

Protocoles de communication dans un réseau

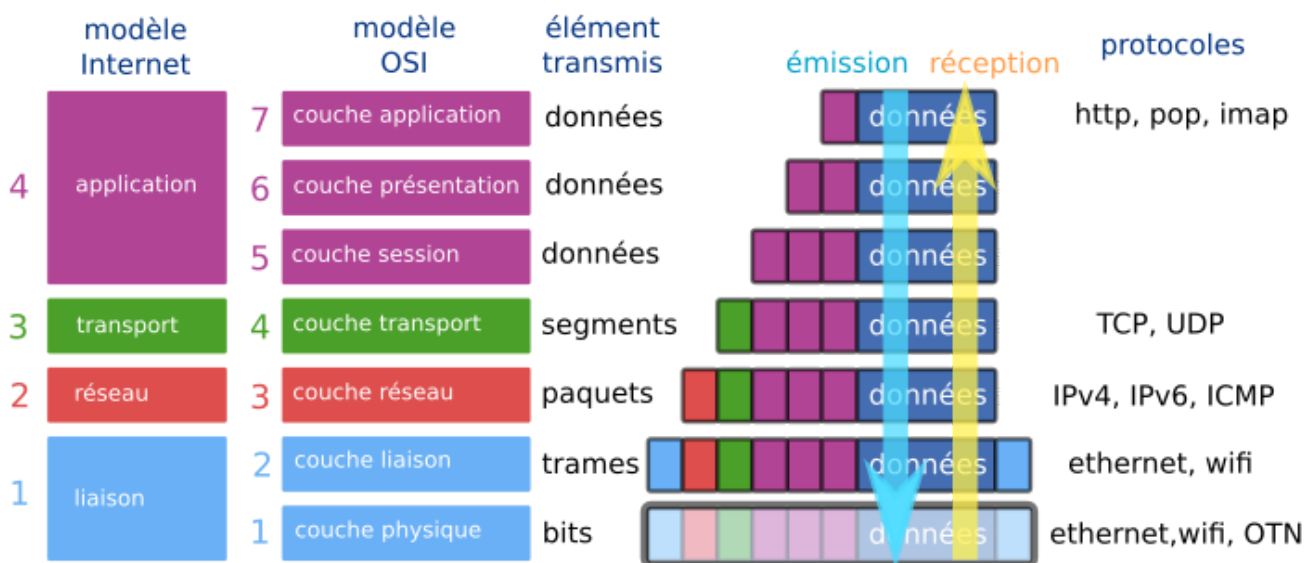
I. Modèle OSI, modèle Internet

Les bits transmis d'un ordinateur à un autre contiennent, en plus des données *utiles* (le mot «bonjour» dans un email), une multitude de données (tout aussi utiles) qui vont aider à l'acheminement de ces bits au bon endroit, puis au bon ordinateur, puis au bon logiciel. Les différents protocoles qui régissent cette transmission sont regroupés dans ce qui est appelé un **modèle**. Deux modèles synthétisent ces protocoles :

- le **modèle Internet** (ou modèle **TCP/IP**, 1974), organisé en **4** couches : liaison, réseau, transport, application.
- le **modèle OSI** (Open Systems Interconnection, 1984), organisé en **7** couches : physique, liaison, réseau, transport, session, présentation, application.

Ces deux modèles coïncident suivant le schéma ci-dessus. Ce sont des modèles théoriques et d'une certaine rigidité. Leur utilisation dans la pratique est parfois plus floue, avec des protocoles à cheval sur plusieurs couches.

Dans la suite de ce cours, nous évoquerons les couches par leur numéro dans le modèle OSI.



Lors de son émission, un message va subir successivement toutes les transformations effectuées par chaque couche, depuis sa création (couche 7) jusqu'à sa transmission physique (couche 1).

Lorsque ce même message sera réceptionné, les transformations seront effectuées dans l'ordre inverse, jusqu'à la présentation du message au destinataire.

- **couches 7-6-5 — couches application-présentation-session** : Ces couches (réunies dans le modèle Internet en une couche unique «application») regroupent les protocoles nécessaires à la bonne mise en forme d'un message (au sens large) avant sa transmission. Ces protocoles peuvent être de nature très différente : protocole HTTP pour la transmission de pages web, protocole FTP pour le transfert de fichiers, protocoles POP ou IMAP pour le courrier électronique...
- **couche 4 — couche transport** :
Le protocole majeur de cette couche est le protocole TCP :

- il s'assure par SYN-ACK que l'émetteur et le récepteur sont prêts à échanger des messages.
 - il découpe en segments numérotés le message à transmettre (côté émetteur) ou bien recompose le message total en remettant les segments dans l'ordre (côté récepteur).
- Les éléments échangés avec la couche inférieure sont des **segments**.

- **couche 3 — couche réseau :**

C'est la couche où chaque segment numéroté est encapsulé dans un paquet qui, suivant le protocole IP, va contenir son adresse source et son adresse de destination. C'est à ce niveau que se décide si le message doit rester dans le réseau local ou être envoyé sur un autre réseau via la passerelle du routeur.

Les éléments échangés avec la couche inférieure sont des **paquets**.

- **couche 2 — couche liaison :**

C'est l'encapsulation finale du message. Suivant le protocole Ethernet, les informations sont transmises d'une carte réseau à une autre, grâce à leur adresse MAC (Media Access Control).

Les éléments échangés avec la couche inférieure sont des **trames**.

- **couche 1 — couche physique :**

C'est la couche où le message est transmis physiquement d'un point à un autre. Par signal lumineux (fibre optique), par ondes (wifi), par courant électrique (Ethernet)... Les éléments transmis sont les **bits**.

Lors de son parcours, une trame peut être partiellement décapsulée et remonter à la couche 3, avant de redescendre et de continuer son chemin. C'est le cas notamment lors du passage dans un routeur. Mais jamais, lors de son acheminement, le contenu réel du message n'est ouvert : les paquets transmis sont acheminés de manière identique, qu'ils contiennent les éléments constitutifs d'une vidéo YouTube ou d'un email à votre cousin.

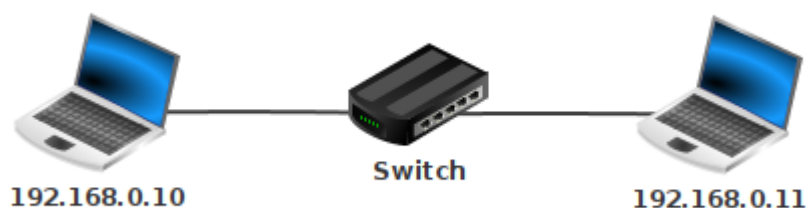
Ce principe fondateur, actuellement menacé par certains acteurs politiques et industriels, est connu sous l'expression «**la neutralité du net**».

II. Observation des trames avec Filius

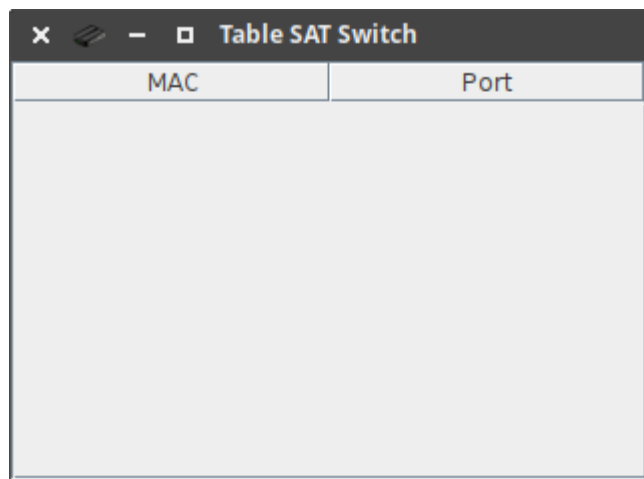
1. Ping à travers un switch

Vous pouvez télécharger le fichier [ping_switch.fls](#).

- Relions une machine **192.168.0.10** d'adresse MAC **BC:81:81:42:9C:31** à une machine **192.168.0.11** d'adresse MAC **2A:AB:AC:27:D6:A7** à travers un switch.

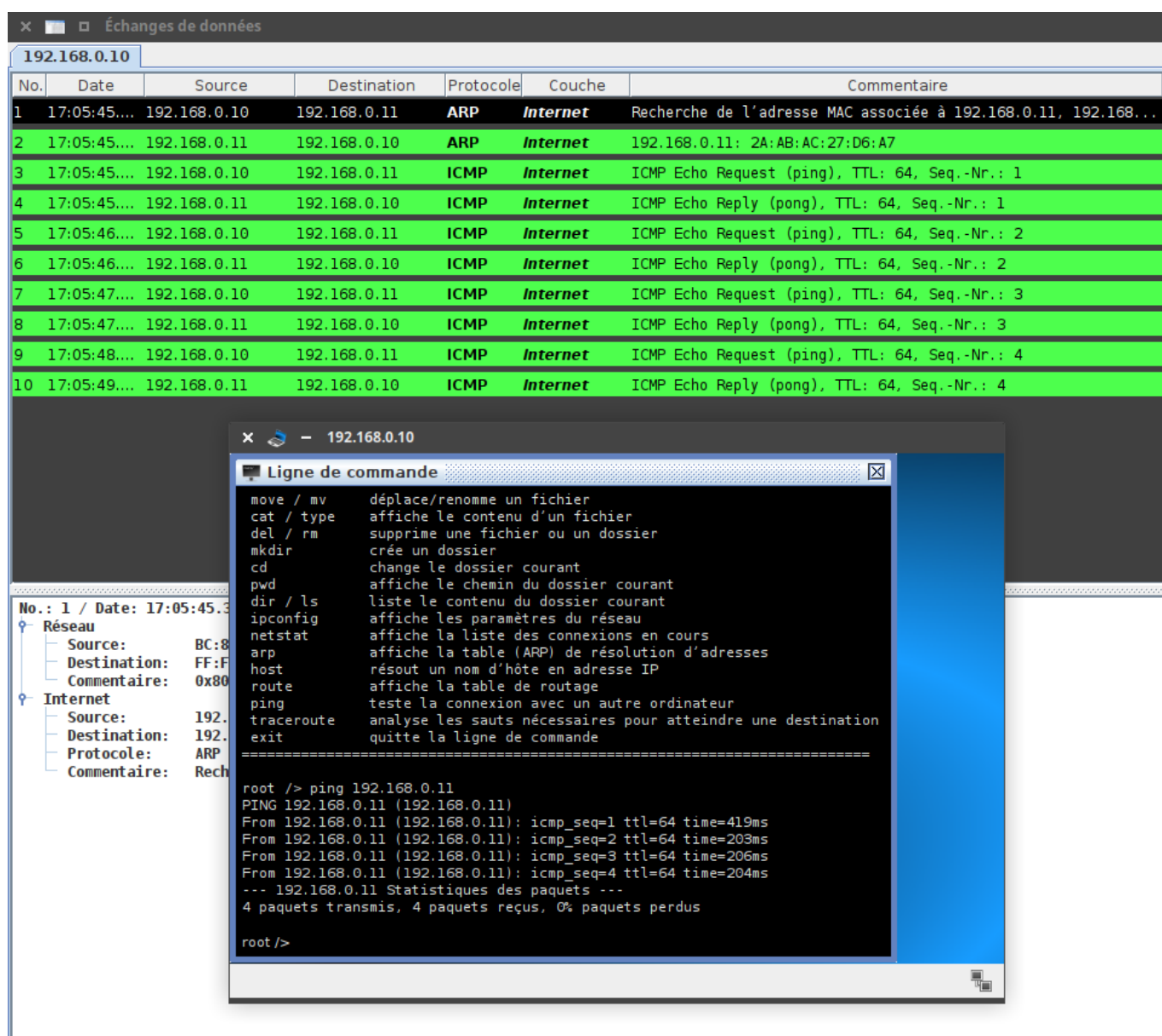


- Observons la table SAT de notre switch : elle est vide, car aucune machine n'a encore cherché à communiquer.



MAC	Port
-----	------

- Lançons un ping depuis 192.168.0.10 vers 192.168.0.11 et observons les données échangées :



No.	Date	Source	Destination	Protocole	Couche	Commentaire
1	17:05:45....	192.168.0.10	192.168.0.11	ARP	Internet	Recherche de l'adresse MAC associée à 192.168.0.11, 192.168...
2	17:05:45....	192.168.0.11	192.168.0.10	ARP	Internet	192.168.0.11: 2A:AB:AC:27:D6:A7
3	17:05:45....	192.168.0.10	192.168.0.11	ICMP	Internet	ICMP Echo Request (ping), TTL: 64, Seq.-Nr.: 1
4	17:05:45....	192.168.0.11	192.168.0.10	ICMP	Internet	ICMP Echo Reply (pong), TTL: 64, Seq.-Nr.: 1
5	17:05:46....	192.168.0.10	192.168.0.11	ICMP	Internet	ICMP Echo Request (ping), TTL: 64, Seq.-Nr.: 2
6	17:05:46....	192.168.0.11	192.168.0.10	ICMP	Internet	ICMP Echo Reply (pong), TTL: 64, Seq.-Nr.: 2
7	17:05:47....	192.168.0.10	192.168.0.11	ICMP	Internet	ICMP Echo Request (ping), TTL: 64, Seq.-Nr.: 3
8	17:05:47....	192.168.0.11	192.168.0.10	ICMP	Internet	ICMP Echo Reply (pong), TTL: 64, Seq.-Nr.: 3
9	17:05:48....	192.168.0.10	192.168.0.11	ICMP	Internet	ICMP Echo Request (ping), TTL: 64, Seq.-Nr.: 4
10	17:05:49....	192.168.0.11	192.168.0.10	ICMP	Internet	ICMP Echo Reply (pong), TTL: 64, Seq.-Nr.: 4

Ligne de commande

```

move / mv      déplace/renomme un fichier
cat / type     affiche le contenu d'un fichier
del / rm       supprime un fichier ou un dossier
mkdir          crée un dossier
cd             change le dossier courant
pwd            affiche le chemin du dossier courant
dir / ls       liste le contenu du dossier courant
ipconfig       affiche les paramètres du réseau
netstat        affiche la liste des connexions en cours
arp            affiche la table (ARP) de résolution d'adresses
host           résout un nom d'hôte en adresse IP
route          affiche la table de routage
ping           teste la connexion avec un autre ordinateur
tracert        analyse les sauts nécessaires pour atteindre une destination
exit           quitte la ligne de commande

```

```

root /> ping 192.168.0.11
PING 192.168.0.11 (192.168.0.11): icmp_seq=1 ttl=64 time=419ms
From 192.168.0.11 (192.168.0.11): icmp_seq=2 ttl=64 time=203ms
From 192.168.0.11 (192.168.0.11): icmp_seq=3 ttl=64 time=206ms
From 192.168.0.11 (192.168.0.11): icmp_seq=4 ttl=64 time=204ms
--- 192.168.0.11 Statistiques des paquets ---
4 paquets transmis, 4 paquets reçus, 0% paquets perdus

root />

```

- Observons de plus près la première ligne de données échangées.

```

No.: 1 / Date: 17:05:45.331
Réseau
├── Source:      BC:81:81:42:9C:31
├── Destination: FF:FF:FF:FF:FF:FF
└── Commentaire: 0x806
Internet
├── Source:      192.168.0.10
├── Destination: 192.168.0.11
├── Protocole:   ARP
└── Commentaire: Recherche de l'adresse MAC associée à 192.168.0.11, 192.168.0.10: BC:81:81:42:9C:31

```

Cette première ligne est une requête **ARP**. ARP est un protocole qui s'interface entre la couche 3 / réseau (appelée dans la capture d'écran *Internet*) et la couche 2 / liaison (appelée dans la capture d'écran *Réseau*). Comme indiqué dans le commentaire, elle consiste à un appel à tout le réseau : "Est-ce que quelqu'un ici possède l'IP **192.168.0.11** ?

Message 1 : « Qui possède l'IP **192.168.0.11** ? »

Il faut comprendre à cette étape que l'adresse IP est totalement inutile pour repérer un ordinateur dans un sous-réseau. Ce sont les adresses MAC qui permettent de se repérer dans un sous-réseau. Les adresses IP, elles, permettront éventuellement d'acheminer le message jusqu'au bon sous-réseau (elles n'intéressent donc que les routeurs).

Revenons à notre ping vers **192.168.0.11**.

La commande `arp -a` effectuée dans un terminal de la machine **192.168.0.10** nous permet de voir qu'elle ne connaît encore personne dans son sous-réseau. La table de correspondance IP \leftrightarrow MAC ne contient que l'adresse de broadcast **255.255.255.255**, qui permet d'envoyer un message à tout le réseau.

```

root /> arp -a
| Adresse IP | Adresse MAC |
|-----|
| 255.255.255.255 | FF:FF:FF:FF:FF:FF |
|-----|
root />

```

Constatant qu'elle ne sait pas quelle est l'adresse MAC de **192.168.0.11**, la machine **192.168.0.10** commence donc par envoyer un message à **tout** le sous-réseau, par l'adresse MAC de broadcast **FF:FF:FF:FF:FF:FF**. Le switch va lui aussi relayer ce message à tous les équipements qui lui sont connectés (dans notre cas, un seul ordinateur)

Message 2 : « Moi ! »

La machine **192.168.0.11** s'est reconnu dans le message de broadcast de la machine **192.168.0.10**. Elle lui répond pour lui donner son adresse MAC.

No.: 2 / Date: 17:05:45.539

```

Réseau
├── Source:      2A:AB:AC:27:D6:A7
├── Destination: BC:81:81:42:9C:31
├── Commentaire: 0x806
Internet
├── Source:      192.168.0.11
├── Destination: 192.168.0.10
├── Protocole:   ARP
└── Commentaire: 192.168.0.11: 2A:AB:AC:27:D6:A7

```

À partir de ce moment, la machine **192.168.0.10** sait comment communiquer avec **192.168.0.11**. Elle l'écrit dans sa table **arp**, afin de ne plus avoir à émettre le message n°1 :

```

root /> arp -a
|  Adresse IP      |  Adresse MAC      |
|-----|-----|
| 255.255.255.255 | FF:FF:FF:FF:FF:FF |
| 192.168.0.11    | 2A:AB:AC:27:D6:A7 |
|-----|-----|

```

Le switch, qui a vu passer sur ses ports 0 et 1 des messages venant des cartes MAC **BC:81:81:42:9C:31** et **2A:AB:AC:27:D6:A7**, peut mettre à jour sa table SAT :

MAC	Port
2A:AB:AC:27:D6:A7	Port 1
BC:81:81:42:9C:31	Port 0

Par la suite, il saura sur quel port rediriger les messages destinés à ces deux adresses MAC. Un switch est un équipement de réseau de la couche 2 du modèle OSI, il ne sait pas lire les adresses IP : il ne travaille qu'avec les adresses MAC.

Message 3 : le ping est envoyé

No.: 3 / Date: 17:05:45.540

```

Réseau
├── Source:      BC:81:81:42:9C:31
├── Destination: 2A:AB:AC:27:D6:A7
├── Commentaire: 0x800
Internet
├── Source:      192.168.0.10
├── Destination: 192.168.0.11
├── Protocole:   ICMP
└── Commentaire: ICMP Echo Request (ping), TTL: 64, Seq.-Nr.: 1

```

Schématisons cette trame Ethernet (couche 2 du modèle OSI) :



Message 4 : le pong est retourné

No.: 4 / Date: 17:05:45.750

Réseau

Source: 2A:AB:AC:27:D6:A7
 Destination: BC:81:81:42:9C:31
 Commentaire: 0x800

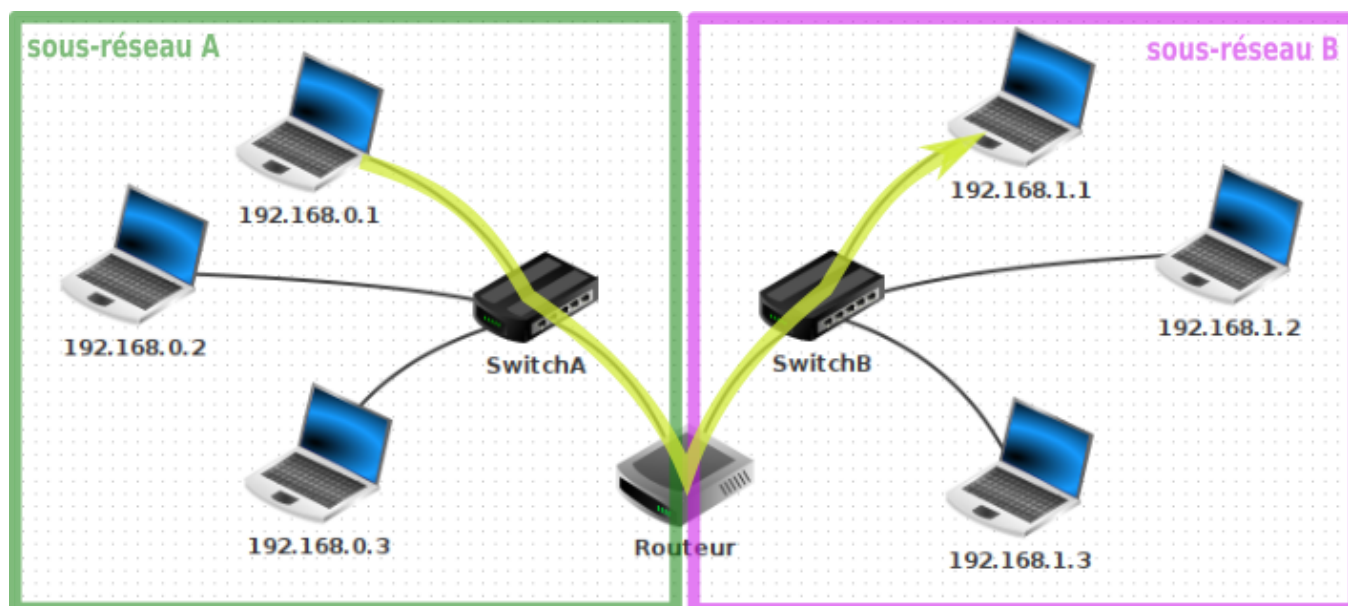
Internet

Source: 192.168.0.11
 Destination: 192.168.0.10
 Protocole: ICMP
 Commentaire: ICMP Echo Reply (pong), TTL: 64, Seq.-Nr.: 1



2. Ping à travers un routeur

Vous pouvez télécharger le fichier [ping_routeur.flx](#).



L'objectif est d'observer les différentes trames lors d'un ping entre :

- la machine 192.168.0.1 / 24 (adresse MAC F9:E1:D6:0B:29:03) et
- la machine 192.168.1.1 / 24 (adresse MAC D3:79:96:B8:5C:A4)

Le routeur est configuré ainsi :

- interface sur le réseau A :
 - IP : 192.168.0.254
 - MAC : 77:C2:22:C9:5C:E7
- interface sur le réseau B :

- IP : 192.168.1.254
- MAC : 66:E5:4E:7D:0B:B0

Étape 0 : le routeur signale sa présence

Lors de l'observation des messages reçus ou émis par la machine 192.168.0.1, on peut être intrigué par ce tout premier message reçu, émis par le routeur :

```
No.: 1 / Date: 18:20:45.824
Réseau
  Source: 77:C2:22:B9:5C:E7
  Destination: FF:FF:FF:FF:FF:FF
  Commentaire: 0x800
Internet
  Source: 192.168.0.254
  Destination: 255.255.255.255
  Protocole: IP
  Commentaire: Protocole:17, TTL: 1
Transport
  Source: 521
  Destination: 520
  Protocole: UDP
Application
  Commentaire:
    192.168.0.254
    192.168.1.254
    16
    75000
    192.168.1.0 255.255.255.0 0
```

On peut y distinguer les 4 couches du modèle Internet. Le routeur, par ce message distribué à tous les éléments du sous-réseau A (il envoie un message équivalent sur son sous-réseau B), déclare sa présence, et le fait qu'il possède deux interfaces, une pour chaque réseau. Il se positionne ainsi comme une passerelle : «c'est par moi qu'il faudra passer si vous voulez sortir de votre sous-réseau». Dans cette trame envoyée figure son adresse MAC, de sorte que tous les membres de son sous-réseau pourront donc communiquer avec lui.

Étape 1 : de 192.168.0.1 vers le routeur

La machine 192.168.0.1 / 24 calcule que la machine 192.168.1.1 / 24 avec laquelle elle veut communiquer n'est **pas** dans son sous-réseau (voir [ce cours](#)). Elle va donc envoyer son message à sa passerelle, qui est l'adresse du routeur dans son sous-réseau.

Cette première trame est :

MAC DESTINATION	MAC SOURCE	IP SOURCE	IP DESTINATION	ICMP ping	TTL
77:C2:22:C9:5C:E7	F9:E1:D6:0B:29:03	192.168.0.1	192.168.1.1		64

Étape 2 : le routeur décapsule la trame

Le routeur est un équipement de réseau de couche 3 (couche réseau). Il doit observer le contenu du paquet IP (sans remonter jusqu'au contenu du message) pour savoir, suivant le procédé de **roulage** (voir cours de Terminale), où acheminer ce paquet.

Dans notre cas, l'adresse IP **192.168.1.1** de destination lui est accessible : elle fait partie de son sous-réseau B.

Le routeur va modifier la valeur du TTL (Time To Live), en la décrémentant de 1. Si, après de multiples routages, cette valeur devenait égale à 0, ce paquet serait détruit. Ceci a pour but d'éviter l'encombrement des réseaux avec des paquets ne trouvant pas leur destination.

Remarque : dans notre cas, le routeur va laisser intacte l'adresse IP Source. Ce n'est pas toujours le cas. Dans le cas classique de la box qui relie votre domicile à internet, le routeur contenu dans celle-ci va remplacer l'adresse locale de votre ordinateur ou smartphone (ex **192.168.0.26**) par son IP publique (celle apparaissant sur whatsmyip.com, par exemple). Elle effectue ce qu'on appelle une translation d'adresse (NAT). Pourquoi ? Parce que sinon la réponse du serveur distant que vous interrogez serait envoyée sur une adresse locale (votre adresse **192.168.0.26**), qui est introuvable depuis un réseau extérieur. Il faut donc remplacer toutes les adresses locales par l'IP publique de votre box. Pour éviter que la réponse du serveur web que vous avez interrogé ne soit affichée sur l'ordinateur de vos parents, le routeur affecte des ports différents à chaque machine de son sous-réseau. Ce port est inclus dans le message transmis au serveur, et il l'est aussi dans sa réponse : le routeur peut donc rediriger le trafic vers la bonne machine du sous-réseau.

Le routeur va ré-encapsuler le paquet IP modifié, et créer une nouvelle trame Ethernet en modifiant :

- l'adresse MAC source : il va mettre l'adresse MAC de son interface dans le sous-réseau B.
- l'adresse MAC de destination : il va mettre l'adresse MAC de **192.168.1.1** (qu'il aura peut-être récupérée au préalable par le protocole ARP)

Cette deuxième trame est donc :



On peut observer dans Filius cette trame, en se positionnant sur l'interface **192.168.1.254** du routeur, ou sur **192.168.1.1** :

No.: 4 / Date: 18:21:03.179

Réseau

Source: 66:E5:4E:7D:0B:B0
Destination: D3:79:96:B8:5C:A4
Commentaire: 0x800

Internet

Source: 192.168.0.1
Destination: 192.168.1.1
Protocole: ICMP
Commentaire: ICMP Echo Request (ping), TTL: 63, Seq.-Nr.: 1

En suivant le même principe, la machine **192.168.1.1** pourra envoyer son *pong*.

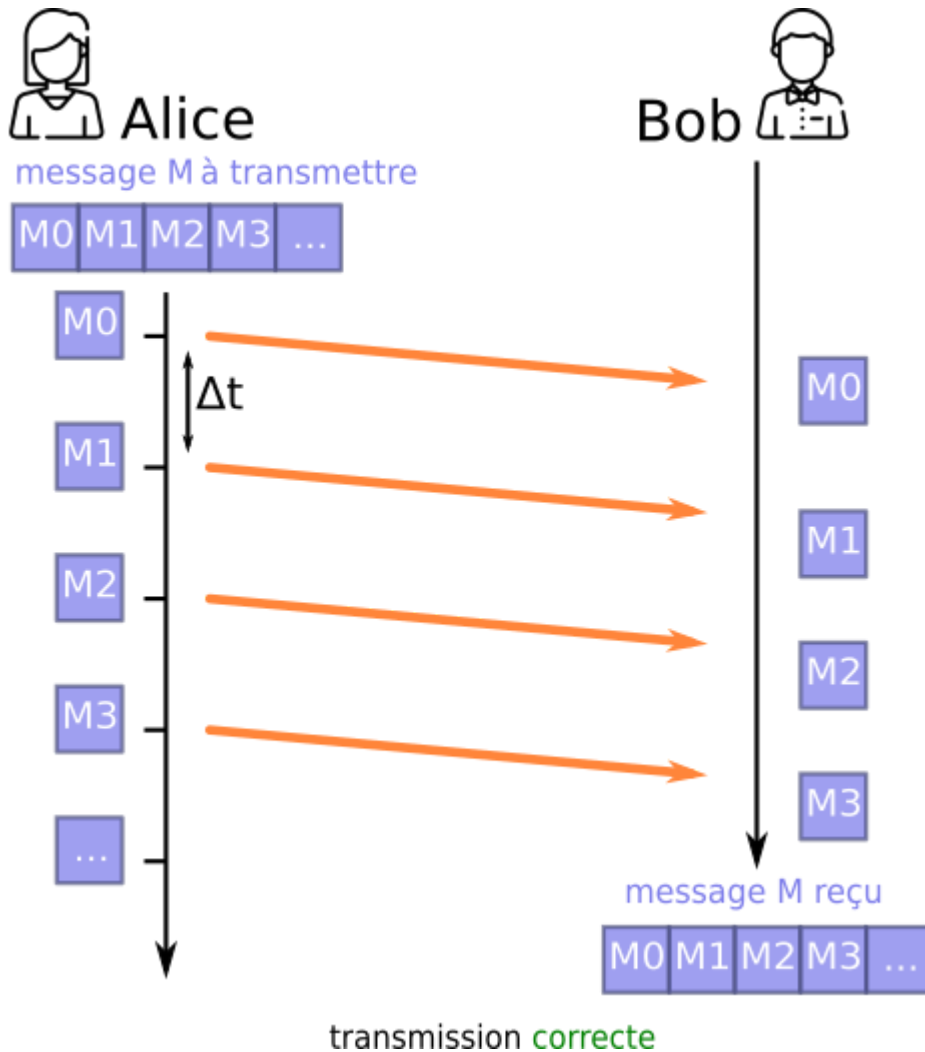
III. Protocole du bit alterné

Ce protocole est un exemple simple de fiabilisation du transfert de données.

1. Contexte

- Alice veut envoyer à Bob un message M, qu'elle a prédécoupé en sous-messages M0, M1, M2,...
- Alice envoie ses sous-messages à une cadence Δt fixée (en pratique, les sous-messages partent quand leur acquittement a été reçu ou qu'on a attendu celui-ci trop longtemps : on parle alors de *timeout*)

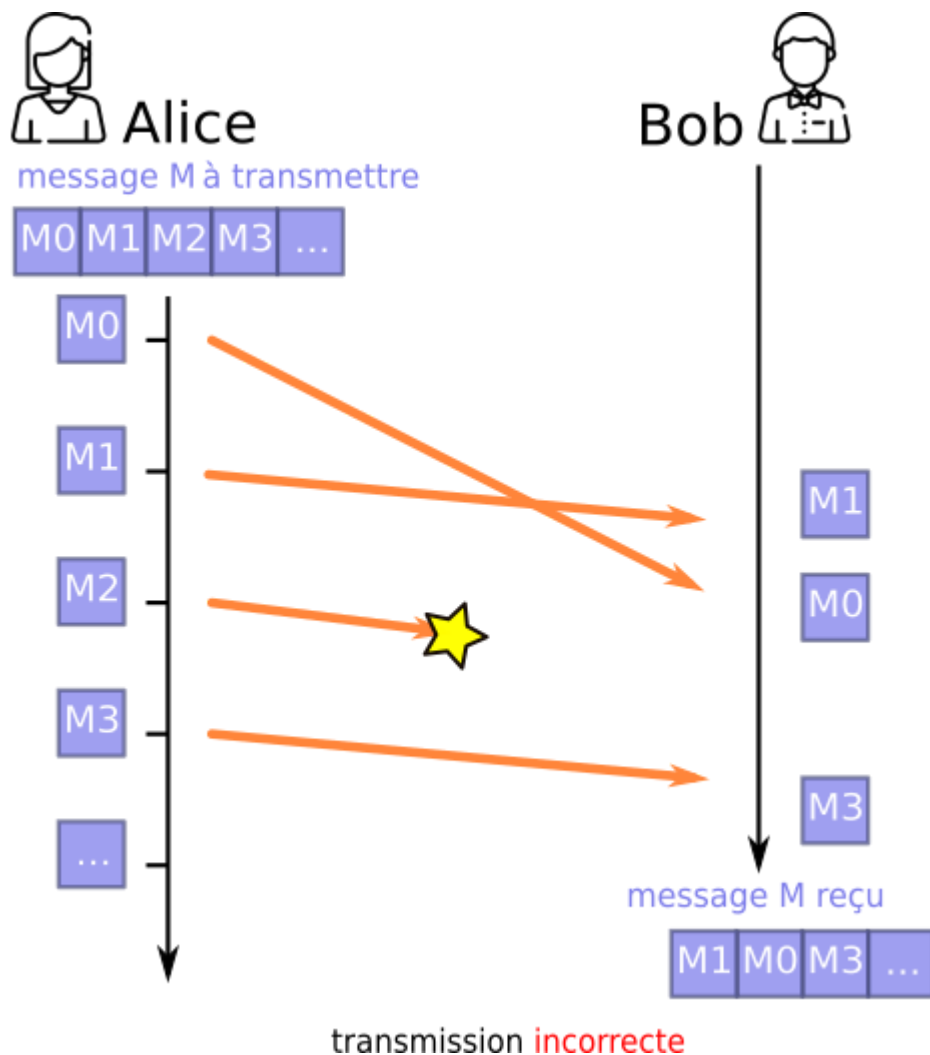
2. Situation idéale



Dans cette situation, les sous-messages arrivent tous à destination dans le bon ordre. La transmission est correcte.

3. Situation réelle

Mais parfois, les choses ne se passent pas toujours aussi bien. Car si on maîtrise parfaitement le timing de l'envoi des sous-messages d'Alice, on ne sait pas combien de temps vont mettre ces sous-messages pour arriver, ni même (attention je vais passer dans un tunnel) s'ils ne vont pas être détruits en route.



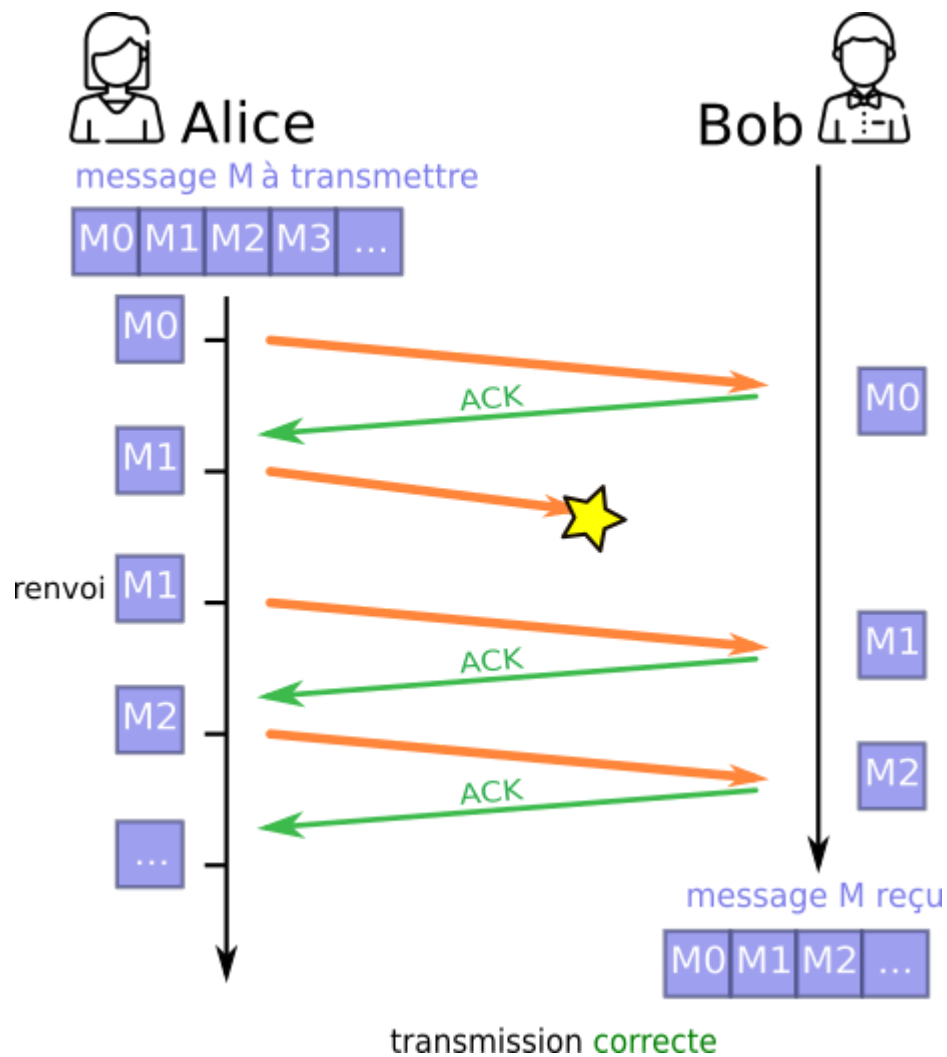
Le sous-message M0 est arrivé après le M1, le message M2 n'est jamais arrivé...

Que faire ?

Écartons l'idée de numéroté les sous-messages, afin que Bob puisse remettre dans l'ordre les messages arrivés, ou même redemander spécifiquement des sous-messages perdus. C'est ce que réalise le protocole TCP (couche 4 — transport), c'est très efficace, mais cher en ressources. Essayons de trouver une solution plus basique.

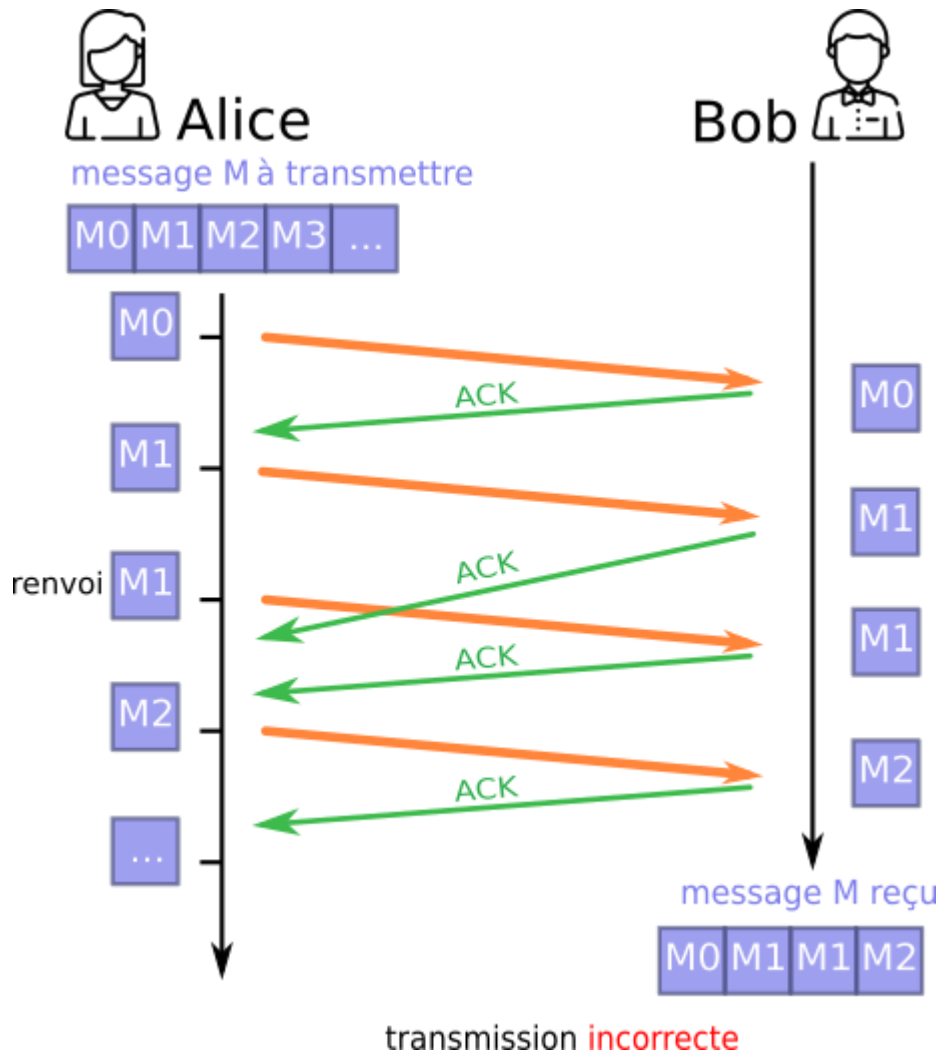
3. Solution naïve...

Pourquoi ne pas demander à Bob d'envoyer un signal pour dire à Alice qu'il vient bien de recevoir son sous-message ? Nous appellerons ce signal ACK (comme *acknowledgement*, traduisible par «accusé de réception»). Ce signal ACK permettra à Alice de renvoyer un message qu'elle considérera comme perdu :



N'ayant pas reçu le ACK consécutif à son message *M1*, Alice suppose (avec raison) que ce message n'est pas parvenu jusqu'à Bob, et donc renvoie le message *M1*.

4. Mais peu efficace...



Le deuxième ACK de Bob a mis trop de temps pour arriver (ou s'est perdu en route) et donc Alice a supposé que son sous-message M1 n'était pas arrivé. Elle l'a donc renvoyé, et Bob se retrouve avec deux fois le sous-message M1. La transmission est incorrecte. En faisant transiter un message entre Bob et Alice, nous multiplions par 2 la probabilité que des problèmes techniques de transmission interviennent. Et pour l'instant rien ne nous permet de les détecter.

5. Bob prend le contrôle

Bob va maintenant intégrer une méthode de validation du sous-message reçu. Il pourra décider de le garder ou de l'écarter. Le but est d'éviter les doublons.

Pour réaliser ceci, Alice va rajouter à chacun de ses sous-messages un bit de contrôle, que nous appellerons FLAG (drapeau). Au départ, ce FLAG vaut 0. Quand Bob reçoit un FLAG, il renvoie un ACK **égal au FLAG reçu**.

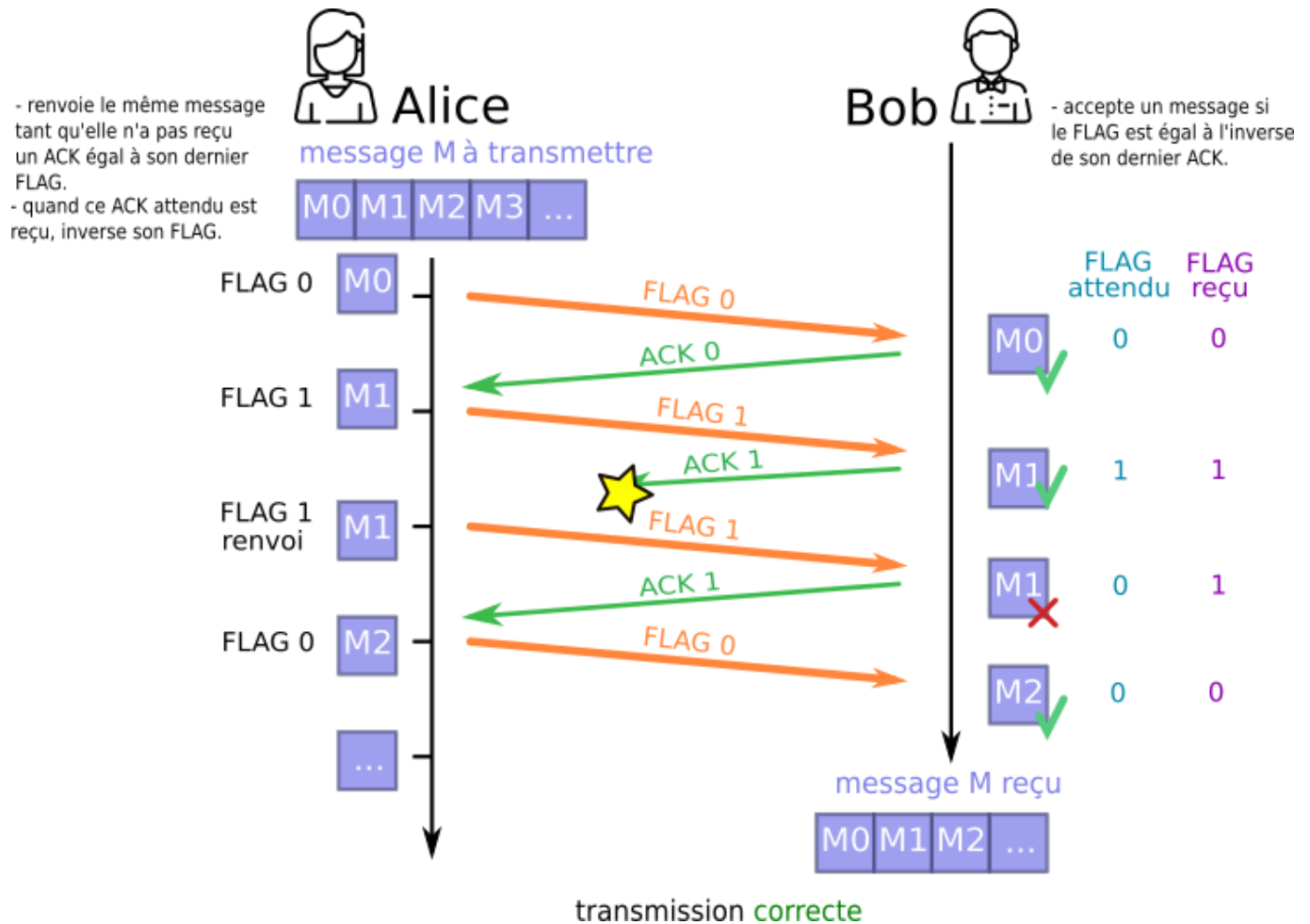
Alice va attendre ce ACK contenant le même bit que son dernier FLAG envoyé :

- tant qu'elle ne l'aura pas reçu, elle continuera à envoyer **le même sous-message, avec le même FLAG**.
- dès qu'elle l'a reçu, elle peut envoyer un nouveau sous-message en **inversant** («alternant») **le bit de son dernier FLAG** (d'où le nom de ce protocole).

Bob, de son côté, va contrôler la validité de ce qu'il reçoit : il ne gardera que **les sous-messages dont le FLAG est égal à son dernier ACK**. C'est cette méthode qui lui permettra d'écarter les doublons.

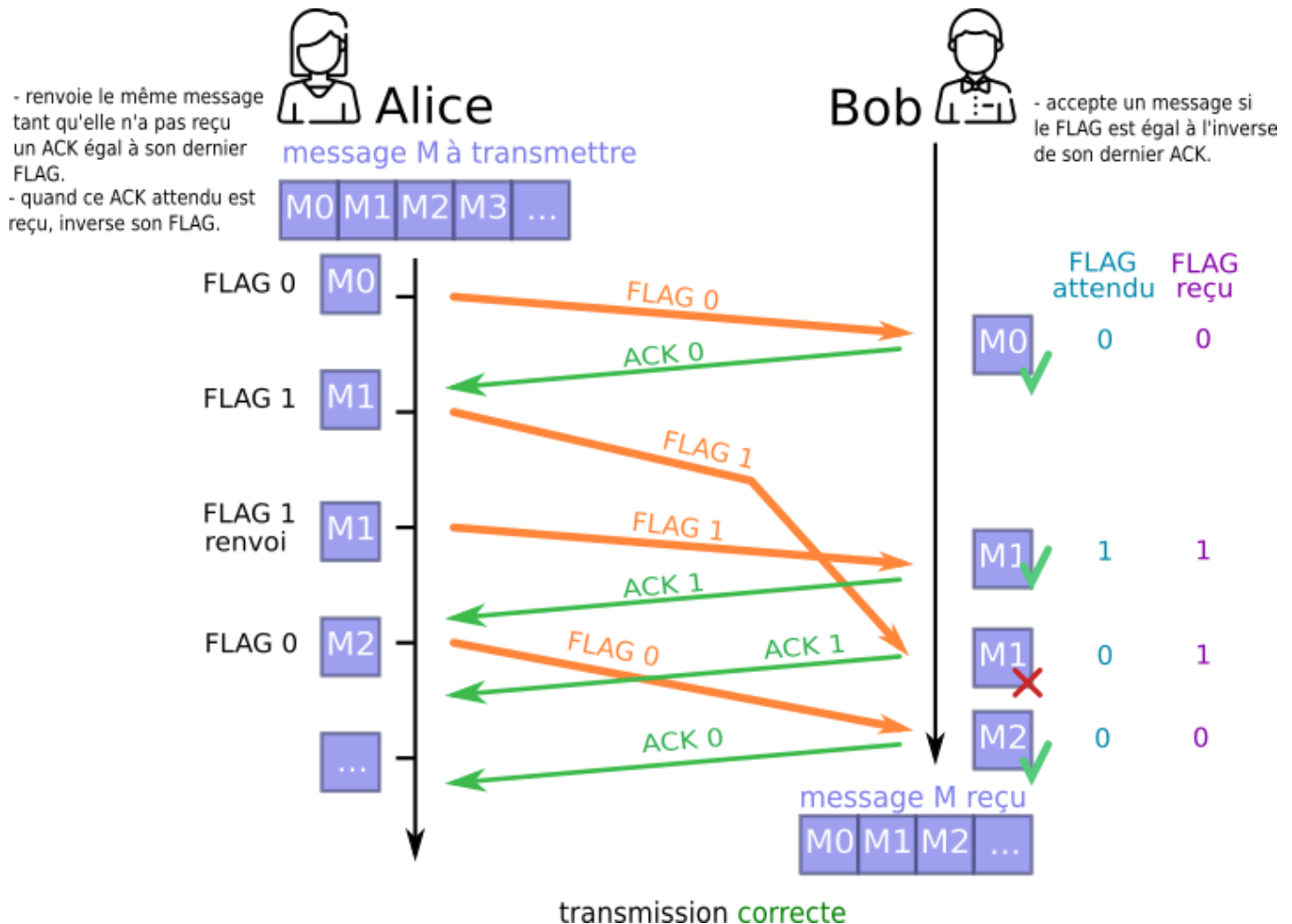
5.1 Cas où le sous-message est perdu





Le protocole a bien détecté le doublon du sous-message M1.

5.3 Cas où un sous-message est en retard



Le protocole a bien détecté le doublon du sous-message M1... mais que se passerait-il si notre premier sous-message M1 était *encore plus* en retard ?

6. Conclusion

Le protocole du bit alterné a longtemps été utilisé au sein de la couche 2 du modèle OSI (distribution des trames Ethernet). Simple et léger, il peut toutefois être facilement mis en défaut, ce qui explique qu'il ait été remplacé par des protocoles plus performants.

Bibliographie

- Numérique et Sciences Informatiques, 1re, T. BALABONSKI, S. CONCHON, J.-C. FILLIATRE, K. NGUYEN, éditions ELLIPSES.
- Prépabac NSI 1ère, C.ADOBET, G.CONNAN, G. ROZSAVOLGYI, L.SIGNAC, éditions Hatier.

