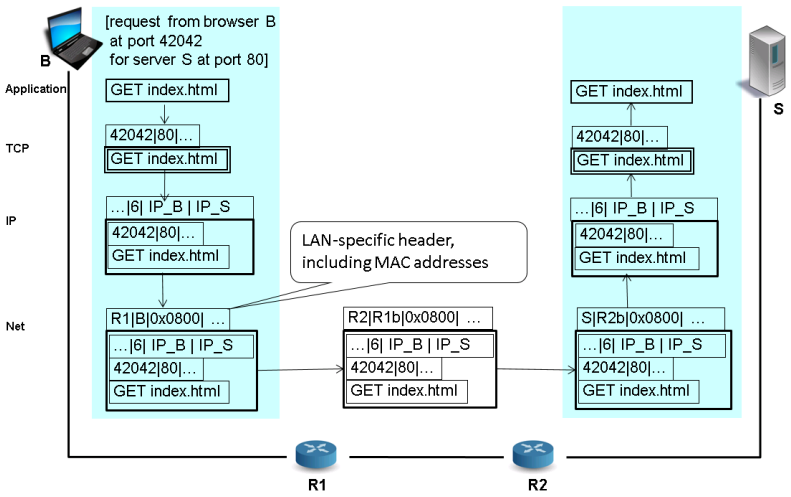

Figure 9: Sample encapsulation of GET request

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 at next higher layer
  - Different protocols use different forms of demux keys (see previous slide)

        * Ethernet header contains **type** field (IPv4 = 0x0800)
        * IP header contains **protocol** field (TCP = 6, UDP = 17)
        * TCP header contains **port** (application id) as demux key

5.4 Review Questions

- While data is transmitted over the Internet, what parts of messages remain stable? What parts of messages change where and why?
- What is the role of a demux key in Internet 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?

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 reuse, 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.)"

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 is less than ideal.
    - Consider transfer of some GB, which may take a long time
      - Small tests may identify individual bit errors early, allowing partial retries

6.5 End-to-End Security

- The above observations also apply to security goals of confidentiality and integrity
- If you want them, you cannot rely on hop-by-hop assurances
  - As offered by, e.g., IPsec, VPN, De-Mail
- You must protect your data inside your applications
  - Recall e-mail and messaging mentioned above

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. What are the communication endpoints? What trust boundaries do exist?

7 In Class Session

7.1 CACS in Practice

- Wikipedia has an article on the Internet protocol suite
  - That article shows a variant of the figure on communication based on the IP stack see earlier

        * Labelling the connection between transport layer peers with "host-to-host"
            · (As of July 12, 2018)
    - Controversy, see talk page
        * Is TCP "host-to-host" or "process-to-process"?

7.1.1 Supporting References

- Standards
    - Basic RFCs (color added by me)
        * RFC 791 (IP, Internet layer)
            · "The internet protocol provides for transmitting blocks of data called datagrams from sources to destinations, where **sources and destinations are hosts** identified by fixed length addresses."
        * RFC 793 (TCP, transport layer)
            · "The TCP provides for reliable **inter-process communication** between **pairs of processes** in host computers attached to distinct but interconnected computer communication networks."
            · Unfortunately, also "The Transmission Control Protocol (TCP) is intended for use as a highly reliable **host-to-host protocol** between hosts in packet-switched computer communication networks, and in interconnected systems of such networks."
            · Clearly, TCP adds functionality beyond the host-to-host connectivity provided by IP, in particular process-to-process communication
        * RFC 1122 (Requirements for Internet Hosts)
            · "The transport layer provides end-to-end communication services for **applications**."
            · "All Internet transport protocols use the Internet Protocol (IP) to carry data **from source host to destination host**."

7.1.2 Reference Used for Article

- In support of "host-to-host," the article references (an older edition of) [Hun10]
    - There, Figure 1-2 contains a "Host-to-Host Transport Layer"
        * "The four-level model illustrated in Figure 1-2 is based on the three layers (Application, Host-to-Host, and Network Access) shown in the DOD Protocol Model in the DDN Protocol Handbook Volume 1, with the addition of a separate Internet layer."
    - The DDN Protocol Handbook Volume 1 does not contain the phrase "DOD Protocol Model"
        * But "The DoD Protocol Architecture" in Figure 3-1 on page 1-22

        * Which contains layers for Application, Host-to-Host, and Network Access Protocols
        * However, IP and TCP are **both** classified as "Host-Host Protocols"
            · As IP is contained, there is no need for "the addition of a separate Internet layer" as claimed in [Hun10]
            · Hence, [Hun10] is not a good source for this particular question

7.1.3 Alternative References

- [PD11]
    - Fig. 1.8 on p. 25 with "Example of layered network system" showing four layers
        * Application programs at top, process-to-process channels on next layer, host-to-host connectivity on next layer, hardware at bottom
    - P. 33: "The transport layer then implements what we have up to this point been calling a process-to-process channel."
- [Cou+11]
    - P. 122
        * "The first characteristic to note is that, whereas IP supports communication between pairs of computers (identified by their IP addresses), TCP and UDP, as transport protocols, must provide process-to-process communication. This is accomplished by the use of ports. [...] Once an IP packet has been delivered to the destination host, the TCP- or UDP-layer software dispatches it to a process via a specific port at that host."

7.2 Wireshark Demo

- Wireshark is free software
    - `https://www.wireshark.org/`
- Analyze network traffic in real-time
    - Trouble-shooting
    - Understanding applications and protocols
        * What data is sent where?
        * How does encapsulation really look like?

7.2.1 Wireshark Filters

- **Capture** filter
    - Specify among capture options, restrict what is being captured

    - Three qualifiers: **type** (`host`, `net`, `port`), **dir** (`src`, `dst`), **proto** (`ip`, `tcp`, `udp`, `arp`, ...)
    - Boolean combinations with `and`, `or`, `not`, ...
  - Examples
    - `port 53`: Source or destination port is 53
    - `host www.uni-muenster.de`: Source or destination host has given name; also IP address instead of name possible
    - `dst host 128.176.0.12 and udp dst port 53`
- **Display** filter
  - Specify "Filter:" on main window, restrict what is being displayed
  - Go to Packet Details portion, select piece of information
    - E.g., TCP flags, right click → "Apply as Filter"

8 Conclusions

8.1 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

References

[BC01] Marjory S. Blumenthal and David D. Clark. "Communications Policy in Transition". In: 2001. Chap. Rethinking the Design of the Internet: The End-to-end Arguments vs. The Brave New World, pp. 91–139. URL: http://dl.acm.org/citation.cfm?id=566696.566700.

[Cou+11] George Coulouris et al. *Distributed Systems: Concepts and Design*. 5th. USA: Addison-Wesley Publishing Company, 2011. URL: http://www.cdk5.net/.

[Hun10] Craig Hunt. *TCP/IP Network Administration*. O'Reilly Media, 2010. URL: https://www.oreilly.com/library/view/tcpip-network-administration/0596002971/ch01.html.

[PD11] Larry L. Peterson and Bruce S. Davie. *Computer Networks, Fifth Edition: A Systems Approach*. 5th. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2011. URL: https://booksite.elsevier.com/9780123850591/.

[SRC84] J. H. Saltzer, D. P. Reed, and D. D. Clark. "End-to-end Arguments in System Design". In: *ACM Trans. Comput. Syst.* 2.4 (1984), pp. 277–288. URL: http://web.mit.edu/Saltzer/www/publications/endtoend/endtoend.pdf.

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

License Information