tms training days

Binding data to client apps

Patrick Prémartin

patrick.premartin@olfsoft.com

Discover our social media pages

Introduction

- In this session, we'll be talking about client/server over local networks or the Internet.
- Many solutions exist for doing this in general, and more specifically in Delphi.
- For the rest, we'll assume that our client applications query one or more servers to obtain or modify the data they process.
- Let's start with a few reminders...

Local and global network

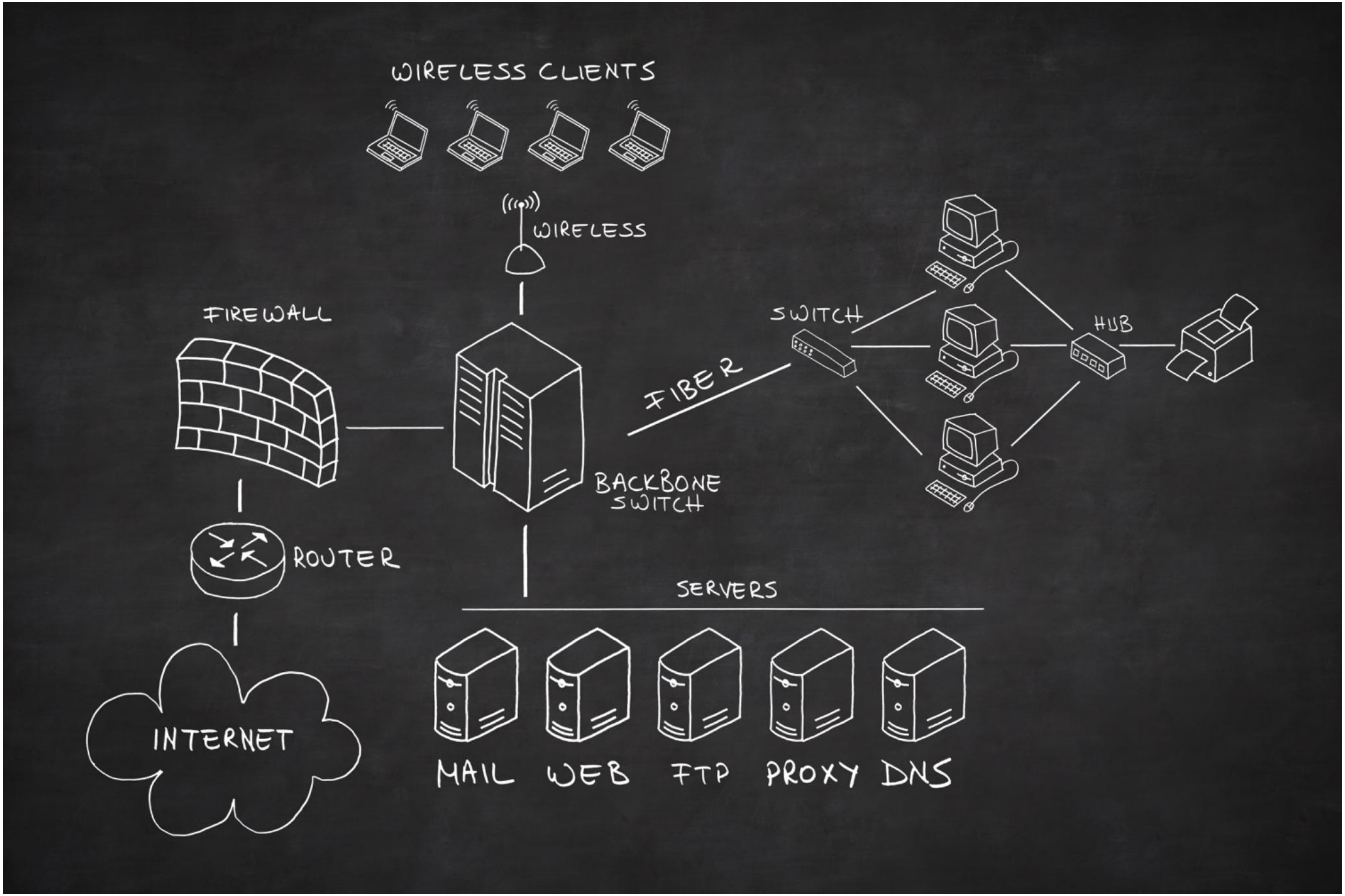
- When we talk about a network, we're talking about a set of devices connected to each other by cable or waves (Wi-Fi, Li-Fi, 3G, 4G, 5G, etc.) in a local area network (LAN) or wide area network (WAN).

- Our current networks use Ethernet. Each piece of equipment has one or more globally unique MAC addresses.

- Our computers use the IP protocol with IPv4 or IPv6 addresses to communicate with each other.

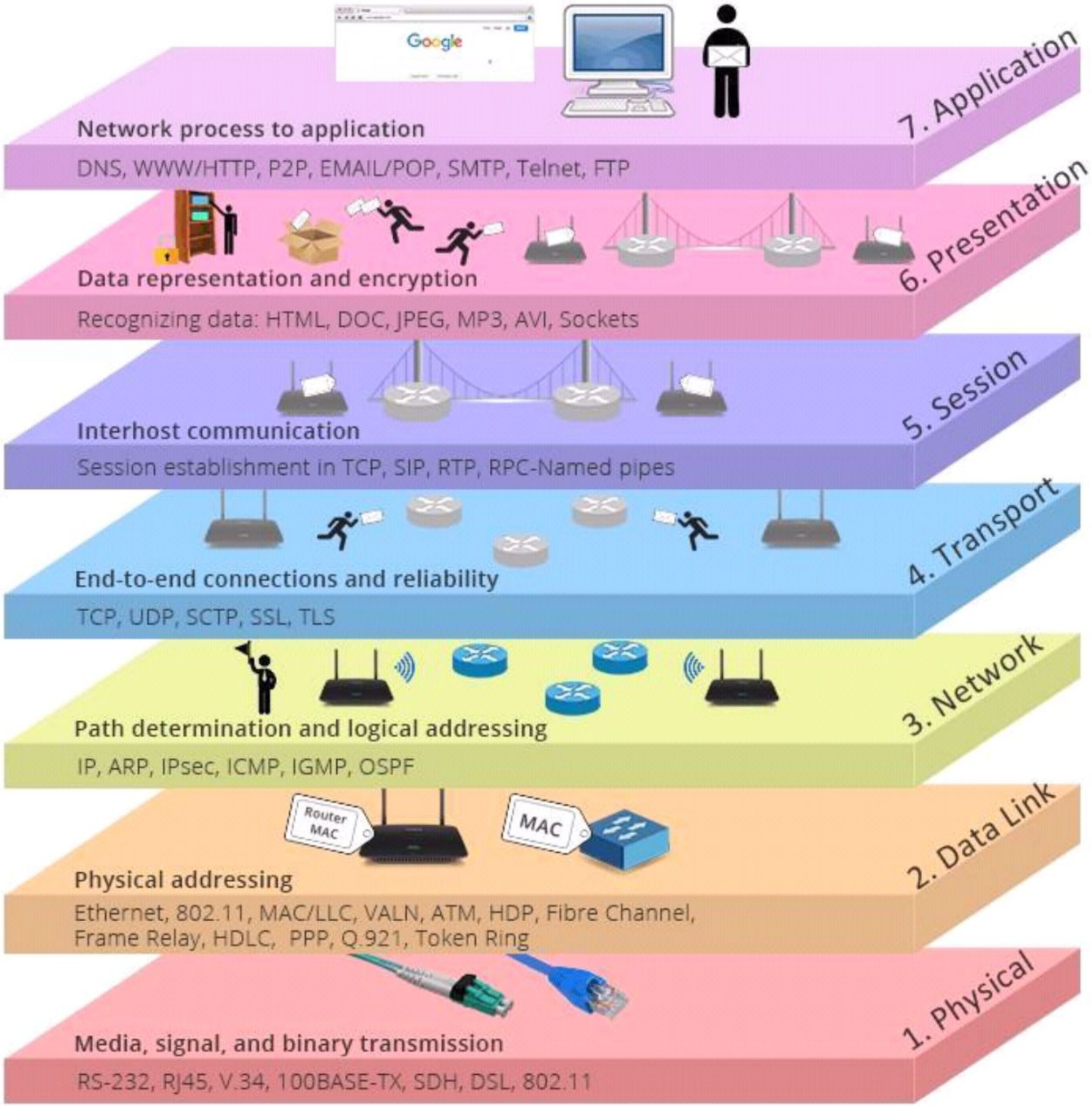
Local and global network

- The Internet is nothing more than a global network of local networks open to the rest of the world.
- Each connected device has one or more private IP addresses visible on its local network.
- Each network is identified by one or more public IP addresses, which are redirected to local IP addresses.
- All this is connected by routers and other specialized network equipment.


WIRELESS CLIENTS
WIRELESS
FIREWALL
SWITCH
HUB
FIBER
BACKBONE SWITCH
ROUTER
SERVERS
INTERNET
MAIL
WEB
FTP
PROXY
DNS

The OSI model

- The OSI model is the communications standard on which all our networks have been based since the 1970s.
- It describes 7 layers: 3 for hardware, 4 for software.
- The higher the level, the simpler the operation is supposed to be.



7. Application

Network process to application

DNS, WWW/HTTP, P2P, EMAIL/POP, SMTP, Telnet, FTP

6. Presentation

Data representation and encryption

Recognizing data: HTML, DOC, JPEG, MP3, AVI, Sockets

5. Session

Interhost communication

Session establishment in TCP, SIP, RTP, RPC-Named pipes

4. Transport

End-to-end connections and reliability

TCP, UDP, SCTP, SSL, TLS

3. Network

Path determination and logical addressing

IP, ARP, IPsec, ICMP, IGMP, OSPF

2. Data Link

Physical addressing

Ethernet, 802.11, MAC/LLC, VALN, ATM, HDP, Fibre Channel,
Frame Relay, HDLC, PPP, Q.921, Token Ring

1. Physical

Media, signal, and binary transmission

RS-232, RJ45, V.34, 100BASE-TX, SDH, DSL, 802.11

OSI MODEL

Layer 7: Application Layer

- Defines interface to user processes
- Provides standardized network services

Layer 6: Presentation Layer

- Specifies architecture-independent data transfer format
- Encodes and decodes data;
  Encrypts and decrypts data;
  Compresses and decompresses data

Layer 5: Session Layer

- Manages user sessions and dialogues
- Controls establishment and termination of logical links between users

Layer 4: Transport Layer

- Provides reliable and sequential end-to-end packet delivery
- Provides connectionless oriented packet delivery

Layer 3: Network Layer

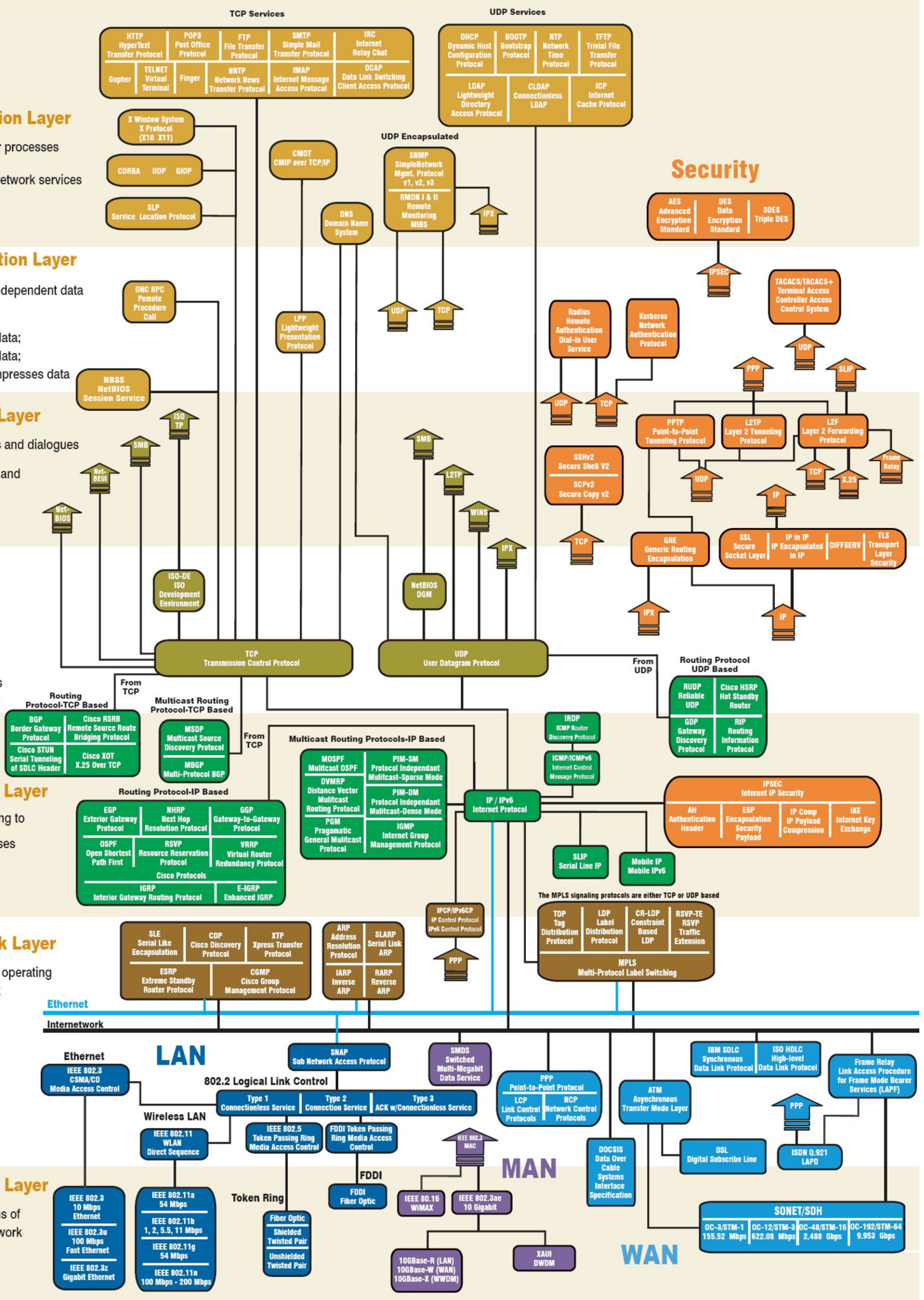
- Routes packets according to unigue network addresses

Layer 2: Data Link Layer

- Defines procedures for operating the communication link
- Provides framing and sequencing

Layer 1: Physical Layer

- Defines physical means of sending data over network devices



TCP Services

HTTP HyperText Transfer Protocol | POP3 Post Office Protocol | FTP File Transfer Protocol | SMTP Simple Mail Transfer Protocol | IRC Internet Relay Chat

Gopher | TELNET Virtual Terminal | Finger | NNTP Network News Transfer Protocol | IMAP Internet Message Access Protocol | DCAP Data Link Switching Client Access Protocol

UDP Services

DHCP Dynamic Host Configuration Protocol | BOOTP Bootstrap Protocol | NTP Network Time Protocol | TFTP Trivial File Transfer Protocol

LDAP Lightweight Directory Access Protocol | CLDAP Connectionless LDAP | ICP Internet Cache Protocol

X Window System X Protocol (X10 X11)

CORBA IIOP GIOP

SLP Service Location Protocol

CMOT CMIP over TCP/IP

DNS Domain Name System

UDP Encapsulated

SNMP SimpleNetwork Mgmt. Protocol v1, v2, v3

RMON I & II Remote Monitoring MIBS

IPX

UDP

TCP

ONC RPC Pemote Procedure Call

LPP Lightweight Presentation Protocol

NBSS NetBIOS Session Service

ISO TP

SMB

Net-BEUI

Net-BIOS

SMB

L2TP

WINS

IPX

ISO-DE ISO Development Environment

NetBIOS DGM

TCP Transmission Control Protocol

UDP User Datagram Protocol

Security

AES Advanced Encryption Standard | DES Data Encryption Standard | 3DES Triple DES

IPSEC

TACACS/TACACS+ Terminal Access Controller Access Control System

Radius Remote Authentication Dial-In User Service

Kerberos Network Authentication Protocol

UDP

PPP

SLIP

UDP

TCP

PPTP Point-to-Point Tunneling Protocol

L2TP Layer 2 Tunneling Protocol

L2F Layer 2 Forwarding Protocol

Frame Relay

UDP

TCP

X.25

SSHv2 Secure Shell V2

SCPv2 Secure Copy v2

IP

TCP

GRE Generic Routing Encapsulation

SSL Secure Socket Layer | IP in IP IP Encapsulated in IP | DIFFSERV | TLS Transport Layer Security

IPX

IP

From UDP

Routing Protocol UDP Based

RUDP Reliable UDP | Cisco HSRP Hot Standby Router

GDP Gateway Discovery Protocol | RIP Routing Information Protocol

From TCP

Routing Protocol-TCP Based

BGP Border Gateway Protocol | Cisco RSRB Remote Source Route Bridging Protocol

Cisco STUN Serial Tunneling of SDLC Header | Cisco XOT X.25 Over TCP

Multicast Routing Protocol-TCP Based

MSDP Multicast Source Discovery Protocol

MBGP Multi-Protocol BGP

From TCP

Multicast Routing Protocols-IP Based

MOSPF Mulitcast OSPF | PIM-SM Protocol Independant Mulitcast-Sparse Mode

DVMRP Distance Vector Mulitcast Routing Protocol | PIM-DM Protocol Independant Mulitcast-Dense Mode

PGM Pragamatic General Mulitcast Protocol | IGMP Internet Group Management Protocol

IRDP ICMP Router Discovery Protocol

ICMP/ICMPv6 Internet Control Message Protocol

IP / IPv6 Internet Protocol

Routing Protocol-IP Based

EGP Exterior Gateway Protocol | NHRP Next Hop Resolution Protocol | GGP Gateway-to-Gateway Protocol

OSPF Open Shortest Path First | RSVP Resource Reservation Protocol | VRRP Virtual Router Redundancy Protocol

Cisco Protocols

IGRP Interior Gateway Routing Protocol | E-IGRP Enhanced IGRP

IPSEC Internet IP Security

AH Authentication Header | ESP Encapsulation Security Payload | IP Comp IP Payload Compression | IKE Internet Key Exchange

SLIP Serial Line IP

Mobile IP Mobile IPv6

The MPLS signaling protocols are either TCP or UDP based

TDP Tag Distribution Protocol | LDP Label Distribution Protocol | CR-LDP Constraint Based LDP | RSVP-TE RSVP Traffic Extension

MPLS Multi-Protocol Label Switching

IPCP/IPv6CP IP Control Protocol IPv6 Control Protocol

PPP

SLE Serial Like Encapsulation | CDP Cisco Discovery Protocol | XTP Xpress Transfer Protocol

ESRP Extreme Standby Router Protocol | CGMP Cisco Group Management Protocol

ARP Address Resolution Protocol | SLARP Serial Link ARP

IARP Inverse ARP | RARP Reverse ARP

Ethernet

Internetwork

LAN

Ethernet

IEEE 802.3 CSMA/CD Media Access Control

SNAP Sub Network Access Protocol

802.2 Logical Link Control

Type 1 Connectionless Service | Type 2 Connection Service | Type 3 ACK w/Connectionless Service

Wireless LAN

IEEE 802.11 WLAN Direct Sequence

IEEE 802.5 Token Passing Ring Media Access Control

FDDI Token Passing Ring Media Access Control

FDDI

FDDI Fiber Optic

Token Ring

Fiber Optic | Shielded Twisted Pair | Unshielded Twisted Pair

IEEE 802.3 10 Mbps Ethernet | IEEE 802.3u 100 Mbps Fast Ethernet | IEEE 802.3z Gigabit Ethernet

IEEE 802.11a 54 Mbps | IEEE 802.11b 1, 2, 5.5, 11 Mbps | IEEE 802.11g 54 Mbps | IEEE 802.11n 100 Mbps - 200 Mbps

SMDS Switched Multi-Megabit Data Service

IEEE 802.3 MAC

MAN

IEEE 80.16 WiMAX

IEEE 802.3ae 10 Gigabit

10GBase-R (LAN) 10GBase-W (WAN) 10GBase-X (WWDM)

XAUI DWDM

PPP Point-to-Point Protocol

LCP Link Control Protocols | NCP Network Control Protocols

DOCSIS Data Over Cable Systems Interface Specification

ATM Asynchronous Transfer Mode Layer

IBM SDLC Synchronous Data Link Protocol | ISO HDLC High-level Data Link Protocol

Frame Relay Link Access Procedure for Frame Mode Bearer Services (LAPF)

PPP

DSL Digital Subscribe Line

ISDN Q.921 LAPD

SONET/SDH

OC-3/STM-1 155.52 Mbps | OC-12/STM-3 622.08 Mbps | OC-48/STM-16 2.488 Gbps | OC-192/STM-64 9.953 Gbps

WAN

One port for each service

- IP addresses enable us to target a computer.
- Each computer offers local or network services.
- When accessible over an IP network, these services are managed by servers listening on one or more ports.

One port for each service

- So, when we're doing client/server, we need an IP to find a device or computer on the network and a port to identify the local server (service) we want to talk to.
- Some ports are reserved for common services (web, FTP, email, etc.), while others are available.
- Here's a list : https://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers

COMMON PORTS

TCP/UDP Port Numbers

7	Echo	**554**	RTSP	**2745**	Bagle.H	**6891-6901**	Windows Live
19	Chargen	**546-547**	DHCPv6	**2967**	Symantec AV	**6970**	Quicktime
20-21	FTP	**560**	rmonitor	**3050**	Interbase DB	**7212**	GhostSurf
22	SSH/SCP	**563**	NNTP over SSL	**3074**	XBOX Live	**7648-7649**	CU-SeeMe
23	Telnet	**587**	SMTP	**3124**	HTTP Proxy	**8000**	Internet Radio
25	SMTP	**591**	FileMaker	**3127**	MyDoom	**8080**	HTTP Proxy
42	WINS Replication	**593**	Microsoft DCOM	**3128**	HTTP Proxy	**8086-8087**	Kaspersky AV
43	WHOIS	**631**	Internet Printing	**3222**	GLBP	**8118**	Privoxy
49	TACACS	**636**	LDAP over SSL	**3260**	iSCSI Target	**8200**	VMware Server
53	DNS	**639**	MSDP (PIM)	**3306**	MySQL	**8500**	Adobe ColdFusion
67-68	DHCP/BOOTP	**646**	LDP (MPLS)	**3389**	Terminal Server	**8767**	TeamSpeak
69	TFTP	**691**	MS Exchange	**3689**	iTunes	**8866**	Bagle.B
70	Gopher	**860**	iSCSI	**3690**	Subversion	**9100**	HP JetDirect
79	Finger	**873**	rsync	**3724**	World of Warcraft	**9101-9103**	Bacula
80	HTTP	**902**	VMware Server	**3784-3785**	Ventrilo	**9119**	MXit
88	Kerberos	**989-990**	FTP over SSL	**4333**	mSQL	**9800**	WebDAV
102	MS Exchange	**993**	IMAP4 over SSL	**4444**	Blaster	**9898**	Dabber
110	POP3	**995**	POP3 over SSL	**4664**	Google Desktop	**9988**	Rbot/Spybot

One port for each service

- We're free to run servers on any port we like, but using “reserved” ports for anything else can have side effects.

Internet is not a safe place !

- Any device connected to the Internet with a public IP is under constant attack from automatons looking for security holes.

- Protect your computers and networks: open only the accesses (IP+ports) you need on your firewalls.

- Systematically test what your servers receive.

- Never assume that what you receive comes from your software!

Internet is not a safe place !

- Never send passwords or API keys in public accesses without previous authentication.

- Be careful of what you have on your web pages and scripts. All is public, all is visible, all is recorder by many unknown bots and can be used against your systems one day.

- If you accidentally share a password or API key, change it! Don’t suppose nobody have seen it.

IP Sockets : low level communication

- Sockets are the basic element of software communication on an IP network.
- A socket is bidirectional.
- Bytes are exchanged in the same way as serial ports.

IP Sockets : low level communication

- The server opens a listening socket on an IP and a port.
- Clients connect to this IP/port, which automatically triggers the opening of a read/write socket on another port reserved to this client.

IP Sockets : low level communication

- Each client has a communication channel with the server.
- Clients can't talk to each other without going through the server.

Demo

Sockets problems

- To use network sockets, we need to be able to read and write our data in byte buffers.

- Sockets are not available in all development languages or on all types of software (e.g. JavaScript on a web browser).

- It's not always easy to get software written in different languages to communicate, depending on the type of data being exchanged.

Client / Server with the Socket Messaging Library

- However, the use of network sockets remains practical when you have the right tools. It's low-level. You can do what you like with it.
- When I was creating the real-time multiplayer video game Sporgloo, I created an open source library for exchanging messages between clients and servers.
- The software defines the messages and generates the code to use them, saving time.

Demo

Client / Server problems in a connected world

- This solution is useful when you want to discuss programs in Delphi, but it's not open to other technologies.

- More accessible solutions are available.

Using web technologies to exchange data

- Rather than reinventing the wheel and defining proprietary protocols by working on sockets, why not start with things that work everywhere?

- Using a web server to complement or replace our data servers is a good compromise.

A web server to serve APIs

- A web server supports the http and https protocols.

- It receives a request from the client.

- It sends a response with a return code indicating whether the response was successful or whether there was an error. These return codes are standardized: https://en.wikipedia.org/wiki/List_of_HTTP_status_codes

A web server to serve APIs

- In addition to the status code, the server transmits the type of information sent in the form of MIME types: https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/MIME_types/Common_types

- There's nothing to stop us using this principle for anything other than HTML, CSS, JavaScript and images!

A web server to serve APIs

- All development languages support http or https requests.

- If needed it's up to us to provide our data in recognized formats such as JSON, XML or text, which can be used in most development languages.

- And depending on our knowledge and needs, we can program the server in PHP, JavaScript, Java, Python and, of course, Delphi!

Using REST API to simplify CRUD operations

- REST is a standard based on the http/s protocol.
- It uses classic web servers to access and modify data.
- REST was originally designed for CRUD, but in reality it can be used for anything.
- Most development languages support a REST API. Delphi and JavaScript are among them.

How to push data to web clients ?

- The main drawback of web APIs (REST or not) is that you're always in :

  client => server => client.

- It is not possible for a server to send data to a client without the client having requested it.

- That's where WebSockets come in!

The WebSocket protocole

- WebSockets are not sockets.
- WebSocket is a bidirectional client/server dialog protocol.
- It relies on http/s to function.
- It's a cheat: the client regularly queries the server to see if it has anything to say.
- The advantage is that it does it all by itself.

The WebSocket protocole

- The WebSocket protocol is an API implemented in recent web browsers. JavaScript can therefore use it without having to use AJAX.

- The WebSocket protocol is simply an http/s-based information exchange protocol. Any development language with access to http/s can implement it.

- TMS Software offers TMS FNC WebSocket for VCL, FireMonkey, Lazarus and Web Core projects.

Demo

Conclusion

- Take advantage of your projects' transition to the web to modernize your data servers.

- Preferably use a REST model for standard cases and JSON for your messages.

- Use WebSocket when you need real-time (e.g. chat) or feedback (e.g. alert or log systems).

Q&A

tms training days

THANK YOU!

Discover our social media pages