Concurrent Computing (Computer Networks)

Daniel Page

Department of Computer Science,
University Of Bristol,
Merchant Venturers Building,
Woodland Road,
Bristol, BS8 1UB. UK.
⟨Daniel.Page@bristol.ac.uk⟩

March 7, 2016

Keep in mind there are *two* PDFs available (of which this is the latter):

1. a PDF of examinable material used as lecture slides, and
2. a PDF of non-examinable, extra material:
   - the associated notes page may be pre-populated with extra, written explaination of material covered in lecture(s), plus
   - anything with a "grey'ed out" header/footer represents extra material which is useful and/or interesting but out of scope (and hence not covered).

Notes:

Notes:

- **Question**:
  1. what is this part of the unit about, and
  2. how does it relate to the rest of the unit?
- **Answer**: it should be obvious that

$$\text{concurrent systems} \supset \text{computer networks} \Rightarrow \text{concurrency} \cup \text{communication}$$

  but beyond this, we need a motivating example ...

Notes:

http://en.wikipedia.org/wiki/File:Ken_Thompson_(sitting)_and_Dennis_Ritchie_at_PDP-11_(2876612463).jpg

Notes:

- This photograph shows (Turing award winners) Ken Thompson and Dennis Ritchie, creators of the UNIX operating system, for example, using a PDP-11 computer. Suspending disbelief wrt. some technical details, you *could* think of this as a simple concurrent system: the PDP-11 itself is used via a Teletype 33 terminal (at which Thompson is sitting). Of course, if *this* PDP-11 was running a proto-UNIX then it captures yet another layer of concurrency by allowing multi-tasking and hence time-shared access (via *multiple* terminals).
- Richie himself documents the photograph at

  http://cm.bell-labs.com/who/dmr/picture.html

  noting that you can find out about the PDP-11 and Teletype 33 via

  http://en.wikipedia.org/wiki/PDP-11

  and

  http://en.wikipedia.org/wiki/Teletype_Model_33

  respectively.
- The acronym **TeleTYpe (TTY)** lives on via Linux device entries such as `/dev/tty*`.

PDP-11 ↔ Teletype 33 (1)

- **TIA-232-F** [8] specifies a **communication medium** used to connect
  - a **Data Terminal Equipment (DTE)** or *master* device, e.g., a workstation, with
  - a **Data Communication Equipment (DCE)** or *slave* device, e.g., a MODEM

  and hence provide a (more abstract) **communication channel** (or **link**).
- Question: what terms can you think of to characterise this channel?

Notes:

- Already some problems emerge with the terminology used. Here we consider a communication channel (or just channel) to be a pathway through which communication can occur, i.e., over which data can be communicated; this includes both **physical channels** (realised using some medium, such as a wire) but also **logical channels** (which are more abstract, and may, for example, be realised by multiple physical channels). Elsewhere the term link is often used (more or less) synonymously, but more often for a physical channel and sometimes just to mean a connection. For instance,
  - in telecommunications, terms such as **uplink** and **downlink** are used to mean channels in one direction or another,
  - in the OSI model there is a **data link layer**.
- The standard name perhaps needs some explanation:
  - The Electronic Industries Association (EIA), later *Alliance* replaced *Association*, published a standard called RS-232 in 1962.
  - RS-232 was later renamed EIA-232; subsequent amendments and updates are indexed by letter, e.g., EIA-232-F.
  - EIA passed responsibility for the standard to the Telecommunications Industries Association (TIA), so now the standard is TIA-232-F.
  - For some reason, the TIA charge you ~ $150 for TIA-232-F!
- Strictly speaking, it seems Teletype 33 terminals only started to use (what was then) TIA-232-F in later generations (given it pre-dates RS-232, for example): a current-loop interface

  http://en.wikipedia.org/wiki/Digital_current_loop_interface

  was used in earlier generations, the design of which is out of scope for this discussion.

PDP-11 ↔ Teletype 33 (1)

- **TIA-232-F** [8] specifies a **communication medium** used to connect
  - a **Data Terminal Equipment (DTE)** or *master* device, e.g., a workstation, with
  - a **Data Communication Equipment (DCE)** or *slave* device, e.g., a MODEM

  and hence provide a (more abstract) **communication channel** (or **link**).
- Question: what terms can you think of to characterise this channel?
- (Incomplete) answer:
  - **serial** (i.e., can communicate 1-bit per-unit of time),
  - (upto) **full duplex** (i.e., communication can occur simultaneously in both directions),
  - **synchronous** *or* **asynchronous** (i.e., can require *or* avoid shared control wrt. timing),
  - **direct** (i.e., there are no intermediate devices),
  - **unicast** (i.e., a single device recieves any transmitted data),
  - ...

Notes:

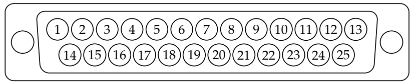
- Already some problems emerge with the terminology used. Here we consider a communication channel (or just channel) to be a pathway through which communication can occur, i.e., over which data can be communicated; this includes both **physical channels** (realised using some medium, such as a wire) but also **logical channels** (which are more abstract, and may, for example, be realised by multiple physical channels). Elsewhere the term link is often used (more or less) synonymously, but more often for a physical channel and sometimes just to mean a connection. For instance,
  - in telecommunications, terms such as **uplink** and **downlink** are used to mean channels in one direction or another,
  - in the OSI model there is a **data link layer**.
- The standard name perhaps needs some explanation:
  - The Electronic Industries Association (EIA), later *Alliance* replaced *Association*, published a standard called RS-232 in 1962.
  - RS-232 was later renamed EIA-232; subsequent amendments and updates are indexed by letter, e.g., EIA-232-F.
  - EIA passed responsibility for the standard to the Telecommunications Industries Association (TIA), so now the standard is TIA-232-F.
  - For some reason, the TIA charge you ~ $150 for TIA-232-F!
- Strictly speaking, it seems Teletype 33 terminals only started to use (what was then) TIA-232-F in later generations (given it pre-dates RS-232, for example): a current-loop interface

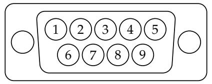
  http://en.wikipedia.org/wiki/Digital_current_loop_interface

  was used in earlier generations, the design of which is out of scope for this discussion.

PDP-11 ↔ Teletype 33 (2) – Communication medium

- The medium itself is formed from
  1. an interface (from a choice of two)





Notes:

- The DTE (resp. DCE) connector has a male (resp. female) gender, with the pins numbered from 1 to $n \in \{9, 25\}$ to match the standard.
- Some pins of the 25-pin connector are reserved or unassigned (e.g., 9, 10 and 11), with the assigned pins associated with six signal classes:
  1. electrical properties,
  2. primary communication channel,
  3. secondary communication channel,
  4. control and status,
  5. timing, and
  6. test and debug.

  The 25-pin connector clearly supports more functionality, by virtue of including more pins, than the 9-pin alternative; some of the associated signals are specific to (or at terminology from) use-cases where the DCE is a MODEM (e.g., "ring indication" or "carrier detect"). Note that the signals are *uni*directional, with an implied direction relative to the DTE.
- The electrical characteristics of the standard imply various constraints, e.g., on signal timings (to ensure signal integrity), signalling rate (of 20000baud at most), and maximum cable length possible (about 50ft based on capacitance limits).
- The 25-pin connector allows a secondary communication channel: this is (or was, it is quite unusual) used for remote control (e.g., diagnostics) of the MODEM, for example.

PDP-11 ↔ Teletype 33 (2) – Communication medium

- The medium itself is formed from

  2. a pin assignment which dictates how the interface is used

Class	Purpose	Mnemonic	Direction	9-pin	25-pin
Electrical properties	Reference (or signal) ground	GND	common	5	7
	Protective (or shield) ground	PG	common		1
Primary communication channel	Transmitted Data	TxD	DTE → DCE	3	2
	Recieved Data	RxD	DCE → DTE	2	3
	Request To Send	RTS	DTE → DCE	7	4
	Clear To Send	CTS	DCE → DTE	8	5
Secondary communication channel	Secondary Transmitted Data	STxD	DTE → DCE		14
	Secondary Recieved Data	SRxD	DCE → DTE		16
	Secondary Request To Send	SRTS	DTE → DCE		19
	Secondary Clear To Send	SCTS	DCE → DTE		13
Control and status	Data Set Ready	DSR	DCE → DTE	6	6
	Data Terminal Ready	DTR	DTE → DCE	4	20
	Carrier Detect	CD	DCE → DTE	1	8
	Secondary Carrier Detect	SCD	DCE → DTE		12
	Ring Indicator	RI	DCE → DTE	9	22
	Data Signal Rate		DTE → DCE		23
Timing	External Transmitter Clock	ETC	DTE → DCE		24
	Transmitter Clock	TC	DCE → DTE		15
	Reciever Clock	RC	DCE → DTE		17
Test and debug	Local Loop-back	LL	DTE → DCE		18
	Remote Loop-back	RL	DTE → DCE		21
	Test Mode	TM	DCE → DTE		25

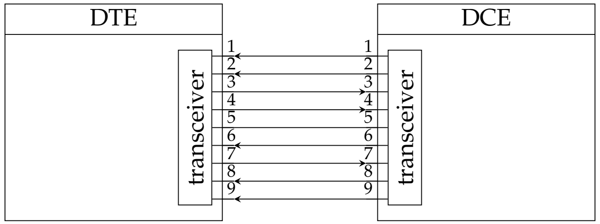
Notes:

- The DTE (resp. DCE) connector has a male (resp. female) gender, with the pins numbered from 1 to $n \in \{9, 25\}$ to match the standard.
- Some pins of the 25-pin connector are reserved or unassigned (e.g., 9, 10 and 11), with the assigned pins associated with six signal classes:
  1. electrical properties,
  2. primary communication channel,
  3. secondary communication channel,
  4. control and status,
  5. timing, and
  6. test and debug.

  The 25-pin connector clearly supports more functionality, by virtue of including more pins, than the 9-pin alternative; some of the associated signals are specific to (or at terminology from) use-cases where the DCE is a MODEM (e.g., "ring indication" or "carrier detect"). Note that the signals are *uni*directional, with an implied direction relative to the DTE.
- The electrical characteristics of the standard imply various constraints, e.g., on signal timings (to ensure signal integrity), signalling rate (of 20000baud at most), and maximum cable length possible (about 50ft based on capacitance limits).
- The 25-pin connector allows a secondary communication channel: this is (or was, it is quite unusual) used for remote control (e.g., diagnostics) of the MODEM, for example.

PDP-11 ↔ Teletype 33 (3) – Communication medium

- The medium is managed using some form of **transceiver**, i.e.,



- Example: depending on the context, either end-point might
  1. rely on **Universal Asynchronous Receiver/Transmitter (UART)** hardware to provide a higher-level interface, or
  2. directly interface with low-level GPIO pins, ensuring correct voltage levels and timing in software (cf. bit-banging).

Notes:

PDP-11 ↔ Teletype 33 (4) – Communication protocol

Low-level protocol

- Fact: TIA-232-F does *not* define a **communication protocol** ...
- ... the ~ 100 year old protocol we *still* use actually stems from the design of electronic typewriters (cf. teletype):

Quote

> *In printing telegraphy, the method of selective signaling which consists in operating a transmitter solely under local control to impart to a line character signals of uniform length and each comprising the same number of positive or negative impulses in quick succession and without perceptible spacing intervals between the impulses, initiating the operation of a selecting receiver switch mechanism in response to transmitted impulses at the beginning of each signal, timing the operation of the switch mechanism in synchronism with the transmitted impulses of each signal independently of the line circuit land restoring the same to a condition of rest at the completion of each signal.*
>
> – *Krum [9, Claim 1]*, 1918 *(!)*

- Translation:
  1. all (binary) data transmitted is encoded as discrete voltage levels (cf. "positive and negative impulses"),
  2. the data (cf. "character") has start and end markers added st. the receiver knows when transmission occurs,
  3. this allows asynchronous communication (cf. "local control").

Notes:

- So ... the DTE and DCE pre-agree a set of **signalling parameters**, e.g.,

$$
\begin{array}{lcl}
\text{baud rate} & \in & \{110, 300, 600, 1200, 2400, 4800, 9600, 19200, \ldots\} \\
\text{number of data bits} & \in & \{5, 6, 7, 8\} \\
\text{parity type} & \in & \{\text{none}, \text{odd}, \text{even}\} \\
\text{number of stop bits} & \in & \{1, 1.5, 2\}
\end{array}
$$

and, as such, agree how data is **framed**, i.e.,

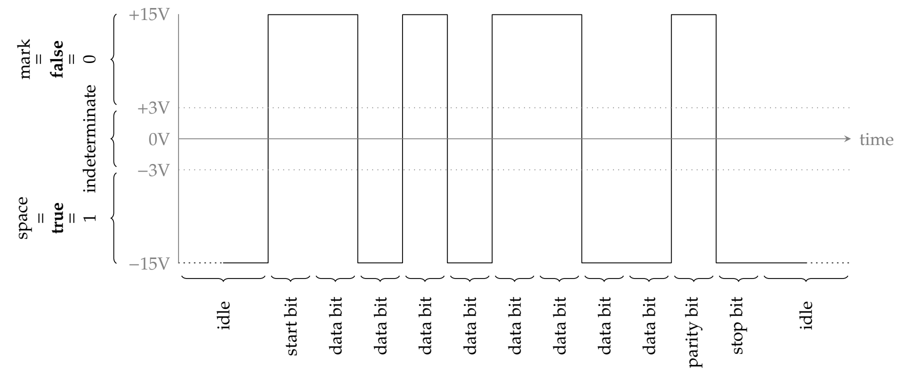
$$D \;\mapsto\; F \;=\; \text{start bit} \parallel D \parallel \text{parity bit} \parallel \text{stop bits}$$

where

  - the **baud rate** the number of transitions per-second of a given signal (e.g., TxD or RxD),
  - the **start bit** mark the start of a frame,
  - the **stop bit(s)** mark the end of a frame,
  - the **data bits** are the data, and
  - the **parity type** supports (optional) error detection.

Notes:

- Considering the same protocol in a more general setting than TIA-232-F, there are some other parameters that need to be fixed:
  - the endianness, i.e., whether D is transmitted LSB or MSB first, and
  - the voltage levels (since the same protocol can be used over other mediums (e.g., using TTL)) and polarity.
- In general, care is needed to avoid confusion between baud rate and raw (or gross) bit rate (or bits per-second). For TIA-232-F, however, there is one transition per-bit so the bit and baud rates are the same.
  1. The baud rate is the number of symbols communicated per-second (or the maximum signal transitions per-second) across the communication channel, so essentially is is the symbol rate. The use of symbols is intentionally abstract versus the use of bits, for example: it allows for communication systems where each symbol might have more than two states, so may encode multiple bits.
  2. The channel efficiency is a measure of the useful, versus the total, information communicated. The two may not be equal where the latter includes some control information, e.g., as the result of framing. This is relevant when characters per-second is quoted as a metric, for example.
- A short-hand specification such as $9600/8/N/1$ captures the parameters for baud rate (9600baud), data bits (8), parity type (N) and stop bits (1); for the parity type, N, O and E mean no, odd and even parity respectively.
- Although it might *seem* there should be a standard set of baud rates, there isn't! As a rough rule, supported rates have typically increased by multiples of 2 (or 1.5) and stem in part from a relationship to clock frequencies used by associated transceivers. Some supported rates have a historical motivation: 110baud, for example, was used by the Teletype 33 and similar terminal equipment.
- Use of anything other than 8-bit data may sound unattractive, but again has historical roots: the 5-bit and 7-bit options match requirements of **Baudot code** [3] and standard, non-extended **ASCII** [1], for example.

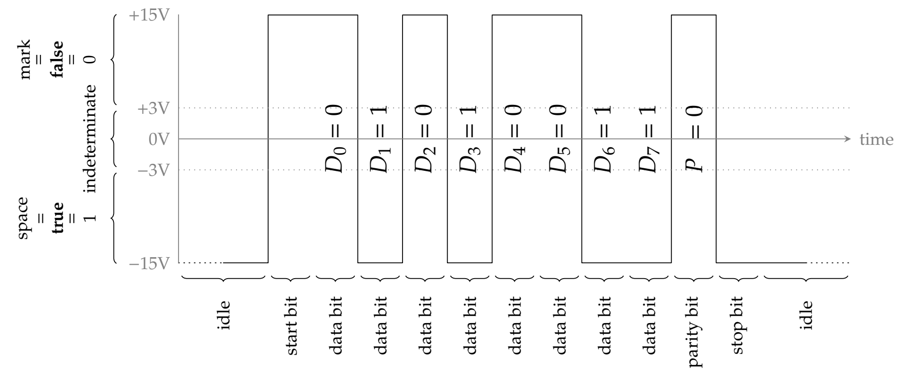


Notes:

- Historical terminology from teleprinters names the high (resp. low) voltage levels used for logic 0 (resp. 1) **mark** (resp. **space**). These are measured relative to the reference ground signal.
- Since there is no real scale along the x-axis, the baud rate here is sort of irrelevant. Note, however, that the parameters specify 8 data bits. The basic idea is as follows:
  - A start bit, which implies a transition from idle (even if the data is zero), signals the start of a frame. This causes the receiver to start sampling data using a clock configured to match the baud rate.
  - The receiver clock has a higher frequency (typically 16-times) than the baud rate: this allows the receiver to sample in the *middle* of each period to avoid skew wrt. the transmitter clock. Put another way, as long as the transmitter and receiver clocks are *locally* synchronised (so they do not become desynchronised in the duration of one frame) they need not be *globally* synchronised.
  - Whatever might cause desynchronisation (e.g., the receiver failing to detect the start bit due to noise), communication *re*synchronises itself. Since the frame size is fixed, there is a fixed period (wrt. the baud rate) between start and stop bits; if it receives a stop bit with the wrong value, it can then wait for a valid start/stop pair to resynchronise with the transmitter.
- Allowing 1.5 stop bits might seem odd, but highlights the fact that a "pause" is required rather than bits per se. In theory we should only ever need 1 stop bit, but larger pauses are needed to cater for slower receivers.

PDP-11 ↔ Teletype 33 (6) – Communication protocol

Low-level protocol



- We have

$$D = \langle D_0, D_1, D_2, D_3, D_4, D_5, D_6, D_7 \rangle = \langle 0, 1, 0, 1, 0, 0, 1, 1 \rangle \mapsto 202_{(10)}$$

noting that

$$\left(\bigoplus_{i=0}^{i<8} D_i \right) \oplus P = 0,$$

which we come back to later ...

Notes:

- Historical terminology from teleprinters names the high (resp. low) voltage levels used for logic 0 (resp. 1) **mark** (resp. **space**). These are measured relative to the reference ground signal.
- Since there is no real scale along the x-axis, the baud rate here is sort of irrelevant. Note, however, that the parameters specify 8 data bits. The basic idea is as follows:
  - A start bit, which implies a transition from idle (even if the data is zero), signals the start of a frame. This causes the receiver to start sampling data using a clock configured to match the baud rate.
  - The receiver clock has a higher frequency (typically 16-times) than the baud rate: this allows the receiver to sample in the *middle* of each period to avoid skew wrt. the transmitter clock. Put another way, as long as the transmitter and receiver clocks are *locally* synchronised (so they do not become desynchronised in the duration of one frame) they need not be *globally* synchronised.
  - Whatever might cause desynchronisation (e.g., the receiver failing to detect the start bit due to noise), communication *re*synchronises itself. Since the frame size is fixed, there is a fixed period (wrt. the baud rate) between start and stop bits; if it receives a stop bit with the wrong value, it can then wait for a valid start/stop pair to resynchronise with the transmitter.
- Allowing 1.5 stop bits might seem odd, but highlights the fact that a "pause" is required rather than bits per se. In theory we should only ever need 1 stop bit, but larger pauses are needed to cater for slower receivers.

PDP-11 ↔ Teletype 33 (7) – Communication protocol

High(er)-level protocol

- TIA-232-F supports (at least) two forms of **flow control**, namely
  1. hardware-based (RTS/CTS), and
  2. software-based (XON/XOFF)

  which you can think of as managing workload of the end-points.
- Example:
  - imagine the DTE is a (fast) workstation, and the DCE is a (slower) printer,
  - if the DCE cannot process the data transmitted fast enough, it will need to either buffer or discard data ...
  - ... *or* it could ask the DTE to slow down, or stop until it has caught up.

Notes:

- RTS/CTS:
  - When the transmitter is ready to transmit, it sets RTS; the receiver primes itself to monitor data signals.
  - When the receiver is ready to receive, it sets CTS; the transmitter commences transmission.
  - If either end-point is unable or unwilling to continue, it clears the associated control signal (i.e., RTS or CTS) so the other end-point stops.

Notes:

- RTS/CTS relies on hardware-based signalling, whereas XON/XOFF is arguably more flexible since it can be realised in software alone: this removes the reliance on 25-pin TIA-232-E itself, making it more generic (e.g., it can also be use via a 9-pin interface). On the other hand, however, the computational overhead of realising XON/XOFF in software may be significant *plus* it operates in-band: if we want to transmit a frame where the data encapsulated just happens to equal XON or XOFF, we need a mechanism to distinguish this from XON/XOFF signalling.
- Before TIA-232-E RTS signals to DCE that the DTE wants to transmit, and CTS to DTE from DCE that it may do so: one set of RTS/CTS signals is available, so the relationship is asymmetric. TIA-232-E presents an alternative that re-purposes the existing signals. Under the new scheme, there is no request and acknowledge, RTS (resp. CTS) simply signals to DCE (resp. DTE) that it can transmit. This alternative is termed RTS/CTS *handshaking*, with careful distinction between the two schemes clearly important.

PDP-11 ↔ Teletype 33 (8) – Communication protocol
High(er)-level protocol

- XON/XOFF:
  - Define two special symbols, e.g., with ASCII

$$\begin{array}{lclcll} XON & = & 19_{(10)} & = & 13_{(16)} & \mapsto & Ctrl-S \\ XOFF & = & 17_{(10)} & = & 11_{(16)} & \mapsto & Ctrl-Q \end{array}$$

    articulated as "transmit on" and "transmit off".
  - When the receiver is unable to accept more symbols, it sends the XOFF symbol; transmitter suspends transmission.
  - When the receiver is able to accept more symbols, it sends the XON symbol; transmitter resumes transmission.

Notes:

- RTS/CTS relies on hardware-based signalling, whereas XON/XOFF is arguably more flexible since it can be realised in software alone: this removes the reliance on 25-pin TIA-232-E itself, making it more generic (e.g., it can also be use via a 9-pin interface). On the other hand, however, the computational overhead of realising XON/XOFF in software may be significant *plus* it operates in-band: if we want to transmit a frame where the data encapsulated just happens to equal XON or XOFF, we need a mechanism to distinguish this from XON/XOFF signalling.
- Before TIA-232-E RTS signals to DCE that the DTE wants to transmit, and CTS to DTE from DCE that it may do so: one set of RTS/CTS signals is available, so the relationship is asymmetric. TIA-232-E presents an alternative that re-purposes the existing signals. Under the new scheme, there is no request and acknowledge, RTS (resp. CTS) simply signals to DCE (resp. DTE) that it can transmit. This alternative is termed RTS/CTS *handshaking*, with careful distinction between the two schemes clearly important.

Daniel Page ⟨Daniel.Page@bristol.ac.uk⟩
Concurrent Computing (Computer Networks) git # 2a1cd85 @ 2016-03-05 University of BRISTOL

- Based on some simple coding theory, i.e.,

Definition (parity)

The **parity** of an n-bit sequence X is defined as

$$\mathcal{P}(X) = \sum_{i=0}^{i<n} X_i \pmod 2 = \bigoplus_{i=0}^{i<n} X_i,$$

st. X has **even parity** (resp. **odd parity**) if $\mathcal{P}(X) = 0$ (resp. $\mathcal{P}(X) = 1$).

Definition (parity code)

An even (resp. odd) **parity code** appends a **parity bit** P to some sequence X st. $X \parallel P$ has even (resp. odd) parity:

$$\text{even parity} \begin{cases} \mathcal{P}(X) = 0 & \Rightarrow & P = 0 & \rightsquigarrow & \mathcal{P}(X \parallel P) = 0 \\ \mathcal{P}(X) = 1 & \Rightarrow & P = 1 & \rightsquigarrow & \mathcal{P}(X \parallel P) = 0 \end{cases}$$

$$\text{odd parity} \begin{cases} \mathcal{P}(X) = 0 & \Rightarrow & P = 1 & \rightsquigarrow & \mathcal{P}(X \parallel P) = 1 \\ \mathcal{P}(X) = 1 & \Rightarrow & P = 0 & \rightsquigarrow & \mathcal{P}(X \parallel P) = 1 \end{cases}$$

a (limited) form of **error detection** is possible

- the transmitter computes $\mathcal{P}(D)$, then appends an appropriate P before transmitting $D \parallel P$,
- the receiver recomputes $\mathcal{P}(D \parallel P)$ and signals an error if this does *not* match expectation.

Notes:

- Using this scheme, 1-bit transmission errors can be detected, but no **error correction** is possible: retransmission of the message is necessary.

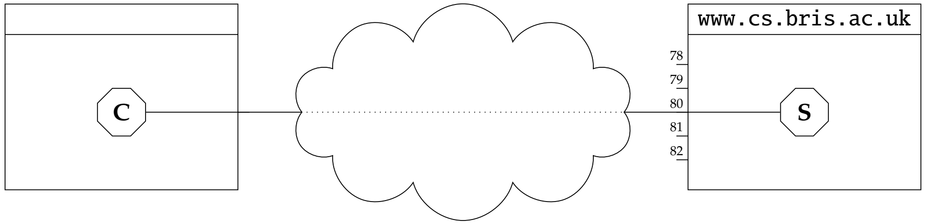
Conclusions

- Take away points:
  - TIA-232-F is a (very) *specific* example of more general concepts and requirements: the goal is a clear, *general* understanding.
  - Put simply, we want to understand how we can communicate data reliably and efficiently via a computer network ...

Notes:

Conclusions

- Remit:
  - understand a simple(ish) computer network

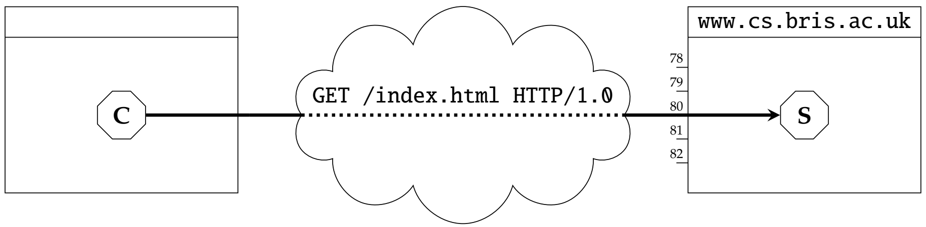


    wrt. support for a HTTP transaction, *but*
  - limit the detail and volume of coverage to fit allocated time.
- *Why*?!
  1. some of you may
     - develop network-related hardware or software, or
     - work in network operations,
  2. *most* of you will develop hardware or software that *depends* on a network, and
  3. many of the concepts and techniques *generalise*.

Notes:

Conclusions

- Remit:
  - understand a simple(ish) computer network

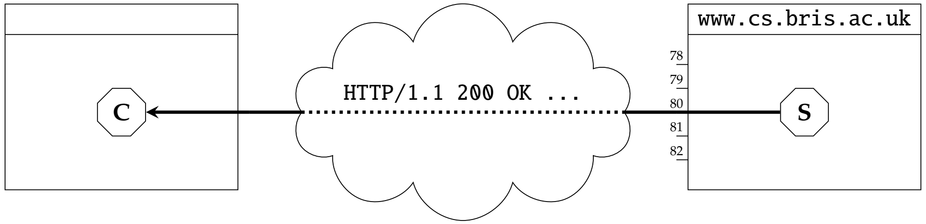


    wrt. support for a HTTP transaction, *but*
  - limit the detail and volume of coverage to fit allocated time.
- *Why*?!
  1. some of you may
     - develop network-related hardware or software, or
     - work in network operations,
  2. *most* of you will develop hardware or software that *depends* on a network, and
  3. many of the concepts and techniques *generalise*.

Notes:

Conclusions

- **Remit**:
  - understand a simple(ish) computer network



    wrt. support for a HTTP transaction, *but*
  - limit the detail and volume of coverage to fit allocated time.
- *Why*?!
  1. some of you may
     - develop network-related hardware or software, or
     - work in network operations,
  2. *most* of you will develop hardware or software that *depends* on a network, and
  3. many of the concepts and techniques *generalise*.

Notes:

References

[1] Wikipedia: ASCII.
http://en.wikipedia.org/ASCII.

[2] Wikipedia: Asynchronous serial communication.
http://en.wikipedia.org/wiki/Asynchronous_start-stop.

[3] Wikipedia: Baudot code.
http://en.wikipedia.org/wiki/Baudot_code.

[4] Wikipedia: Flow control.
http://en.wikipedia.org/wiki/Flow_control_(data).

[5] Wikipedia: Parity bit.
http://en.wikipedia.org/wiki/Parity_bit.

[6] Wikipedia: Serial port.
http://en.wikipedia.org/wiki/Serial_port.

[7] Wikipedia: Teleprinter.
http://en.wikipedia.org/wiki/Teletypewriter.

[8] Interface between data terminal equipment and data circuit terminating equipment employing serial binary data interchange.
Telecommunications Industry Association (TIA) TIA-232-F, 2002.
http://www.tiaonline.org/standards/.

Notes:

[9] H.L. Krum.
Electric selective system.
U.S. Patent 1,286,351.
http://www.google.com/patents/US1286351.

Notes: