Networks and Communications

"Introduction and Basic Concepts"

Konstantinos Gkoutzis
Imperial College

Outline

Imperial College London

- General Information
- Introduction
  - The Internet
- Basic Concepts
  - Circuit Switching & Packet Switching
  - Protocols & Services
  - Connection-oriented & Connectionless
  - Latency & Throughput
- The OSI Model
- The TCP/IP Model

Course Instructors

Imperial College London

- **Konstantinos Gkoutzis**
  - kgk{*at*}ic.ac.uk
  - www.doc.ic.ac.uk/~kgk
  - Huxley 228
- **Peter Pietzuch**
  - prp{*at*}doc.ic.ac.uk
  - www.doc.ic.ac.uk/~prp
  - Huxley 442 (*by appointment*)
- **William Culhane**
  - Research Associate
- **Federico Morini**
  - PhD Teaching Scholar

Course Schedule

Imperial College London

Week	Topic(s)	Instructor
2	Introduction and Basic Concepts	Konstantinos
3	Application Layer	Konstantinos
4	Transport Layer	Konstantinos
5	Security	Konstantinos
6	Practical Demonstration: Wireshark	Federico
		William
	Future	Konstantinos

Week	Topic(s)	Instructor
7	Network Layer	Peter
8	Information Theory	Peter
9	Physical Layer	Peter
10	Data Link Layer	Peter

- Always check CATe and the DoC/IC websites for up-to-date information and resources
- DoC: www.doc.ic.ac.uk/~kgk/212/
- IC: http://www.imperial.ac.uk/computing/current-students/courses/212/lectures-and-tutorials/

Course Structure

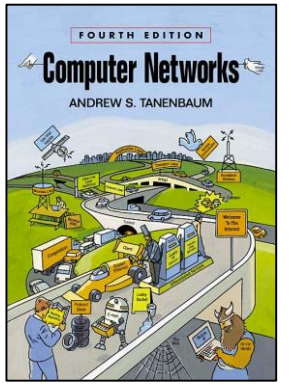
- **Course format**
  - Lectures and unassessed worksheets (*sometimes worksheet materials mixed into lectures*)
  - One assessed coursework exercise – early February (*Week 5*)

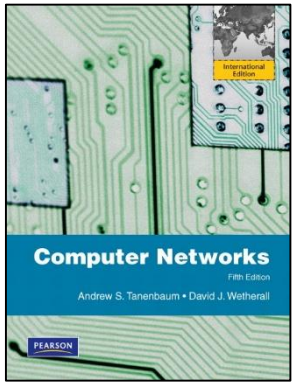
- **Exam questions**
  - Not about low-level details:
    - **Q**: "What is the 50th bit in the IP packet header?"
    - **A**: "It's the `Don't fragment' bit`" (well done; full marks)
  - But rather about principals and design trade-offs:
    - **Q**: "You need to design a transport layer for a network with the following characteristics: [...] How would you do this?"
    - **A**: "I'd use a reliable transport service, similar to TCP, because..."
  - Always explain your reasoning
    - Some questions will involve calculations
  - The worksheets are a good preparation
    - Please ask questions!

Bibliography – Core

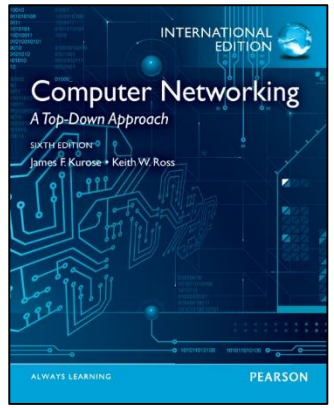
Imperial College London

- "Computer Networks", by Andrew S. Tanenbaum
  - and David J. Wetherall in latest edition
  - 5th edition released in 2010 by Pearson
  - but if you use the 4th instead (2002), it also covers most topics





- "Computer Networking: A Top-Down Approach", by James Kurose and Keith Ross
  - 7th edition just released (2016) by Pearson
  - 6th (2013) is also fine
  - E-book available via the Imperial College Library website



- You are not expected to read these from cover to cover
  - start by focusing on the topics addressed in the slides
  - expand from there

- Full Reading List can be found here: http://bit.ly/212ReadingList

Bibliography – Supplementary

- "Computer Networks: A Systems Approach"
  - by Larry L. Peterson (with contributions by Bruce S. Davie)
  - Morgan Kaufmann, 5th edition, 2012

- "Data and Computer Communications"
  - by William Stallings (with contributions by Moumita Mitra Manna)
  - Pearson, 10th edition, 2014

- "Network Security Essentials: Applications and Standards"
  - by William Stallings (with contributions by B.R. Chandavarkar)
  - Pearson, 5th edition, 2013

- Many free online resources also available on the Web
  - Example: http://cnp3book.info.ucl.ac.be/secondedition.html

What is Computer Networking?

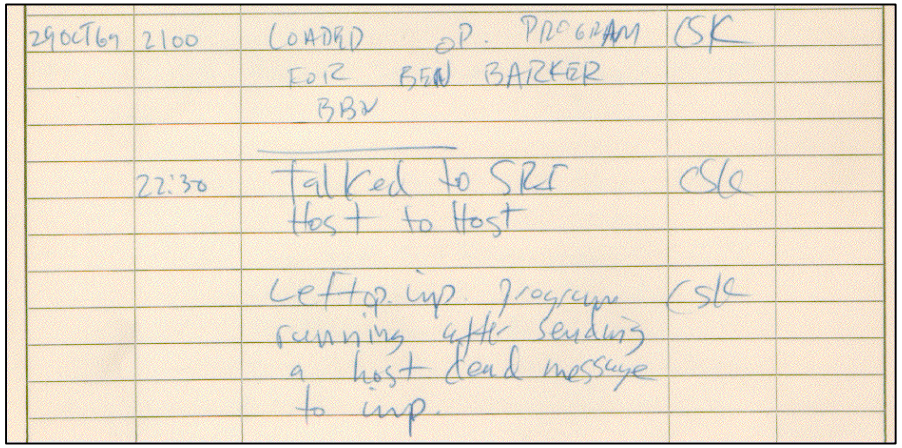
- **Computer Networking** is the process of *interconnecting* computer systems via **telecommunication** methods
  - to *exchange data* and *share resources*
- Computer networks are becoming pervasive (*i.e. they are almost everywhere*)
- Most mainstream software systems are distributed systems
  - e.g. cloud computing (*Amazon AWS EC2*), smartphone apps
- **Performance often depends on network usage**
  - every (*serious*) developer must know the basics of networking
- The Internet is used daily by countless people
  - Web, Email, Voice/Video, Games, BitTorrent
  - and it is getting bigger and bigger!

Question:
How old is the Internet?

Answer:
It turns 48 this year!

Internet Evolution

- 1969
  - September
    - 1 node
  - October
    - First message sent on the ARPANET at about 22:30 on the 29th
    - Message sent from host in UCLA to host in SRI (Stanford)
    - Message text was the word login
    - After the "l" and the "o" were transmitted, the system crashed
    - Upon reboot, one hour later, it worked fine :-)
  - December
    - 4 nodes



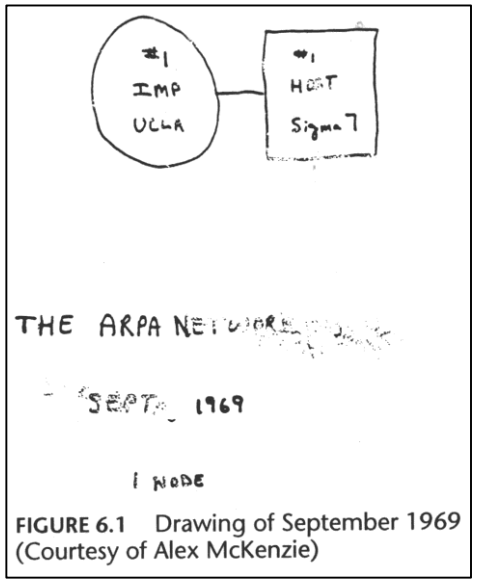


FIGURE 6.1 Drawing of September 1969 (Courtesy of Alex McKenzie)

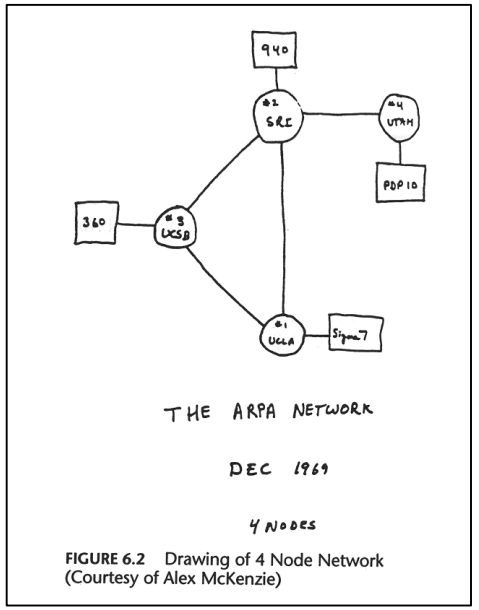
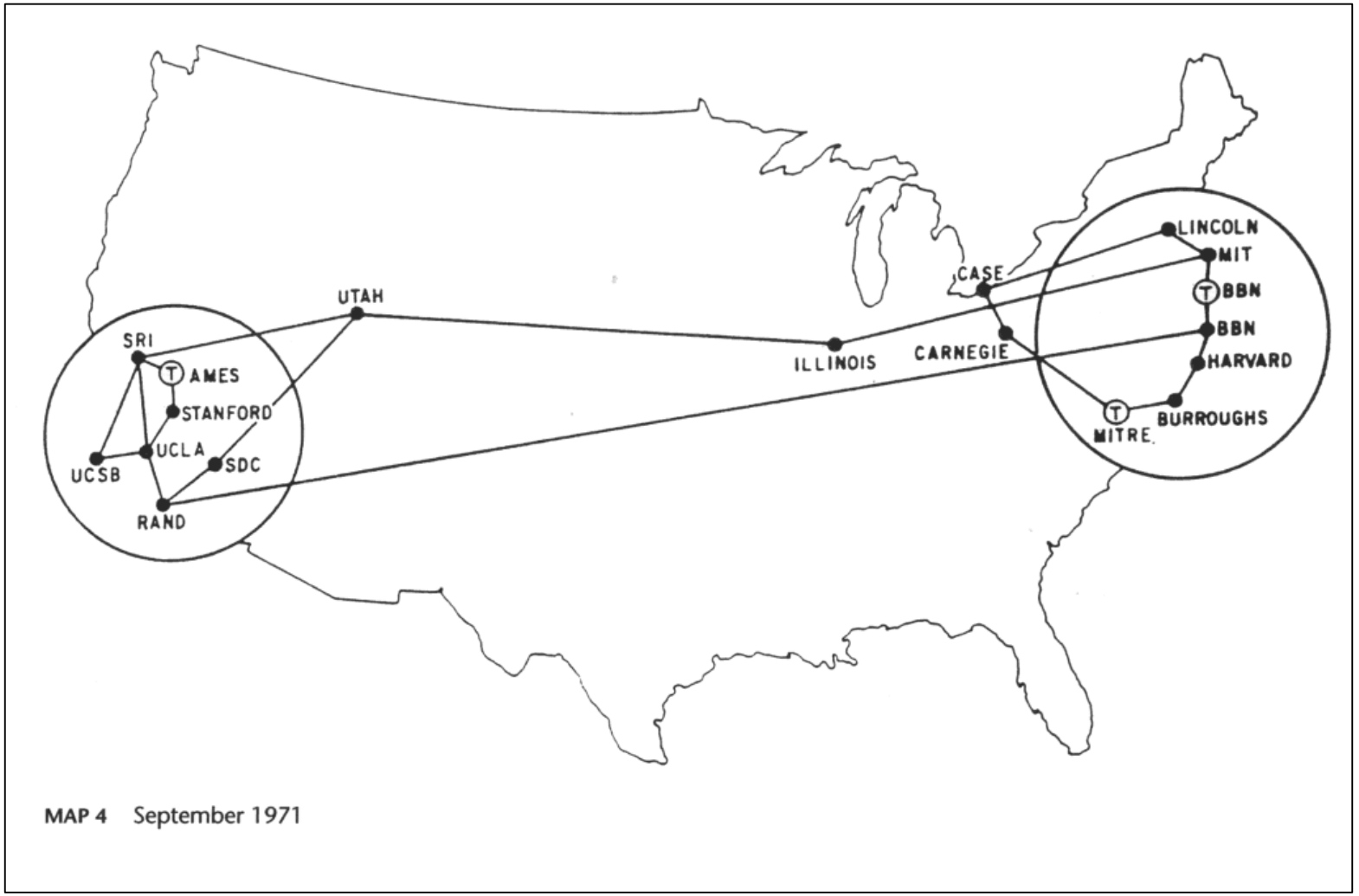


FIGURE 6.2 Drawing of 4 Node Network (Courtesy of Alex McKenzie)

Internet Evolution (cont'd)

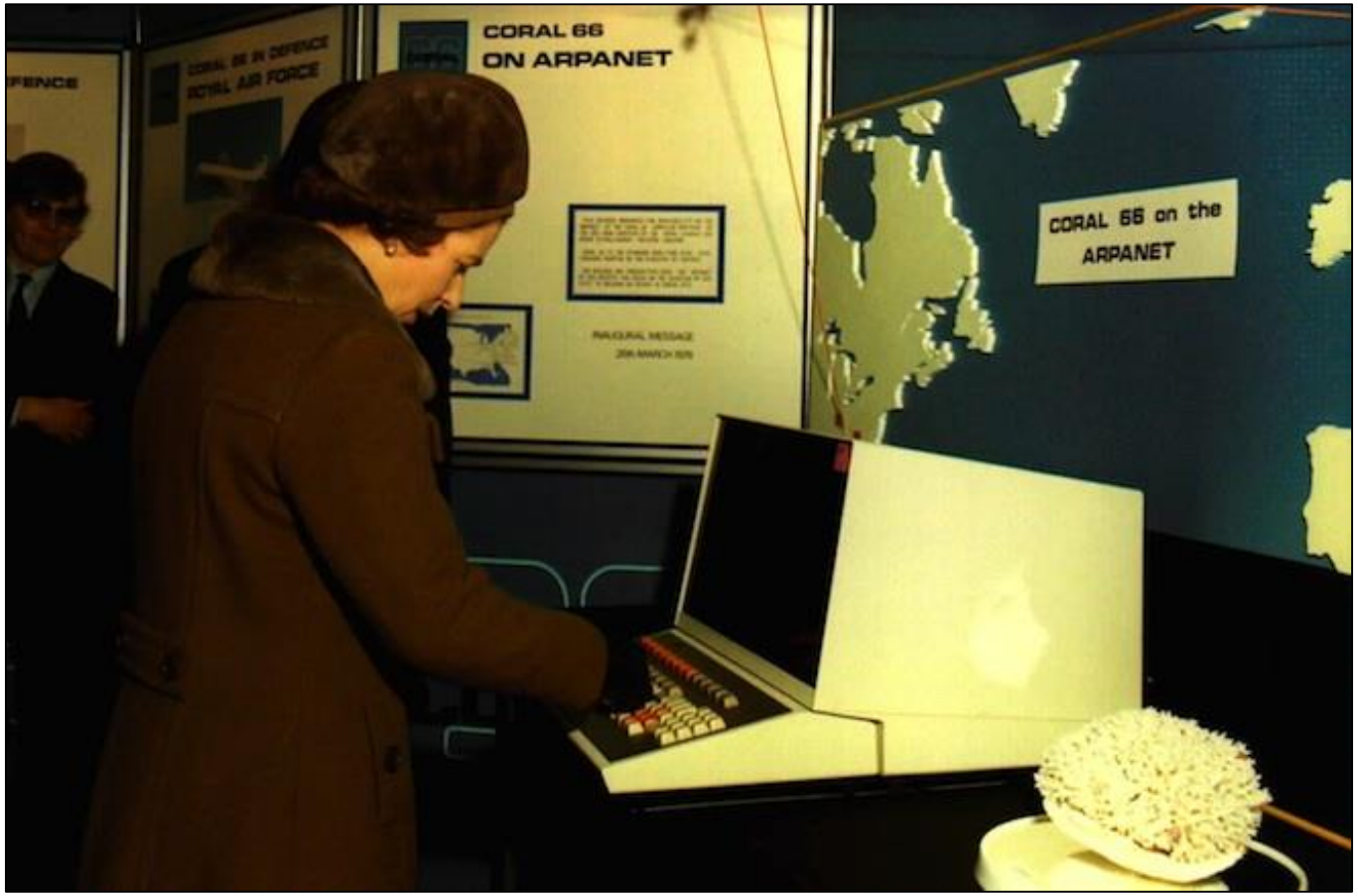
- 1971



*Salus, P.H. and Vinton, G., 1995. Casting the Net: From ARPANET to Internet and Beyond. Addison-Wesley Longman Publishing.

Internet Evolution (cont'd)

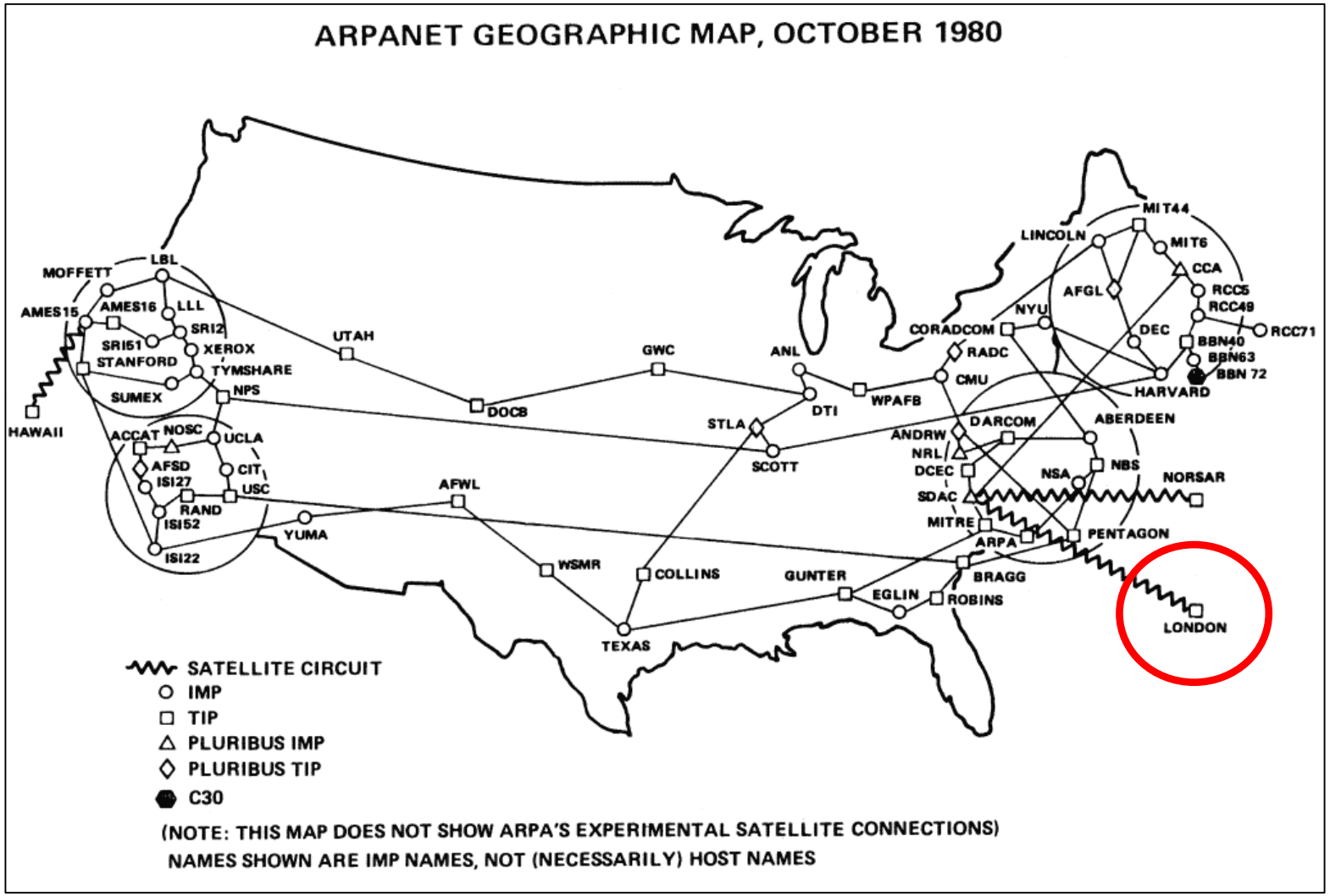
- 1976
  - March 26: Queen Elizabeth II sends an e-mail



*More interesting facts: http://www.computerhistory.org/timeline/networking-the-web/

Internet Evolution (cont'd)

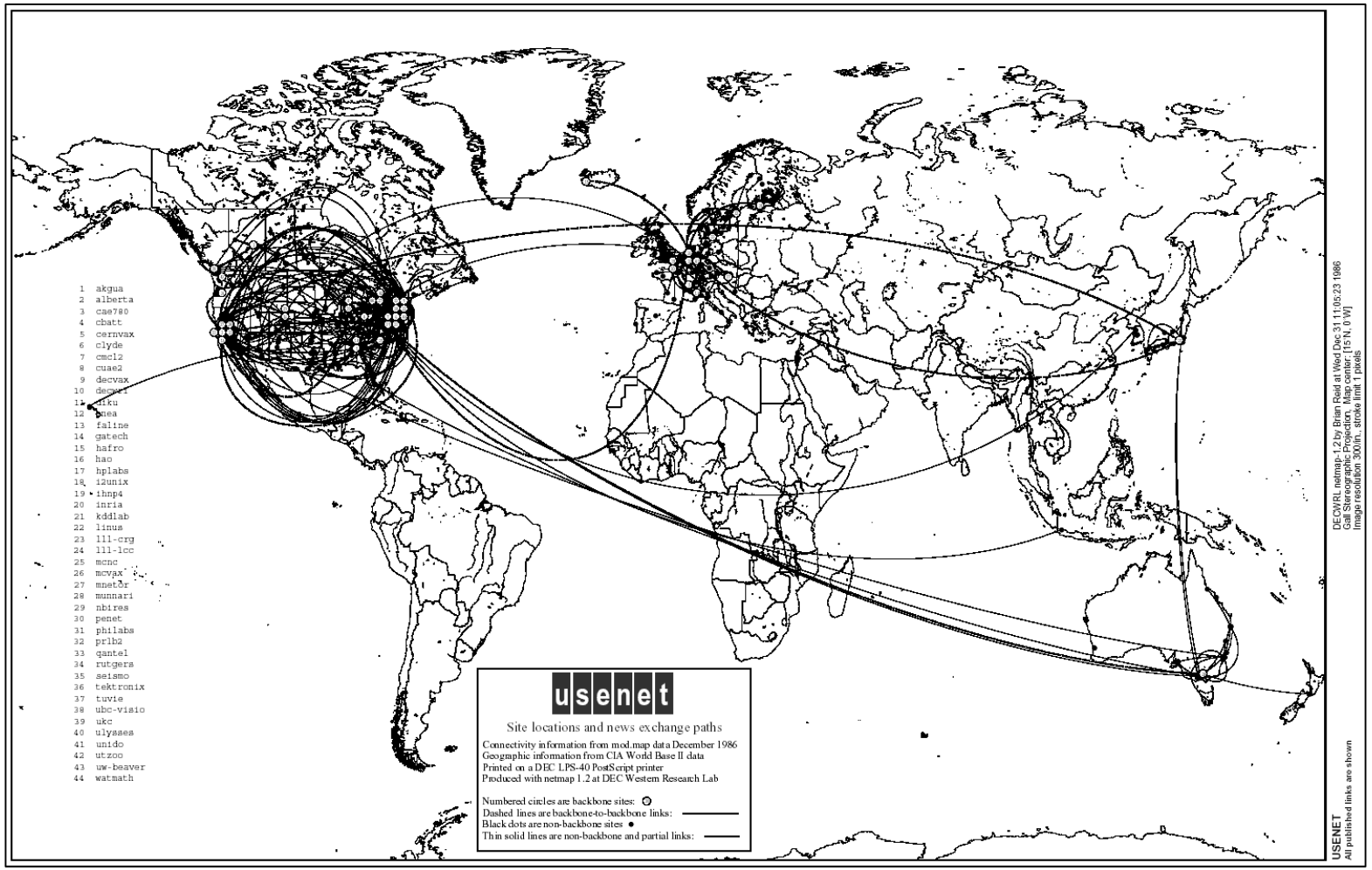
- 1980



**Salus, P.H. and Vinton, G., 1995. Casting the Net: From ARPANET to Internet and Beyond. Addison-Wesley Longman Publishing.*

Internet Evolution (cont'd)

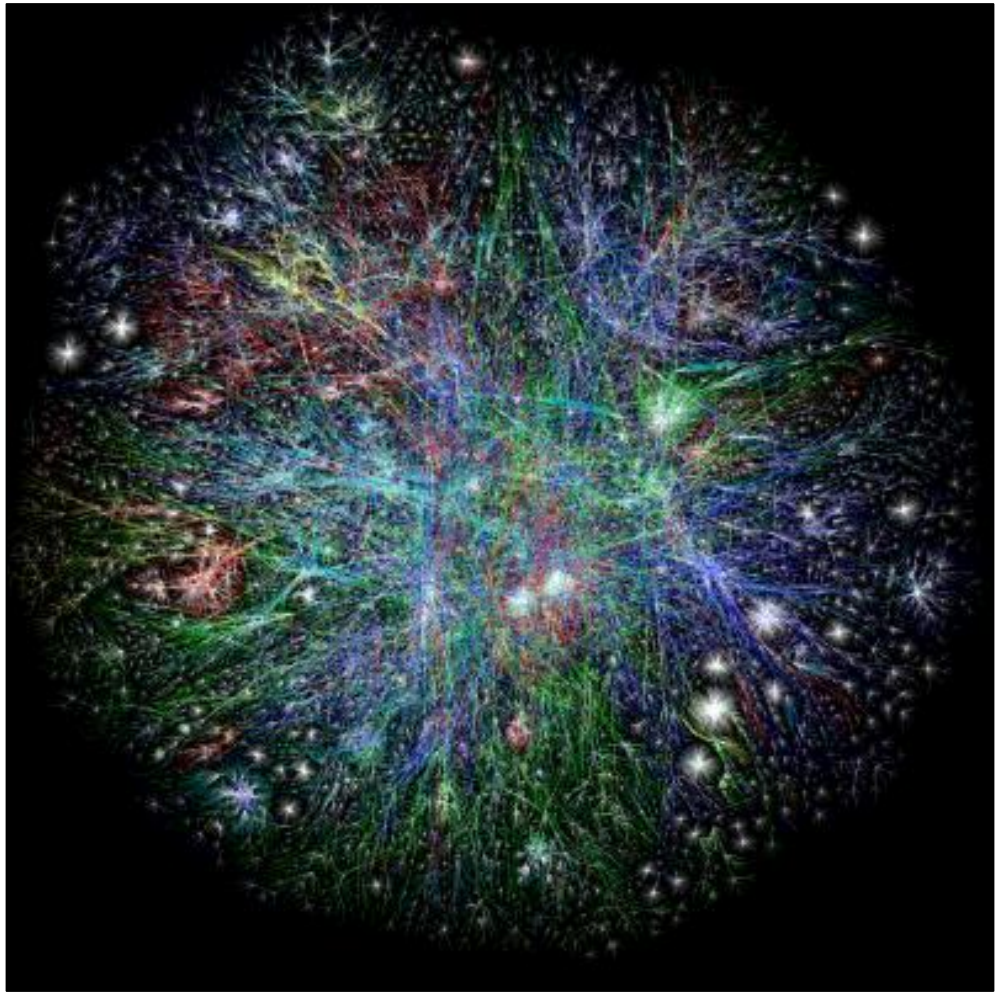
- 1986



*More maps can be found here: http://mappa.mundi.net/maps/maps_021/

Internet Evolution (cont'd)

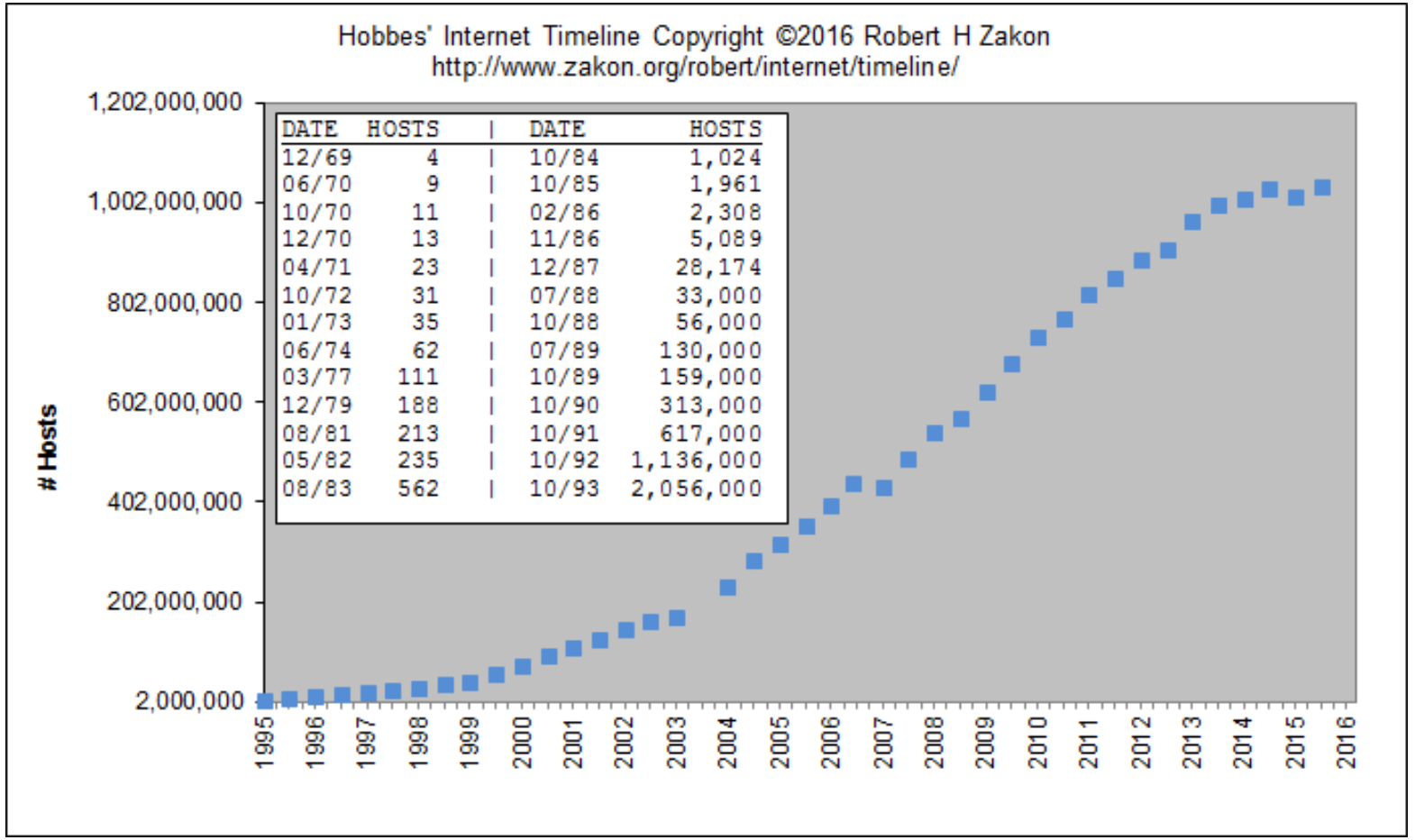
- 2005



*More maps can be found here: https://www.wired.com/2015/06/mapping-the-internet/

Internet Evolution (cont'd)

- 2016
  - Number of Hosts

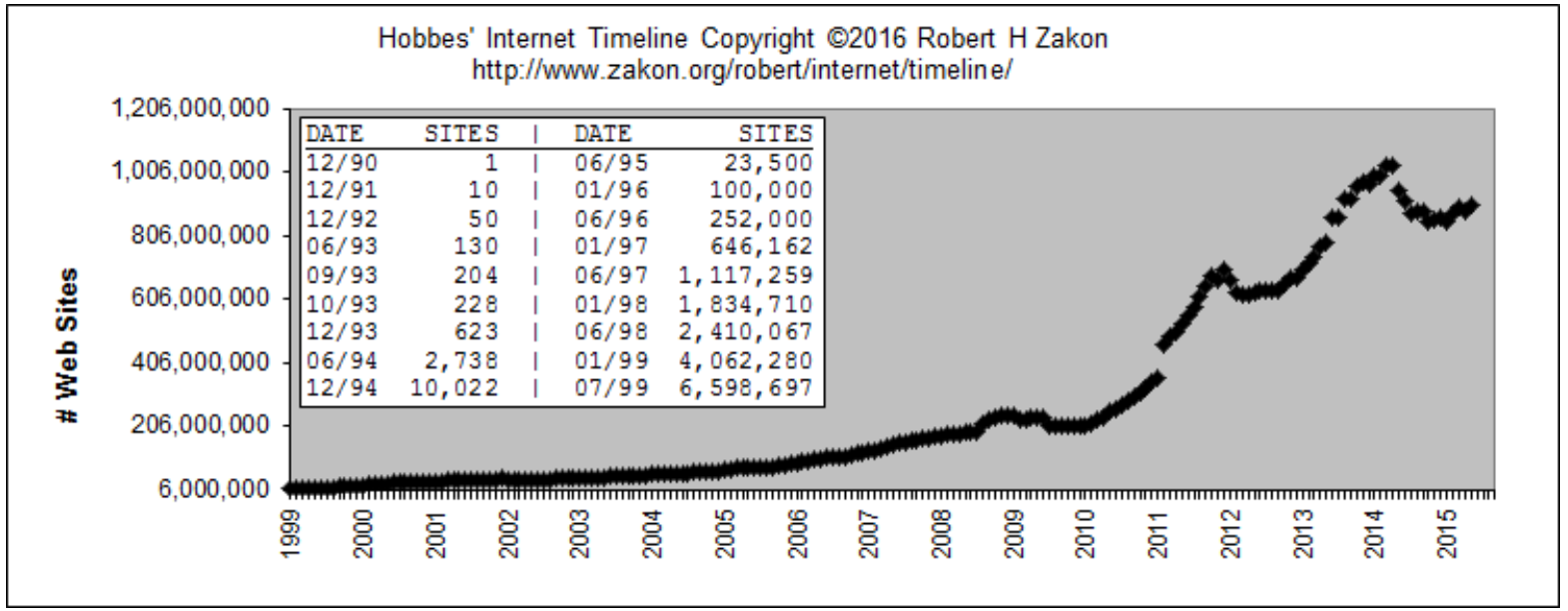


DATE	HOSTS	DATE	HOSTS
12/69	4	10/84	1,024
06/70	9	10/85	1,961
10/70	11	02/86	2,308
12/70	13	11/86	5,089
04/71	23	12/87	28,174
10/72	31	07/88	33,000
01/73	35	10/88	56,000
06/74	62	07/89	130,000
03/77	111	10/89	159,000
12/79	188	10/90	313,000
08/81	213	10/91	617,000
05/82	235	10/92	1,136,000
08/83	562	10/93	2,056,000

*Source: https://www.zakon.org/robert/internet/timeline/

Internet Evolution (cont'd)

- 2016
  - Number of Websites



DATE	SITES	DATE	SITES
12/90	1	06/95	23,500
12/91	10	01/96	100,000
12/92	50	06/96	252,000
06/93	130	01/97	646,162
09/93	204	06/97	1,117,259
10/93	228	01/98	1,834,710
12/93	623	06/98	2,410,067
06/94	2,738	01/99	4,062,280
12/94	10,022	07/99	6,598,697

- Early 2017: 1.800+ Billion hostnames (*but only 172 Million are active**)

**More interesting facts: https://news.netcraft.com/archives/category/web-server-survey/*

Networks are Everywhere!

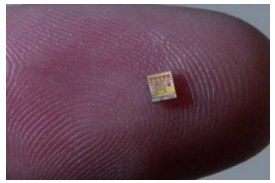
- Mobile devices

- Home networks
  - Laptops, TVs, game consoles, audio systems, ...

- Sensor networks
  - embedded computing
  - environmental monitoring

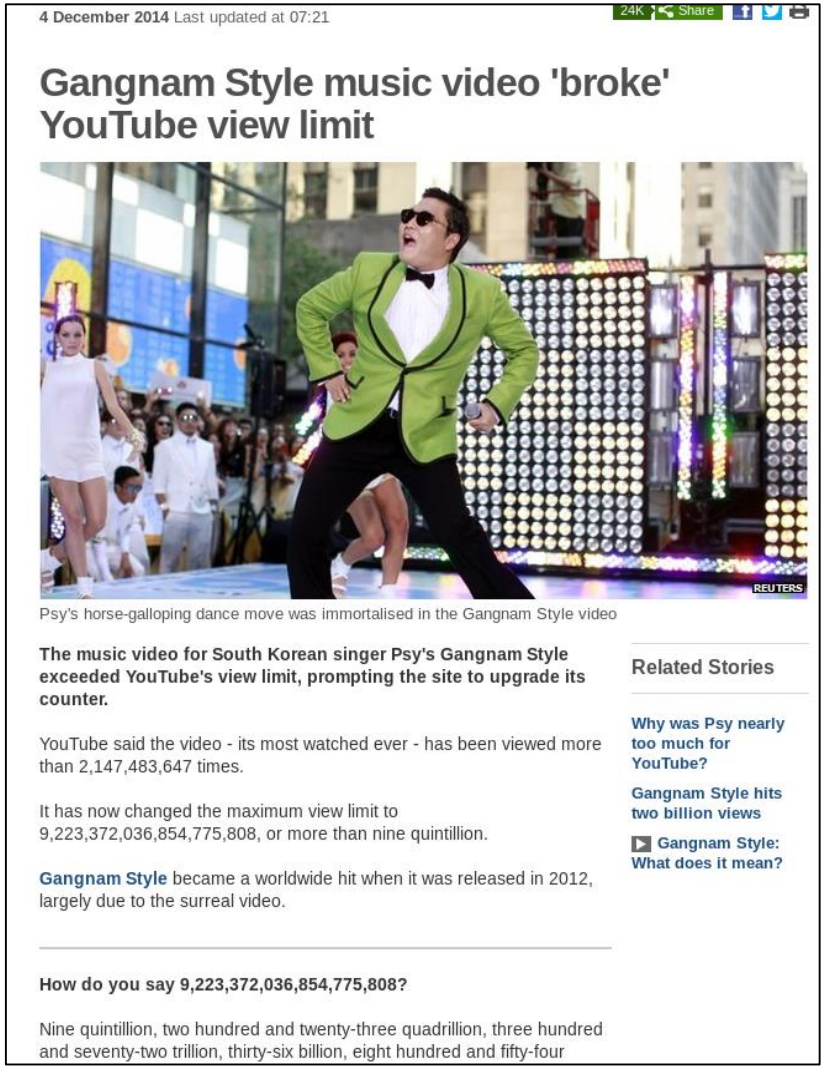
- Vehicular Networks
  - self-driving cars

- Banks, train stations, street lights, and many-many more!



Networks are Everywhere! (cont'd)

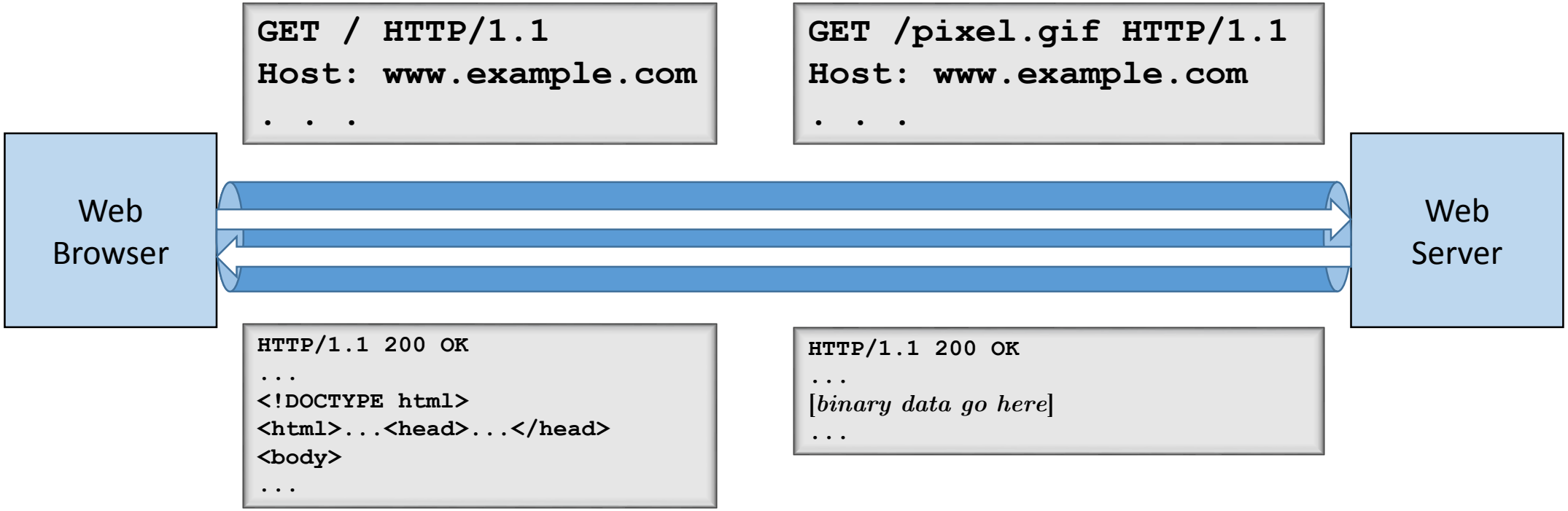
- Example:



Internet's "Killer App"

- The [**W**orld **W**ide] **W**eb (*WWW*)
- Invented by Sir Tim Berners-Lee while working at CERN
- Based on HTTP (**H**yper**T**ext **T**ransfer **P**rotocol)

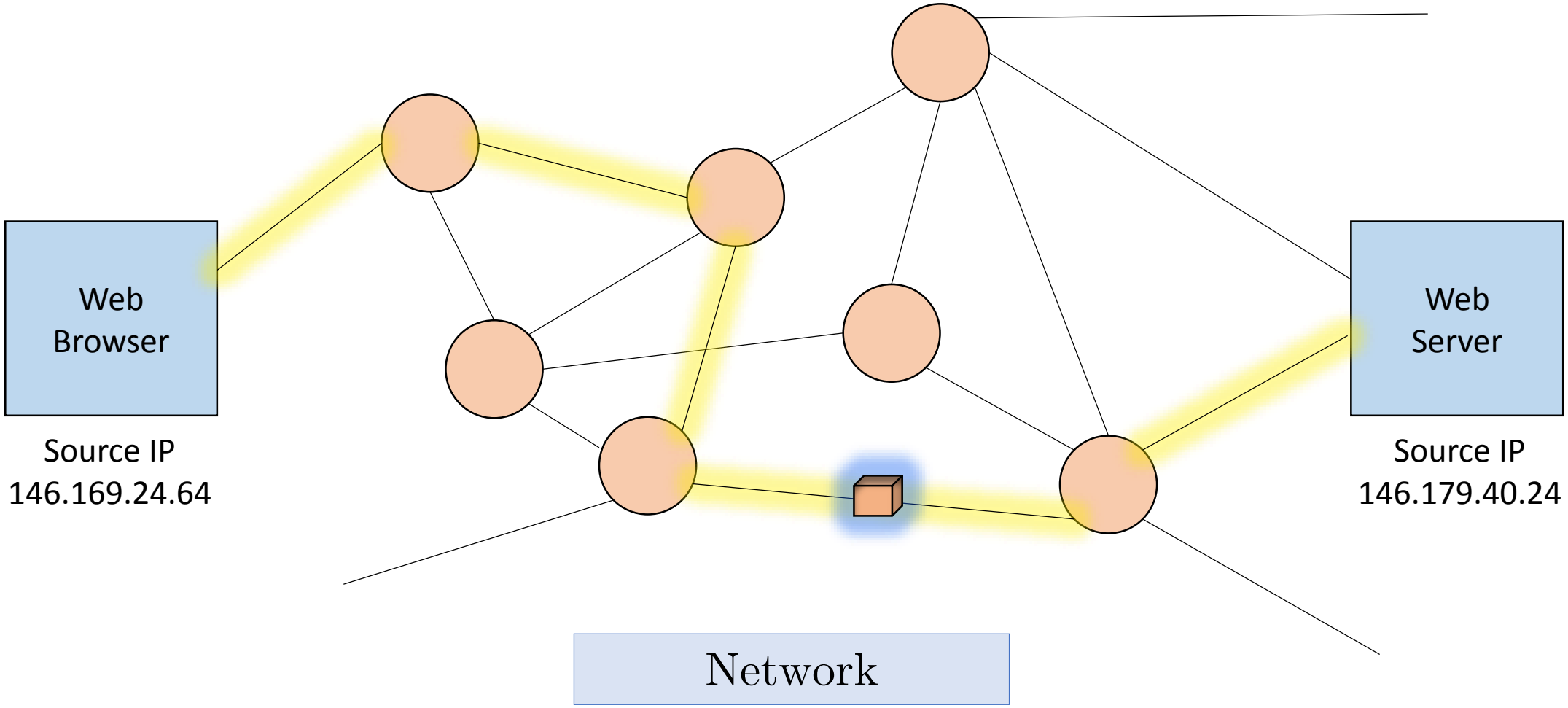
Application



Packets, Segments, Datagrams

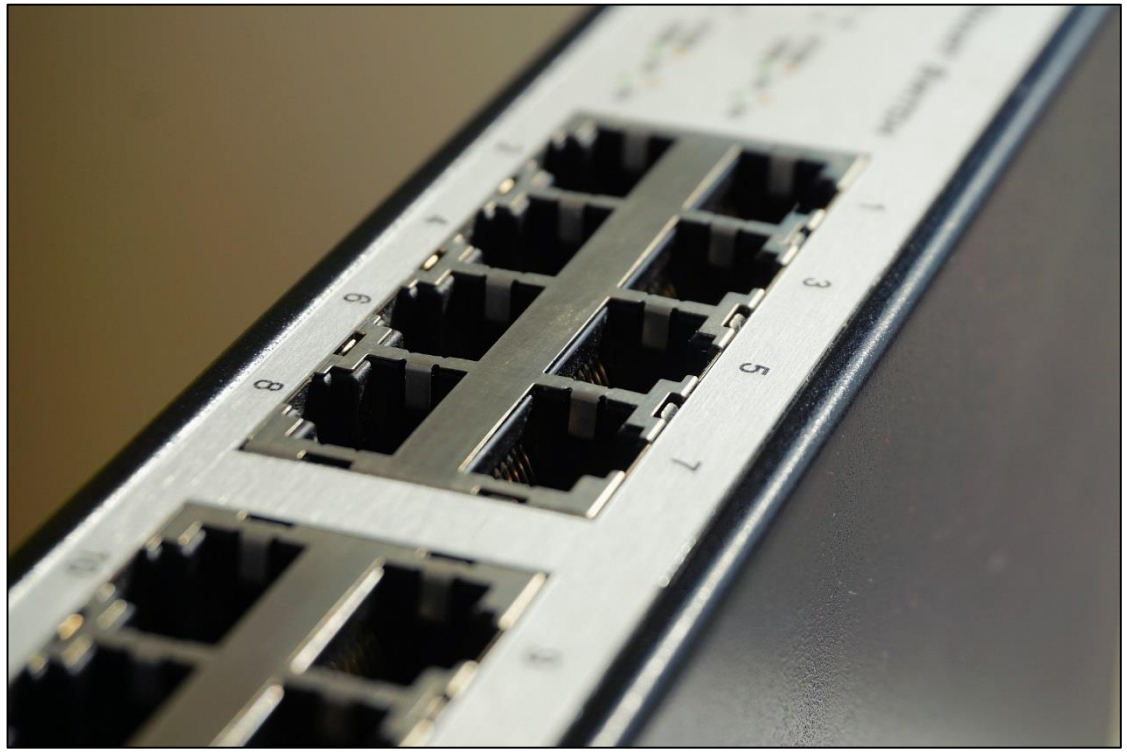
Transport

Interconnections and Paths



Communication Links

- Network Interface Controllers (NICs)
  - Wired
    - Ethernet
    - Fiber Optic network card
    - [...]
  - Wireless
    - WiFi Access Points
    - 4G USB dongle
    - [...]



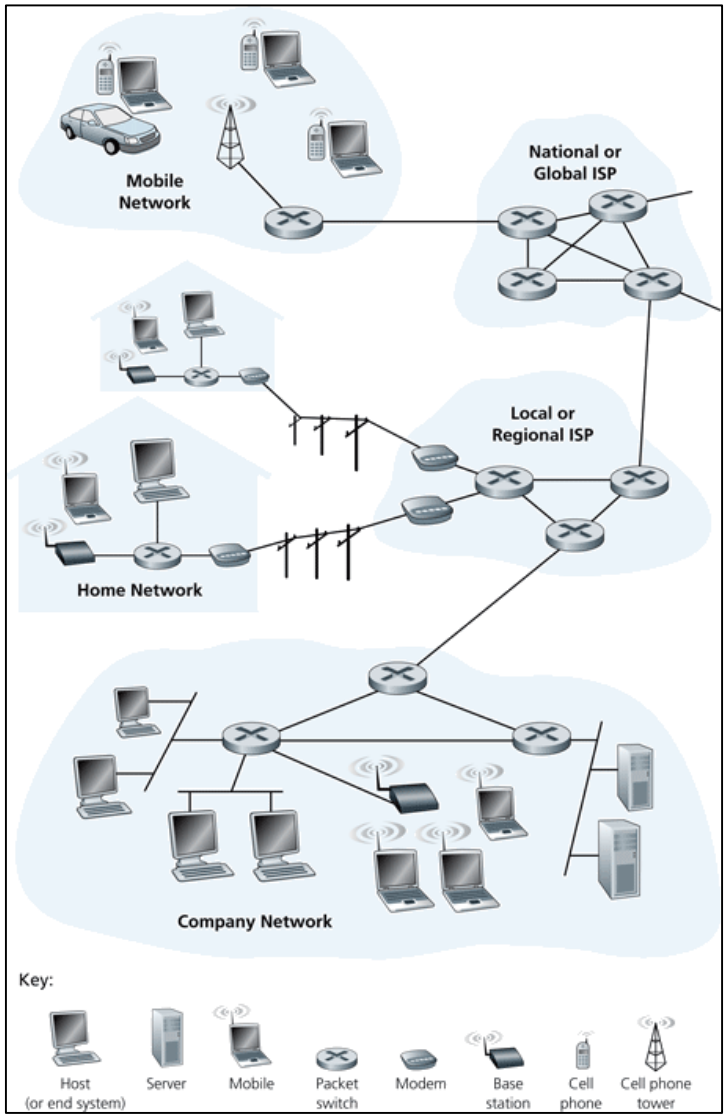
Link

Communication Media

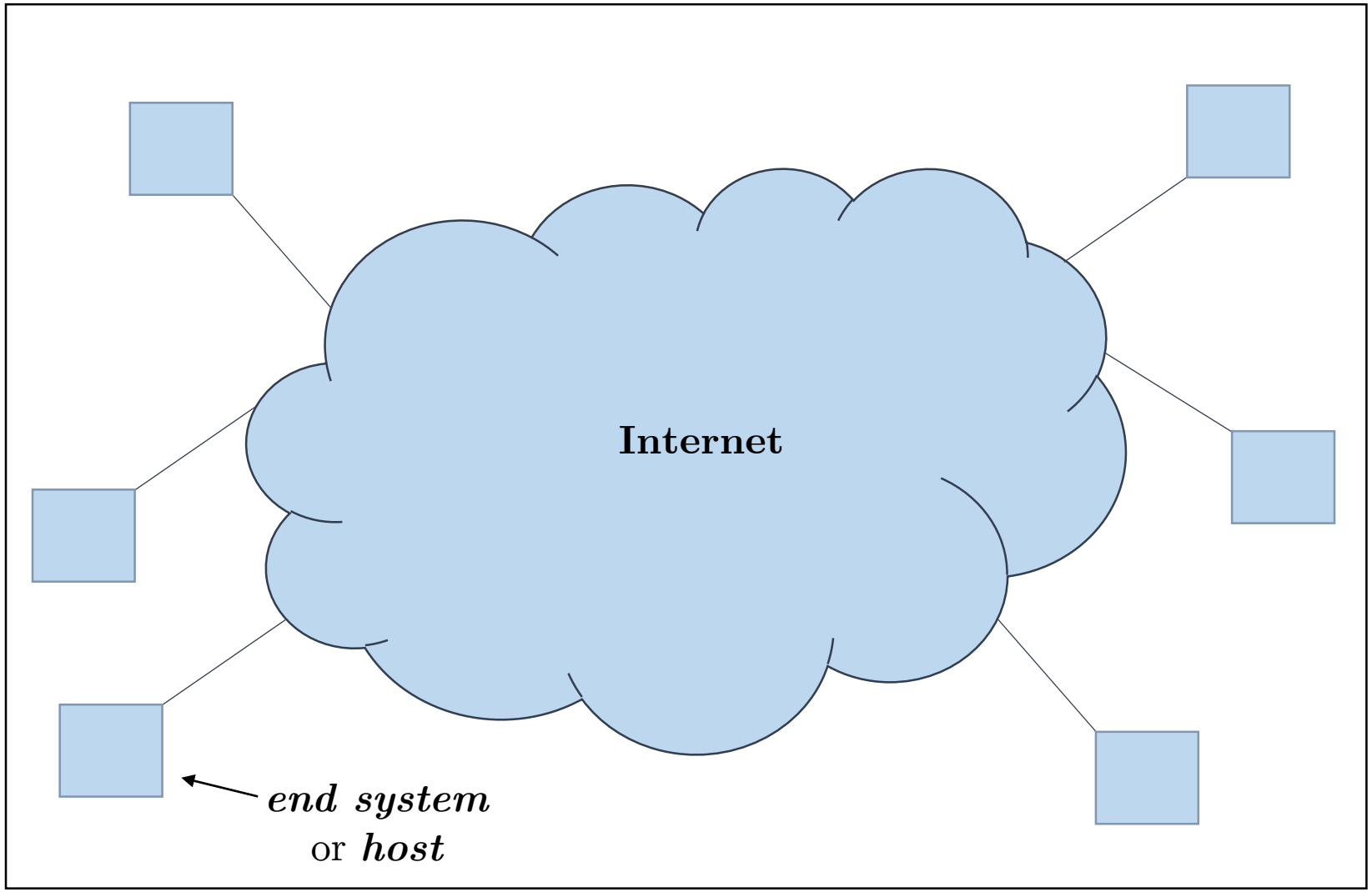
- Various types and forms of media:
  - Fiber-optic cable
  - Twisted-pair copper wire
  - Coaxial cable
  - Wireless local-area links (*e.g. 802.11, Bluetooth*)
  - Satellite channel
  - [...]

Physical

What is the Internet?



What is the Internet? (cont'd)



End Systems

- End system
  - a Web server
  - a laptop
  - a smartphone
  - . . .
  - a camera (a.k.a. webcam)
  - a temperature sensor
  - . . .
  - a car
  - a television set
  - a picture frame
  - a toaster
  - . . .
  - a toilet seat?
  - an arm and a leg?

IP picture frame

Internet refrigerator

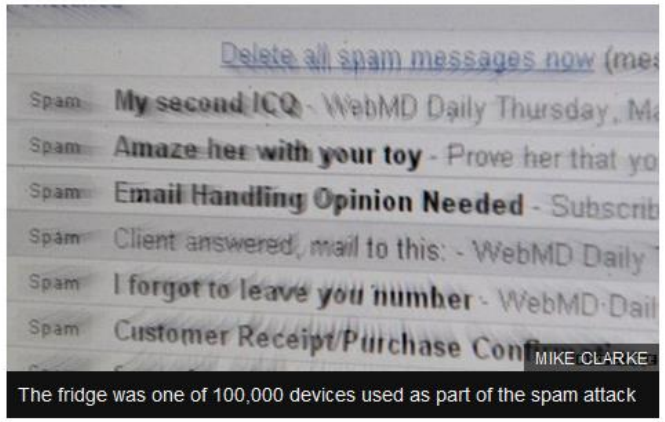
Web-enabled toaster + weather forecaster

End Systems (cont'd)

- Ease-of-access
  - vs. Security
- Big Data
  - vs. Privacy

Fridge sends spam emails as attack hits smart gadgets

17 January 2014 | Technology



The fridge was one of 100,000 devices used as part of the spam attack

A fridge has been discovered sending out spam after a web attack managed to compromise smart gadgets.

The fridge was one of more than 100,000 devices used to take part in the spam campaign.

Uncovered by security firm Proofpoint the attack compromised computers, home routers, media PCs and smart TV sets.

The attack is believed to be one of the first to exploit the lax security on devices that are part of the "internet of things".

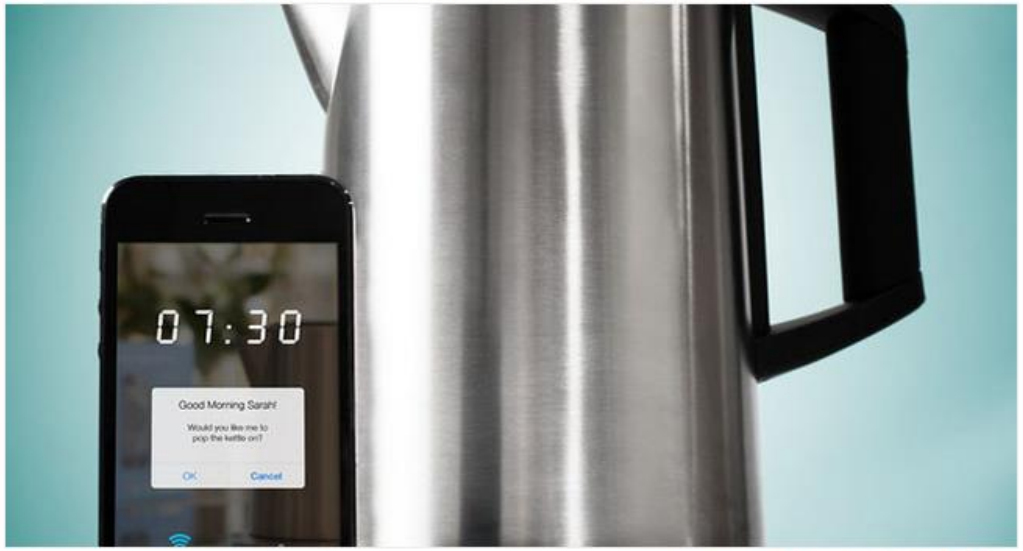
Poor protection

The spam attack took place between 23 December 2013 and 6 January this year, **said Proofpoint in a statement**. In total, it said, about 750,000 messages were sent as part of the junk mail campaign. The emails were routed through the compromised gadgets.

Security

Connected kettles boil over, spill Wi-Fi passwords over London

Pen-tester's killer cuppas made in cracked iKettle



19 Oct 2015 at 05:57, Darren Pauli

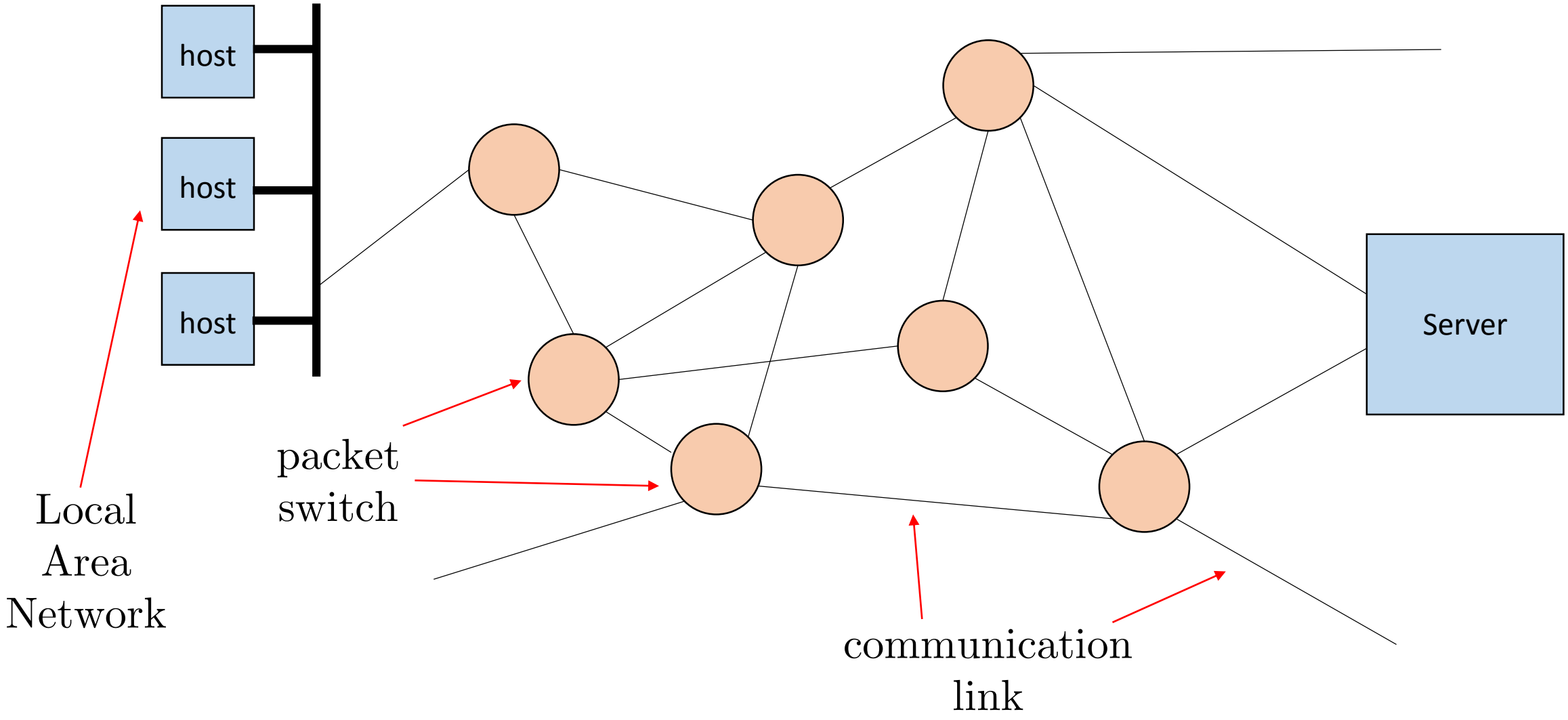
A security man has mapped and hacked insecure connected kettles across London, proving they can leak WiFi passwords.

The iKettle is designed to save users precious seconds spent waiting for water to boil by allowing the kitchen staple to be turned on using a smartphone app.

Pen Test Partners bod Ken Munro says hackers can make more than a cuppa, however: armed with some social engineering data, a directional antenna, and some networking gear they can "easily" cause the iKettle to spew WiFi passwords.

What is "inside" the Internet?

Imperial College London

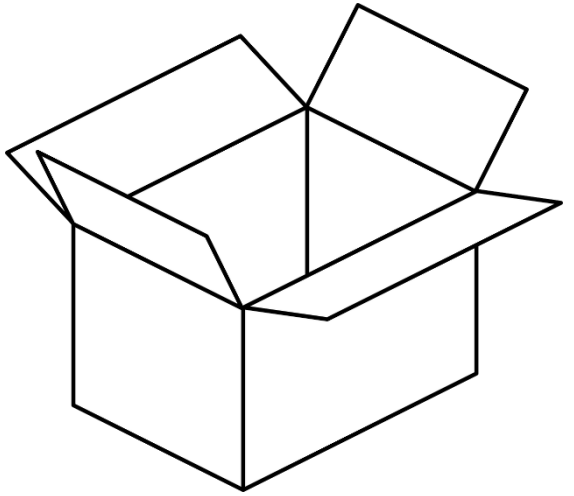


Terminology

- **Packet switch**: a link-layer switch or a **router**
- **Communication link**: a connection between packet switches and/or end systems
  - Fiber-optic cable
  - Twisted-pair copper wire
  - Coaxial cable
  - Wireless local-area links (*e.g. 802.11, Bluetooth*)
  - Satellite channel
  - . . .
- **Route**: sequence of switches that a packet goes through (*a.k.a. path*)
- **Protocol**: control the sending and receiving of information to and from end systems and packet switches

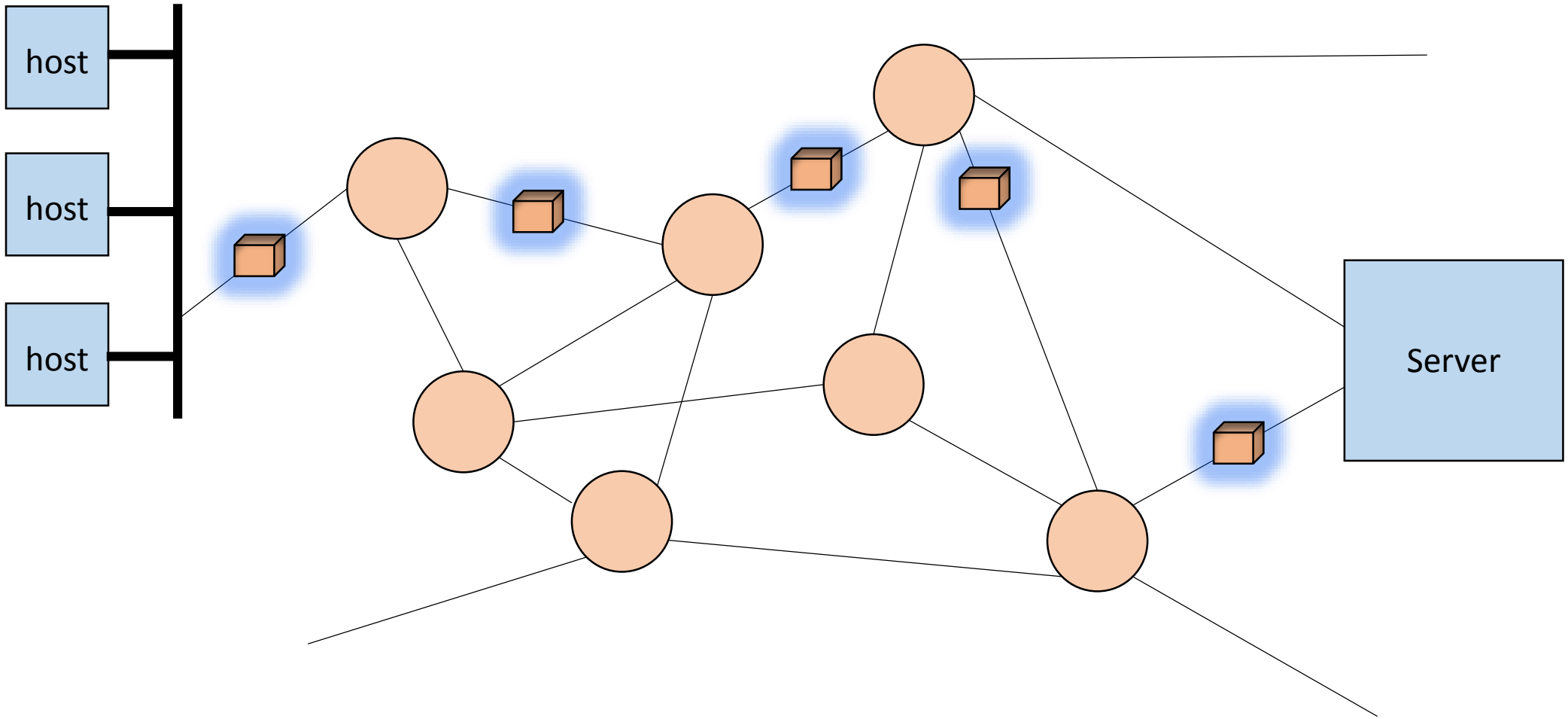
Packet Switching

- The Internet is a **packet-switched** network
- Information is transmitted in **packets**
  - A packet is a formatted unit of data
- **Switches/Routers** operate on individual packets



- A switch/router receives packets and *forwards* them along to other switches or to end systems
- Every forwarding decision is taken on the basis of the information contained in the packet

Packet Switching (cont'd)

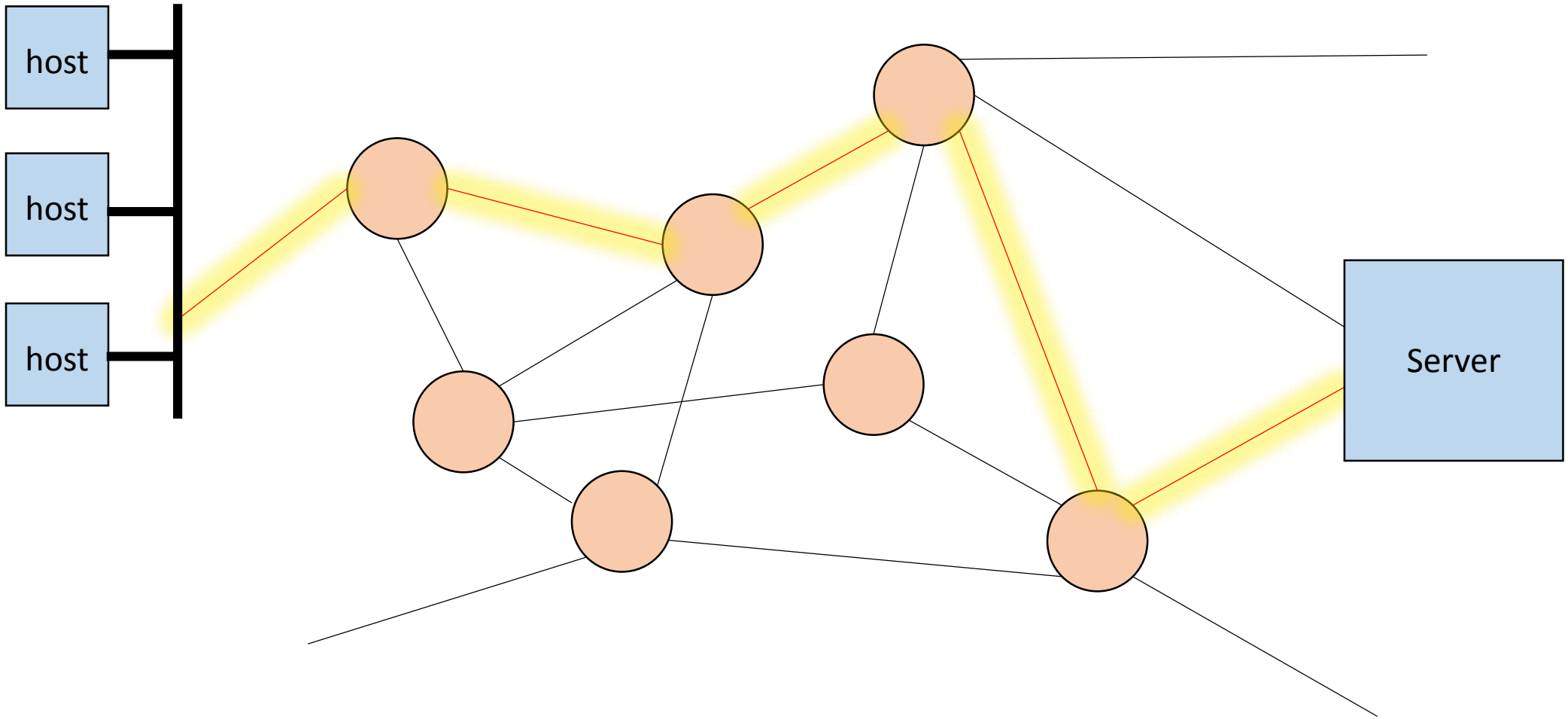


Circuit Switching

- The telephone network is a typical circuit-switched network

- Communication requires a connection setup phase
  - Network reserves all the necessary resources for that connection (links, buffers, switches, etc.)
- After a successful setup, the communicating systems are connected by a set of links that are dedicated to their connection for the entire duration of their conversation
- When the conversation ends, the network destroys the connection, freeing the corresponding resources (links, buffers, etc.) for other connections

Circuit Switching (cont'd)



Circuit vs. Packet Switching

- Circuit switching requires an *expensive setup phase*
  - However, once the connection is established, little or no processing is required

- Packet switching *does not incur any setup cost*
  - However, it always incurs a significant processing and space overhead, on a per-packet basis
    - *processing cost* for forwarding
    - *space overhead* because every packet must be self-contained

Circuit vs. Packet Switching (cont'd)

- Circuit switching admits a straightforward implementation of **Q**uality **O**f **S**ervice guarantees
  - network *resources are reserved* at connection setup time

- Guaranteeing any Quality Of Service with packet switching is very difficult
  - no concept of a "connection"
  - and again, processing, space overhead, etc.

Circuit vs. Packet Switching (cont'd)

- Circuit switching allows only a limited sharing of communication resources
  - once a connection is established, the resources are blocked even though there might be long silence periods
  - i.e. circuit switching is an *inefficient* way to use the network

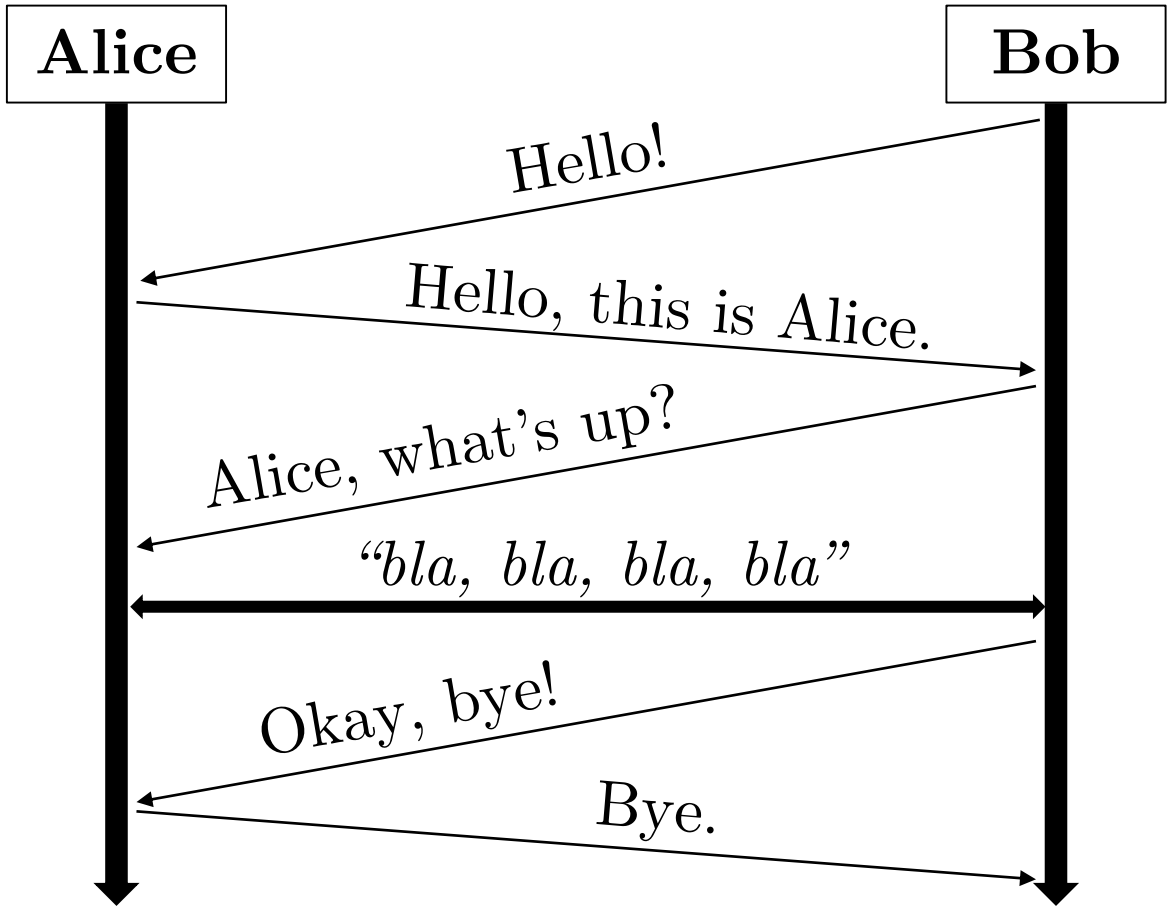
- Packet switching achieves a much better utilisation of network resources
  - it is designed specifically to *share* links

- **But how do end systems, or packet switches, communicate?**

Communication Protocols

- End systems, as well as packet switches, run **protocols**. What is a protocol?

- Let's consider a phone call: **Bob** calls **Alice**



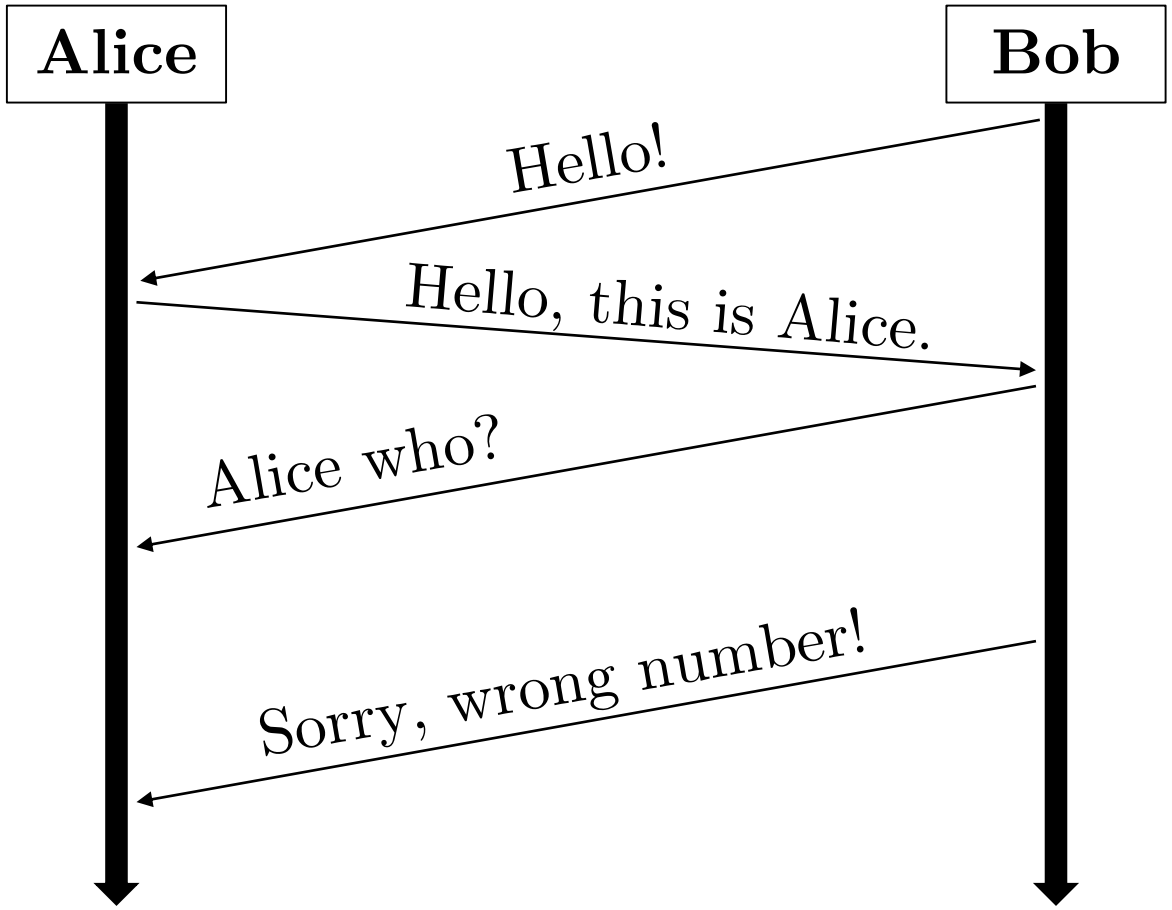
Communication Protocols (cont'd)

- A communication **protocol** is an *agreement* between the communicating parties on how communication is to proceed

- Phases of the protocol
  - **handshake**: establishes the identities and/or the context
  - **conversation**: free-form exchange
  - **closing**: terminates the conversation

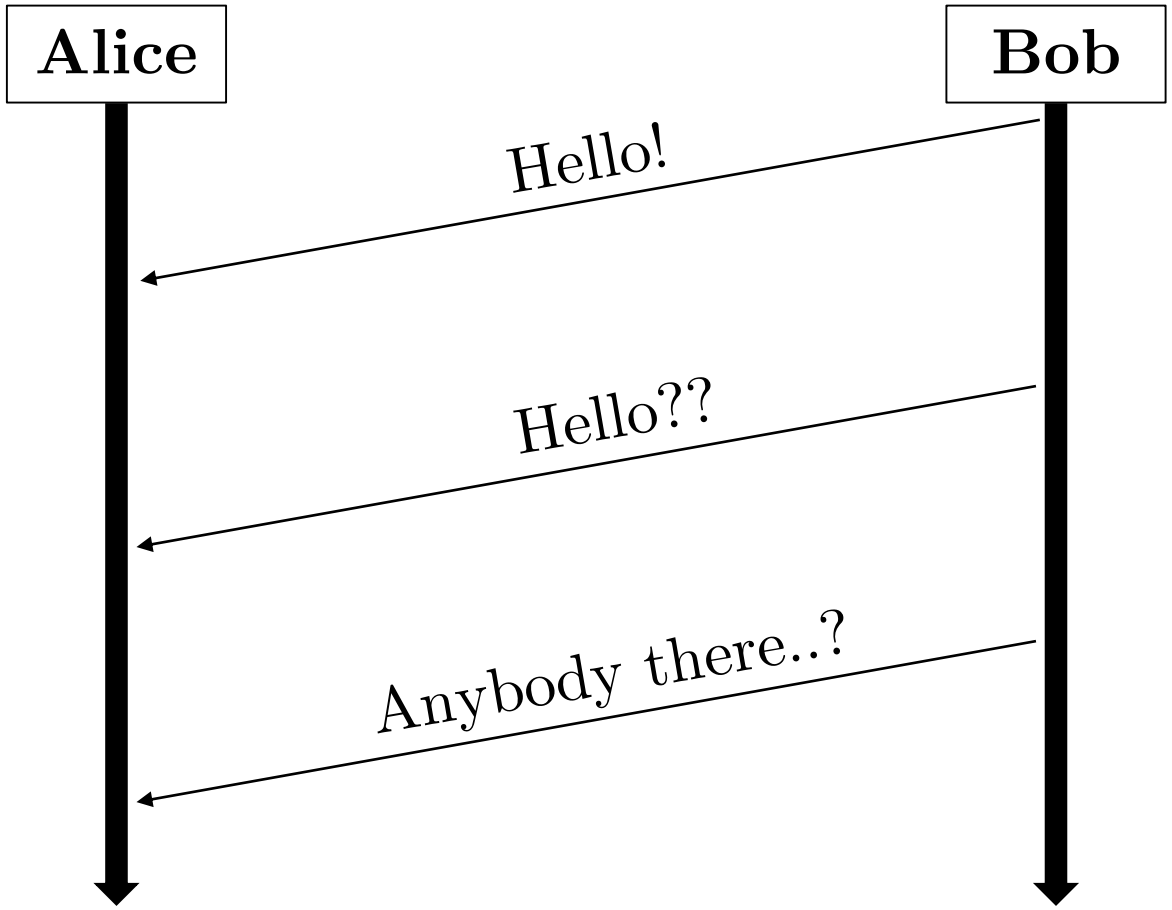
Communication Protocols (cont'd)

- Let's revisit the phone-call protocol



Communication Protocols (cont'd)

- Another run of the phone-call protocol

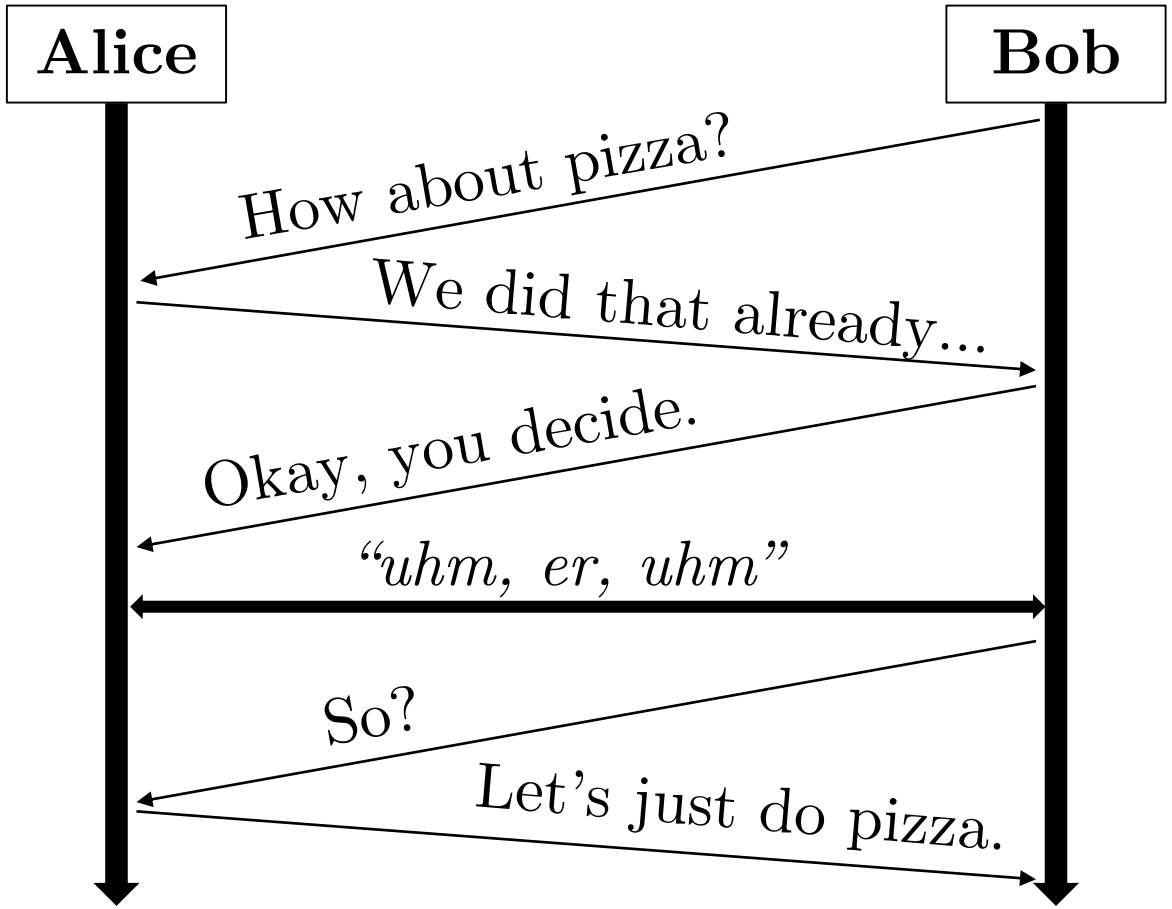


Communication Protocols: Principles

- A **protocol** is a *set of rules*
- It is an *executable* specification
- It must be *unambiguous*
- It must be *complete*
  - i.e. it must include actions and/or responses for all possible situations and all possible messages
- A network protocol must also define all the necessary *message formats*

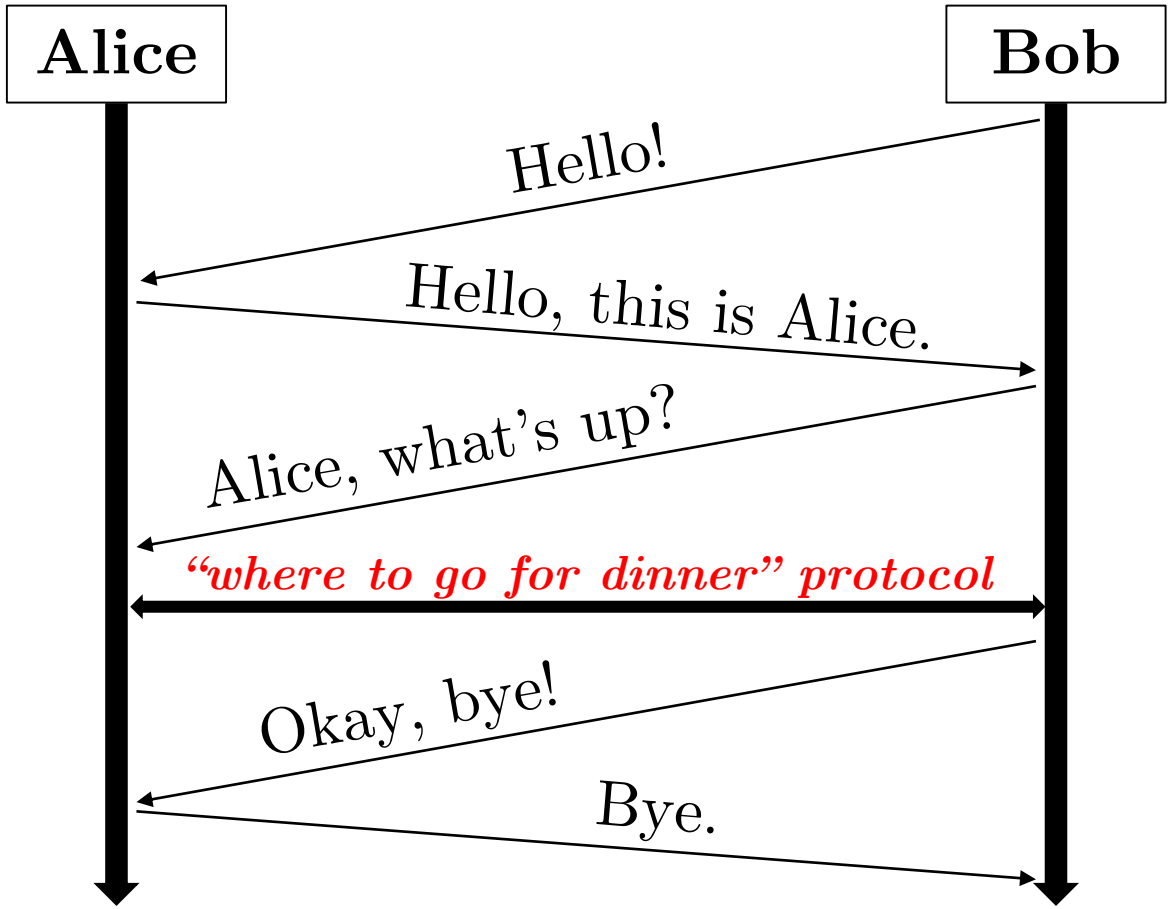
Communication Protocols: Layering

- Another protocol: deciding "**where to go for dinner**"



Communication Protocols: Layering

- Now Bob calls Alice to decide where to go for dinner



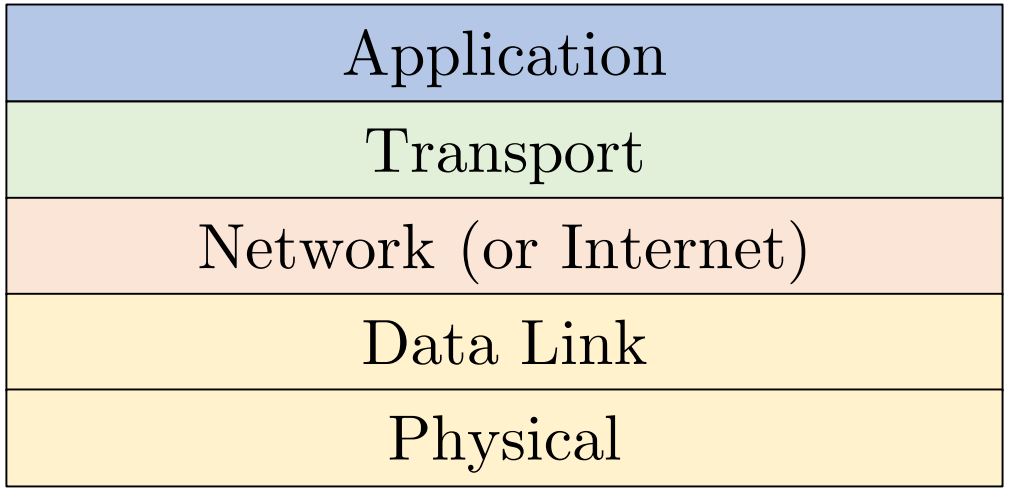
Communication Protocols: Layering (cont'd)

- Protocols within (*based on*) protocols

"where to go for dinner" protocol
phone call protocol
call setup
voice over the Internet (Protocol)
...

Internet Protocol Stack

- Hybrid 5-layer version



Internet Protocol Design

- Examples of Design Issues:
  - **Addressing**
    - how to denote the intended recipient
  - **Error Control**
    - how to detect (*and possibly fix*) transmission errors
  - **Flow Control**
    - how to prevent a fast sender from swamping a slow receiver
  - **Multiplexing / Demultiplexing**
    - how to support multiple communications in parallel
  - **Routing**
    - how to route packets to the final destination

Services: To Connect, Or Not To Connect?

- (*Most*) Network layers offer one or both of the following types of services:
  - **Connection-oriented**
    - This is the telephone model: you first *establish* a connection, then *exchange* data, and finally *release* the connection.
      - *Circuit-switch* is a form of connection-oriented protocol
  - **Connectionless**
    - The postal model: your data are put into some kind of *envelope*, on which the destination address has been written. The envelope and its contents are sent to the destination, and that's it.
      - *Packet-switch* is a form of connectionless protocol

Internet Protocol Stack – Model Variations

7 Layer OSI

Application
Presentation
Session
Transport
Network
Data Link
Physical

5 Layer Hybrid

Application
Transport
Network/Internet
Data Link
Physical

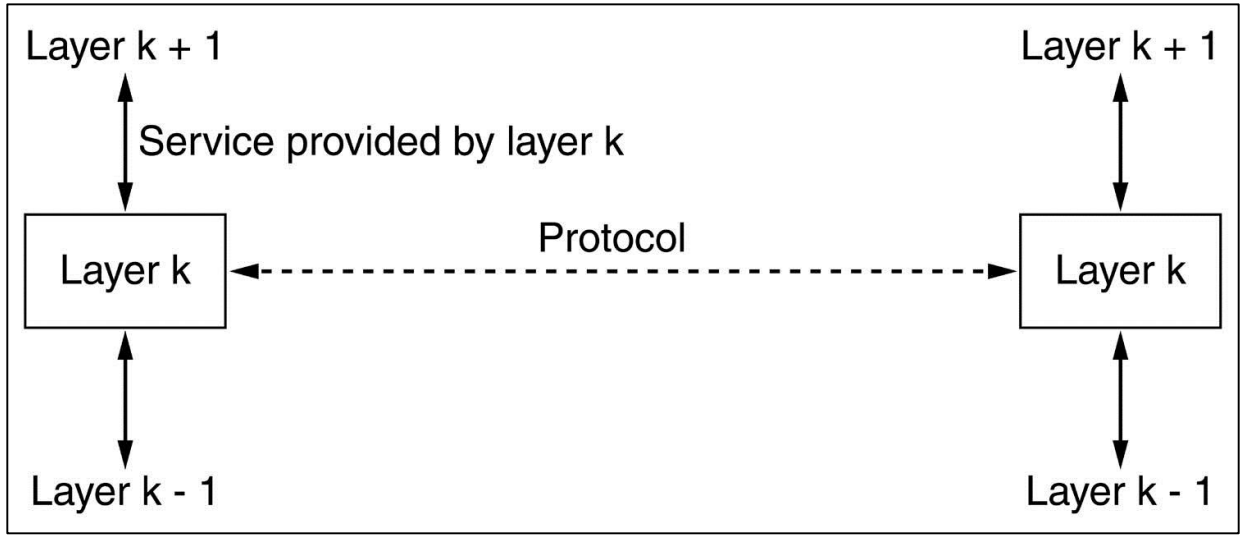
4 Layer TCP/IP

Application
Transport
Internet
Network Access

Services vs. Protocols

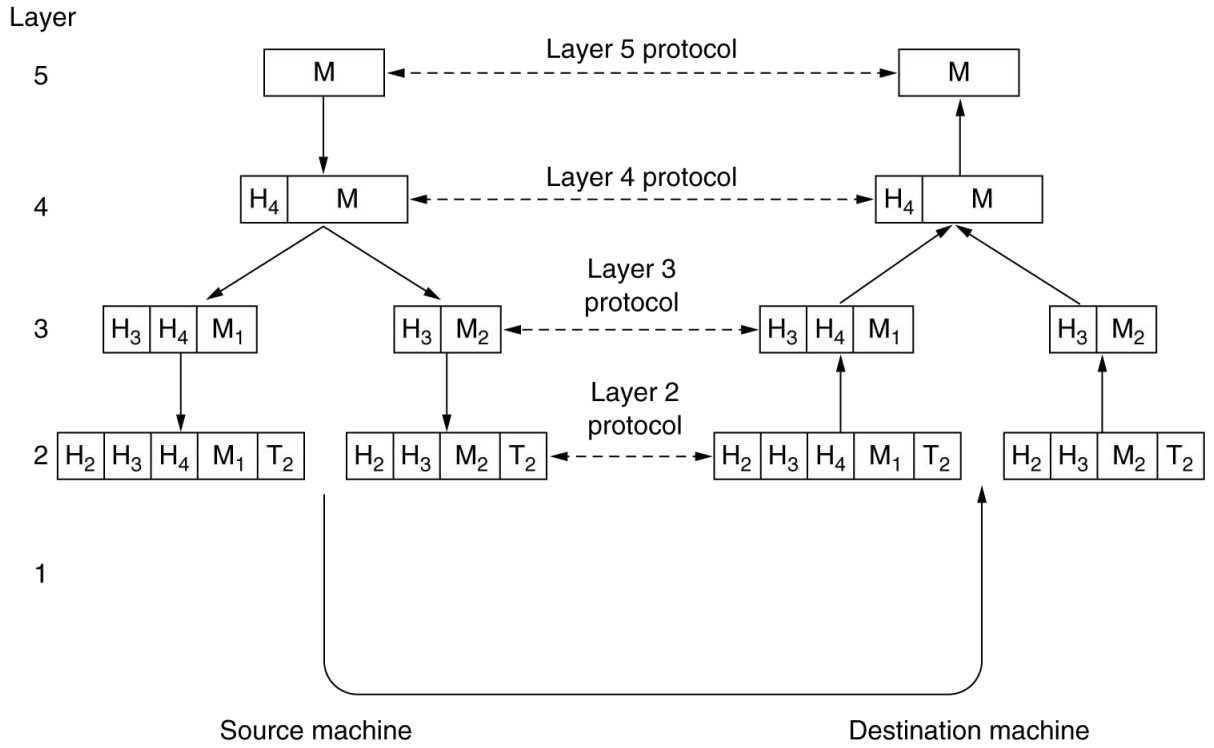
- **Service**: set of primitives that a layer provides to the layer above it.

- **Protocol**: set of rules that prescribe the layout and meaning of packets, and often the order in which specific packets should be sent.



Note: The same service can be realised by **different protocols**; each protocol can be implemented differently (using different data structures, programming languages, etc.)

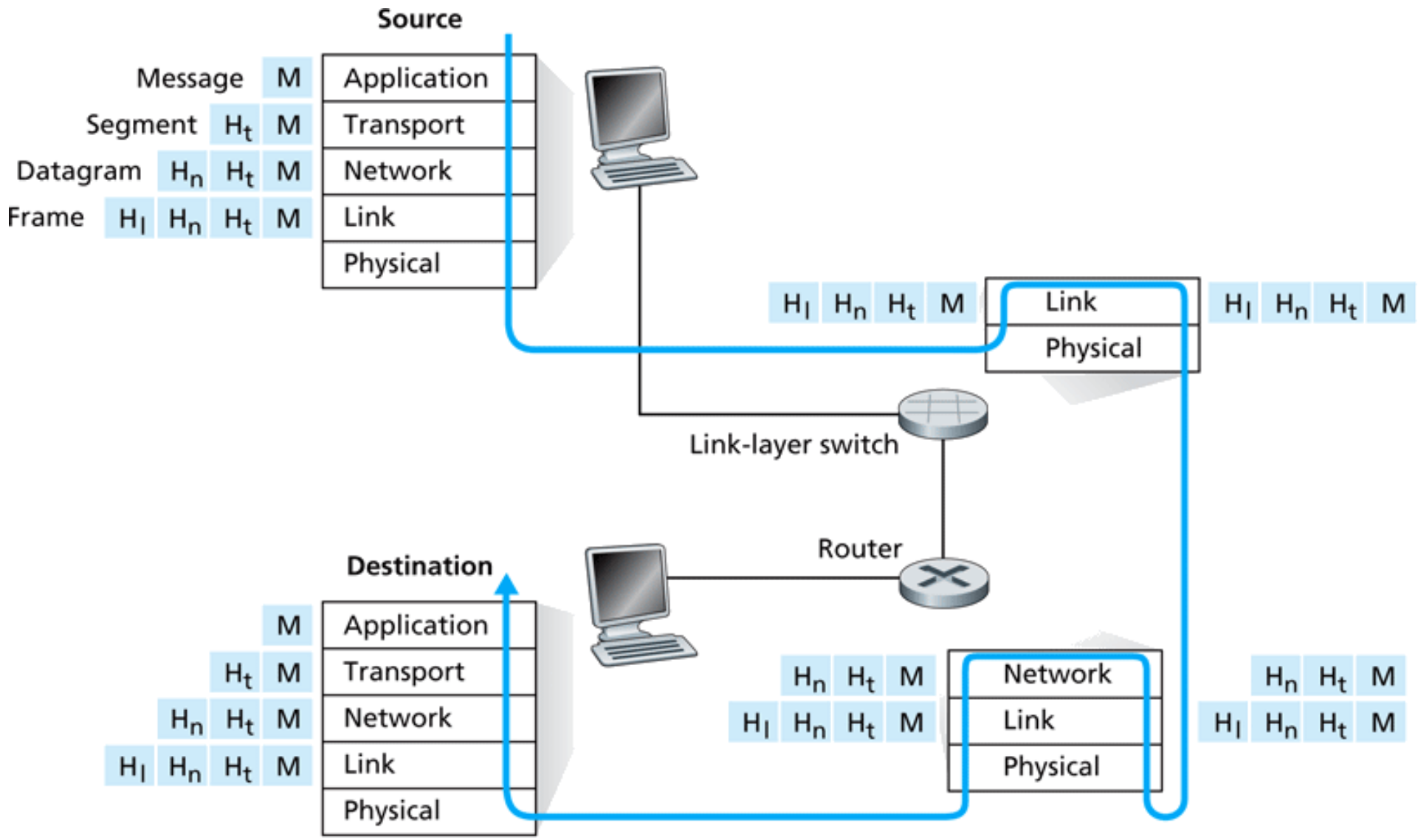
Protocol Layering: Example #1



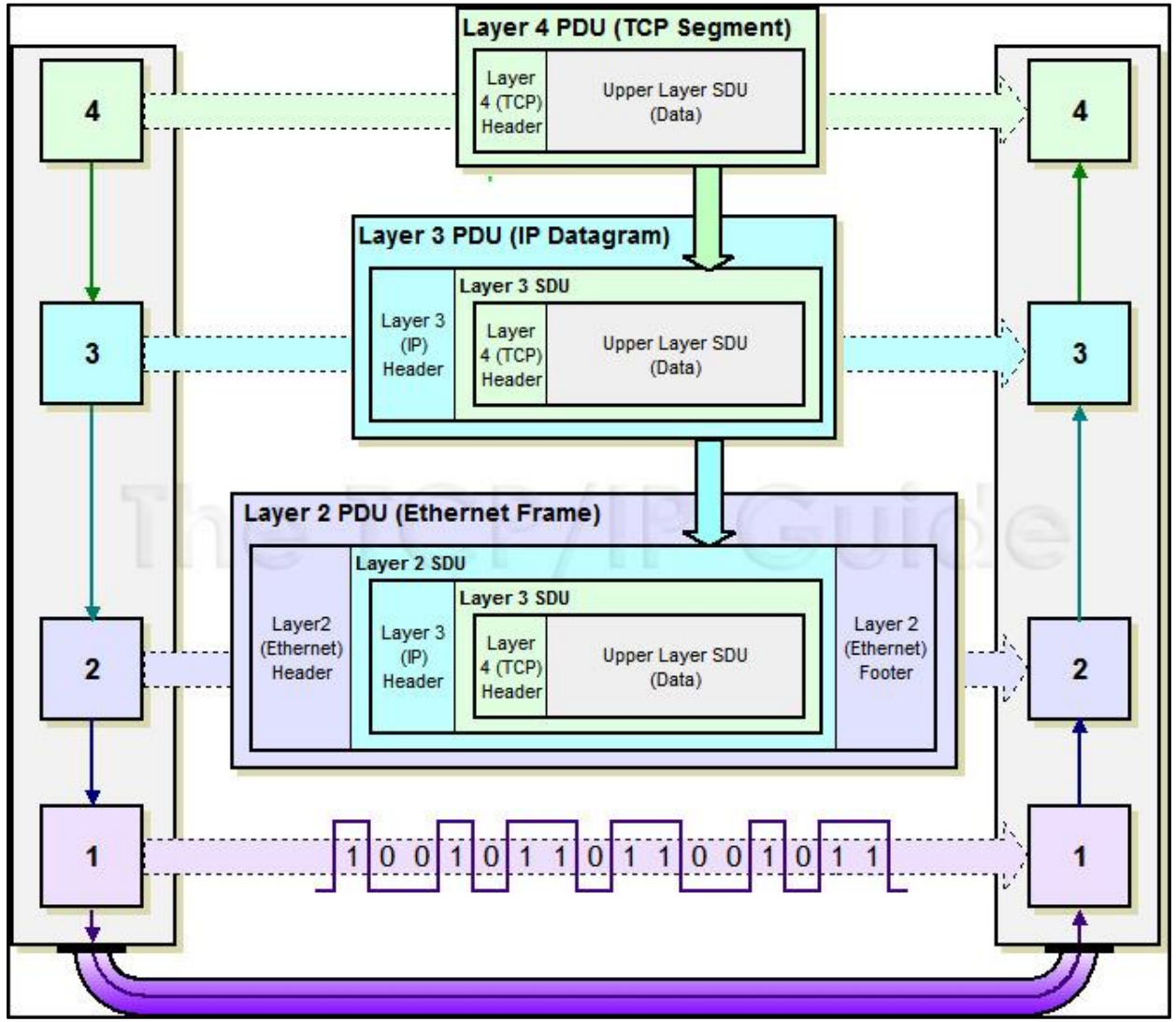
- In a protocol stack, Layer k puts its *entire* packet as **data** into a Layer k – *1* packet; the latter may add a header and/or a trailer.

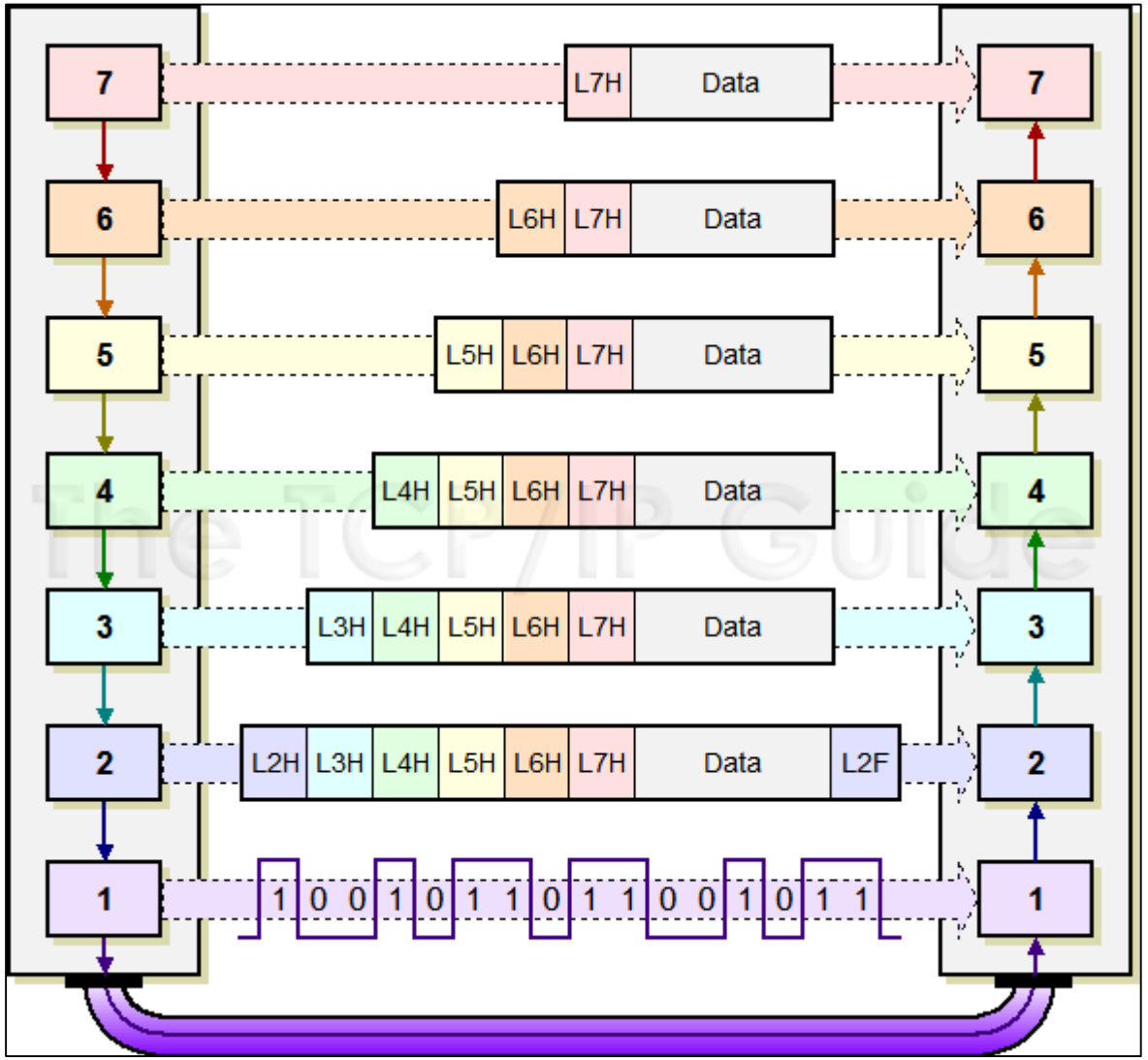
- It may occur that *Layer* k data have to be split across several Layer k – *1* packets (**fragmentation**)

Protocol Layering: Example #2



Protocol Layering: Example #3





*Source: [The TCP/IP Guide]

Application Layer

- Defines applications functionality and message formats
  - **Traditional**: Name services (DNS), E-mail (SMTP), FTP, Telnet, News (NNTP), Web (HTTP/HTTPS)
  - **Modern**: All types of **middleware protocols** to support **distributed systems**:
    - New transfer protocols for object systems like Java RMI, Apache Thrift, Google Protocol Buffers
    - Special protocols to handle replication, fault tolerance, caching, data persistence, etc.
  - **High-level protocols**: Special **application-level protocols** for e-commerce, banking, etc.
  - **Peer-to-Peer protocols**: BitTorrent, Skype, ...

Transport Layer

- **Observation**: Generally offers **connection-oriented** as well as **connectionless** services, and varying degrees of reliability. This layer provides the actual network interface to **applications**.
  - Often provides network interface through **sockets** (UNIX, Windows)
  - Allows to set up a connection to another application, and subsequently deliver data **reliably**, and **in the order** that it was sent
  - Provide the mechanisms so that fast senders don't overwhelm slow receivers (**flow control**)
  - Often also support for **secure connections**
  - Also support for **datagrams**: unreliable message passing on a per-message basis

Network Layer

- **Essence**: Describes how routing (and congestion) is to be done.
  - How do we find out which computers/routers are in the network?
  - How do we calculate the **best route** from A to B?
  - What happens when a computer/router goes down?
  - Should **multicasting/broadcasting** be supported?
  - What happens if a router becomes overloaded and starts dropping packets (**congestion**)?
  - Can we detect and avoid "hot spots"?

Data Link Layer

- **Observation**: We need to at least **detect bit transmission errors**
  - Add **redundant bits** in frames to detect something went wrong
  - Examples:
    - Add a **parity bit** to every 7 transmitted bits: 1 says there were odd number of '1's; 0 says there were an even number of '1's.
    - Add a **checksum** (**C**yclic **R**edundancy **C**heck) which should match the bits before it

- **Observation**: We also need to specify how a number of computers can **share** a common channel
  - **M**edium **A**ccess **C**ontrol sub-layer (MAC)
  - Examples:
    - Listen to each other; retreat when you hear someone else, and try again later
    - Wait your turn by passing a **token** between all stations
    - **Well-known protocols**: Ethernet, FDDI (*also: token ring, token bus*)

Physical Layer

- **Essence**: Describes the transmission of **raw bits**, in terms of **mechanical** and **electrical** issues.

  - Example:
    - Connect two computers by means of a wire:
      - Setting -3V on the wire corresponds to a binary **1**;
      - Setting +4V on the wire corresponds to a binary **0**;
      - The wire is quite short;
      - You may change the voltage at most 20,000 times per second;

Question	**Answer**
▪ What is the transfer rate?	▪ 20 Kbps (Kilo**bits** Per Second)

Quantifying Data Transfer

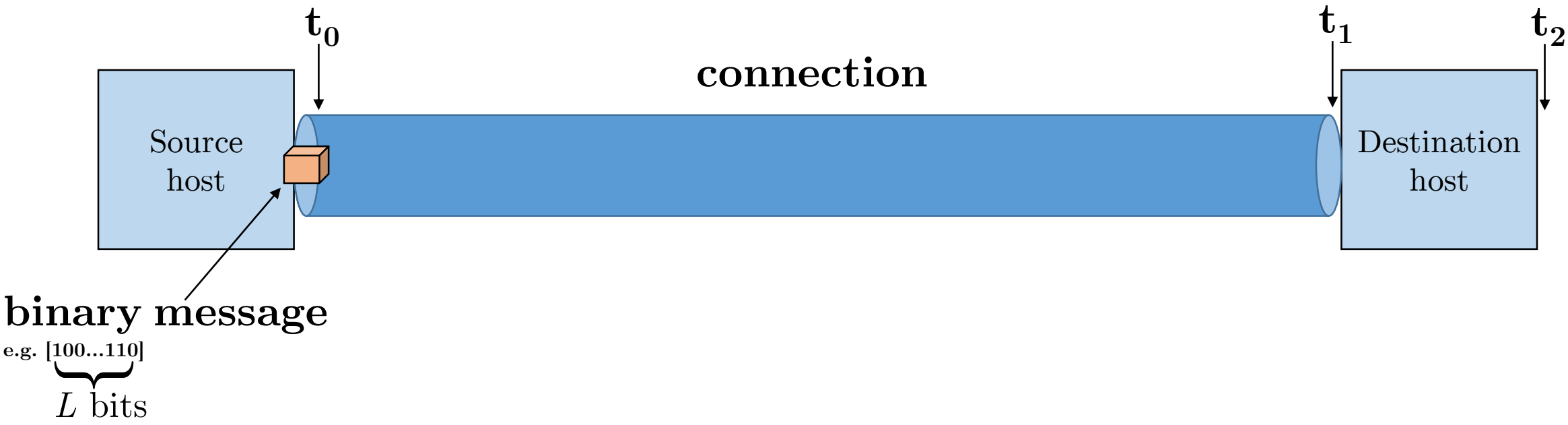
- How do we measure the "**speed**" and "**capacity**" of a network connection?

- Use your intuition:
  - water moves in a pipeline
  - cars move on a road

- **Latency**
  - the time it takes for one bit to go through the connection (*from one end to the other*)

- **Throughput**
  - the amount of information that can get into (*or out of*) the connection in a time unit
  - at "*steady-state*" we assume zero accumulation of traffic, therefore the **input** throughput is the same as the **output** throughput

Latency and Throughput



- Latency (*propagation delay*): $d = t_1 - t_0$ sec (*general formula:* $\frac{distance}{wave\ propagation\ speed}$)
- Throughput (*link bandwidth*): $R = \frac{L}{t_2 - t_1}$ bits/sec
- Packetization (*transmission delay*): $\frac{L}{R}$ (*a.k.a. store-and-forward delay*)
- Transfer time: Δ = propagation delay + transmission delay = $d + \frac{L}{R}$ sec

Examples

- How long does it take to transfer a file between, let's say, London and Edinburgh?
- How big is this file? And how fast is our connection?
- Example: assume it is a (*short*) e-mail message:
- $L = 4\text{KB}$
- $d = 500\text{ms}$
- $R = 1\text{MB/s}$
- $\Delta = 500\text{ms} + 4\text{ms} = 504\text{ms}$

Examples (cont'd)

- How about a big file? (*e.g. the size of a full CD*)

- $L = 700MB$
- $d = 500ms$
- $R = 1MB/s$
- $\Delta = 500ms + 700s = 700.5s = 11' 40.5''$

- How about a bigger file? (*e.g. 10 full DVDs*)

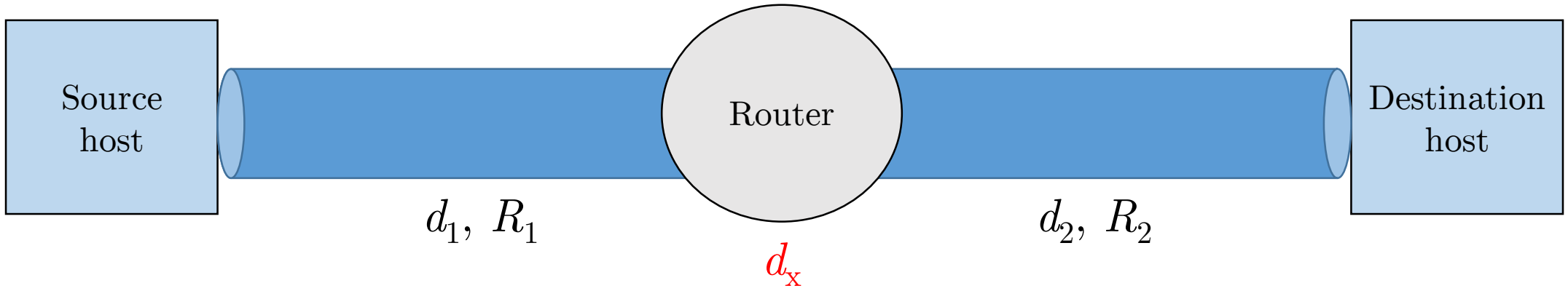
- $L = 40GB$
- $d = 500ms$
- $R = 1MB/s$
- $\Delta = \varepsilon + 40000s = 11h\ 6'\ 40''$

Examples (cont'd)

- You have a HDD with a large database backup (630GB) which needs to be transferred from London to Edinburgh.
  - Assume it takes you 1 second to take the HDD out of your backpack;
  - Assume your Vespa will get you there in exactly 7 hours;

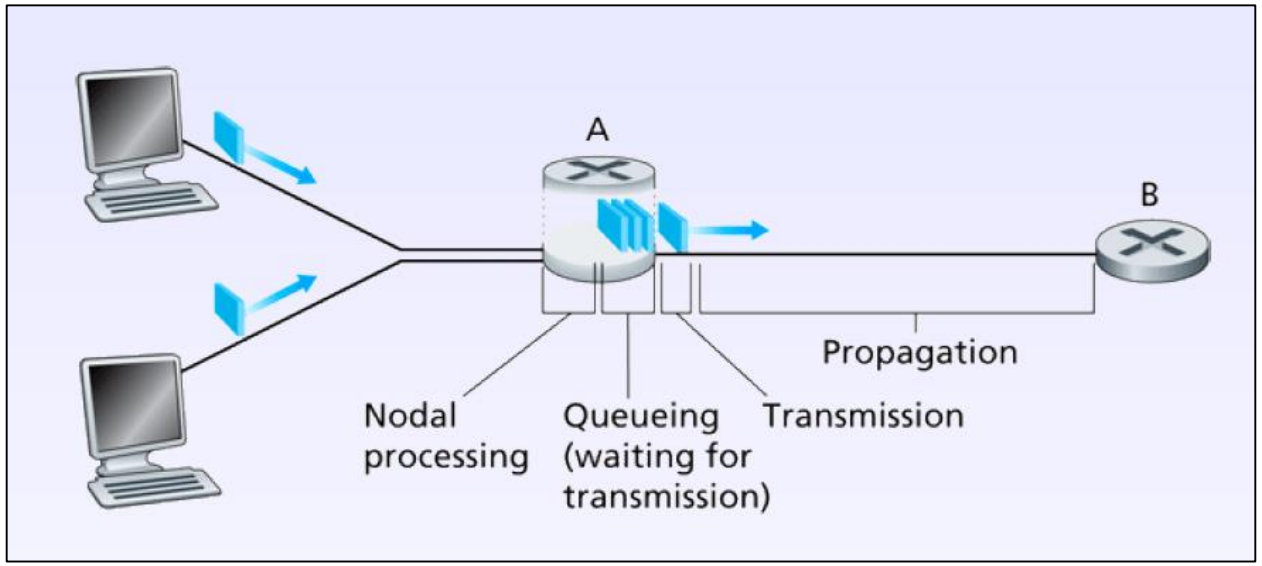
- $L = 630\text{GB}$
- $d = 7\text{h}$
- $R = 630\text{GB}$
- $\Delta = 7\text{h}\ (+\ 1\ \text{sec})$
- If you need to transfer 630GBs from London to Edinburgh and time is of the essence, then you're better off riding your Vespa to Edinburgh rather than using the Internet..!

Two Hops



- $d_{1,2}$ = prop. delay; $R_{1,2}$ = link b/w; L = packet length (bits); L/R = trans. delay
- (R1 < R2) $\quad d_{\text{end-end}} = d_1 + \frac{L}{R_1} + d_x + d_2 \quad$ sec
- (R1 ≥ R2) $\quad d_{\text{end-end}} = d_1 + d_x + \frac{L}{R_2} + d_2 \quad$ sec
- So, $\quad d_{\text{end-end}} = d_1 + d_x + d_2 + \frac{L}{\min(R_1, R_2)} \quad$ sec
- **But what is d_x?**

What is d_x?



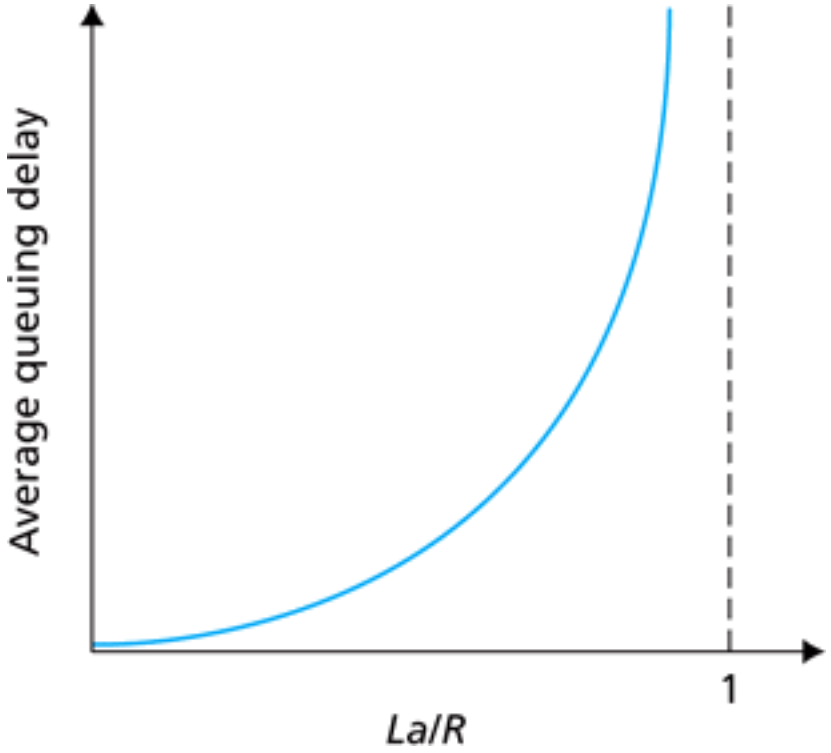
Router delay ($\mathbf{d_x}$) has two components:

- **Processing delay** ($\mathbf{d_{proc}}$)
  - check bit errors
  - determine output link
  - typically negligible ($<$ msec)
- **Queueing delay** ($\mathbf{d_q}$)
  - time waiting at output link for transmission
  - depends on congestion level of router

Queueing Delay

- Traffic intensity
  - R: link bandwidth (bps)
  - L: packet length (bits)
  - a: average packet arrival rate
  - $\frac{La}{R}$: ***traffic intensity***

- $\frac{La}{R} \approx 0$: avg. queueing delay small
- $\frac{La}{R} \to 1$: avg. queueing delay large
- $\frac{La}{R} > 1$: more "work" arriving than can be serviced
  - *avg. queueing delay infinite!*



*See also: Delay Simulation Java Applets

Q&A

- You will find the first **exercise worksheet** on the course website(s) today
  - *Solutions will be uploaded on the DoC website at the end of next week*
- **Suggested reading**: Tanenbaum, Ch#1
- Please provide *anonymous* feedback on www.menti.com using the code **40 02 50**
  - *always active throughout the term*
- You can also provide *eponymous* feedback or ask questions via email (*username:* ***kgk***)
- Thank you for your attention
- **Movie of the week**: Lo and Behold, Reveries of the Connected World
- **Next time**: Applications of the Internet

Acknowledgements

- *Many thanks to:*
  - *Anandha Gopalan*
  - *Paolo Costa*
  - *Peter McBrien*

- *Special thanks to:*
  - *Maarten van Steen*
  - *Antonio Carzaniga*