Network Applications

Andy Carpenter

(Andy.Carpenter@manchester.ac.uk)

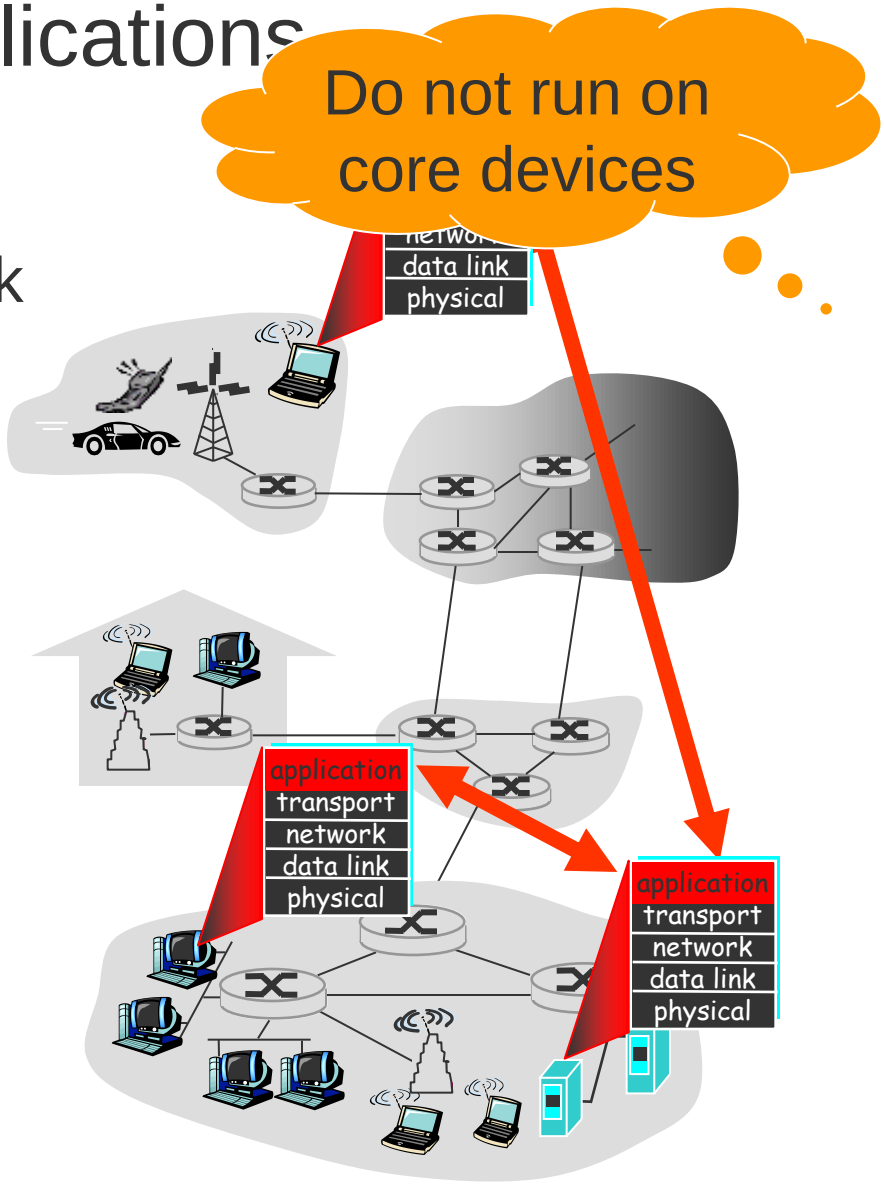
Elements these slides come from Kurose and Ross, authors of "Computer Networking: A Top-down Approach", and are copyright Kurose and Ross

Example Network Applications

- e-mail
- web
- instant messaging
- remote login
- P2P file sharing
- multi-user network games
- streaming stored video clips
- voice over IP (VoIP)
- real-time video conferencing
- grid computing

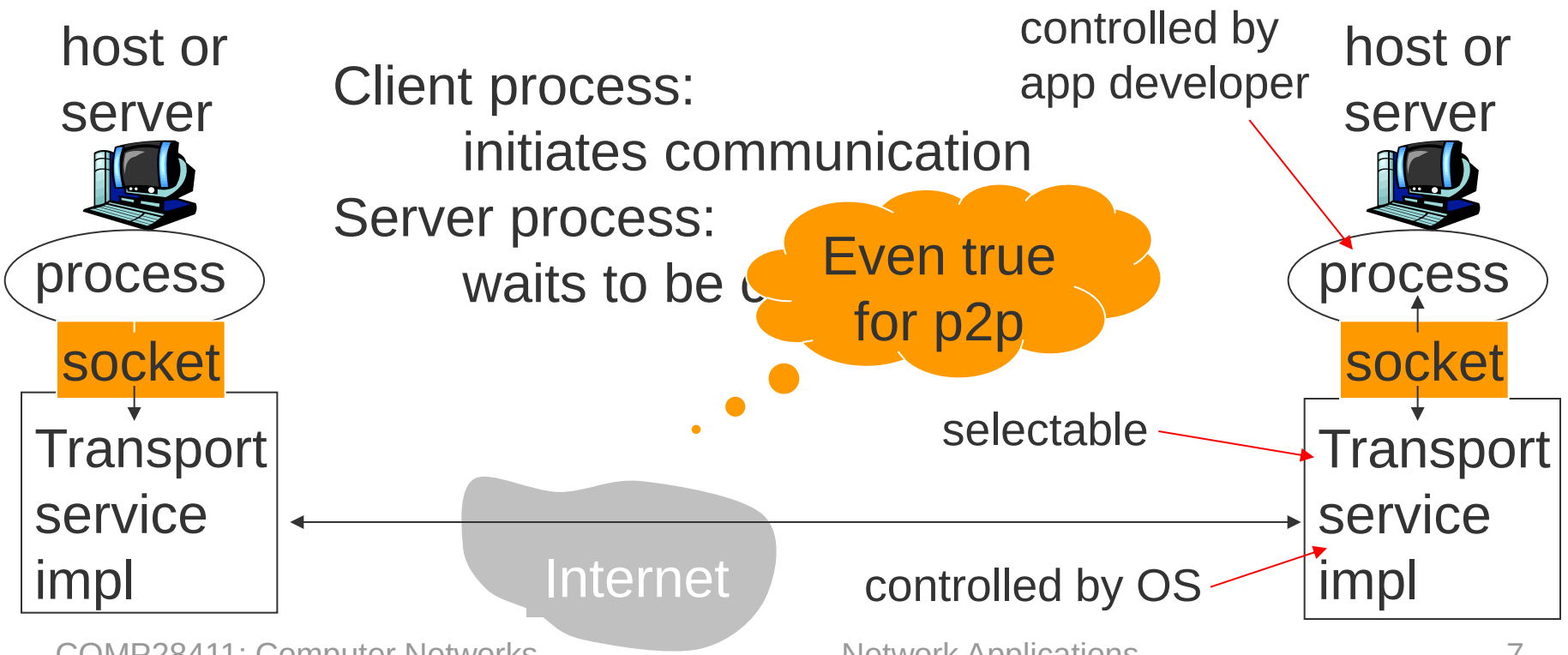
Network Applications

- Programs that:
  - communicate over network
  - run on end systems
- Issues:
  - architecture
  - comms infrastructure
  - protocols, addressing
  - control vs. data
  - understanding data
  - buffering, state
  - extensibility, scalability



Architecture: End-points

- Application end-point is a process
- Communicate by exchanging messages
- Messages sent/received via socket



Application QoS: (Some) Params

Data loss

- some apps (e.g., audio) can tolerate some loss
- other apps (e.g., file transfer, telnet) require 100% reliable data transfer

Timing

- some apps (e.g., Internet telephony, interactive games) require low delay to be "effective"

Throughput

- some apps (e.g., multimedia) require minimum amount of throughput to be "effective"
- other apps ("elastic apps") make use of whatever throughput they get

Security

- Encryption, data integrity, …

Application QoS: Requirements

Application	Data loss	Throughput	Time Sensitive
file transfer	no loss	elastic	no
e-mail	no loss	elastic	no
web	no loss	elastic	no
real-time audio/video	loss-tolerant	Audio: 5kbps-1Mbps Video: 10kbps-5Mbps	yes, 100's msec
stored audio/video	loss-tolerant	Same as above	yes, few secs
interactive games	loss-tolerant	few kbps upwards	yes, 100's msec
instant messaging	no loss	elastic	yes and no

Internet Transport Service Models

TCP service:

- *connection-oriented:* setup required between client and server processes
- *reliable transport* between sending and receiving process
- *flow control:* sender won't overwhelm receiver
- *congestion control:* throttle sender when network overloaded
- *does not provide:* timing, minimum throughput guarantees, security

UDP service:

- unreliable data transfer between sending and receiving process
- does not provide: connection setup, reliability, flow control, congestion control, timing, throughput guarantee, or security

Application Protocols

- Application protocols enhance transport service model to precise communication service needs of application
- Define:
  - types of message exchanged; e.g. request
  - message syntax;
    - fields present and their delineation
  - message semantics; meaning of fields
  - message exchange rules
- Ways protocols are defined:
  - RFCs, open-standards allowing interoperability
  - proprietary implementations, e.g. Skype

Application Protocol Examples

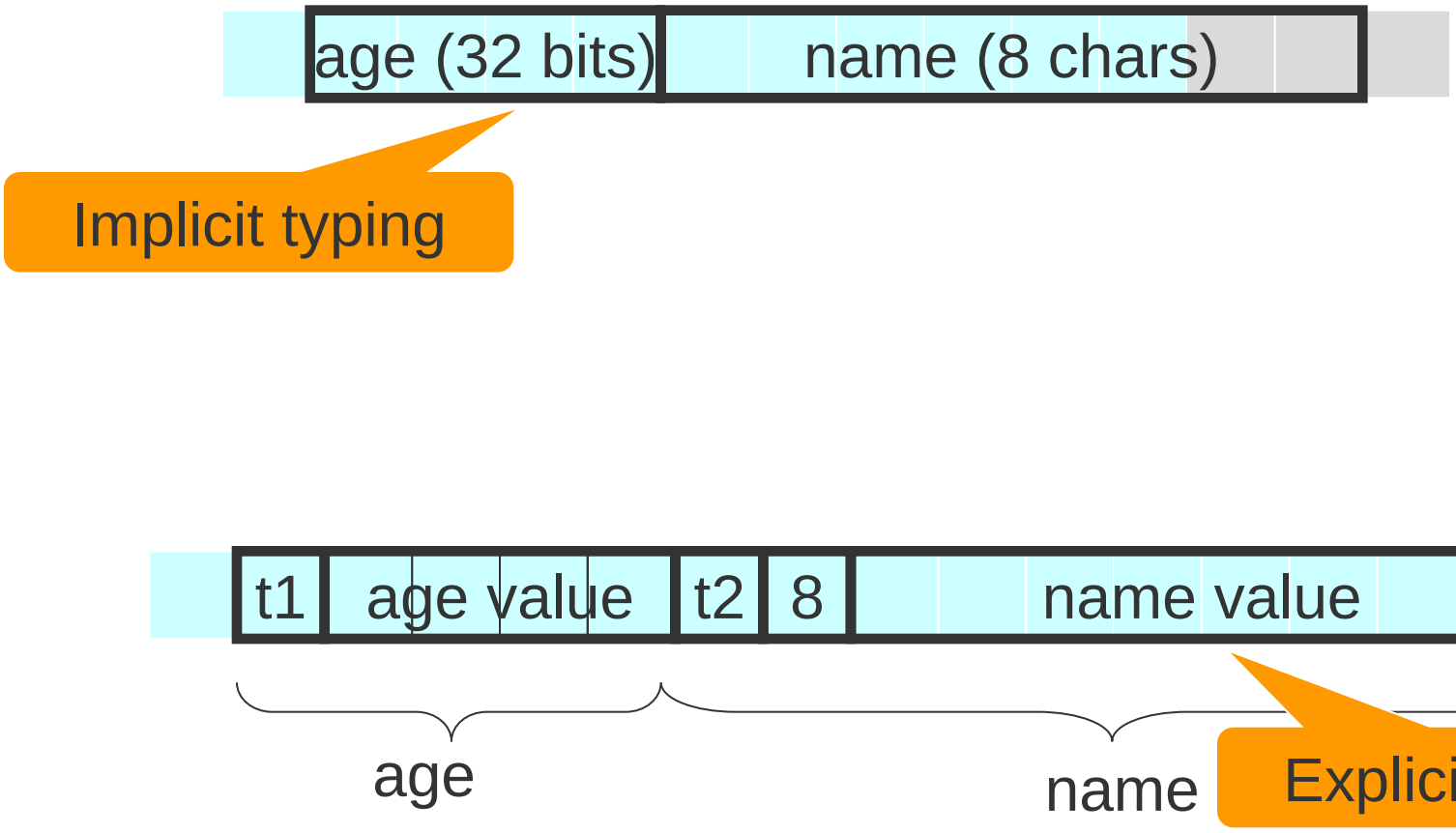
Application	Application layer protocol	Transport layer protocol
e-mail	SMTP [RFC 2821]	TCP
remote terminal access	Telnet [RFC 854]	TCP
web	HTTP [RFC2616]	TCP
file transfer	FTP [RFC 959]	TCP
streaming multimedia	HTTP (e.g. Youtube), RTP [RFC 1889]	TCP or UDP
Internet telephony	SIP, RTP proprietary (e.g. Skype)	Usually UDP

Why UDP?

Application Data

- What does this decimal byte sequence mean?
  - 72 101 108 108 111 32 99 108 97 115 115 32
- Application source and destination must:
  - each make same interpretation
- Also want efficient transmission (encoding) of data
- Compression minimises size on cable; not considered further
- Issues:
  - type and meaning of data (understanding)
  - representation for data in transit
  - when presentation encoding/decoding performed

Data: Implicit/Explicit Typing



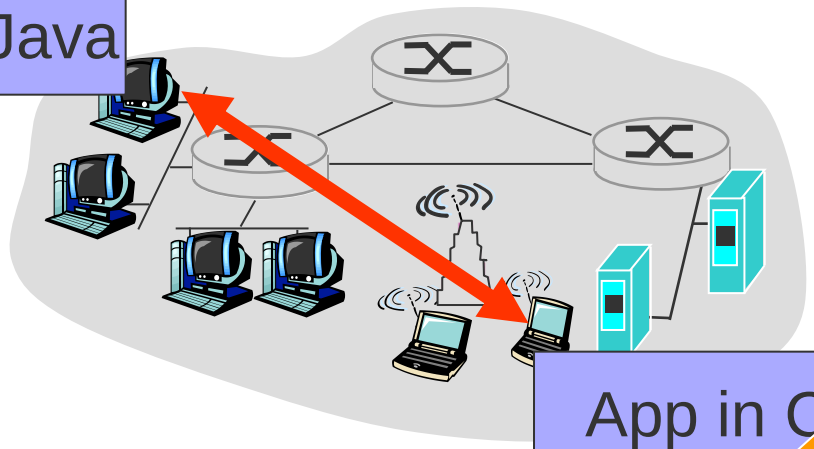
Data: Need for Conversion

App in Java

Age: (32-bit int)

Name: (chars)

Addr: (ref)



App in C

Age: (16-bit int)

Name: (chars)

0 <= age <= 150

What representation is transmitted?

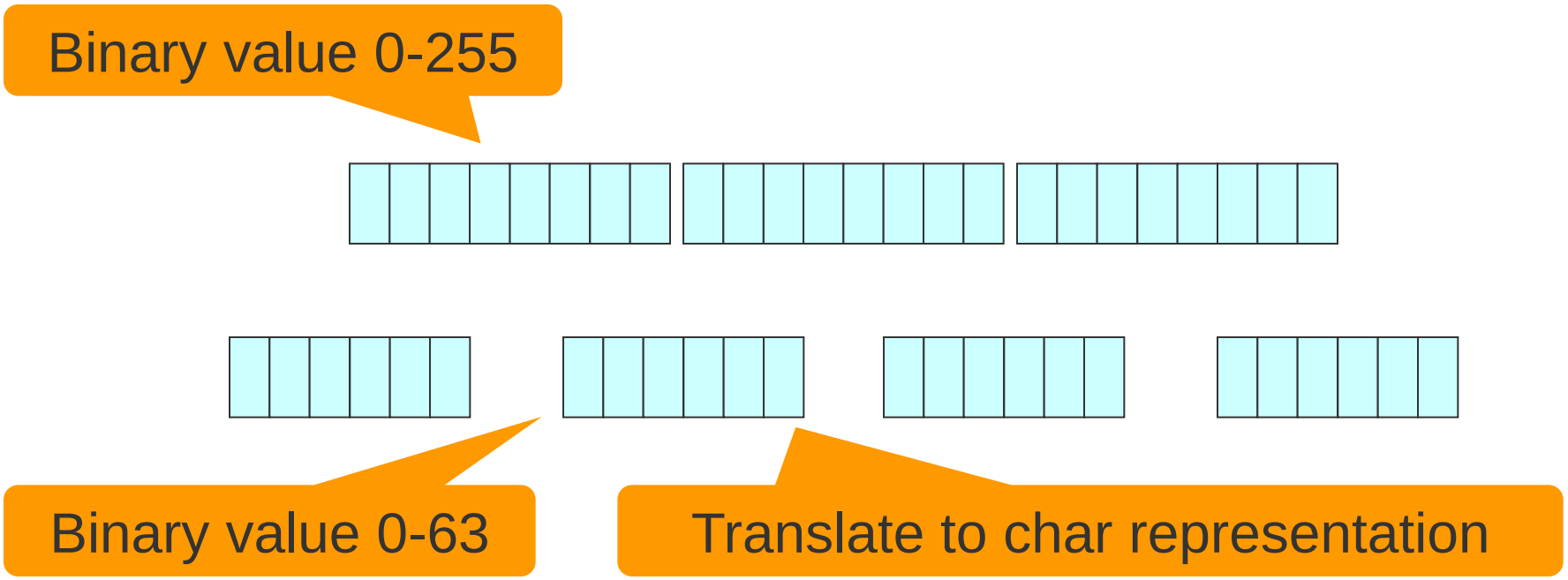
How many characters are sent?

Data: Conversion Strategies

- Canonical (standard) network representation
  - source translates to, destination translates from
  - need to understand two translations
  - for same architectures, may be two translations
  - implicit or explicit typing of primitive values
- Receiver-makes-right representation
  - transmit in internal format
  - avoids unnecessary conversion
  - must understand N translations
  - must have explicit description of what receive

Data: Encoding

- Data may be encoded for transmission, e.g.
  - Email sends binary data using MIME base64
  - (assumes that can only transfer 7-bit ASCII data)
- MIME base64: sends 3 bytes as 4 ASCII chars



Data: MIME base64 Encoding

	0	1	2	3	4	5	6	7
0	A	B	C	D	E	F	G	H
8	I	J	K	L	M	N	O	P
16	Q	R	S	T	U	V	W	X
24	Y	Z	a	b	c	d	e	f
32	g	h	i	j	k	l	m	n
40	o	p	q	r	s	t	u	v
48	w	x	y	z	0	1	2	3
56	4	5	6	7	8	9	+	/

Example:

Encode value 42

42 = 40 + 2

42 ≡ q

Compression is also a form of encoding

Control vs. Data: Telnet

- Periodically client and server exchange commands
- Commands are embedded within application data
- Use escape character, Interpret As Command, to
- indicates next char is a command

Source

Data	Data

Send: Source + Command

Data	IAC	Cmd	Data

- If a command has an option, it follows the command
- If IAC occurs in application data stream, duplicated

Source

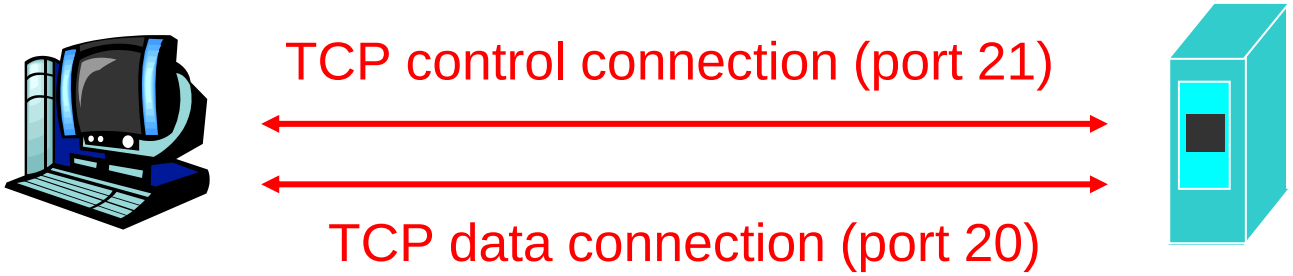
Data	IAC	Data

Send

Data	IAC	IAC	Data

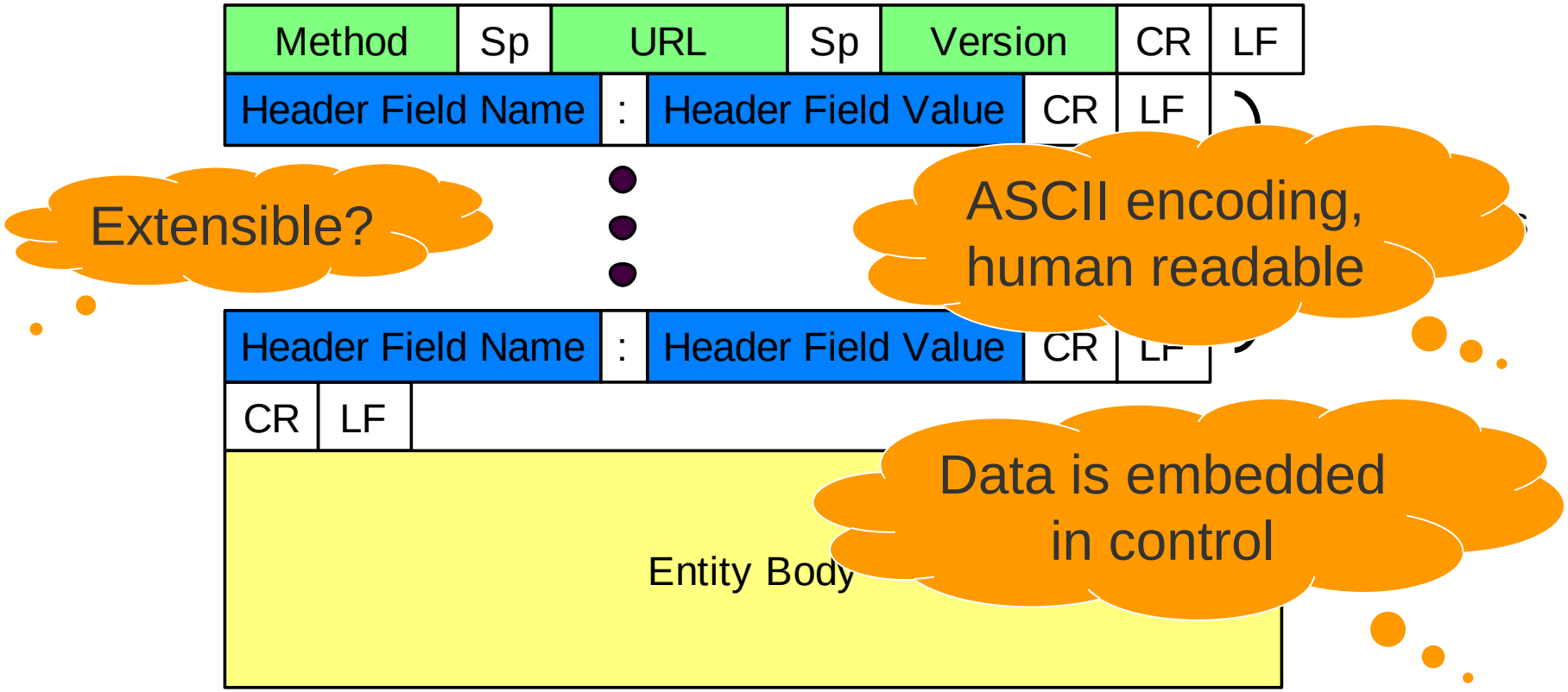
Same as \ in Java strings

Control vs. Data: FTP



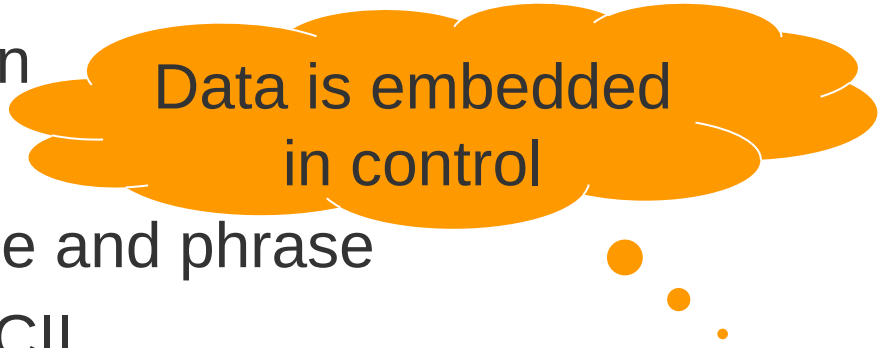
- FTP client contacts FTP server at port TCP 21
- Client authorized over control connection
- When server receives file transfer command
  - server opens 2nd TCP connection (for file) to client
- After one transfer, server closes data connection.
- FTP server maintains “state”:
- current directory, earlier authentication

Control vs. Data: HTTP

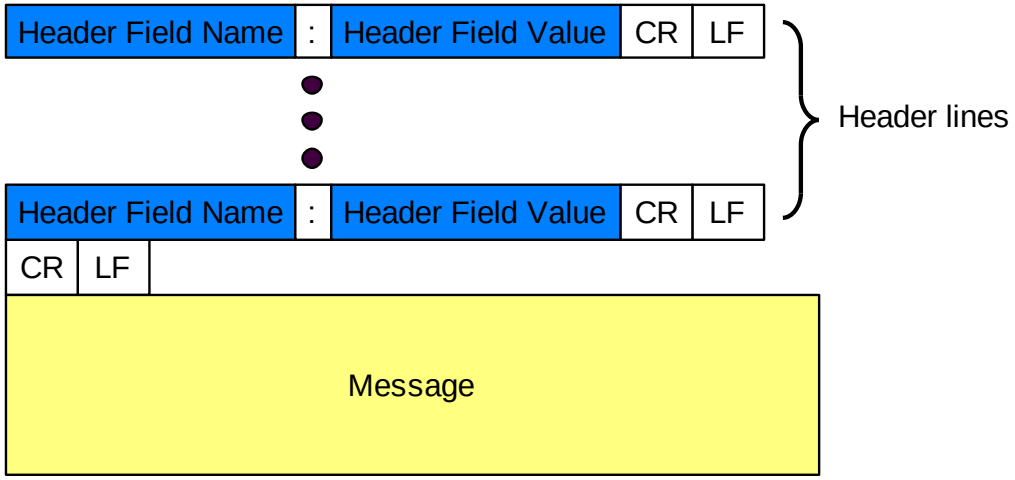


Control vs. Data: Email

- Simple Mail Transfer Protocol (RFC 5321)
- Uses persistent TCP connections to server port 25
- Three phases of transfer:
  - handshaking (greeting)
  - transfer of messages
  - closure
- Command/response interaction
  - commands: ASCII text
  - response: ASCII status code and phrase
- Messages must be in 7-bit ASCII
- SMTP is push; HTTP is pull



Control vs. Data: RFC822



Summary

- Begun to look at applications
- What is important is underlying principles
  - not how a particular application works
- Demands placed on underlying network infrastructure
- Architecture: p2p or client-server
- Understanding data
- Relationship of data and control
- More next time