Last Lecture

- How to send data from end to end: two switching methods
  - Circuit switching
  - Packet switching
- Packet loss and delay in a packet switched network

This Lecture

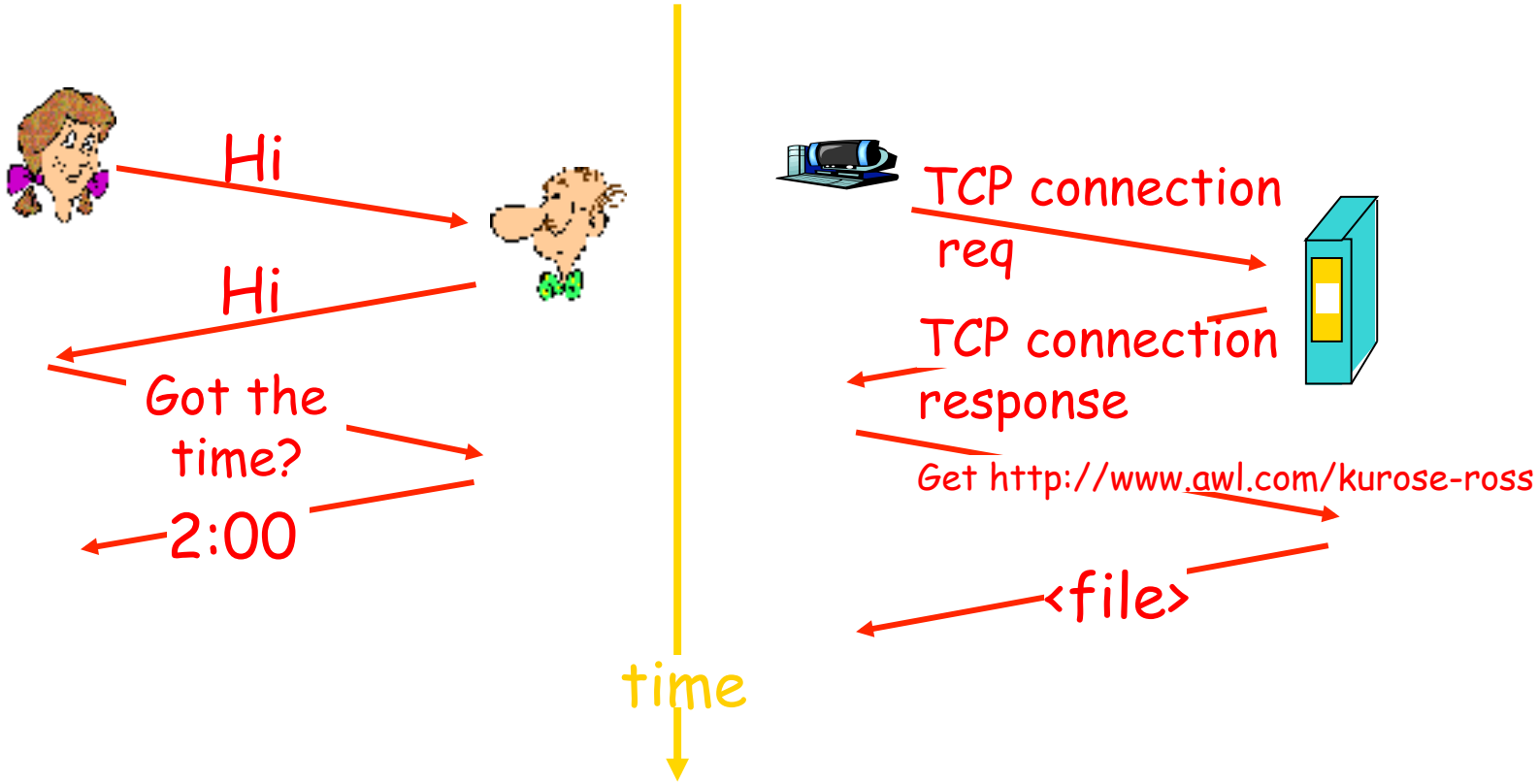
- Modular view of the Internet
  - Layering, protocol architecture
  - The TCP/IP reference model

What Is a Protocol?

- We have just seen an example of a protocol (the HTTP protocol)
- Protocol:
  - A formal description of a set of rules and conventions that govern how peer entities on a network exchange information.
- Peer entities: same-level network entities like processes, routers, modems, …
- A protocol's key elements:
  - **Syntax**: data format, signal levels
  - **Semantics**: control information and error handlings
  - **Timing**: speed matching and sequencing

What Is a protocol?

a human protocol and a computer network protocol:



An Example Protocol: HTTP

```
GET /dir/page.html HTTP/1.1
HOST: www.buffalo.edu
Connection: close
User-agent: Mozilla/4.0
Accept-language:fr
'\n'
```

HTTP Request Message

Optionally:

```
Cookie: 1634679
```

```
HTTP/1.1 200 OK
Date: Wed, 29 Aug 2001 08:14:59
   GMT
Server: Apache/1.3.20 Ben-SSL/
   1.44 (Unix) PHP/4.0.6
Connection: close
Content-Type: text/html

(requested file ..)
```

HTTP Reply Message

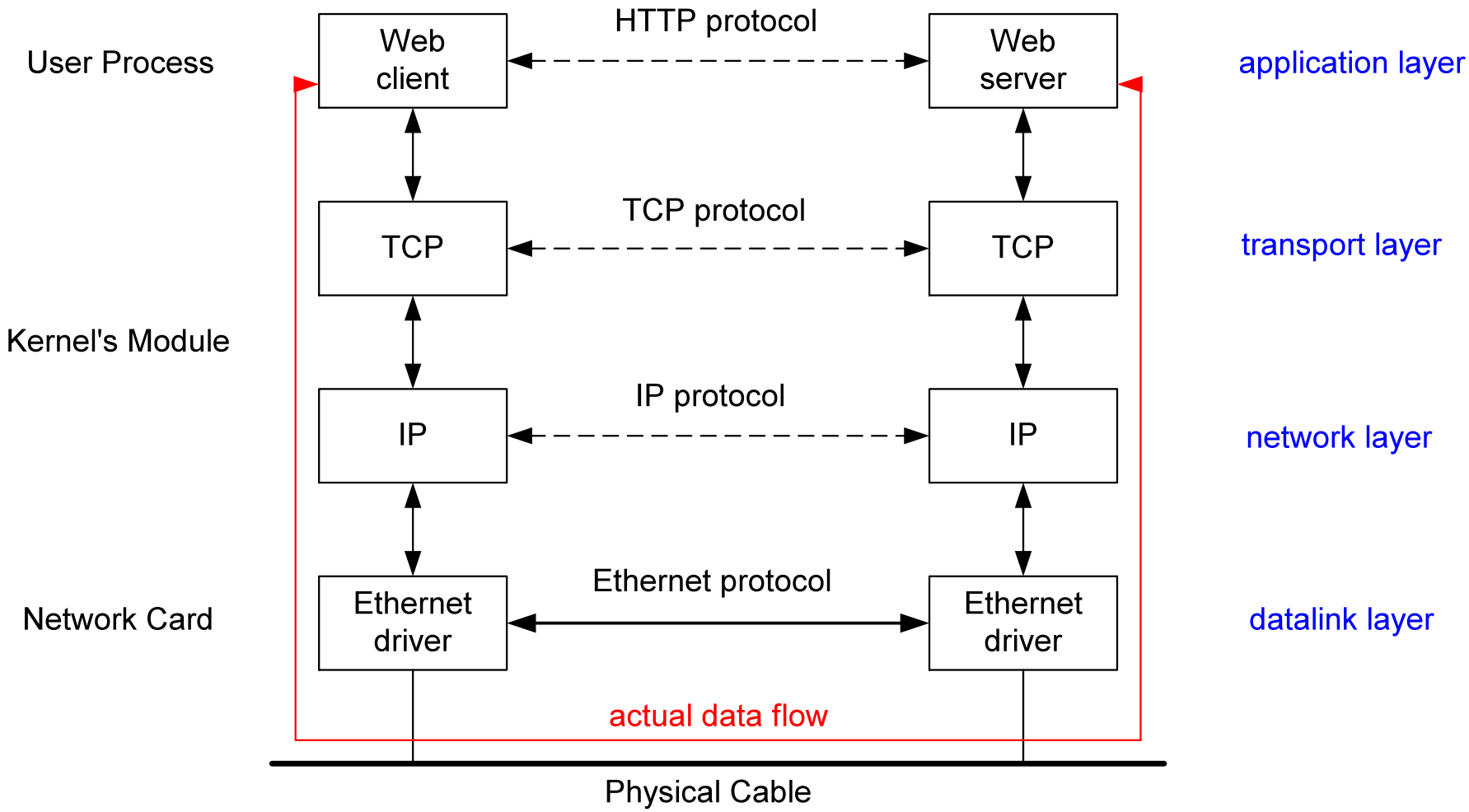
Optionally:

```
Set-cookie: 1634679
```

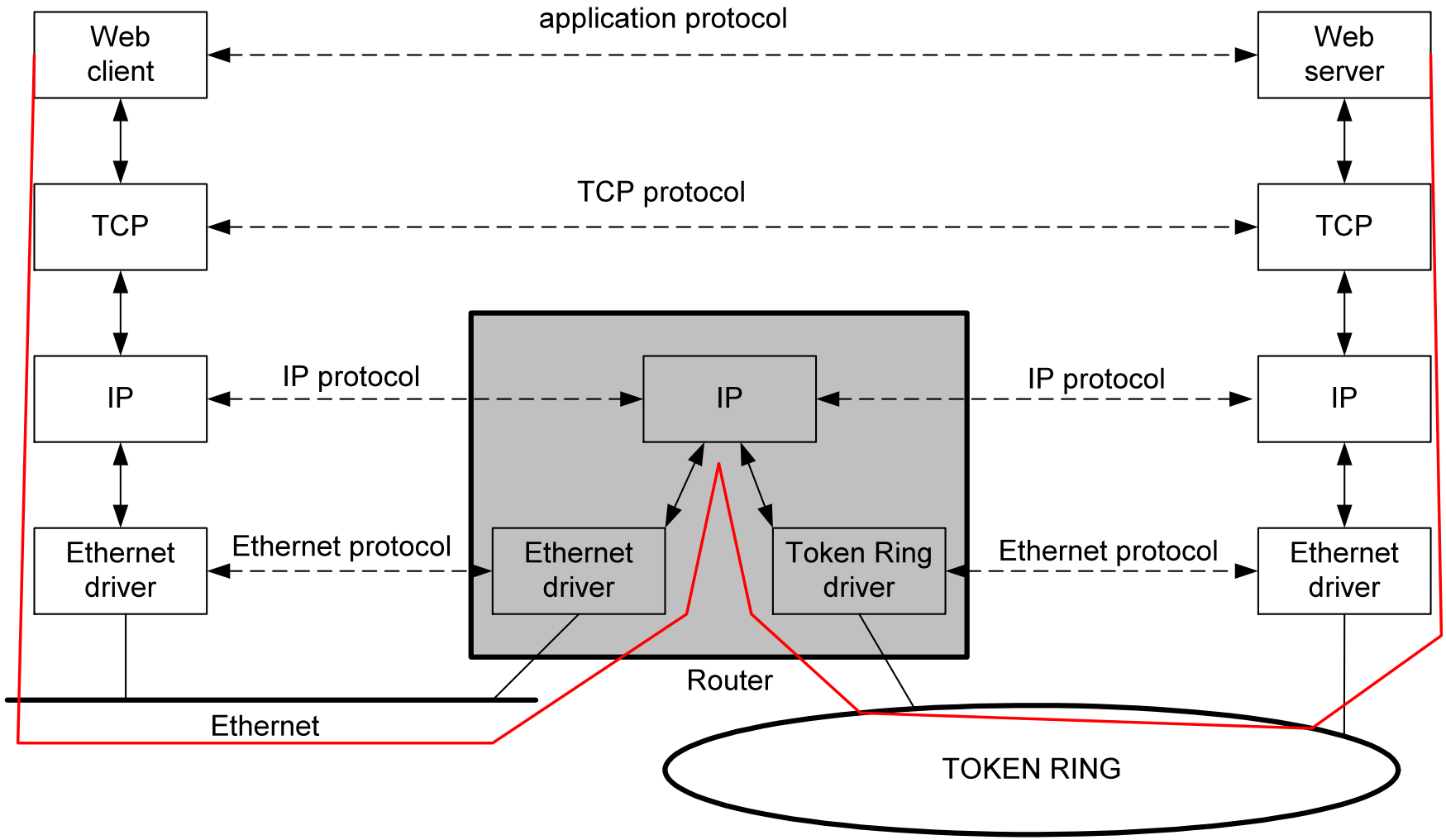
Protocol Architectures

- The Internet is extremely complicated
  - Just imagine what happens in every detail how those "*simple*" HTTP request and response get through the network
- Key idea in CS: *Modularize*
- To simplify network design complexity,
  - Organize protocols and the hardwares/softwares that implement the protocols in to *layers*
    - Each layer is a software and/or hardware module
  - Upper layers use *services* provided by lower layers
  - The protocol layers form a *protocol stack* (protocol suite, protocol architecture)

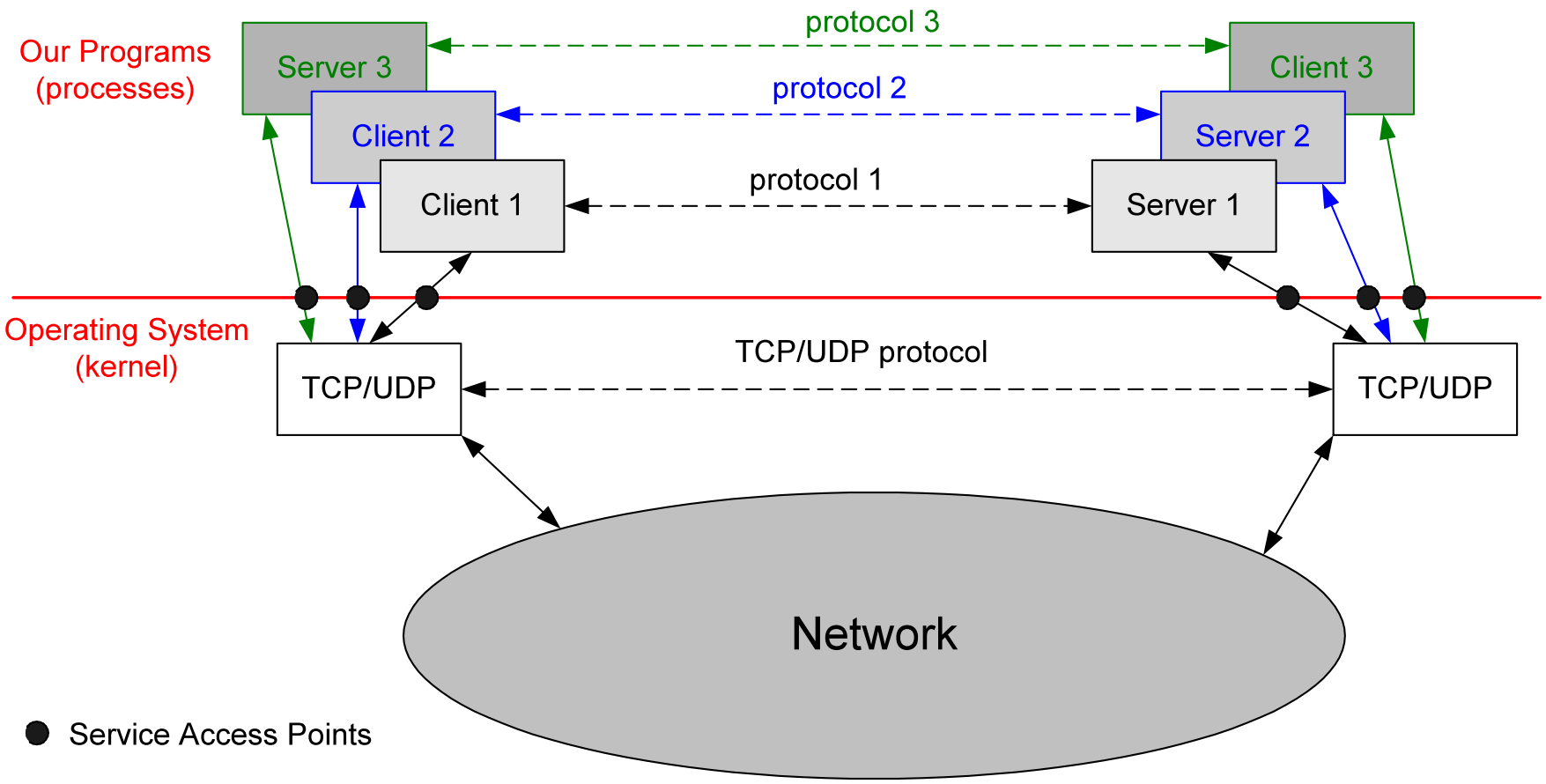
A Practical Example



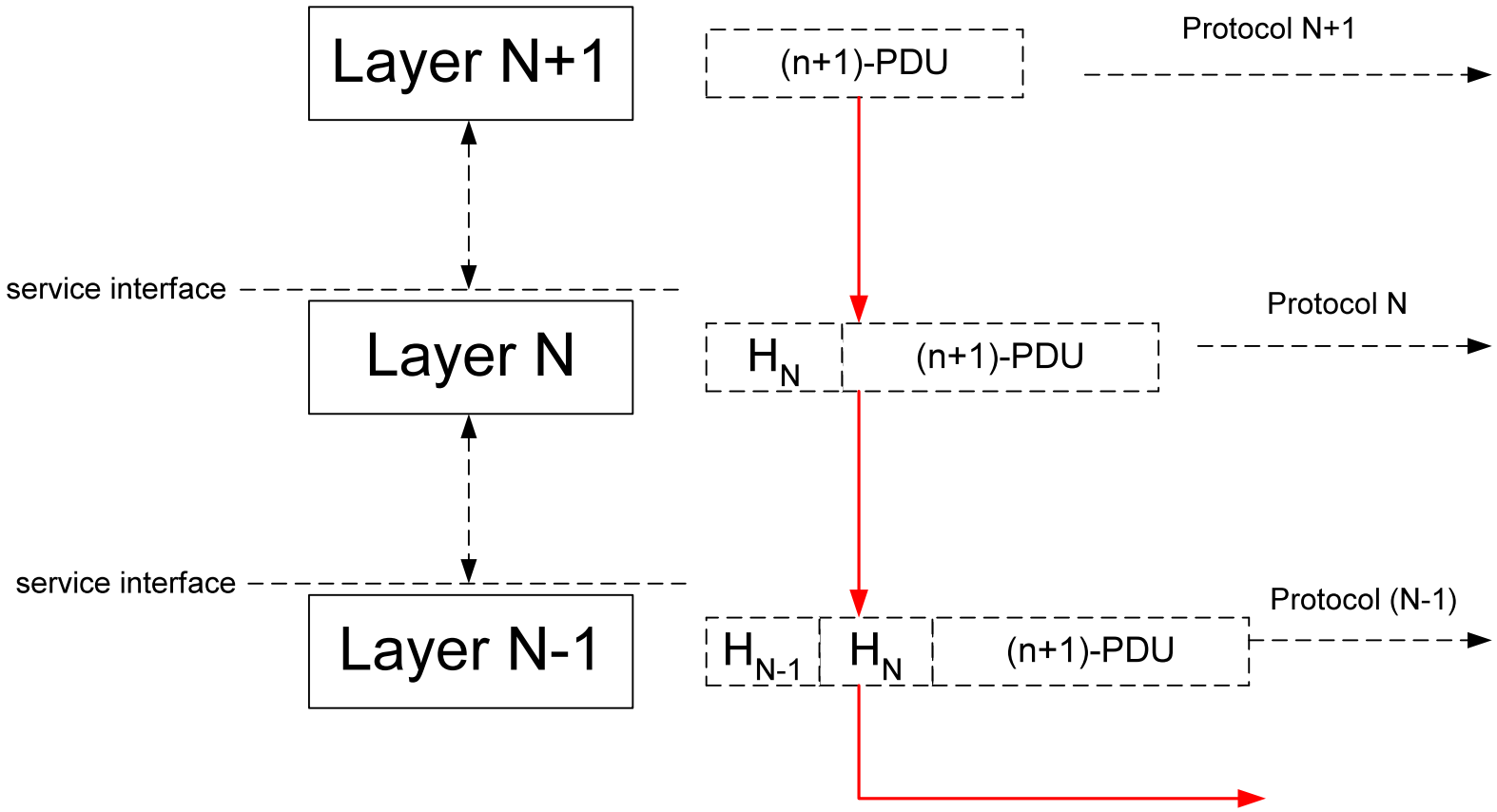
A More Practical Example



Our view in the first few weeks



A Theoretical Figure



PDU: Protocol Data Unit, H: header (control information)

Typical Functions of a Layer

- Connection setup: handshaking with peer
- Error Control
- Flow Control
- Segmentation & Reassembly
- Multiplexing

Service Characteristics

- Connection-Oriented
  - Connection established, used, and released (think of telephony systems)
- Connectionless
  - Each packet carries the destination address, that's it (think of postal systems)
- Reliable
  - I ensure the packet gets there, sooner or later
- Unreliable
  - I will try my best to serve you

In Reality

99% of services are

- Reliable, connection-oriented (TCP)
- Unreliable, connectionless (UDP)
  also called *datagram service*

Questions:

- Discuss Pros and Cons of these two types
- When to use what ?

Service Primitives

- The set of operations provided by the lower layer to the upper layer to perform a service
- Example:
  - Connection-oriented services:
    `connect, send, receive, disconnect`
  - Connectionless services:
    `send, receive`

Service Access Points

- How do two processes on two computers identify themselves to each other ?
- Answer: use a triple

  (protocol, ip_address, port_number) = **socket**

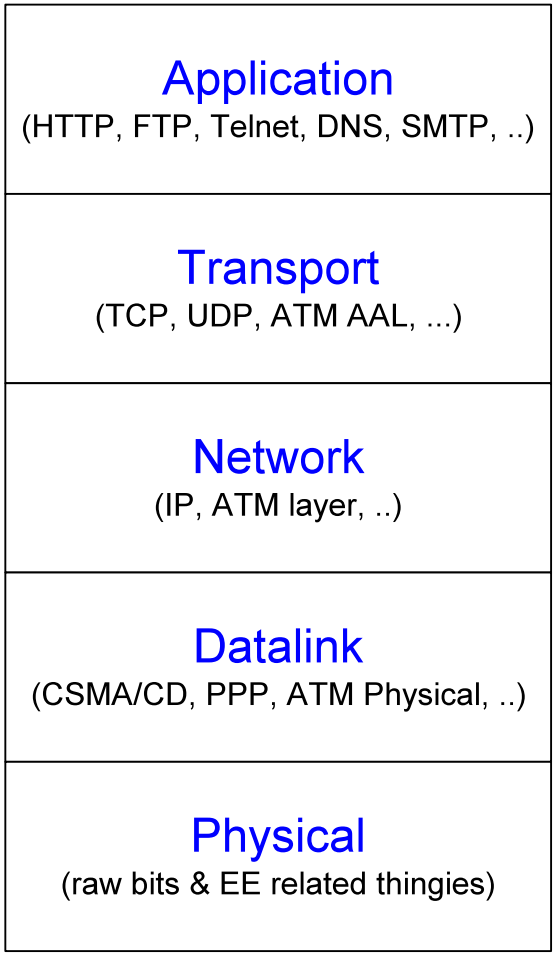
  e.g. (TCP, 192.168.0.1, 80)
- *(Actually a socket is a quintuple, more later)*
- In general: each entity of a layer access lower layer's services via *Service Access Points*

Services vs. Protocols

- *Service*: a set of function prototypes of a module
- *Protocols*: algorithms to implement those functions

- Algorithms can be changed without affecting users of the functions

Our Protocol Stack



Supports Network Applications

Transports applications' messages
TCP: connection-oriented, reliable
UDP: connectionless, unreliable

Routes data packets from hosts to hosts
IP: Internet Protocol, and many routing protocols

Deals with algorithms to achieve reliable, efficient communication between two adjacent machines

Moves raw bits (0/1) between adjacent nodes depending on the physical medium used

Summary

- Lots of basic concepts introduced

- Essential ones:
  - Circuit switching vs. packet switching
  - Small packets vs. large packets
  - Protocols
  - Protocol Architectures, TCP/IP Stack
  - Services