Designing Application Protocols for TCP/IP

Stephen Cleary

Who is Stephen Cleary?

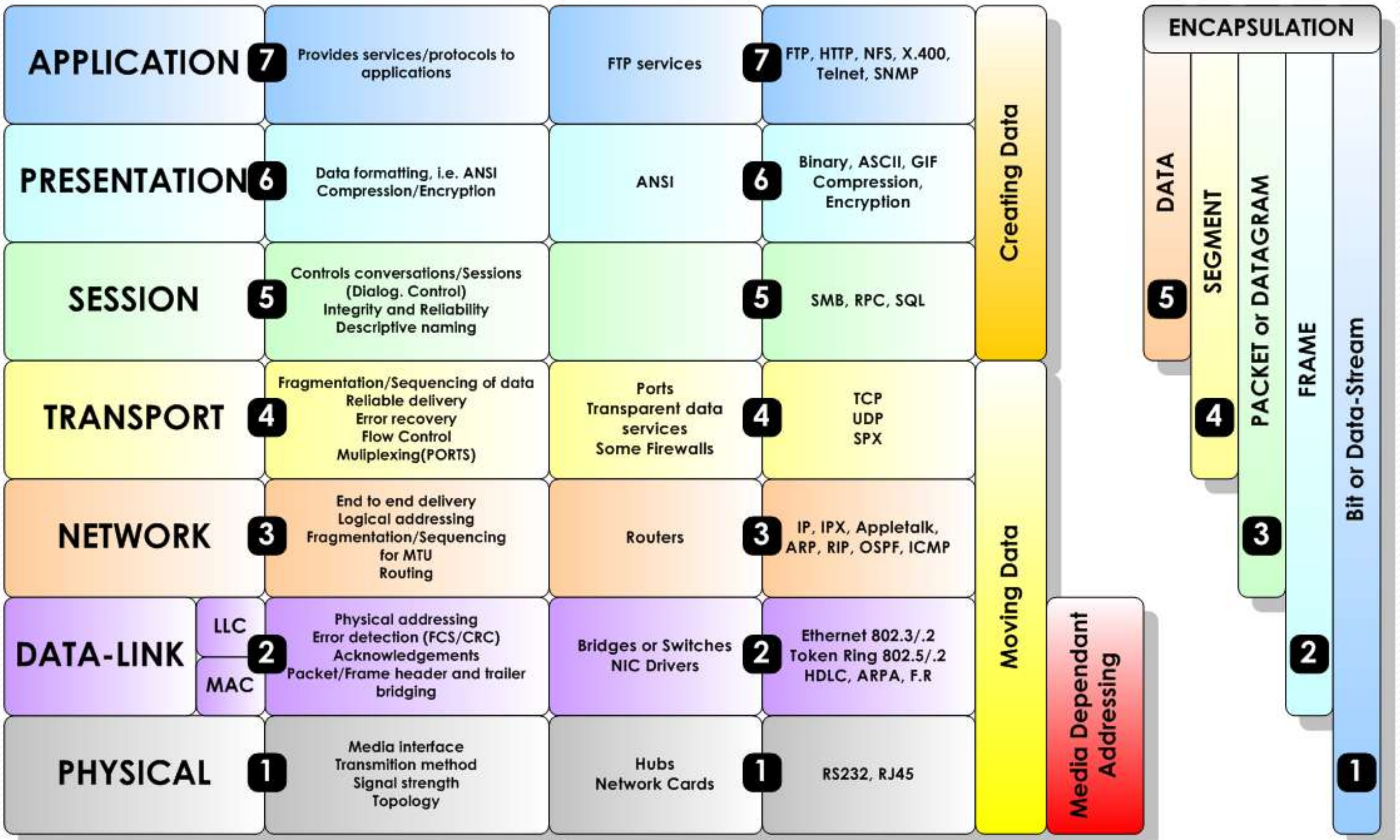
- Most TCP/IP experience came from 7 years of work as a systems integrator for Jervis B. Webb Company.
- "The TCP/IP guy" – designed current AGV protocol.
- Clients: GM (auto assembly lines), Syracuse News (paper delivery), Estee Lauder, RR Donnelley (bottling), Ricoh (toner), BlueScope Steel. (all 24x7)
- Components: AGVs, printing presses, hot backup systems, smart clients; many "bridge" devices to translate TCP/IP to/from serial or non-IP networks.
- Wrote C# socket class wrappers (part of Nito.Async).

http://nitoprograms.blogspot.com

What is an “Application Protocol”?

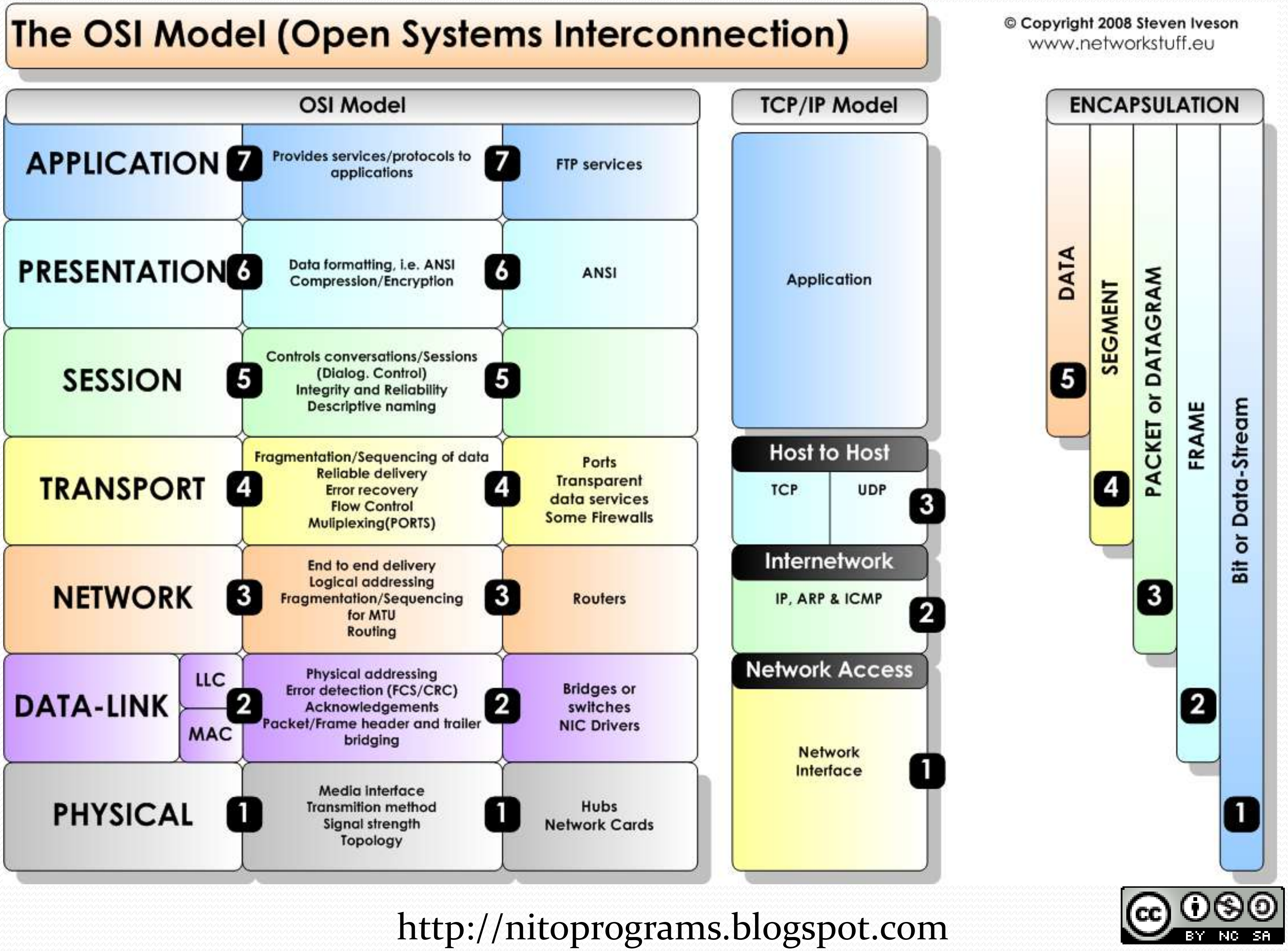
- An application protocol is all the communication *above* the TCP/IP layer.
- Examples: HTTP, FTP, POP, SOAP/HTTP.
- We will cover: Four important guidelines for TCP/IP application protocol design; XML over TCP/IP.
- We won’t cover: UDP, implementation specifics (e.g., optimal error recovery for the WinSock API).
- For C# implementation specifics, my blog has a TCP/IP .NET Sockets FAQ.

The OSI Model (Open Systems Interconnection)



Layer	No.	Description	Devices	No.	Protocols / Standards			ENCAPSULATION
APPLICATION	7	Provides services/protocols to applications	FTP services	7	FTP, HTTP, NFS, X.400, Telnet, SNMP	Creating Data		DATA
PRESENTATION	6	Data formatting, i.e. ANSI Compression/Encryption	ANSI	6	Binary, ASCII, GIF Compression, Encryption	Creating Data		DATA
SESSION	5	Controls conversations/Sessions (Dialog. Control) Integrity and Reliability Descriptive naming		5	SMB, RPC, SQL	Creating Data		DATA 5
TRANSPORT	4	Fragmentation/Sequencing of data Reliable delivery Error recovery Flow Control Muliplexing(PORTS)	Ports Transparent data services Some Firewalls	4	TCP UDP SPX	Moving Data		SEGMENT 4
NETWORK	3	End to end delivery Logical addressing Fragmentation/Sequencing for MTU Routing	Routers	3	IP, IPX, Appletalk, ARP, RIP, OSPF, ICMP	Moving Data		PACKET or DATAGRAM 3
DATA-LINK (LLC / MAC)	2	Physical addressing Error detection (FCS/CRC) Acknowledgements Packet/Frame header and trailer bridging	Bridges or Switches NIC Drivers	2	Ethernet 802.3/.2 Token Ring 802.5/.2 HDLC, ARPA, F.R	Moving Data	Media Dependant Addressing	FRAME 2
PHYSICAL	1	Media interface Transmition method Signal strength Topology	Hubs Network Cards	1	RS232, RJ45	Moving Data	Media Dependant Addressing	Bit or Data-Stream 1

http://nitoprograms.blogspot.com



The OSI Model (Open Systems Interconnection)

OSI Model

Layer		Function		Examples
APPLICATION	7	Provides services/protocols to applications	7	FTP services
PRESENTATION	6	Data formatting, i.e. ANSI Compression/Encryption	6	ANSI
SESSION	5	Controls conversations/Sessions (Dialog. Control) Integrity and Reliability Descriptive naming	5	
TRANSPORT	4	Fragmentation/Sequencing of data Reliable delivery Error recovery Flow Control Muliplexing(PORTS)	4	Ports Transparent data services Some Firewalls
NETWORK	3	End to end delivery Logical addressing Fragmentation/Sequencing for MTU Routing	3	Routers
DATA-LINK (LLC / MAC)	2	Physical addressing Error detection (FCS/CRC) Acknowledgements Packet/Frame header and trailer bridging	2	Bridges or switches NIC Drivers
PHYSICAL	1	Media interface Transmition method Signal strength Topology	1	Hubs Network Cards

TCP/IP Model

Layer	Protocols	
Application		
Host to Host	TCP, UDP	3
Internetwork	IP, ARP & ICMP	2
Network Access	Network Interface	1

ENCAPSULATION

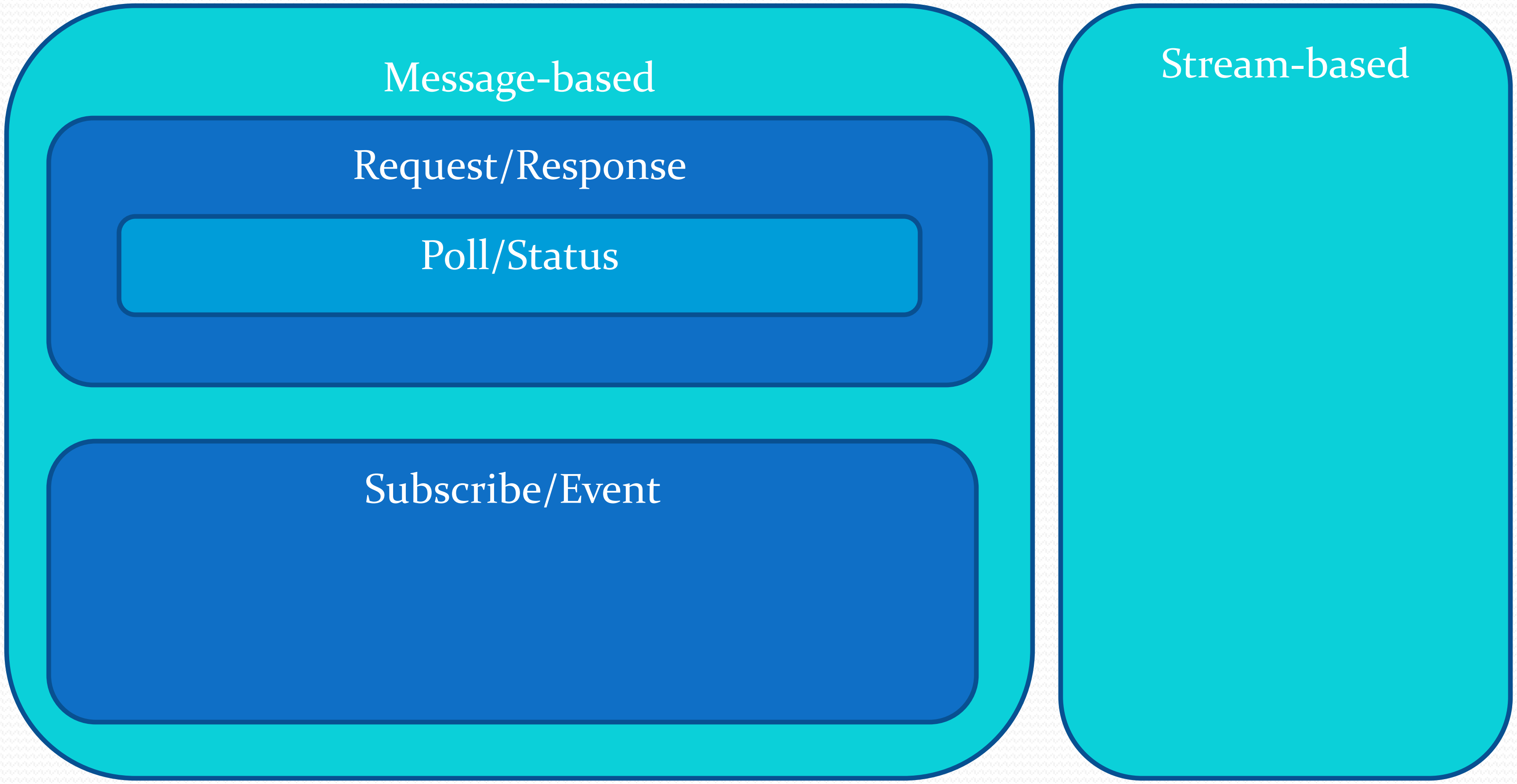
Unit	
DATA	5
SEGMENT	4
PACKET or DATAGRAM	3
FRAME	2
Bit or Data-Stream	1

http://nitoprograms.blogspot.com

What TCP/IP Provides

- Affects the application protocol design:
  - Concept of a “Connection” (client connects to server, after which the two sides are identical).
  - Reliability: acknowledgements, checksums, retransmission, discarding duplicate packets.
  - Sequencing: packets are sorted to match original sending order.
- Does not affect the application protocol design:
  - Flow and congestion control, adaptive timeouts, other stuff: Nagle algorithm, delayed ACKs, etc.

Types of App Protocols



1 – Write a Spec

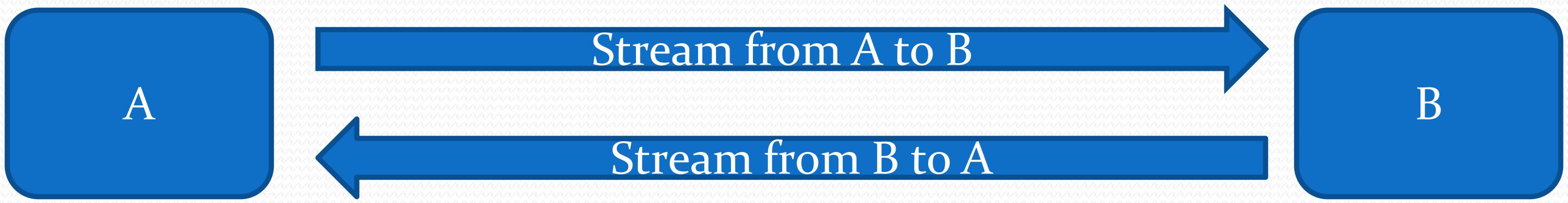
- Having a clearly-defined specification written down will reduce errors on both sides.
- Terminology:
  - "MUST" and "MAY" (RFC 2119).
  - Include a glossary referencing established standards.
- First Contact: decide who is client and who is server.
- Choosing the Port (configurable if possible):
  - IANA: 0-1023 is off limits; 1024-49151 is "off limits".
  - Preventing ephemeral conflicts: KB812873

2 – App Protocol Versioning

- Plan for the future now: enable future backwards compatibility. It's easier now than later!
- Application protocol documents should always have a version number defined in the document.
- The version of the app protocol used at runtime should be negotiated, not assumed.
- Don't over-engineer! A simple implementation is to send a list of supported versions, and let the other side choose. Negotiating feature sets is more complex.
- Decide in advance which version number changes are backwards-compatible and which are not.

3 – Message Framing

- TCP does *not* preserve message boundaries (!)
- From the app’s perspective, **TCP does not operate on *packets* of data; it operates on *streams* of data.**
- “Send” places bytes in the outgoing stream; “Receive” reads bytes from the incoming stream.
  - Send and Receive are not 1:1
- “Receive” was designed to allow partial reads.



3 – Message Framing

- Most protocols are based on messages (e.g., query / response), so we need message framing.
- Solution A: Length Prefixing
  - Specify length and endianness of length prefix.
  - (May be hidden as a "message ID" if message lengths are known, and may be at a fixed offset instead of a prefix.)
- Solution B: Delimiters
  - Escape sequences may be necessary.
  - Requires flexible buffer scheme to receive efficiently.
- Both solutions must consider DoS protection.

4 – Keepalives

- TCP does *not* provide detection of dropped connections. It is an idle protocol.
- TCP will detect a dropped connection if data is sent. The receiving side will not get a notification; this results in a *half-open connection*.
- Causes: router/computer crash, wireless lost.
- Wrong solutions: ping or a second connection.
- Correct solutions: a timer sending an empty message frame or actual keepalive message. Or TCP option.
- Keepalives must be done on both sides unless polling.

Miscellaneous Notes

- Put plenty of good examples in the protocol specification document to reduce ambiguity.
- Implementation:
  - There is nothing more important than logging.
    - Have a full tracing system that can be turned on at runtime on production machines. You'll need it.
    - Dump every byte received and sent, as well as its interpretation.
  - When in doubt, error out. Connections can always be re-established.

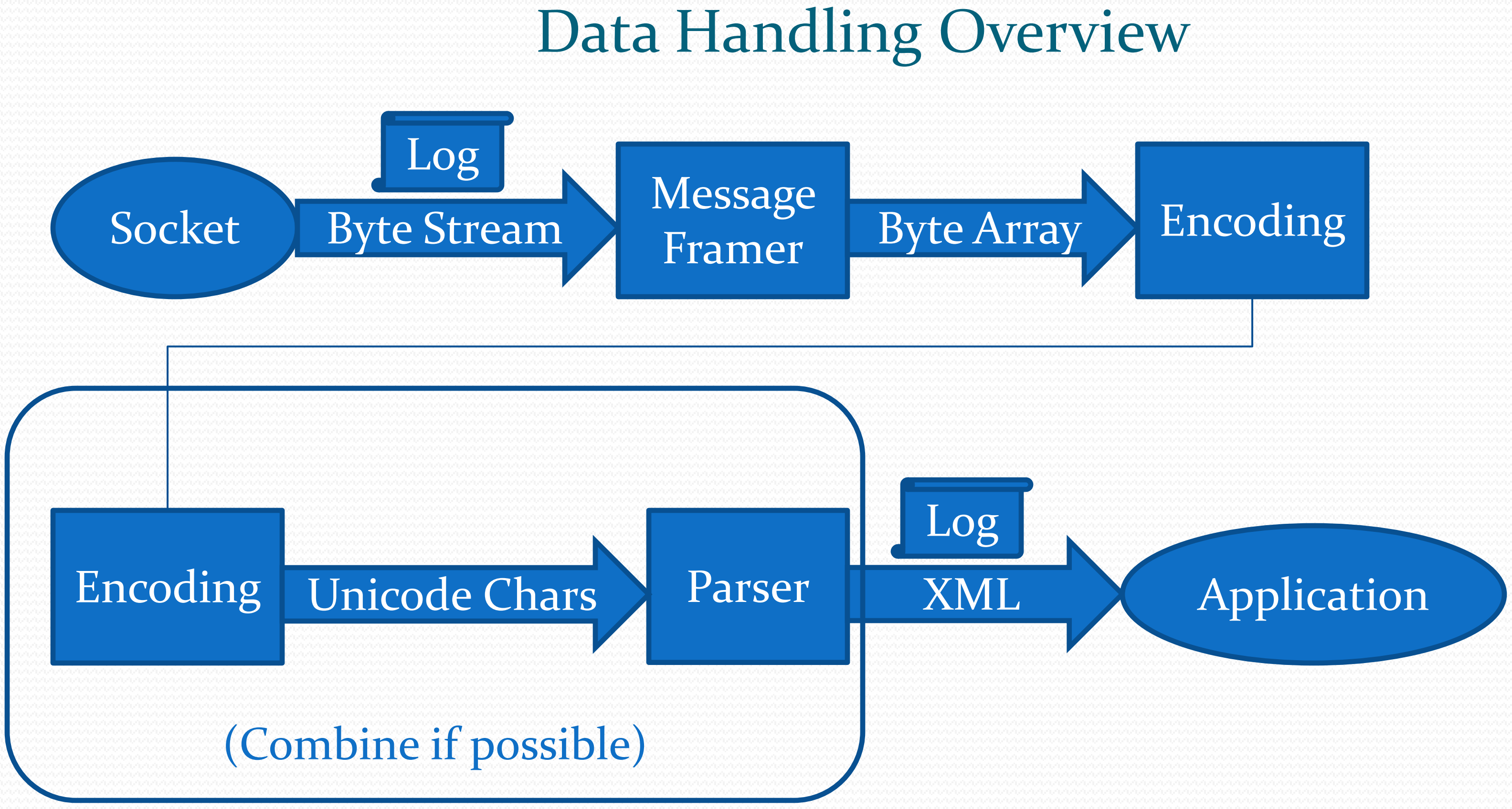
XML over TCP/IP

- Be familiar with the XML standard: www.w3.org.
- Most protocols don't use: entities (except for escaping), processing instructions, XSDs/DTDs, namespaces; some don't use text data.
- Message framing is technically not required but highly recommended (it greatly simplifies parsing).
- Keepalives are required, as any other protocol.
- Protocol versioning is highly recommended. XSDs/DTDs may interfere with compatibility.
  - A common rule: ignore unknown elements and attributes, instead of treating them as errors.

XML over TCP/IP – Encoding

- Each message becomes an XML document.
- XML documents are sequences of Unicode characters; TCP works on sequences (streams) of bytes. The Encoding is what translates one to the other.
- Three encoding decisions:
  - Encoding to use: specify or auto-detect.
  - Byte Order Mark: required by UTF-16; optional for UTF-8.
  - XML Prolog. May cause problems if encoding attribute is included (it's easy to get wrong).
- Do not perform an intermediate conversion to string.

XML over TCP/IP – Pipeline



http://nitoprograms.blogspot.com

XML over TCP/IP – Messages

- For each element, be sure to include:
  - When the message is meaningful (e.g., a "Response" should only be sent in response to a "Request").
  - Which attributes and elements are required and which are optional. This includes complex relations (e.g., a "Log" element must contain at least one "Message" element and exactly one "Source" element). Be sure to use terms with specific definitions ("at least one", "exactly one", etc).
- Document the format of any non-string data such as dates, booleans, and integers.
- Messages are data and commands, not behavior.

XML over TCP/IP – Naming

- Use PascalCasing for element/attribute names.
- Two-letter acronyms are in all caps: IO; compound words are treated as a single word: Lifetime.
- Avoid abbreviations, except Id and Ok.
- Avoid language keywords: Event.
- Prefer readability: MessageType, not TypeOfMessage.
- Avoid “magic values”.
- Attributes vs. child elements – be consistent.

The Most Important Point

- Some people are mostly good; others are mostly bad – but everyone has done *something* wrong.
- The Bible says there is a penalty for "wrong-doing": hell.
- Jesus Christ, the Son of God, lived a perfect life and then chose to die for you.
- His blood can pay the penalty for your wrongdoing.
- Repent of your sin and trust in Him to take you to Heaven instead of hell.
  - To the mostly good: *everyone* needs Jesus.
  - To the mostly bad: if you receive Him, He *will* receive you.
- Jesus offers you peace, and freedom from your sin.
  - He's the server socket; we have to connect to Him. Knowing the protocol isn't enough.

Thank you!

Stephen Cleary