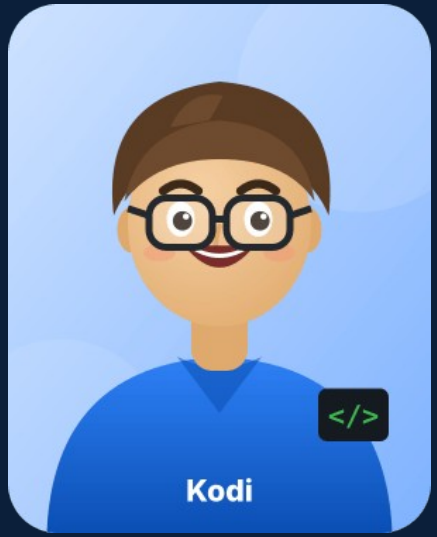
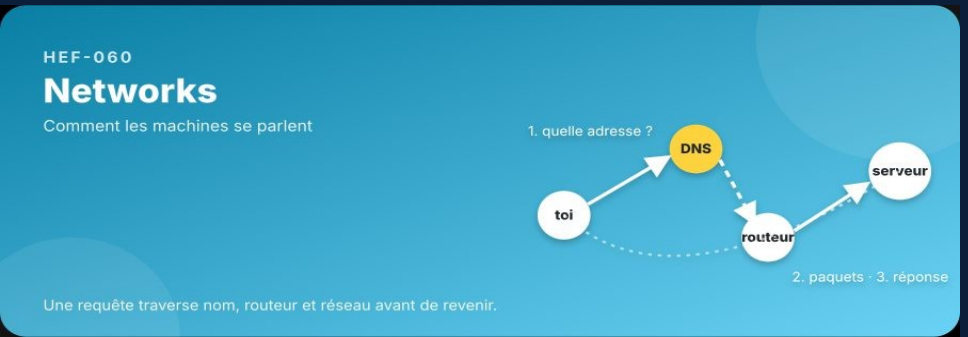
HEF-060 · MISSION 01 / 10

Les modèles réseau : OSI et TCP/IP

Un message qui traverse internet passe par des couches, comme un colis qui change d'emballage à chaque étape.





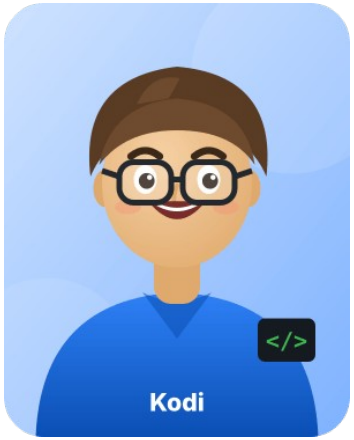
Badge : Cartographe du réseau

70 min · intermédiaire

L'histoire

Kodi envoie un message à un ami de l'autre côté du monde. Il appuie sur « Envoyer » et, une fraction de seconde plus tard, le message est là. Il se demande : « Mais par où est-il passé ? Qui s'est occupé de quoi ? »

> *« Personne ne s'occupe de tout, dit Sam. Chaque couche fait une petite tâche et fait confiance à celle du dessous. C'est ce qui permet à des milliards d'appareils différents de se comprendre. »*



Dans cette mission, tu découvres les couches qui organisent tous les réseaux.

Ta mission

À la fin, je serai capable de…

1. expliquer pourquoi on découpe les réseaux en couches
2. citer les couches du modèle TCP/IP et leur rôle
3. décrire l'encapsulation d'un message
4. relier un protocole (HTTP, TCP, IP, Ethernet) à sa couche
5. distinguer modèle OSI (théorique) et TCP/IP (utilisé)

Défi d'échauffement

Défi d'échauffement

Tu envoies un colis à un ami à l'étranger. Tu écris une lettre (contenu), tu la mets dans une enveloppe avec l'adresse (destinataire), un transporteur la place dans un carton avec un numéro de suivi, puis un camion la transporte. Quels sont les **emballages successifs** ? Qui lit quoi à l'arrivée ? Le camionneur a-t-il besoin de lire ta lettre ?

Le concept

Pourquoi des couches ?

Un réseau doit faire beaucoup de choses : transmettre des signaux, adresser, router, fiabiliser, chiffrer, présenter une page web. Plutôt que de tout mélanger, on **sépare en couches**. Chaque couche :

- offre un **service** à la couche du dessus ;
- utilise le service de la couche du dessous ;
- peut être **remplacée** sans changer les autres (le Wi-Fi remplace le câble, la page web ne le sait pas).

C'est la même idée que l'abstraction (HEF-000) et que les interfaces (HEF-020).

Le concept

Le modèle TCP/IP (4 couches utilisées en pratique)

Couche	Rôle	Exemples
Application	ce que l'utilisateur voit : web, mail, noms	HTTP, HTTPS, DNS, SMTP, SSH
Transport	communication entre deux **programmes** (ports), fiabilité	TCP, UDP
Internet (réseau)	adressage et acheminement entre **machines**	IP, ICMP
Accès réseau (liaison)	transmission sur **un lien local**	Ethernet, Wi-Fi

Le modèle OSI (7 couches, de référence)

#	Couche OSI	Équivalent TCP/IP
7	Application	Application

Le concept

Le modèle OSI (7 couches, de référence) (suite)

#	Couche OSI	Équivalent TCP/IP
6	Présentation (formats, chiffrement)	Application
5	Session	Application
4	Transport	Transport
3	Réseau	Internet
2	Liaison de données	Accès réseau
1	Physique (câbles, ondes)	Accès réseau

Le modèle OSI sert de **vocabulaire** (« problème de couche 2 », « pare-feu de couche 7 ») ; TCP/IP décrit ce qui tourne réellement.

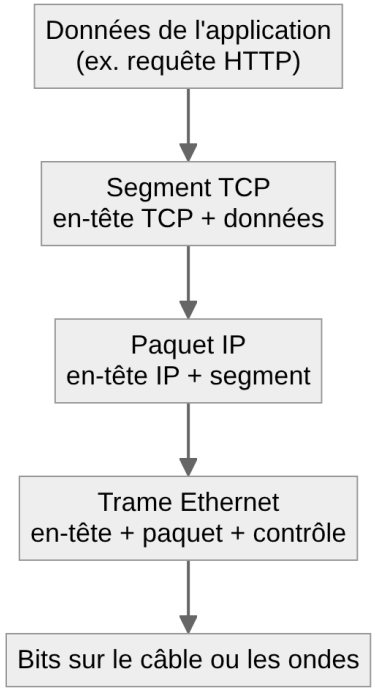
Le concept

L'encapsulation

À l'envoi, chaque couche **ajoute son en-tête** (et parfois une fin) autour des données de la couche supérieure. À la réception, chaque couche **retire** le sien.

Le concept

L'encapsulation (suite)



Le concept

L'encapsulation (suite)

Description du schéma : les données de l'application sont placées dans un segment TCP (avec son en-tête), qui est placé dans un paquet IP (avec son en-tête), qui est placé dans une trame Ethernet, qui part sous forme de bits sur le support physique.

Qui lit quoi ?

Équipement	Regarde jusqu'à	Pourquoi
Switch	l'adresse Ethernet (couche 2)	acheminer sur le réseau local
Routeur	l'adresse IP (couche 3)	choisir le prochain saut
Pare-feu	IP, ports (couches 3-4), parfois plus	filtrer
Machine finale	toutes les couches	remettre les données au bon programme

Le concept

Un en-tête, concrètement

Un en-tête est une suite d'octets avec des champs de taille fixe. Python peut le fabriquer :

```
import struct

donnees = b"GET / HTTP/1.1"
# En-tête TCP simplifié : port source (2 octets), port destination (2 octets)
segment = struct.pack("!HH", 54321, 80) + donnees
# En-tête IP simplifié : adresse source (4 octets), adresse destination (4 octets)
paquet = bytes([192, 168, 1, 10]) + bytes([93, 184, 216, 34]) + segment

print(len(donnees), len(segment), len(paquet))     # 14 18 26
print(paquet[:8].hex(" "))                         # c0 a8 01 0a 5d b8 d8 22
```

Le concept

Un en-tête, concrètement (suite)

🧠 ***Astuce*** *: quand un problème réseau survient, demande « à quelle* ***couche*** *? » Cela réduit tout de suite l'espace de recherche.*

Exemple pas à pas

Problème : simuler l'encapsulation et la décapsulation d'un message, couche par couche.

```
def encapsuler(message, ip_src, ip_dst, port_src, port_dst, mac_src, mac_dst):
    segment = {"couche": "transport", "port_src": port_src, "port_dst": port_dst,
   "donnees": message}
    paquet = {"couche": "internet", "ip_src": ip_src, "ip_dst": ip_dst, "contenu":
    segment}
    trame = {"couche": "liaison", "mac_src": mac_src, "mac_dst": mac_dst, "contenu
   ": paquet}
    return trame

def decapsuler(trame, ma_mac, mon_ip, mon_port):
    if trame["mac_dst"] != ma_mac:
        return "rejeté par la couche liaison (pas pour ma carte)"
    paquet = trame["contenu"]
    if paquet["ip_dst"] != mon_ip:
        return "rejeté par la couche internet (pas pour mon adresse IP)"
    segment = paquet["contenu"]
```

Exemple pas à pas (suite)

```
    if segment["port_dst"] != mon_port:
        return "rejeté par la couche transport (aucun programme à ce port)"
    return "livré à l'application : " + segment["donnees"]

trame = encapsuler("bonjour", "192.168.1.10", "192.168.1.20", 54321, 80,
                   "aa:aa:aa:aa:aa:01", "bb:bb:bb:bb:bb:02")

print(decapsuler(trame, "bb:bb:bb:bb:bb:02", "192.168.1.20", 80))
print(decapsuler(trame, "cc:cc:cc:cc:cc:03", "192.168.1.20", 80))
print(decapsuler(trame, "bb:bb:bb:bb:bb:02", "192.168.1.99", 80))
print(decapsuler(trame, "bb:bb:bb:bb:bb:02", "192.168.1.20", 443))
```

Lecture : chaque couche fait **son** contrôle et ne regarde que **son** en-tête. Un message peut être rejeté à l'une des trois couches, pour une raison différente.

C'est exactement ce qui permet de diagnostiquer : si le message arrive à la carte mais pas au programme, le problème est en haut ; s'il n'arrive pas du tout, il est en bas.

À toi, avec aide

À toi, avec aide

Classe ces éléments dans la bonne couche TCP/IP :

Élément	Couche
HTTPS	?
Adresse IP	?
Port 443	?
Adresse MAC	?
DNS	?
Câble Ethernet / ondes Wi-Fi	?

À toi, avec aide · réponse

Indices progressifs

Indice 1

HTTPS et DNS sont des protocoles que les programmes utilisent directement.

Indice 2

Un port identifie un programme sur une machine.

Indice 3

L'adresse MAC ne sert que sur le réseau local.

À toi, avec aide · réponse

Réponse

Élément	Couche
HTTPS	Application
Adresse IP	Internet (réseau)
Port 443	Transport
Adresse MAC	Accès réseau (liaison)
DNS	Application
Câble / Wi-Fi	Accès réseau (physique)

À toi, en solo

Voici trois pannes. Pour chacune, dis à quelle couche le problème se situe probablement et propose une vérification :

1. Le câble est débranché : la carte réseau n'a plus de lumière.
2. La machine a une adresse IP, peut joindre `8.8.8.8` mais pas `exemple.fr`.
3. Le serveur est joignable, mais la connexion au port 443 est refusée.

À toi, en solo · réponse

Pistes de correction

1. **Couche physique/liaison** : vérifier câble, voyant, `ip link`.
2. **Couche application (DNS)** : la connectivité IP marche, la résolution de noms non. Vérifier `nslookup exemple.fr` ou le serveur DNS configuré.
3. **Couche transport/application** : aucun programme n'écoute sur ce port, ou un pare-feu bloque. Vérifier que le service tourne (`ss -tlnp`) et les règles de pare-feu.

La démarche est toujours la même : de bas en haut, jusqu'à la première couche qui échoue.

Pièges classiques

Piège	Pourquoi c'est un problème	Comment l'éviter
Mélanger les couches	On cherche au mauvais endroit	Toujours nommer la couche concernée
Croire que le routeur lit le contenu	Il lit l'en-tête IP, pas la lettre	Chaque équipement regarde sa couche
Apprendre OSI par cœur sans comprendre	Aucune utilité en dépannage	Relier chaque couche à un exemple concret
Confondre MAC et IP	MAC : réseau local ; IP : entre réseaux	L'adresse MAC change à chaque saut, l'IP reste

Mini-boss

Le voyage d'un message. Alice (192.168.1.10) envoie « salut » au serveur 203.0.113.50 (port 80) via son routeur 192.168.1.1.

1. Dessine (Mermaid) le trajet : Alice → switch → routeur → internet → serveur.
2. Pour chaque saut, écris les adresses MAC et IP source/destination de la trame. Que remarques-tu sur les adresses MAC et IP au fil des sauts ?
3. Écris un programme qui simule ce voyage avec `encapsuler` et un routeur qui **réécrit les adresses MAC** mais conserve les adresses IP.

Mini-boss · réponse

Pistes de correction

```
def trame(mac_src, mac_dst, ip_src, ip_dst, message):
    return {"mac_src": mac_src, "mac_dst": mac_dst, "ip_src": ip_src, "ip_dst": ip
   _dst, "msg": message}

def routeur(trame_entrante, mac_routeur_sortie, mac_prochain_saut):
    t = dict(trame_entrante)
    t["mac_src"], t["mac_dst"] = mac_routeur_sortie, mac_prochain_saut      # nouv
   elles adresses MAC
    return t                                                                # IP i
   nchangées

t1 = trame("aa:01", "rr:in", "192.168.1.10", "203.0.113.50", "salut")       # Alic
   e -> routeur
t2 = routeur(t1, "rr:out", "fai:01")                                        # rout
   eur -> routeur du FAI
t3 = routeur(t2, "fai:02", "srv:01")                                        # FAI
   -> serveur
for nom, t in (("Alice→routeur", t1), ("routeur→FAI", t2), ("FAI→serveur", t3)):
    print(nom, t)
```

Mini-boss · réponse

Pistes de correction (suite)

```
assert t1["ip_src"] == t3["ip_src"] and t1["ip_dst"] == t3["ip_dst"]
assert t1["mac_src"] != t3["mac_src"]
```

Observation : les **adresses MAC changent à chaque saut** (elles ne servent que sur le lien local), alors que les **adresses IP restent** de bout en bout (hors NAT, qu'on verra plus tard).

Quiz éclair

Quiz éclair

1. L'encapsulation consiste à :

A. ajouter un en-tête à chaque couche
B. chiffrer les données
C. compresser le message

2. Un routeur prend ses décisions à partir de :

A. l'adresse MAC de l'émetteur
B. l'adresse IP de destination
C. le contenu de la page web

3. Un port identifie :

A. un câble

Quiz éclair

Quiz éclair (suite)

B. une machine
C. un programme sur une machine

4. 🔁 *Révision HEF-050, mission 09* : Un service (démon) tourne :

A. en arrière-plan, supervisé par le système
B. uniquement dans un terminal
C. sans processus

5. 🔁 *Révision HEF-050, mission 08* : Un tube (pipe) relie :

A. deux disques
B. la sortie d'un processus à l'entrée d'un autre
C. deux écrans

Quiz éclair · réponse

Réponses

1. **A**.
2. **B**.
3. **C**.
4. **A**.
5. **B**.

Je réfléchis

1. Quelle situation de ta vie ressemble à une encapsulation (message dans une enveloppe, dans un carton, dans un camion) ?
2. Pourquoi dire « à quelle couche ? » aide-t-il à résoudre un problème ?

À retenir

À retenir

1. Les réseaux sont organisés en couches indépendantes.
2. TCP/IP : application, transport, internet, accès réseau.
3. Chaque couche ajoute son en-tête (encapsulation).
4. Chaque équipement regarde sa couche : switch (MAC), routeur (IP).
5. Diagnostiquer = trouver la première couche qui échoue.

MISSION ACCOMPLIE

+100 XP

Badge débloqué : Cartographe du réseau

Prochaine mission : Ethernet, adresses MAC et réseau local

