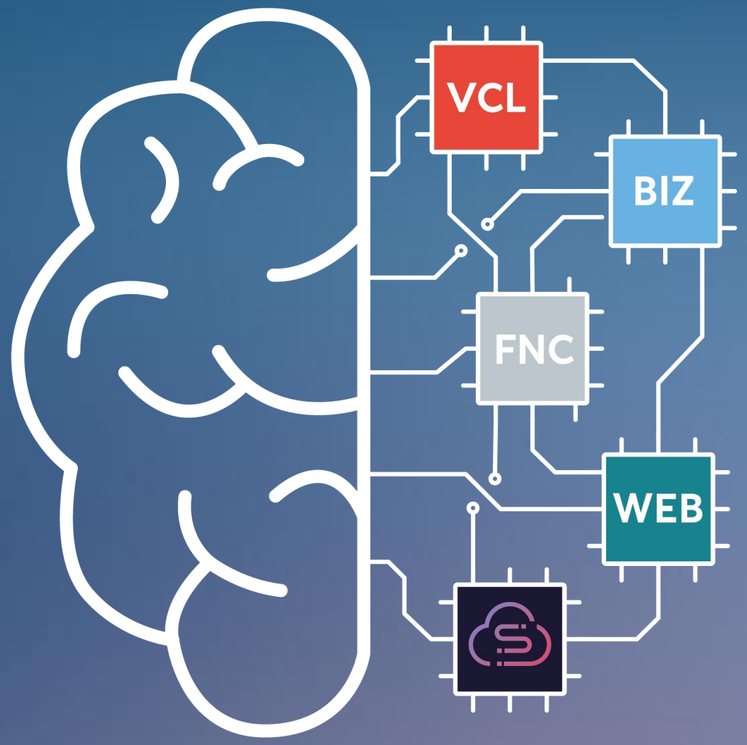


tms training days

Binding data to client apps

Patrick Prémartin

patrick.premartin@olfsoft.com

Discover our social media pages

Introduction

- Dans cette session nous allons parler de client / serveur sur des réseaux locaux ou par Internet.
- De nombreuses solutions existent pour faire ça d’une façon générale et plus particulièrement en Delphi.
- Pour la suite nous partirons du principe que nos applications clientes interrogent un ou plusieurs serveurs pour obtenir ou modifier les données qu’elles traitent.
- Commençons par quelques rappels…

Local and global network

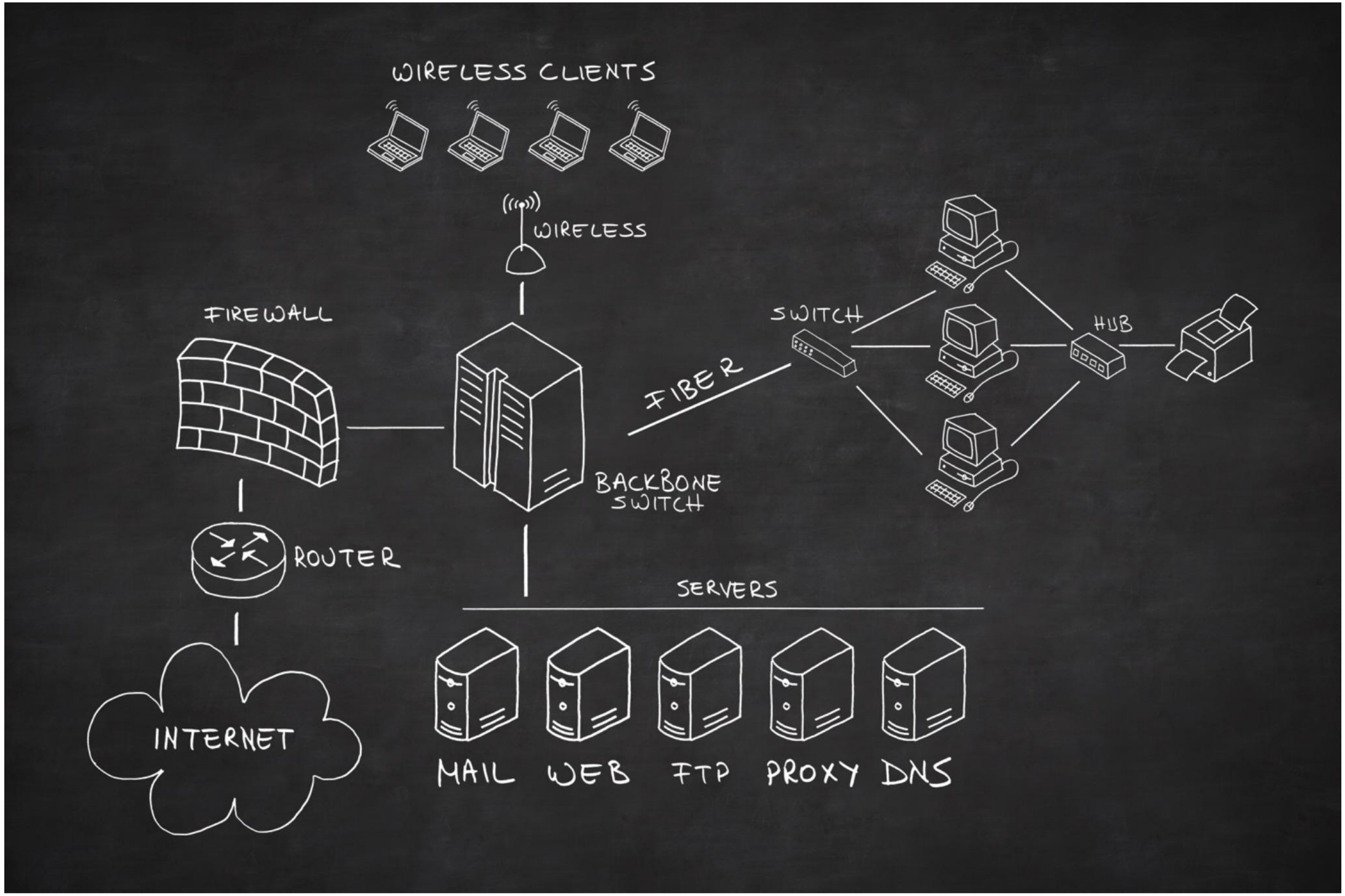
- Quand on parle de réseau on va parler d'un ensemble d'appareils connectés entre eux par câble ou ondes (Wi-Fi, Li-Fi, 3G, 4G, 5G, …) en réseau local (LAN) ou en réseau étendu (WAN).

- Nos réseaux actuels fonctionnent en Ethernet. Chaque équipement a une ou plusieurs adresses MAC.

- Nos ordinateurs utilisent le protocole IP avec des adresses IPv4 ou IPv6 pour communiquer entre eux.

Local and global network

- Internet n’est qu’un réseau global regroupant des réseaux locaux ouverts sur le reste du monde.
- Chaque appareil connecté a une ou plusieurs adresses IP privées visibles sur son réseau local.
- Chaque réseau est identifié par une ou plusieurs adresses IP publiques qui sont redirigées vers des IP locales.
- Le tout est connecté par des routeurs et d’autres équipements réseau spécialisés.

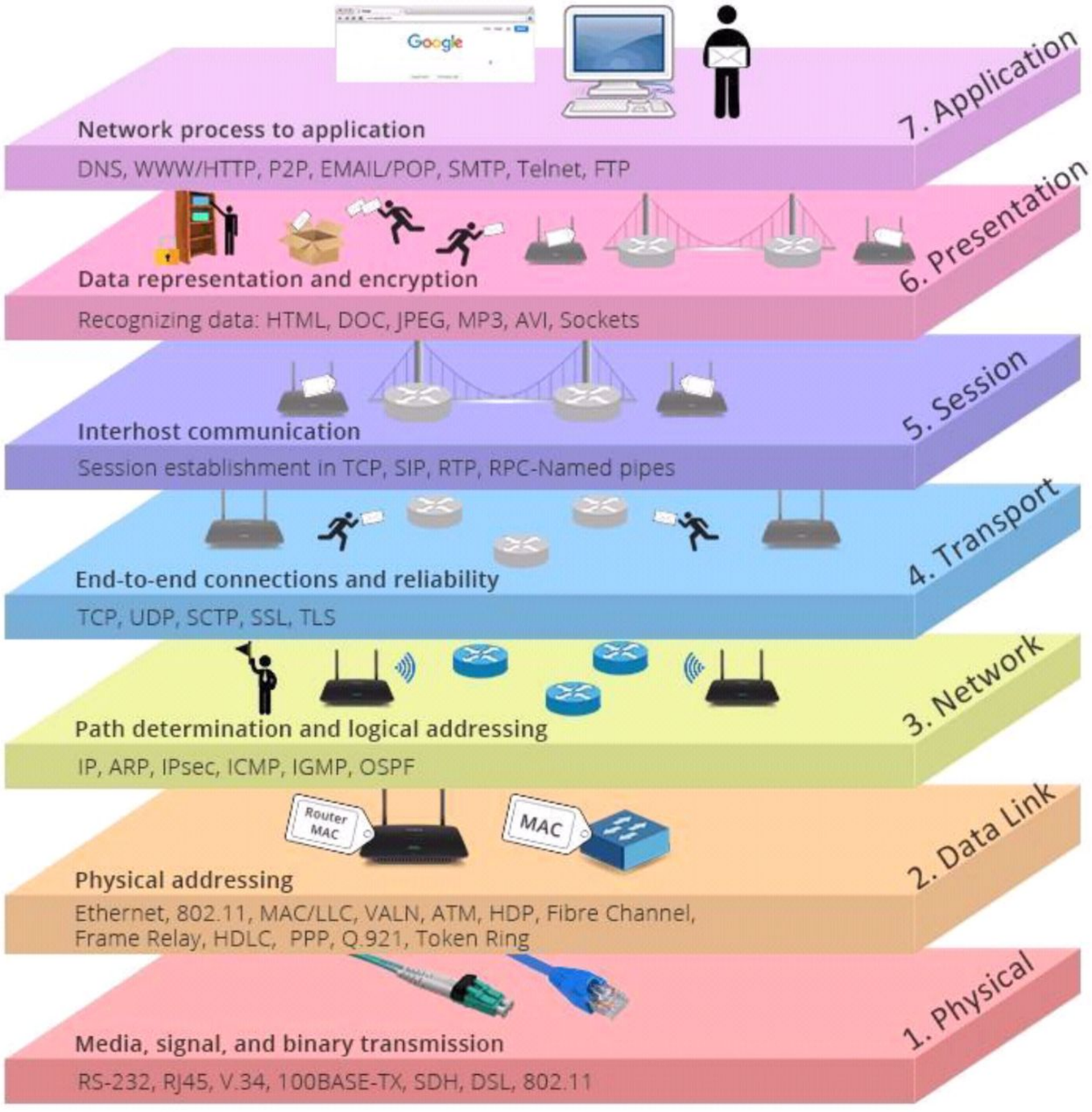

WIRELESS CLIENTS
WIRELESS
FIREWALL
SWITCH
HUB
FIBER
BACKBONE SWITCH
ROUTER
SERVERS
INTERNET
MAIL
WEB
FTP
PROXY
DNS

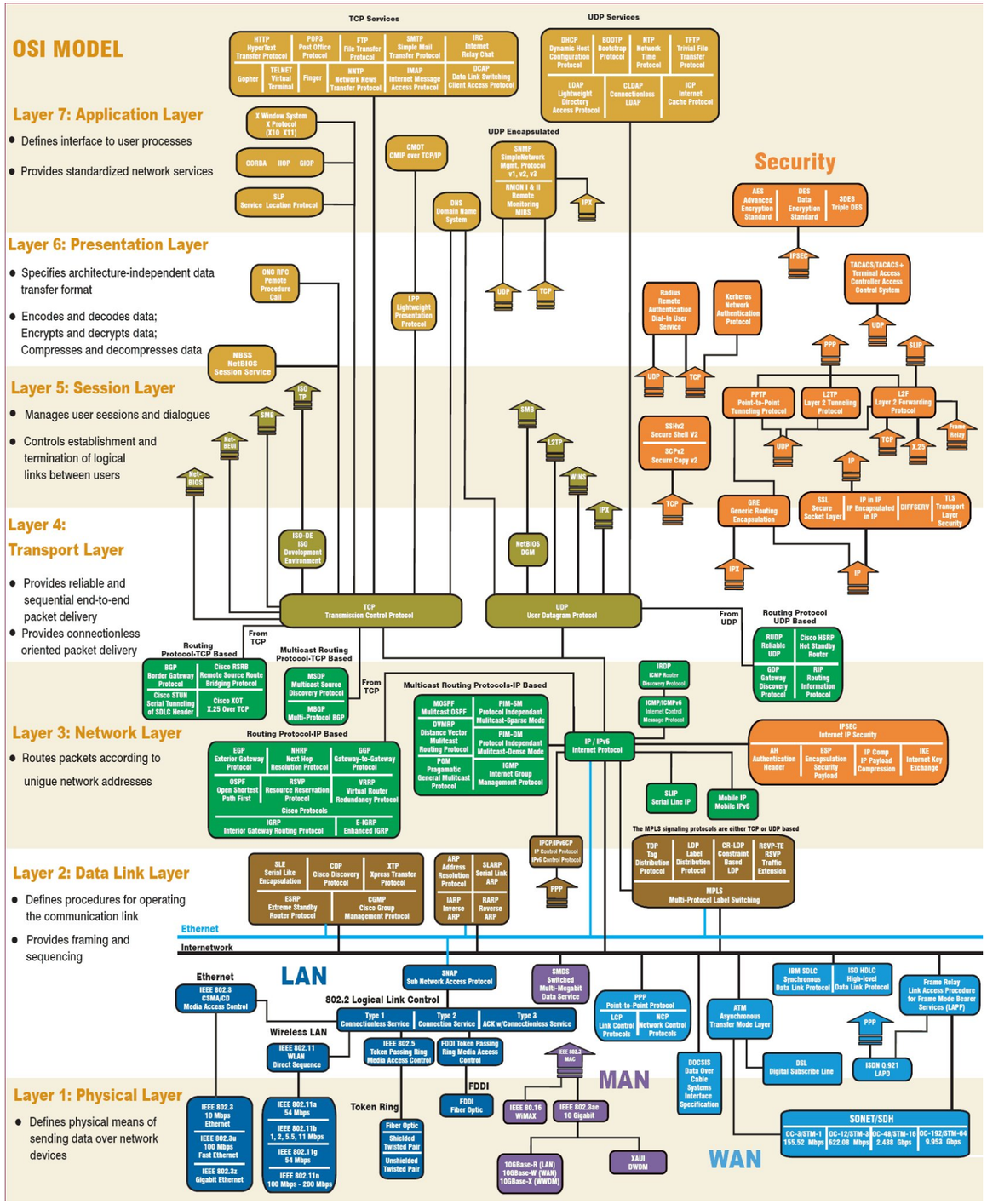
The OSI model

- Le modèle ISO est la norme de communication sur laquelle sont basés tous nos réseaux depuis les années 1970.

- Elle décrit 7 couches : 3 pour la partie matérielle, 4 pour la partie logicielle.

- Plus on monte dans les niveaux, plus le fonctionnement est supposé être simple.


Google
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
Router MAC
MAC
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
TCP Services
UDP Services
Layer 7: Application Layer
Defines interface to user processes
Provides standardized network 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
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
Security
AES Advanced Encryption Standard
DES Data Encryption Standard
3DES Triple DES
IPSEC
TACACS/TACACS+ Terminal Access Controller Access Control System
Layer 6: Presentation Layer
Specifies architecture-independent data transfer format
Encodes and decodes data; Encrypts and decrypts data; Compresses and decompresses data
ONC RPC Remote Procedure Call
LPP Lightweight Presentation Protocol
UDP
TCP
Radius Remote Authentication Dial-In User Service
Kerberos Network Authentication Protocol
PPP
SLIP
NBSS NetBIOS Session Service
PPTP Point-to-Point Tunneling Protocol
L2TP Layer 2 Tunneling Protocol
L2F Layer 2 Forwarding Protocol
Layer 5: Session Layer
Manages user sessions and dialogues
Controls establishment and termination of logical links between users
ISO TP
SMB
Net BIOS
NFS
SSHv2 Secure Shell V2
SCPv2 Secure Copy v2
X.25
Frame Relay
GRE Generic Routing Encapsulation
SSL Secure Socket Layer
IP in IP IP Encapsulated in IP
DIFFSERV
TLS Transport Layer Security
Layer 4: Transport Layer
Provides reliable and sequential end-to-end packet delivery
Provides connectionless oriented packet delivery
ISO-DE ISO Development Environment
NetBIOS DGM
WINS
TCP Transmission Control Protocol
UDP User Datagram Protocol
From TCP
From UDP
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
Routing Protocol UDP Based
RUDP Reliable UDP
Cisco HSRP Hot Standby Router
GDP Gateway Discovery Protocol
RIP Routing Information Protocol
Layer 3: Network Layer
Routes packets according to unigue network addresses
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
Multicast Routing Protocols-IP Based
MOSPF Multicast OSPF
DVMRP Distance Vector Multicast Routing Protocol
PGM Pragmatic General Multicast Protocol
PIM-SM Protocol Independent Multicast-Sparse Mode
PIM-DM Protocol Independent Multicast-Dense Mode
IGMP Internet Group Management Protocol
IP / IPv6 Internet Protocol
IRDP ICMP Router Discovery Protocol
ICMP/ICMPv6 Internet Control Message Protocol
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
IPCP/IPv6CP IP Control Protocol/IPv6 Control Protocol
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
Layer 2: Data Link Layer
Defines procedures for operating the communication link
Provides framing and sequencing
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
SMDS Switched Multi-Megabit Data Service
PPP Point-to-Point Protocol
LCP Link Control Protocols
NCP Network Control Protocols
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)
MAN
FDDI
Token Ring
FDDI Fiber Optic
IEEE 802.16 WiMAX
IEEE 802.3ae 10 Gigabit
DOCSIS Data Over Cable Systems Interface Specification
DSL Digital Subscribe Line
ISDN Q.921 LAPD
Layer 1: Physical Layer
Defines physical means of sending data over network devices
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
Fiber Optic
Shielded Twisted Pair
Unshielded Twisted Pair
10GBase-R (LAN) 10GBase-W (WAN) 10GBase-X (WWDM)
XAUI DWDM
WAN
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

One port for each service

- Grâce aux adresses IP nous pouvons cibler un ordinateur.
- Chaque ordinateur propose des services locaux ou réseau.
- Lorsqu’ils sont accessibles sur un réseau IP ces services sont gérés par des serveurs en écoute sur un ou plusieurs ports.

One port for each service

- Quand nous faisons du client/serveur nous avons donc besoin d'une IP pour trouver une machine sur le réseau et d'un port qui identifie son serveur local avec lequel nous désirons discuter.

- Certains ports sont réservés à des services courants (web, FTP, email, …), d'autres sont disponibles.

- En voici une liste : https://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers

COMMON PORTS

TCP/UDP Port Numbers

7 Echo
19 Chargen
20-21 FTP
22 SSH/SCP
23 Telnet
25 SMTP
42 WINS Replication
43 WHOIS
49 TACACS
53 DNS
67-68 DHCP/BOOTP
69 TFTP
70 Gopher
79 Finger
80 HTTP
88 Kerberos
102 MS Exchange
110 POP3

554 RTSP
546-547 DHCPv6
560 rmonitor
563 NNTP over SSL
587 SMTP
591 FileMaker
593 Microsoft DCOM
631 Internet Printing
636 LDAP over SSL
639 MSDP (PIM)
646 LDP (MPLS)
691 MS Exchange
860 iSCSI
873 rsync
902 VMware Server
989-990 FTP over SSL
993 IMAP4 over SSL
995 POP3 over SSL

2745 Bagle.H
2967 Symantec AV
3050 Interbase DB
3074 XBOX Live
3124 HTTP Proxy
3127 MyDoom
3128 HTTP Proxy
3222 GLBP
3260 iSCSI Target
3306 MySQL
3389 Terminal Server
3689 iTunes
3690 Subversion
3724 World of Warcraft
3784-3785 Ventrilo
4333 mSQL
4444 Blaster
4664 Google Desktop

6891-6901 Windows Live
6970 Quicktime
7212 GhostSurf
7648-7649 CU-SeeMe
8000 Internet Radio
8080 HTTP Proxy
8086-8087 Kaspersky AV
8118 Privoxy
8200 VMware Server
8500 Adobe ColdFusion
8767 TeamSpeak
8866 Bagle.B
9100 HP JetDirect
9101-9103 Bacula
9119 MXit
9800 WebDAV
9898 Dabber
9988 Rbot/Spybot

One port for each service

- Nous sommes libres de faire tourner les serveurs sur le port qu'on veut mais utiliser des ports "réservés" pour autre chose peut avoir des effets indésirables.

Internet is not a safe place !

- Tout appareil connecté à Internet avec une IP publique est attaqué en permanence par des automates à la recherche de failles de sécurités.

- Protégez vos ordinateurs et vos réseaux: n’ouvrez que les accès nécessaires sur vos pare-feux.

- Testez systématiquement ce que reçoivent vos serveurs.

- Ne partez jamais du principe que ce que vous recevez provient de vos logiciels !

Internet is not a safe place !

- N'envoyez jamais de mots de passe ou de clés API dans des accès publics sans authentification préalable.

- Faites attention à ce que vous mettez sur vos pages web et vos scripts. Tout est public, tout est visible, tout est enregistré par de nombreux robots inconnus et peut être utilisé un jour contre vos systèmes.

- Si vous partagez accidentellement un mot de passe ou une clé API, changez-le ! Ne supposez pas que personne ne l'a vu.

IP Sockets : low level communication

- Les sockets sont l'élément de base de la communication entre logiciels sur un réseau IP.
- Une socket est bidirectionnelle.
- On s'y échange des octets comme on le fait sur des ports série.

IP Sockets : low level communication

- Le serveur ouvre un socket en écoute sur une IP et un port.

- Les clients se connectent sur cet IP/port ce qui déclenche automatiquement l'ouverture d'une socket en lecture/écriture sur un autre port pour prendre en charge la communication avec ce client.

IP Sockets : low level communication

- Chaque client a un canal de communication dédié avec le serveur.

- Les clients ne peuvent pas discuter entre eux sans passer par le serveur.

Demo

Sockets problems

- Pour utiliser les sockets réseau il faut pouvoir lire et écrire nos données dans des buffers d’octets.

- Ce n’est pas disponible dans tous les langages de développement ou sur tout type de logiciel (par exemple JavaScript sur un navigateur web).

- Il n’est pas toujours facile de faire communiquer des logiciels écrits dans des langages différents selon le type de données échangées.

Client / Server with the Socket Messaging Library

- L’utilisation des sockets réseau reste cependant pratique quand on a les bons outils. C’est du bas niveau. On fait ce qu’on veut dessus.

- A l’occasion de la création d’un jeu vidéo multijoueur temps réel j’ai créé une librairie open source prenant en charge l’échange de messages entre des clients et serveurs.

- Un logiciel permet de définir les messages et générer le code pour les utiliser afin de gagner du temps.

Demo

Client / Server problems in a connected world

- Cette solution est pratique quand on veut faire discuter des programmes sous Delphi mais elle n’est pas ouverte pour d’autres technologies.

- Des solutions plus accessibles sont disponibles.

Using web technologies to exchange data

- Plutôt que réinventer la roue et définir des protocoles propriétaires en travaillant sur des sockets, pourquoi ne pas partir sur des choses qui fonctionnent partout ?

- Utiliser un serveur web en complément ou pour remplacer nos serveurs de données est un bon compromis.

A web server to serve APIs

- Un serveur web prend en charge les protocoles http et https.

- Il reçoit une demande du client.

- Il envoi une réponse avec un code de retour indiquant si la réponse est bonne ou s'il y a eu une erreur. Ces codes de retour sont normalisés : https://en.wikipedia.org/wiki/List_of_HTTP_status_codes

A web server to serve APIs

- En plus du code de retour le serveur transmet le type d'information envoyée sous forme de type MIME dont voici une liste : https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/MIME_types/Common_types

- Rien nous empêche d'utiliser ce principe pour autre chose que du HTML, du CSS, du JavaScript et des images !

A web server to serve APIs

- Tous les langages de développement permettent de faire une requête http ou https.

- A nous de fournir nos données dans des formats reconnus comme JSON, XML ou du texte exploitables dans la plupart des langages de développement.

- Et selon nos connaissances ou besoins on peut programmer le serveur en PHP, en JavaScript, en Java, en Python et bien sûr en Delphi !

Using REST API to simplify CRUD operations

- REST est une norme basée sur le protocole http/s.
- On utilise des serveurs web classiques afin d'accéder et modifier des données.
- De base REST était prévu pour faire du CRUD mais en réalité on peut tout faire sur ce principe.
- La plupart des langages de développement permettent de prendre en charge une API REST. Delphi et JavaScript en font bien entendu partie.

How to push data to web clients ?

- L’inconvénient principal des API web (REST ou pas) c’est qu’on est toujours en mode :

  client => serveur => client.

- Il n’est pas possible au serveur d’envoyer des données à un client sans que le client ne les ait demandées.

- C’est là qu’interviennent les WebSockets !

The WebSocket protocole

- Les WebSockets ne sont pas des sockets.
- WebSocket est un protocole de dialogue client/serveur bidirectionnel.
- Il est basé sur http/s pour fonctionner.
- C’est de la triche : le client va interroger régulièrement le serveur pour savoir si celui-ci a quelque chose à lui dire.
- L’avantage c’est que ça se fait tout seul.

The WebSocket protocole

- Le protocole WebSocket est une API implémentée dans les navigateurs web récents. JavaScript peut donc s'en servir sans nous obliger à passer par AJAX.

- Le protocole WebSocket n'est qu'un protocole d'échange d'informations basé sur http/s. Tout langage de développement ayant accès à http/s peut donc l'implémenter.

- TMS Software propose TMS FNC WebSocket pour les projets VCL, FireMonkey, Lazarus et Web Core.

Demo

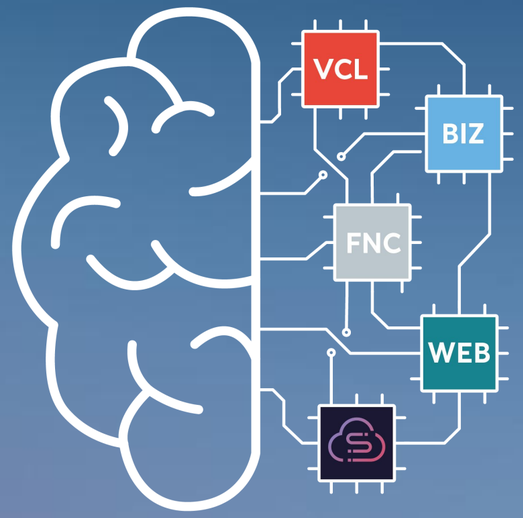
Conclusion

- Profitez du passage au web de vos projets pour moderniser vos serveurs de données.

- Passez de préférence sur un modèle REST pour les cas standards et sur JSON pour vos messages.

- Utilisez WebSocket quand vous avez besoin de temps réel (par exemple de la discussion) ou de remontée d'informations (par exemple des systèmes d'alerte ou de log).

Q&A



tms training days

THANK YOU!

Discover our social media pages