Modern Internet architecture, technology & philosophy

Advanced Internet Services

Dept. of Computer Science
Columbia University

Henning Schulzrinne

Spring 2015
02/16/2015

PROTOCOLS & LAYERS

Key concepts & questions

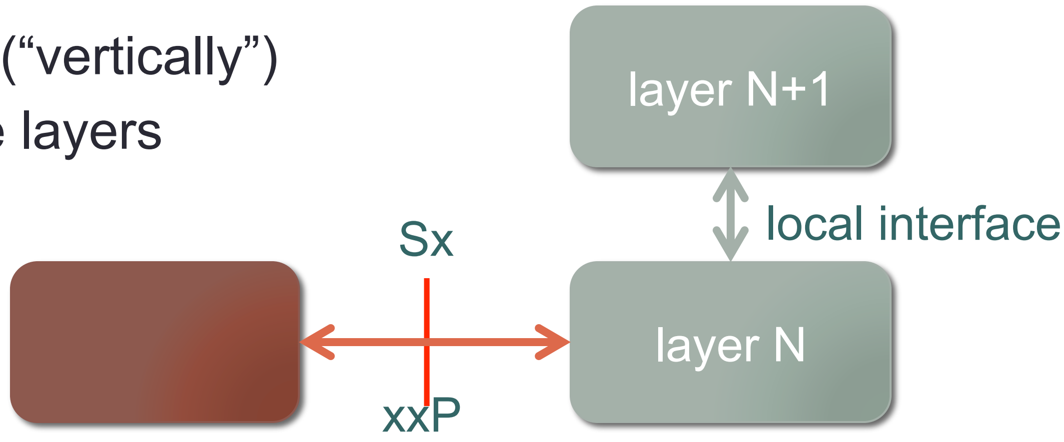
- Protocols as contracts
- Layering as abstraction & complexity reducer
- Layers: behavior + data structure
- How many layers should there be?
- The end-to-end principle: where should functions be performed?
- Why has the layer model changed?
- What is serialization and why do we need it?

Why layering?

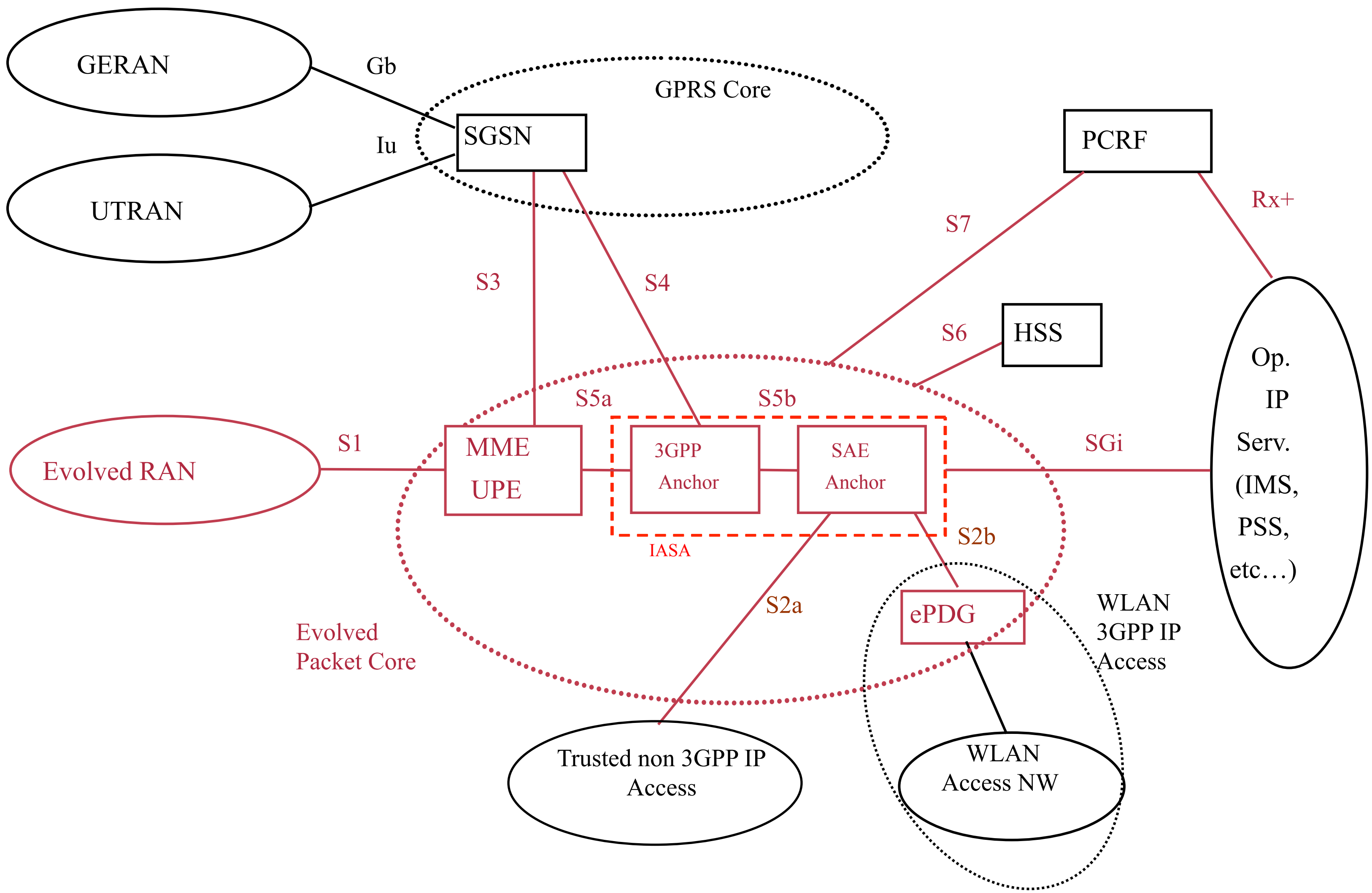
- Perform functions once
  - upper layers rely on lower layers
  - in theory (see: “end-to-end principle”)
- Common in engineering and society
  - postal system, operating systems & other APIs, buildings, …
  - but not always formal or deep
  - model of a (legal) contract
    - *“The elements of a contract are "offer" and "acceptance" by "competent persons" having legal capacity who exchange "consideration" to create "mutuality of obligation.”* (Wikipedia)

Why layering?

- Change implementation without affecting relying parties
  - minimize communications, “information hiding”, “isolation”
  - “black box”
- Topological, economic and administrative scoping
  - single *physical* connection technology
  - single vs. multiple *administrative* domains
- Related to interfaces:
  - interfaces define layers (“vertically”)
  - but not all interfaces are layers

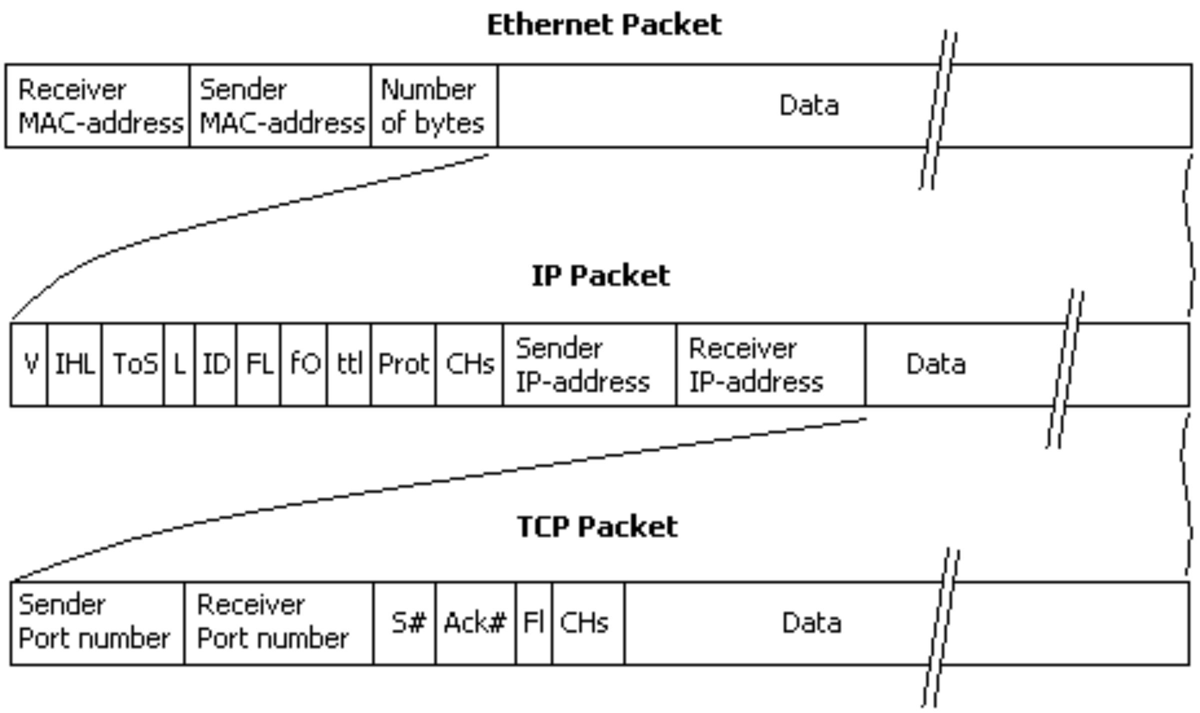


Example for 3GPP (LTE)



Layers → (sometimes) wrapping

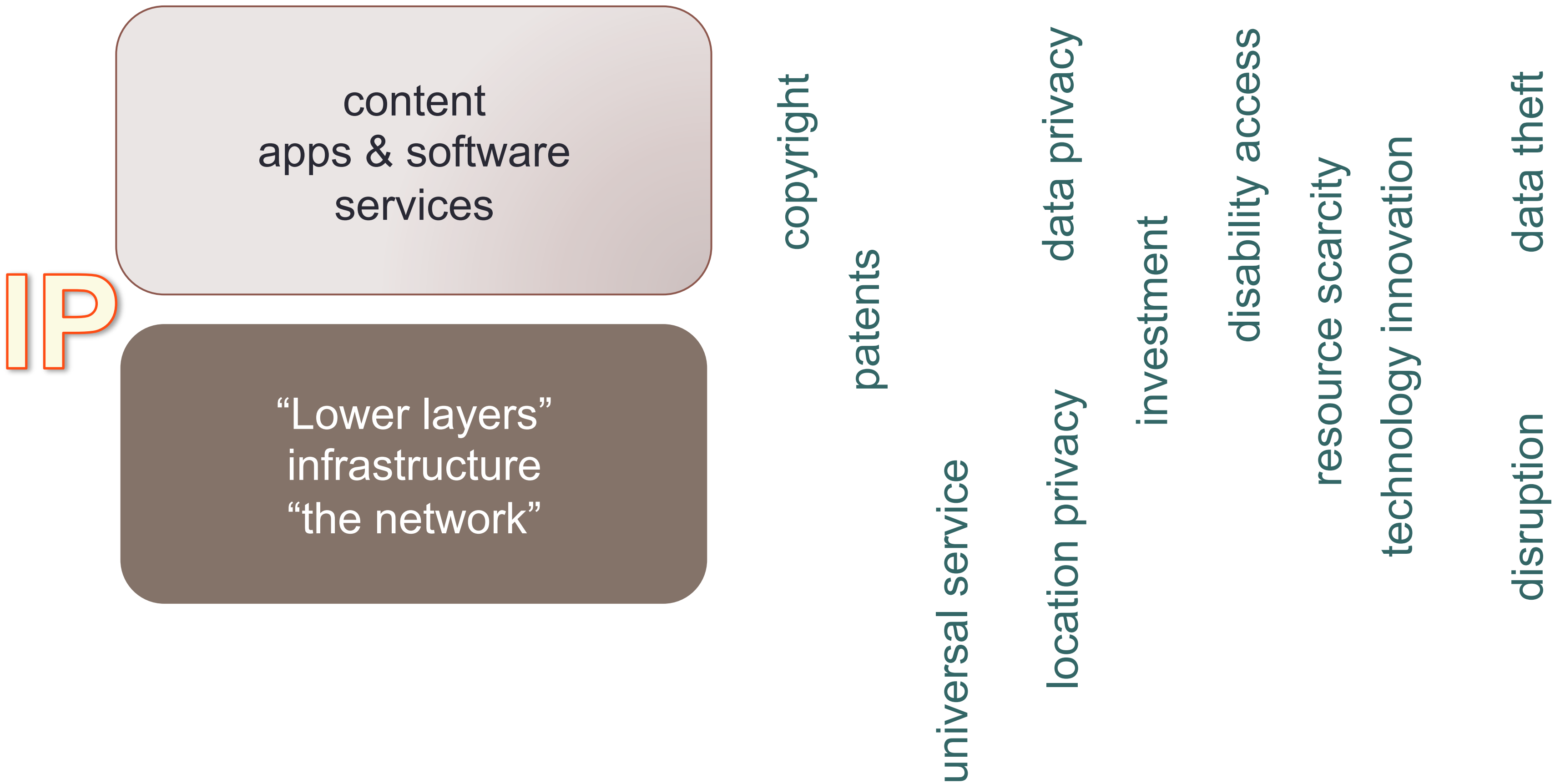
different from APIs!



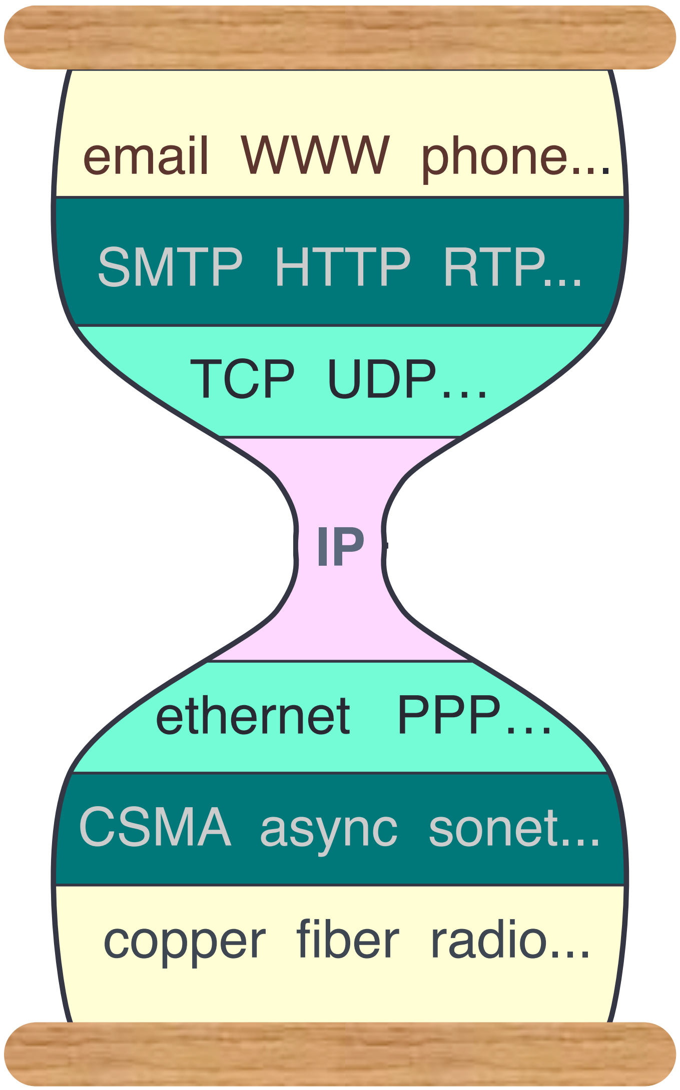
How many layers

- 2! → industry structure
- 4! → core Internet protocols
- 7! → classical layering
- 9! → sub-layers

The two-layer model



The Internet Protocol Hourglass (S. Deering)



Why the hourglass architecture?

- Why an internet layer?
  - make a bigger network
  - global addressing
  - virtualize network to isolate end-to-end protocols from network details/changes
- Why a *single* internet protocol?
  - maximize interoperability
  - minimize number of service interfaces
- Why a *narrow* internet protocol?
  - assumes least common network functionality to maximize number of usable networks

Deering, 1998

Layer splitting

- Traditionally, L2 (link), L3 (network = IP), L4 (transport = TCP), L7 (applications)
- Layer 2: Ethernet → PPPoE (DSL)
- Layer 2.5: MPLS, L2TP
- Layer 3: tunneling (e.g., GPRS)
- Layer 4: UDP + RTP
- Layer 7: HTTP + real application

Why 4 core layers?

Layer	Colloquial name	Function
1	PHY	photons & electrons → **bits**
2	MAC	bits → **packets** on one technology
3	L3	packets **end-to-end**, on heterogeneous technologies, to **interface**
4	L4	unreliable → **reliable** host/interface → **application**
(5)	Presentation, data	application data structure encoding
7	Application	Application behavior (email, web)

Internet layer functions

Layer	Key protocols	Control protocol	Transmission technologies	Administrative domains	Main function	Addresses
PHY	Ethernet, 4G		single, but may be diverse (fiber, copper)	1	analog-to-digital	none
MAC	Ethernet	3GPP	same	1	framing	MAC address
network	IPv4, IPv6	DHCP, OSPF, BGP	agnostic	many	end-to-end delivery	IP addresses
transport	UDP, TCP	built-in	agnostic	2 (ends)	reliability, congestion control	ports
application	HTTP, RTP	SIP	agnostic (except for properties)	2 (ends)	framing, description, sessions	URLs, email addresses

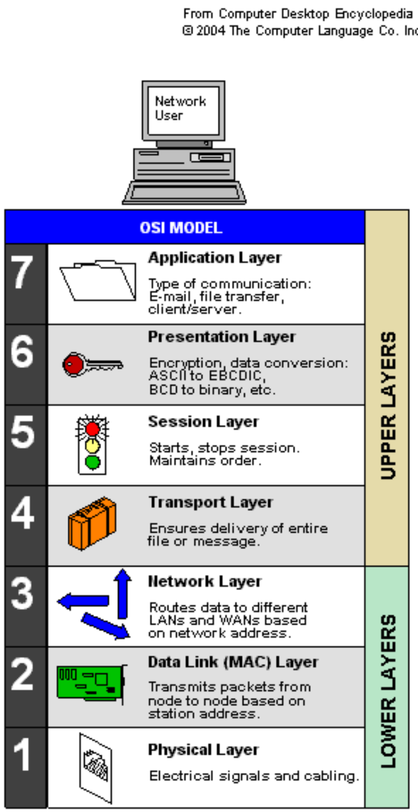
The real model

Layer violations

- Layers offer abstraction → avoid "Internet closed for renovation"
- Cost of information hiding
  - wireless networks
  - cost in $ and performance
- Cost of duplication of information when nothing changes
  - fundamental design choice of Internet = difference between circuit and datagram-oriented networks
- Assumption: packets are large and getting larger
  - wrong for games and audio
- Cost prohibitive on wireless networks
  - will see: 10 bytes of payloads, 40 bytes of packet header
  - header compression → compress into state index on one link

Internet acquires presentation layer

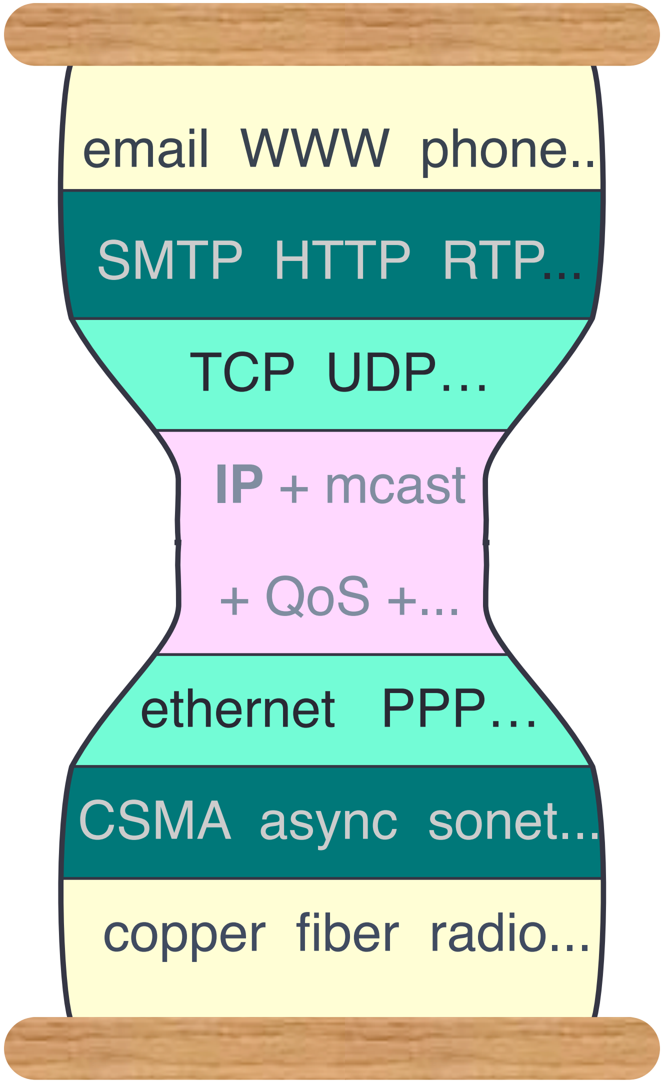
- All learn about OSI 7-layer model
- OSI: ASN.1 as common rendering of application data structures
  - used in LDAP and SNMP (and H.323)
- Internet never really had presentation layer
  - approximations: common encoding (TLV, RFC 822 styles)
- Now, XML (& JSON?) as the design choice by default



Internet acquires session layer

- Originally, meant for data sessions
- Example (not explicit): ftp control connection
- Now, separate data delivery from session setup
    - address and application configuration
    - deal with mobility
    - will see later as RTSP, SIP and H.323

Putting on Weight



requires more functionality from underlying networks

Classical Internet philosophy

- Almost all functions except packet routing are in end systems
  - reliability, security, mobility
  - → facilitate edge innovation
- The network is *common carrier*
  - common carrier: "Under the law, a common carrier is required to make its infrastructure available to everyone willing to pay to access it."
  - does not differentiate (or discriminate against) traffic types or applications or customers
  - → facilitate edge innovation
  - see "network neutrality" discussion

Network philosophy: End-to-end argument

- "All functions need to be performed at the edge"
  - edge (end) needed for correctness
  - middle can get in the way or sometimes help (performance)
  - → design vs. "moral" argument ("networks should respect e2e")
  - → transparency
  - → *The Rise of the Dumb Network*
    - common carriage = don't discriminate against

see also http://www.aarnet.edu.au/engineering/wgs/video/presentations/2004Feb/clark.ppt

End-to-end argument – but…

- ISPs and carriers prefer to sell services, not bit pipes → price & service differentiation
- network protection against "bad" users
- user protection against "bad" data
- dealing with badly designed protocols (e.g., caching, wireless)

Some end-vs-middle issues

Issue	Why middle?	Why end?
Error recovery (FEC, ARQ)	Smaller recovery time (link RTT << e-e RTT)	“real” reliability (including router-induced losses)
Congestion control	Faster feedback more accurate link congestion information	simpler end system
Caching	Disruption tolerance	Access control, visibility
Buffering	Higher TCP throughput (but: buffer bloat)	Cheaper memory
Encryption	Traffic source/destination hiding	Untrustworthy networks

Internet architecture documents (readings)

- http://www.ietf.org/rfc/rfcXXXX.txt
  - http://www.zvon.org/ in HTML format with cross-referencing
- RFC 1287 (*Towards the Future Internet Architecture*)
- RFC 2101 (*IPv4 Address Behaviour Today*)
- RFC 2775 (*Internet Transparency*)
- RFC 3234 (*Middleboxes: Taxonomy and Issues*)

Guidelines

- Middleboxes:
  - firewalls, network address translators, transparent caches
  - connection disruptions
  - application surprises
  - energy consumption (refresh)
  - see (e.g.,) SIGCOMM 2011 "*An Untold Story of Middleboxes in Cellular Networks*"
- Minimize pain →
  - Coordinated
  - Discoverable (not today)
  - Discoverable behavior
    - e.g., timeouts, port blocking, NAT behavior

SERIALIZATION

Serialization

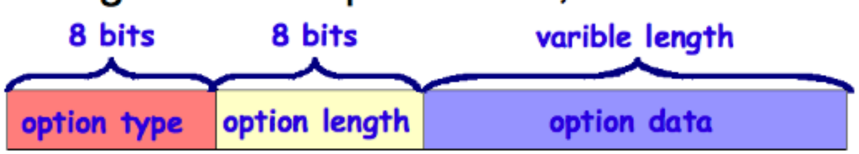
- It lets you take an object or group of objects, put them on a disk or send them through a wire or wireless transport mechanism, then later, perhaps on another computer, reverse the process: resurrect the original object(s). The basic mechanisms are to flatten object(s) into a one-dimensional stream of bits, and to turn that stream of bits back into the original object(s).
  - Like the Transporter on Star Trek, it's all about taking something complicated and turning it into a flat sequence of 1s and 0s, then taking that sequence of 1s and 0s (possibly at another place, possibly at another time) and reconstructing the original complicated "something.” [C++ FAQ]

Serialization: TLV

IPv6

TLV-encoding (Type-Length-Value)

- Type: identifier of type of option
- Two highest bits of Type: unrecognised option processing:
  - 00 – skip over the option and continue
  - 01 – discard the packet
  - 10 – discard the packet and send ICMPv6
  - 11 – discard the packet and send ICMPv6 only if destination isn't IPv6 multicast address
- Third highest-order bit of Type: whether (1) or not (0) Option Data can change en-route to the final destination
- Length: length of the Option Data, in octets



Serialization: text-based

```xml
<?xml version="1.0"?>
<dept-tickets>
 <dept-chief>Greg Sanguinetti"/>
 <dept-id>12389289/>
 <ticket id="034567910" code="301">
  <offender>
   <name>John Smith</name>
   <license-number>10003887</license-number>
   <plate-number>9AER9876</plate-number>
  </offender>
  <offence-date>09/30/2005</offence-date>
  <location>
   <state>CA</state>
   <city>SJ</city>
   <intersection>West Tasman Dr.-Great America Pkwy.</intersection>
  </location>
  <officer>
   <officer-name>Paul Greene</officer-name>
   <officer-badge>7652323</officer-badge>
   <cruiser-plate-number>6TYX0923</cruiser-plate-number>
  </officer>
  <description>Failure to stop at red light</description>
  <fine>100</fine>
 </ticket>
 <ticket id="..." code="...">
 ...
 </ticket>
 <ticket id="..." code="...">
 ...
 </ticket>
</dept-tickets>
```

XML

```json
{
  "firstName": "John",
  "lastName": "Smith",
  "isAlive": true,
  "age": 25,
  "height_cm": 167.6,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021-3100"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "office",
      "number": "646 555-4567"
    }
  ],
  "children": [],
  "spouse": null
}
```

JSON

Serialization: RFC 822

Delivered-To: hgs10@lionmailmx.cc.columbia.edu
Received: by 10.140.158.132 with SMTP id e126csp131562qhe;
Thu, 28 Aug 2014 14:01:48 -0700 (PDT)
Return-Path: etickets@amtrak.com
Return-Path: etickets@amtrak.com
Received: from unknown (HELO etvswas01p) ([10.14.128.202])
by phlsmtprelay01.amtrak.com with ESMTP; 28 Aug 2014 16:55:42 -0400
Date: Thu, 28 Aug 2014 17:01:30 -0400 (EDT)
From: etickets@amtrak.com
To: HGS@cs.columbia.edu, HENNING.SCHULZRINNE@FCC.GOV
Message-ID: <633700356.JavaMail.TDDServerProd@amtrak.com>
Subject: Amtrak: eTicket and Receipt for Your 09/10/2014 Trip
MIME-Version: 1.0
Content-Type: multipart/mixed;
boundary="----=_Part.1409259690306”
MIME-Version: 1.0
Content-Type: multipart/mixed;

Serialization: ASN.1

```
FooProtocol DEFINITIONS ::= BEGIN

    FooQuestion ::= SEQUENCE {
        trackingNumber INTEGER,
        question       IA5String
    }

    FooAnswer ::= SEQUENCE {
        questionNumber INTEGER,
        answer         BOOLEAN
    }

END
```

serialization = convert data structure into (linear) byte stream

like C, without pointers…

```
myQuestion FooQuestion ::= {
    trackingNumber     5,
    question           "Anybody there?"
}
```

30 13 02 01 05 16 0e 41 6e 79 62 6f 64 79 20 74 68 65 72 65 3f

Serialization trade-offs

Property	TLV	ASN.1	JSON, XML, RFC 822
Space	compact	somewhat	inefficient
Time	efficient	not aligned	inefficient
Energy	efficient	somewhat	inefficient
Structured	yes	yes	JSON, XML
Self-describing	backwards-compatible	backwards-compatible	labeled
Signable	mostly	DER	canonical formats

Serialization: specification

- Part of XML, JSON, ASN.1 definition
- ABNF for low-level textual specification
- RFC 5234

```
from            =   "From:" mailbox-list CRLF

sender          =   "Sender:" mailbox CRLF

reply-to        =   "Reply-To:" address-list CRLF

addr-spec       =   local-part "@" domain

local-part      =   dot-atom / quoted-string / obs-local-part

domain          =   dot-atom / domain-literal / obs-domain

domain-literal  =   [CFWS] "[" *([FWS] dtext) [FWS] "]" [CFWS]

dtext           =   %d33-90 /          ; Printable US-ASCII
                    %d94-126 /         ;  characters not including
                    obs-dtext          ;  "[", "]", or "\"
```

RFC 5322 (email)