The Internet

Jens Lechtenbörger

Neuland im Internet 2019

Contents

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 to inspect network traffic

1.3 General Importance of Internet

- The Internet is everywhere
  - Decentralized, heterogeneous, evolving

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

  - 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?

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, fibre, nothing)
- or two or more networks connected by one or more nodes (with necessary links)
  - (e.g., gateway, router)

Figure 2: Figure under CC0 1.0

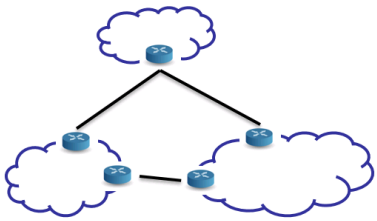

Figure 3: Figure under CC0 1.0

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**

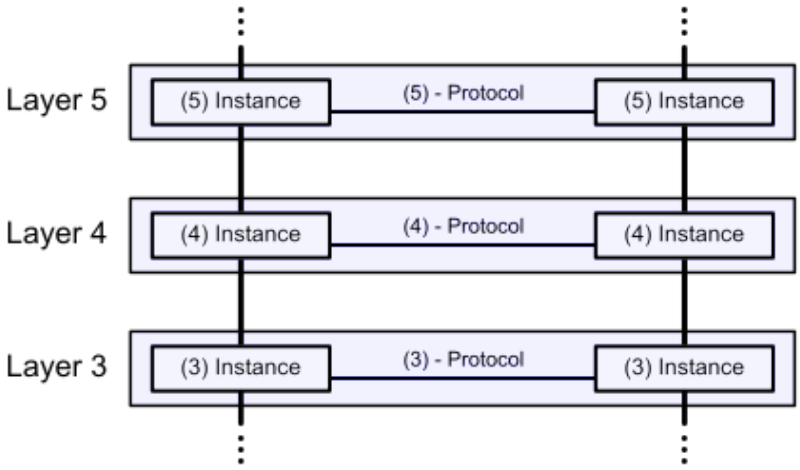

Figure 4: "Layered Communication in OSI Model" by Runtux under Public domain; from Wikimedia Commons

  - 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

(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 Drawing for OSI Model

Warning! External figure **not** included: "Networking layers" © 2016 Julia Evans, all rights reserved from julia's drawings. Displayed here with personal permission.
(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 from source host over multiple links to destination host
    - Host layers
      4. Provides end-to-end delivery
      5. Manages task dialogs
      6. Converts different representations
      7. Provides functions needed by users/applications

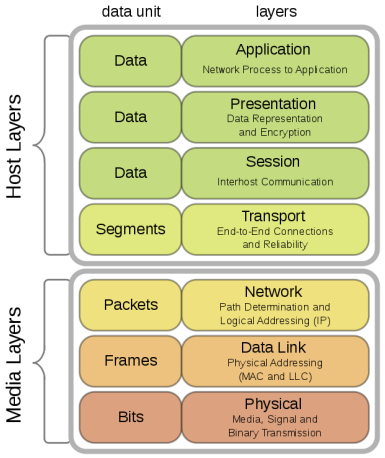

Figure 5: "OSI Model" by Offnfopt under CC0 1.0; from Wikimedia Commons

4.3 OSI Model on Internet

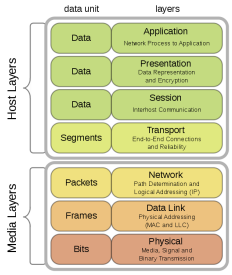

Figure 6: "OSI Model" by Offnfopt under CC0 1.0; from Wikimedia Commons

- OSI vs Internet
  - 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
  - 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."

4.4.1 Internet Architecture

- "Hourglass design"

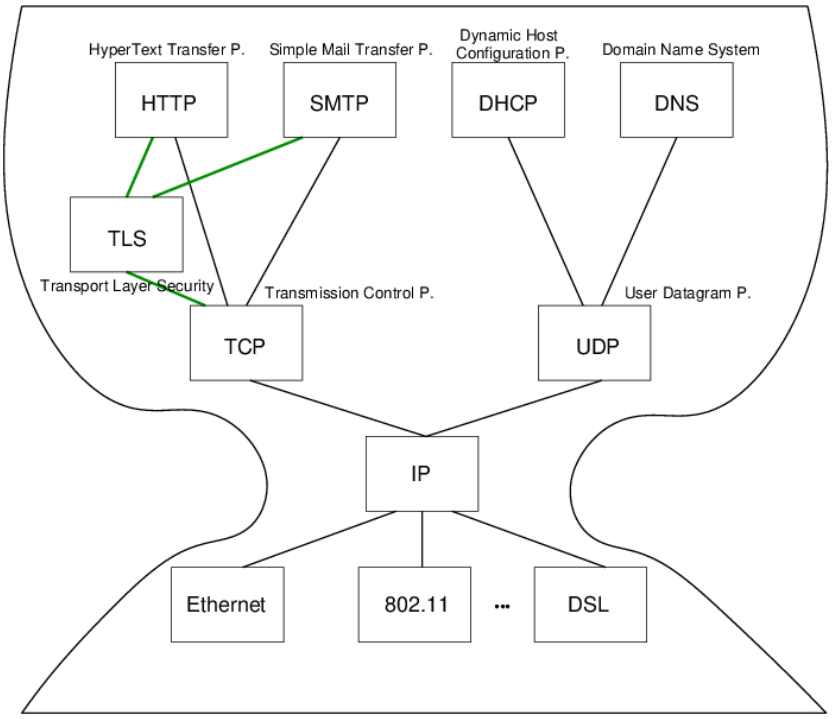

Figure 7: Internet Architecture with narrow waist

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

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 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.

4.4.3 Drawing on TCP

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

5 Internet Communication

5.1 IP Stack Connections

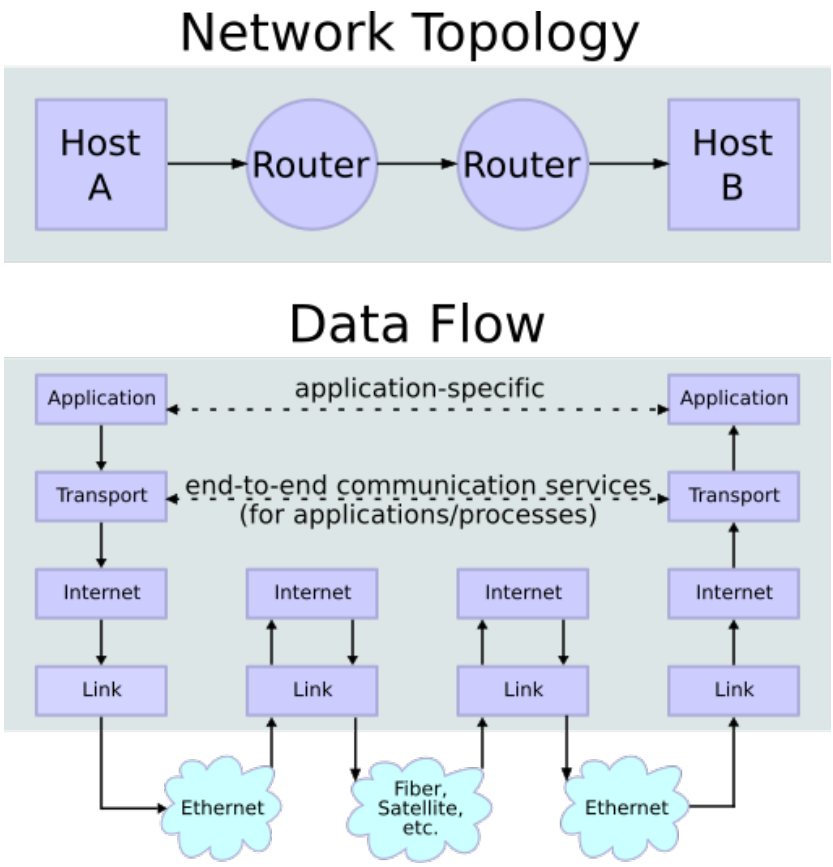

Figure 8: "IP stack connections" by Jens Lechtenbörger under CC BY-SA 4.0; based on work under CC BY-SA 3.0 by en:User:Kbrose and en:User:Cburnett by changing arrow labels; from GitLab

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

Warning! External figure **not** included: "What's a MAC address?" © 2016 Julia Evans, all rights reserved from julia's drawings. Displayed here with personal permission.
(See HTML presentation instead.)

5.1.2 Drawing of Packet

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

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 (media access control, hardware) addresses
    - MAC address: Hardware address of network card (e.g., Ethernet, WiFi)
      - E.g., 02:42:fa:5c:4a:4a
    - **Lookup** of MAC addresses for IP 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 IP address 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

– Routers repeat steps (4) - (6) to forward M to final destination

5.2 Encapsulation

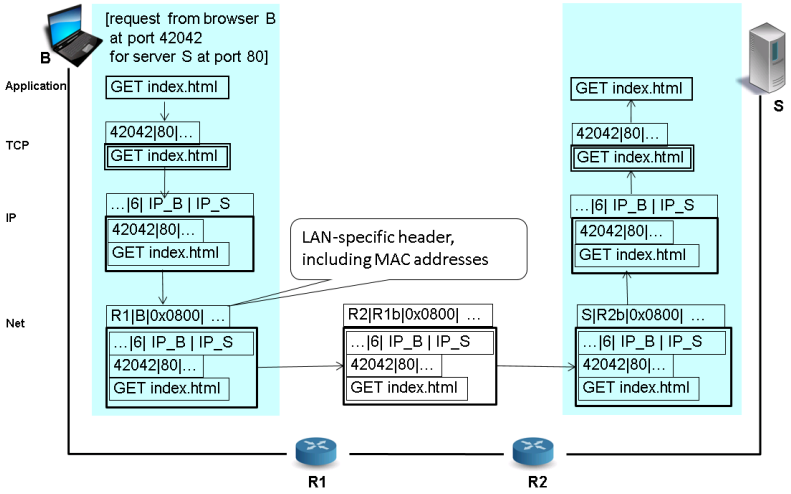

Figure 9: Sample encapsulation of GET request

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 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

6 In Class Session

6.1 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?

6.1.1 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"

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

[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/.

[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