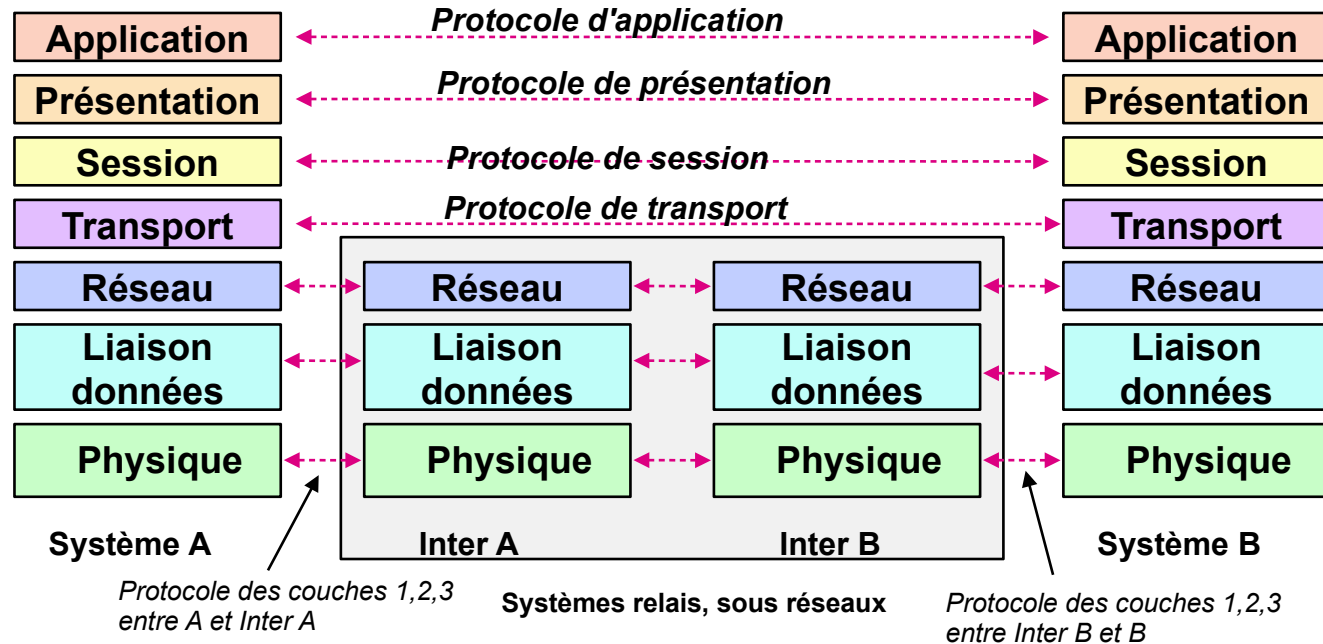
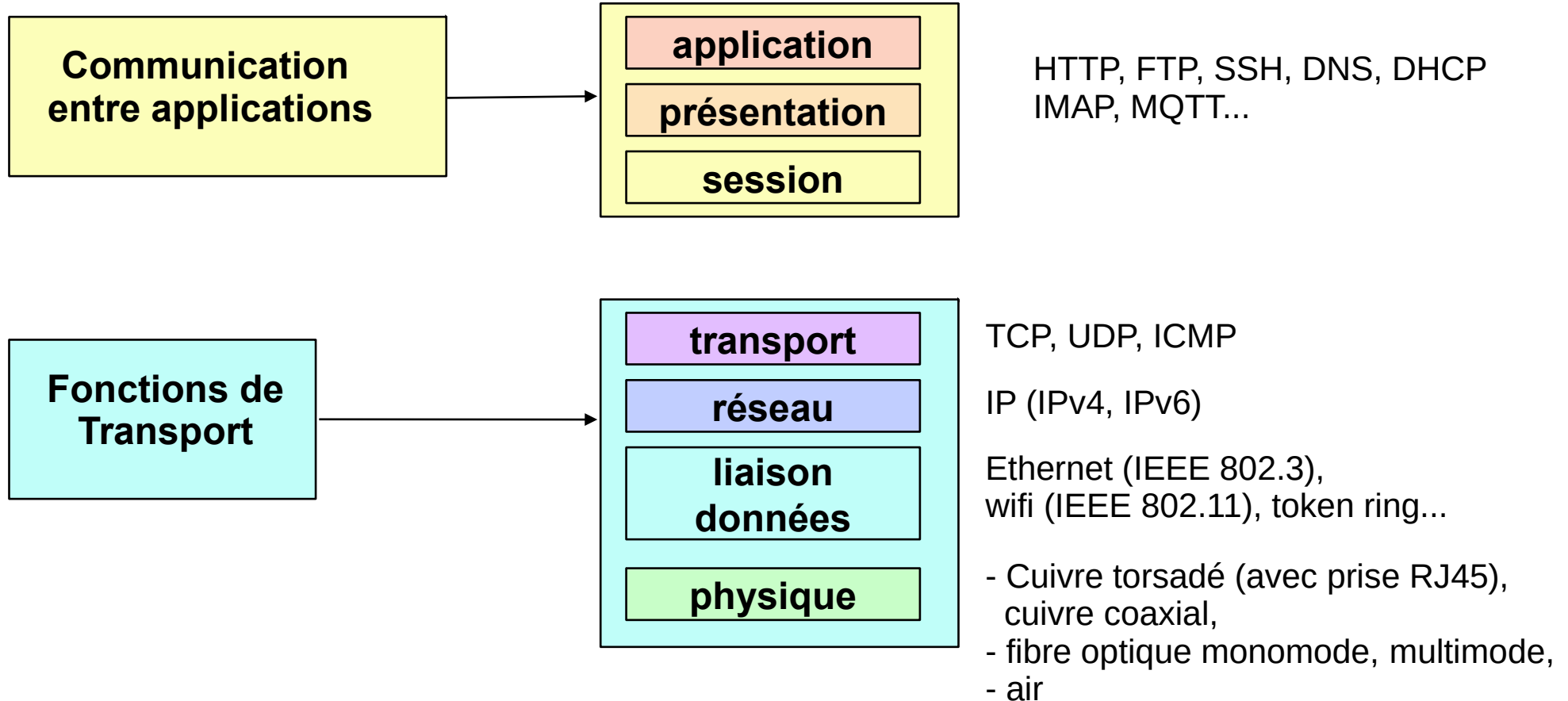


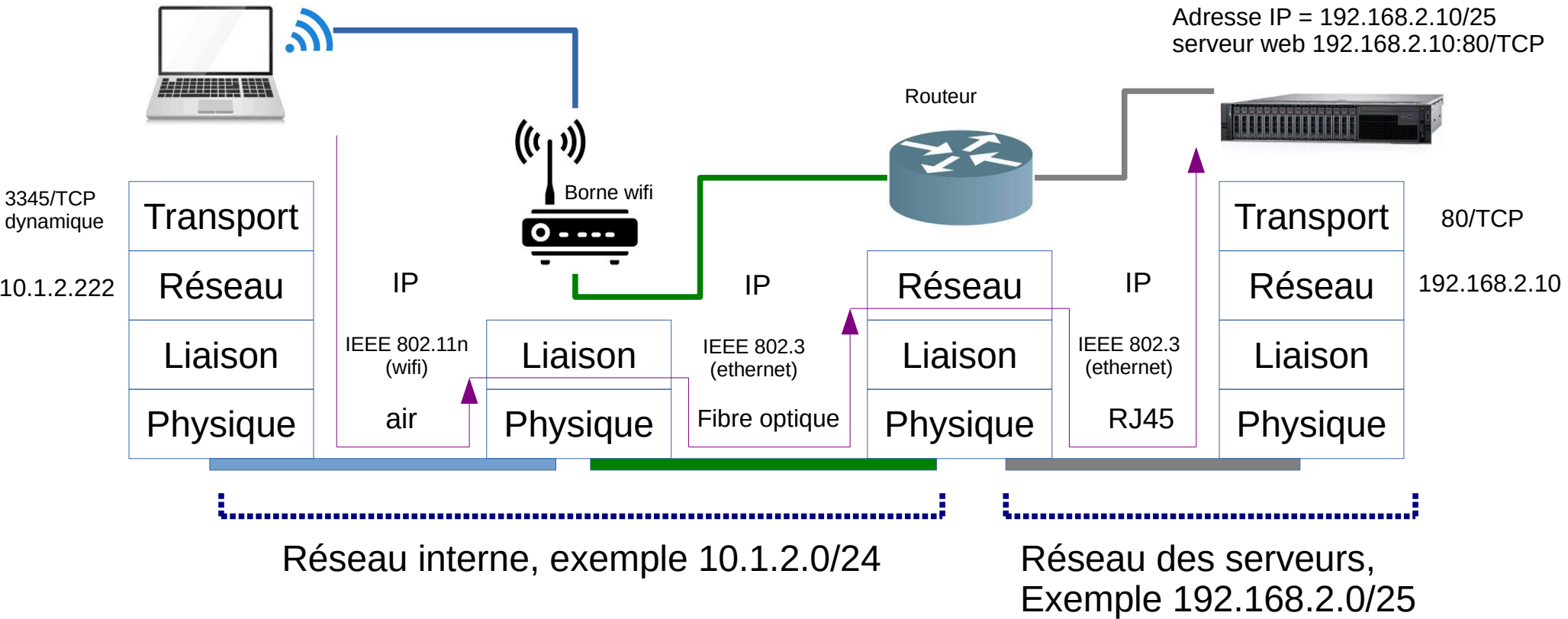
Couches réseau : modèle OSI (Open System Interconnection)



Modèle OSI avec exemples

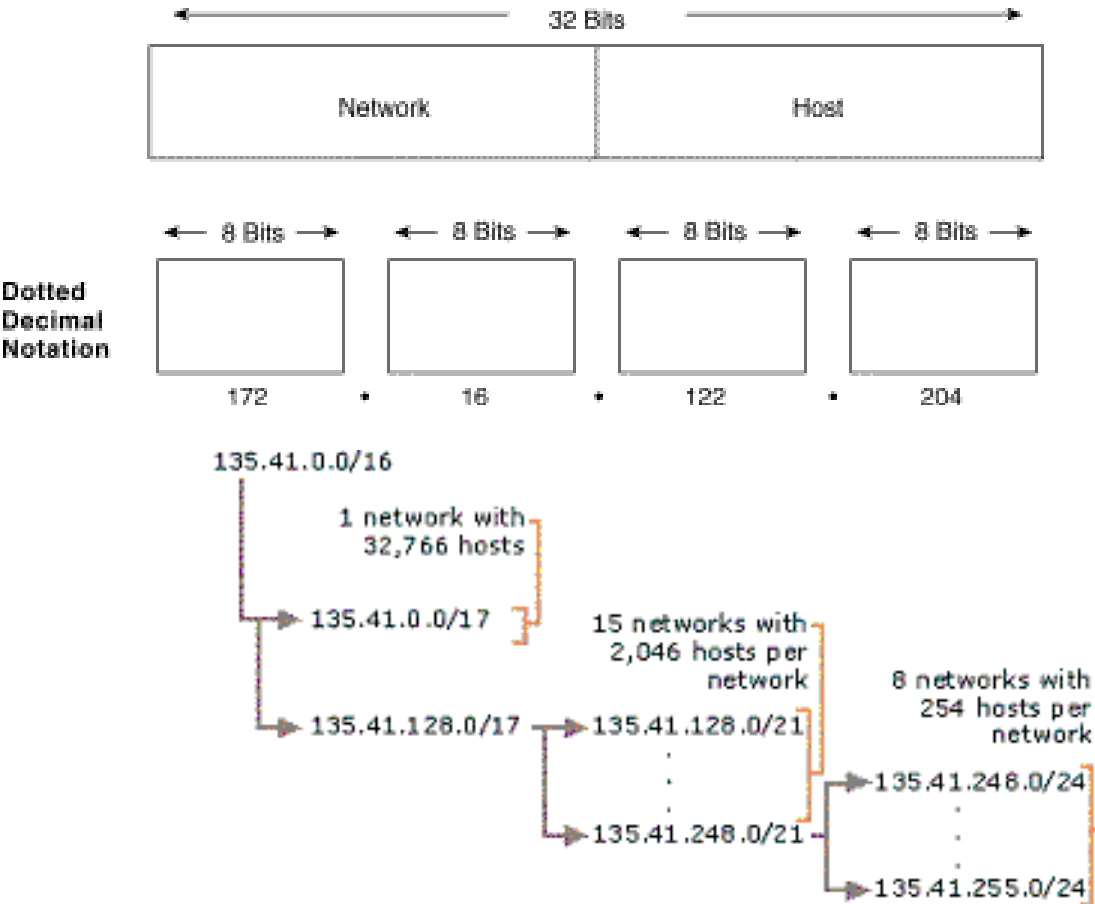


Adresse IP = 10.1.2.222/24
Client web http://192.168.2.10:80/index.html



Couche 3 : Réseau

IPv4 : adressage sur 4 octets (32 bits)



Il n'y a plus de notion de classe d'adresse IPv4 (classes A, B, C), mais de CIDR :

- * 192.0.1/24 : préfixe de réseau
- * 192.0.1.5/24 : adresse avec le masque

Exemple : 192.0.1.5/24

- 24 bits pour le numéro de réseau
- 8 bits pour l'adressage local
- $2^8 - 2$ adresses IPv4 locales possibles

Couche 3 : Réseau

IPv6 : adressage sur 16 octets (128 bits)

Le même principe de CIDR : Prefix / prefix length

- 2001:DB8:3003::/48 <= Prefix
- 2001:db8:3003:2:a00:20ff:fe18:964c/64 <= Address

Différentes notations possibles :

- 2001:0DB8:3003:0001:0000:0000:6543:210F (format complet)
- 2001:0DB8:3003:0001::6543:210F (format compact)

Si on souhaite se connecter à un serveur en utilisant directement les adresses IP :

- IPv4 : <http://192.0.1.5:80/index.html>
- IPv6 : [http://\[2001:0DB8:3003:0001::6543:210F\]:80/index.html](http://[2001:0DB8:3003:0001::6543:210F]:80/index.html)

Sachant que

- <http://192.0.1.5> par défaut sur le port 80/TCP
- <https://192.0.1.5> par défaut sur le port 443/TCP

Couche 3 : Réseau : routage et adressage des paquets

Pour qu'une machine puisse communiquer sur un réseau IP, il faut généralement :

- **une adresse IP + préfixe réseau** : exemple 172.16.122.204/24

Ou

- **une adresse IP + « masque de sous réseau » ou « netmask »**

Exemple : adresse IP 172.16.122.204 , netmask : 255.255.255.0

Et (obligatoire pour sortir du réseau) : **Une passerelle** dans le réseau local, lui permettant de communiquer avec d'autres machines en dehors de son réseau, exemple : 172.16.122.1

Et (facultatif, mais utile pour utiliser les noms plutôt que les adresses IP) : **L'adresse IP d'un serveur DNS** : 9.9.9.9

La couche 3 route les paquets IP = les fait transiter d'un réseau aux autres sur Internet
(Internet = interconnexion de réseaux IP)

Un routeur IP, de base, **ne sait faire que 2 choses** :

- * **envoyer** un paquet reçu, sur une de ses interfaces

- * **détruire** le paquet reçu

Tests de connectivité IP :

```
$ ping 9.9.9.9
```

```
$ traceroute 9.9.9.9
```

```
$ telnet 172.16.122.204 80
```

Couche 4 : Transport : gère les communications de bout en bout entre les systèmes

On sait que la couche 3 (IP) ne garantit pas que les paquets arrivent à leur destination

Mais comment les applications peuvent-elles communiquer ?

En utilisant la couche transport qui lui offre plusieurs services :

- * un service de transport des paquets avec connexion : **TCP** (Transmission Control Protocol)
- * un service de transport des paquets sans connexion **UDP** (User Datagram Protocol)
- * un service de contrôle : **ICMP** (Internet Control Message Protocol)

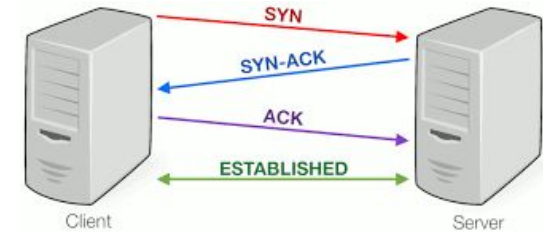
- **TCP (Transmission Control Protocol)**

- mode connecté,
- reprise automatique en cas d'erreur de transmission, de perte de paquets ou de panne de liaison entre émetteur et récepteur ,
- identifie le flux de données comme UDP,
- assure une transmission fiable,
- assure un contrôle de congestion,
- connexion uniquement en point à point (unicast) en mode duplex.

Il ouvre la connexion en 3 temps : « **TCP 3 way handshake** » => connexion établie

- **UDP (User Datagram Protocol)**

- mode déconnecté,
- transport non fiable,
- très efficace, (mais) peut utiliser toute la bande passante



TCP 3 way handshake

Que se passe t'il quand on utilise une application réseau ?

Exemple client ssh : **Linux1 (10.1.2.222)** qui va se connecter à un serveur **serveur1.local**

Le serveur DNS configuré dans Linux1 est **9.9.9.9**

```
$ ssh pugnere@serveur1.local
```

=> Linux1 ne connaît pas l'adresse IP de serveur1.local

=> **requête DNS de l'OS de Linux1** de vers serveur DNS 9.9.9.9 pour connaître l'adresse IP de serveur1.local
(par convention le serveur DNS écoute sur le port 53/UDP du serveur)

=> résolveur DNS de Linux1 : **10.1.2.222:1456/UDP -> 9.9.9.9:53/UDP** : quelle est l'adresse IP de serveur1.local ?

=> réponse du serveur DNS : **9.9.9.9:53/UDP -> 10.1.2.222:1456/UDP** : l'adresse IP de serveur1.local est **192.168.2.10**

=> le client ssh se connecte alors sur serveur1.local : **10.1.2.222:44500/TCP -> 192.168.2.10:22/TCP**
(par convention le client SSH se connecte sur le port 22/TCP du serveur)

=> « **TCP 3 way handshake** » => connexion établie => socket ouverte

=> L'application SSH peut maintenant utiliser la connexion

=> utilisation du canal de communication : négociation protocole, chiffrement, compression...

=> échanges de clés et d'empreintes entre le client SSH et le serveur SSH :

```
$ ssh pugnere@serveur1.local
```

```
The authenticity of host 'serveur1.local (192.168.2.10)' can't be established.
```

```
ECDSA key fingerprint is SHA256:hM+ajJw3UN0l8beTwP4zR0lUkLw0Xvg6tJvJErN6rWs.
```

```
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
```

=> authentification de l'utilisateur

=> Création du shell sur serveur1.local

```
[iot@serveur1.local ~]$
```



Quelques questions

- Pourquoi IP est non fiable ?
- Dans quelles conditions peut-on avoir des pertes de paquets ?

Quelques questions

- Comment fiabiliser une connexion IP entre un client et un serveur pour une application de transmission de données ?
- Comment tester si un serveur (ou un device) est connecté sur un réseau (ou Internet) ?
- Quel numéro de port utiliser pour tester un service réseau ?