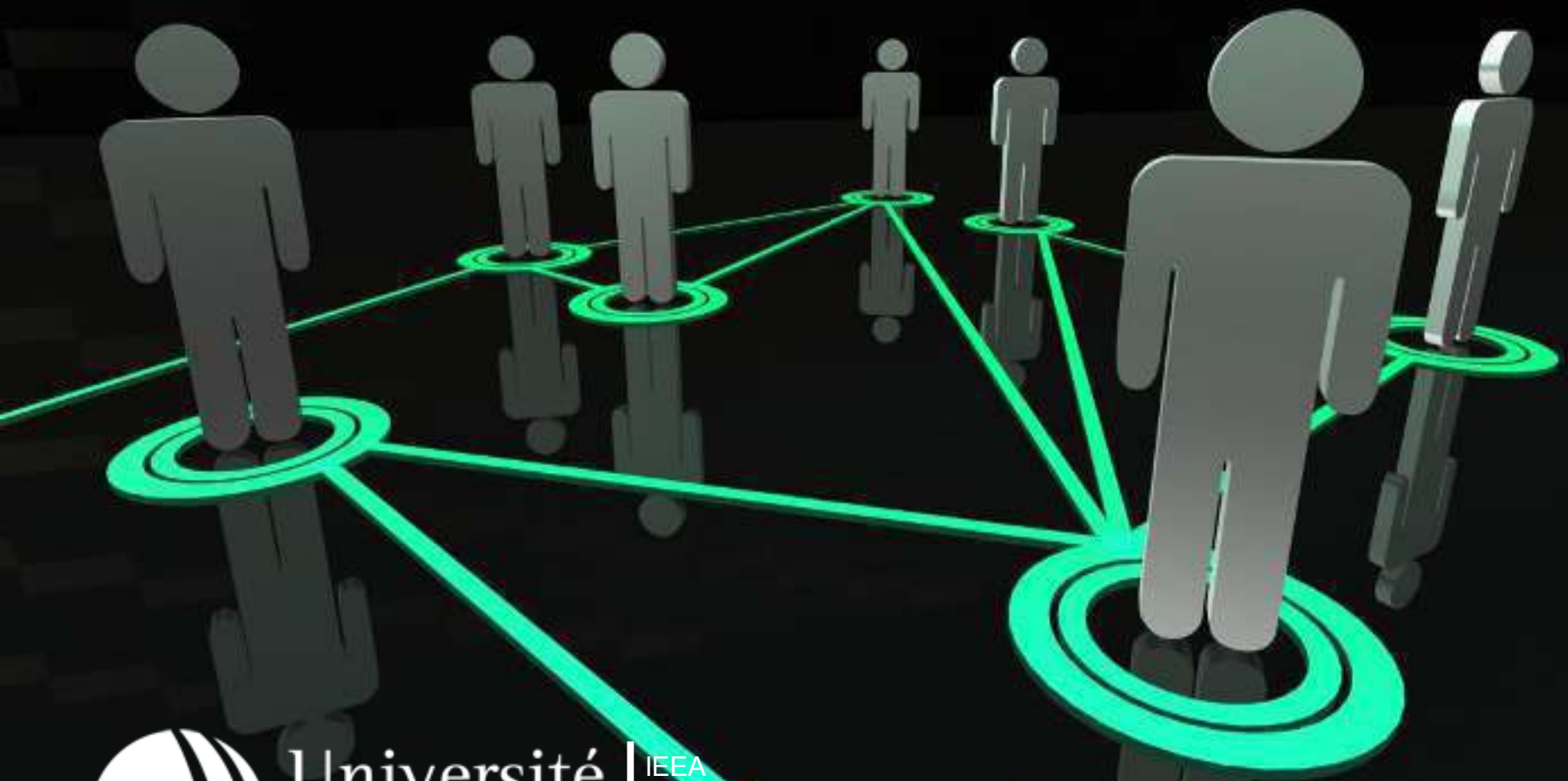
Réseaux et routage



Université Lille1
Sciences et Technologies

IEEA
Informatique
Electronique
Electrotechnique
Automatique

Gilles Grimaud
Laurent Noé

Le routage

La couche routage a pour objectif de créer des liens virtuels entre toutes les machines qui souscrivent au même réseau. Le lien virtuel est en fait le résultat d'un ensemble de retransmissions de lien réel en lien réel.

Adressage des machines - exemple de l'OSI (*NSAP*) -

[RFC 1237, RFC 1629]

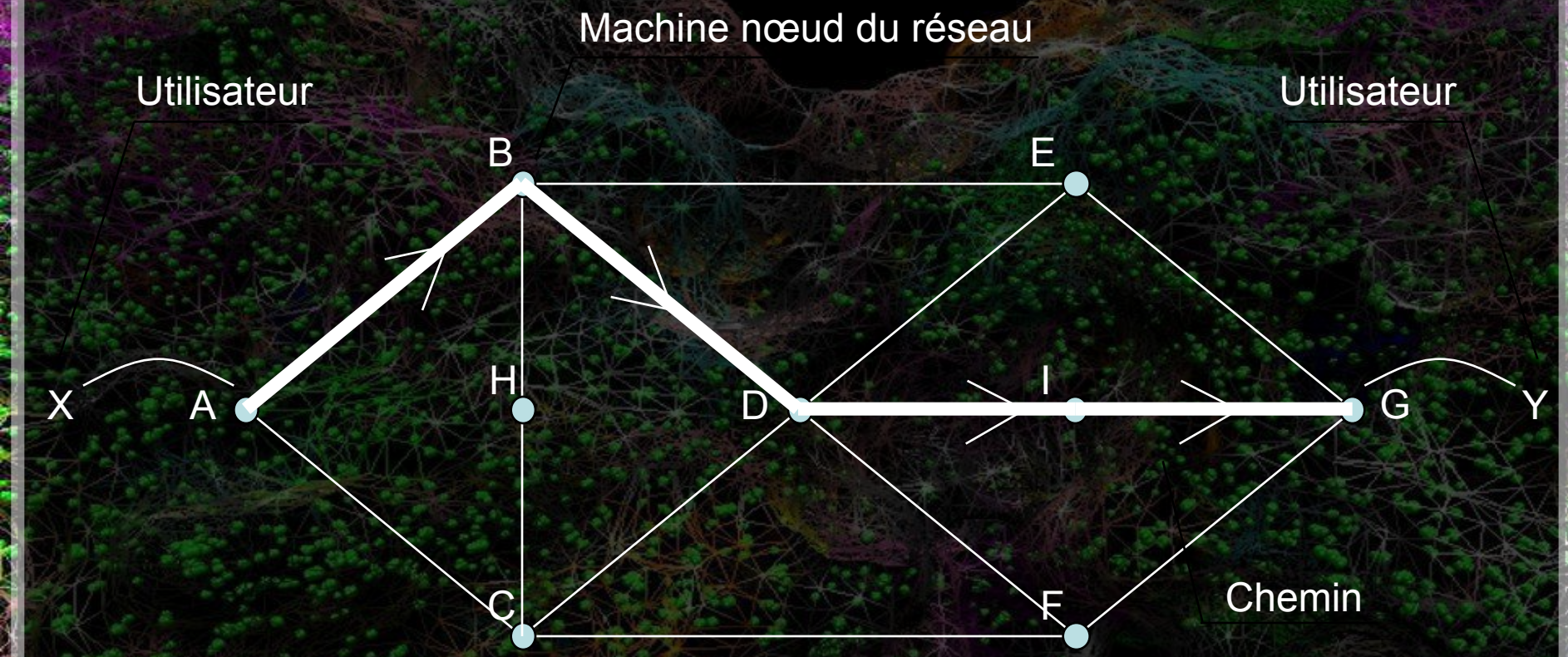
G.D.P. : *Global Domain Part*
(Définit la forme des adresses présentes dans D.S.P…)
A.F.I. : *Authority and Format Identifier*
I.D.I. : *Initial Domain Identifier*
D.S.P. : *Domain Specific Part*
(L'adresse selon le mécanisme d'adressage du réseau visé…)

Analogie avec le téléphone :

(X)Y ZZ ZZ ZZ ZZ

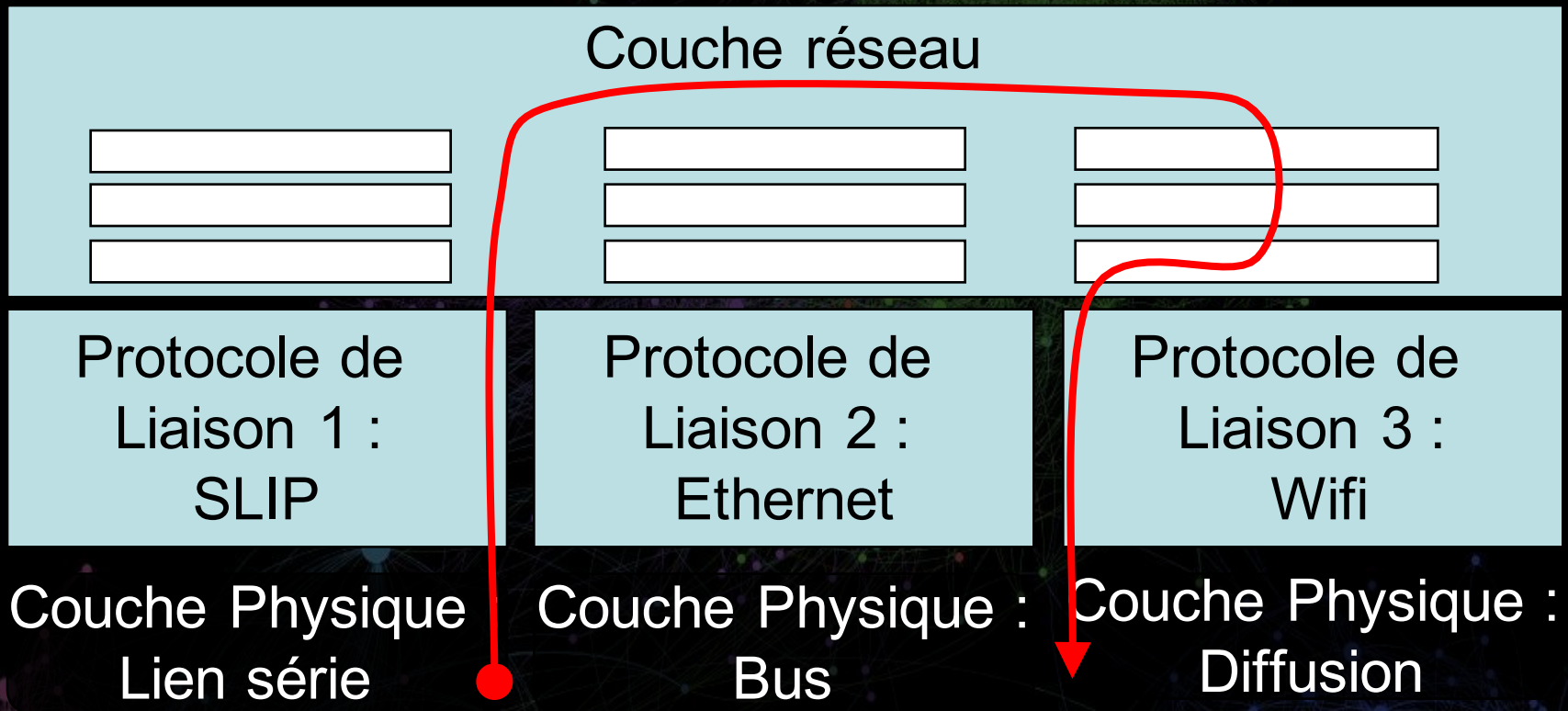
Identifiant de numéro à l'intérieur de la région
Région d'appel
Identifiant de fournisseur de réseau global

Structure d'un réseau



Les machines utilisateurs et les machines nœuds du réseau peuvent être confondues.

Mécanisme de relais d'information



Le temps moyen T_{moy} de traversée d'une file d'attente est donné par :

$T_{moy} = 1/(\mu C - \lambda)$,

avec $1/\mu$ la taille moyenne d'un paquet (en bits/?) , C débit sortant de la ligne (en bits/s) et λ quantité moyenne de paquets entrant chaque seconde

c.f. *théorie des files d'attente « M/M/1»*

Routage connecté *vs* non-connecté

préoccupation	Service avec connexion	Service sans connexion
Initialisation :	Nécessaire.	Impossible.
@ du destinataire :	Nécessaire uniquement à l'installation.	Nécessaire dans chaque paquet.
Séquencement des paquets:	Garanti.	Non Garanti.
Contrôle d'erreur :	À la charge du réseau.	À la charge des utilisateurs.
Contrôle de flux :	À la charge du réseau.	A la charge des utilisateurs.
Possibilité de négociations :	Oui	Non

Table de routage

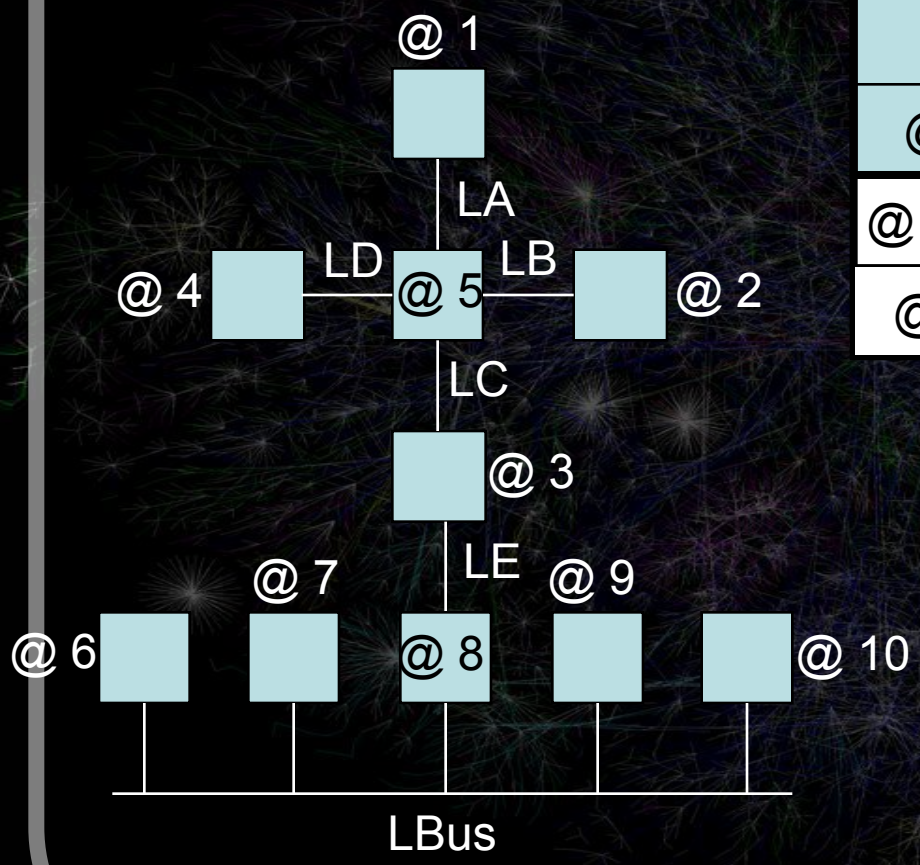
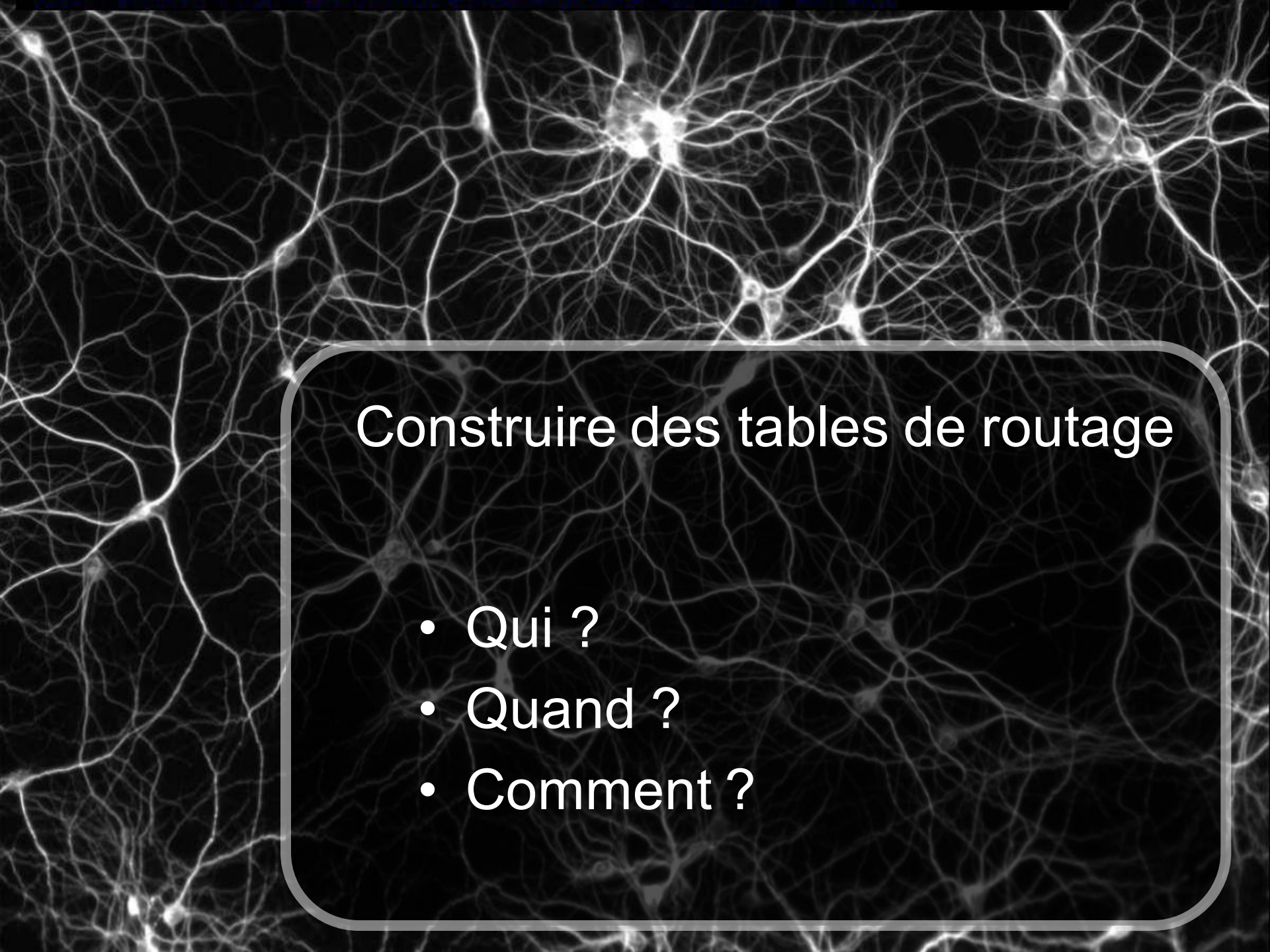


Table @3	
@ des	Liaison
@ 1-2;4-5	LC
@ 6-10	LE

Table @5	
@ des	Liaison
@ 1	LA
@ 2	LB
@ 3;6-10	LC
@ 4	LD

Table @8	
@ des	Liaison
@ 1-4	LE
@ 6-7;9-10	LBus



Construire des tables de routage

- Qui ?
- Quand ?
- Comment ?

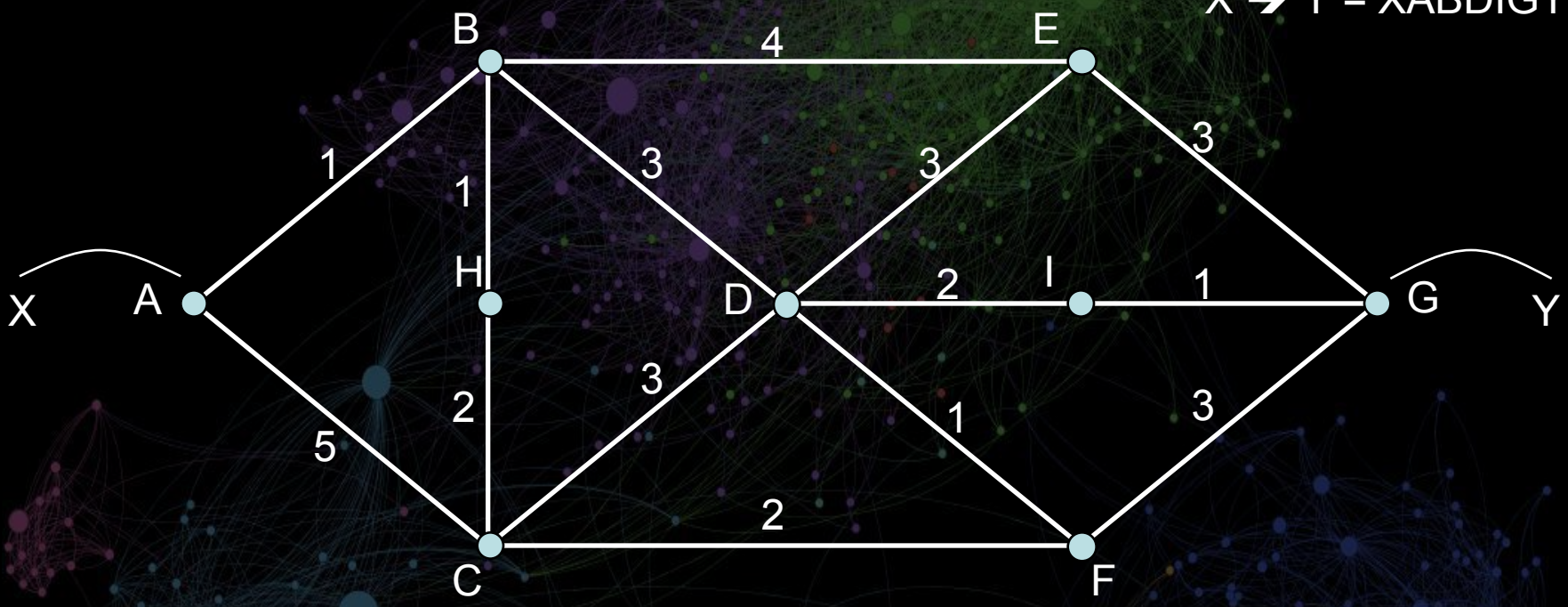
Propriétés souhaitables pour un algorithme de routage :

- Exactitude ;
- Simplicité ;
- Robustesse (aux MaJ & Défaillances des Machines) ;
- Stabilité (garantie de convergence) ;
- Justice (vis-à-vis des usagers) ;
- Optimisation (minimiser le temps de traversée, mais aussi, maximiser le flux de transmission) .

(Propriétés parfois contradictoires)

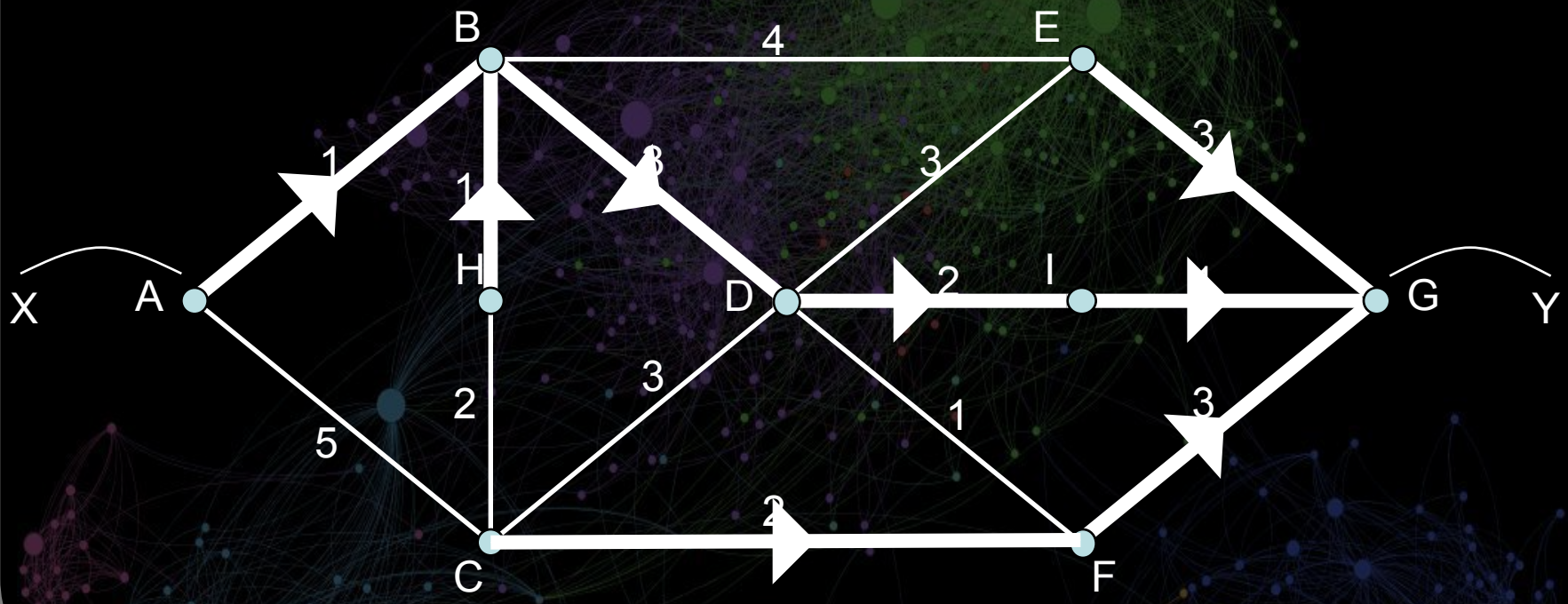
Algorithme du plus court chemin

Chemin retenu :
X ➔ Y = XABDIGY



A partir du graphe du réseau, déterminer une valeur pour chaque liaison/arc (1 fois pour toutes ou périodiquement avec des messages de contrôle de la couche liaison par exemple), dans l'unité de mesure choisie (distance, temps de transmission, charge supportée,…).

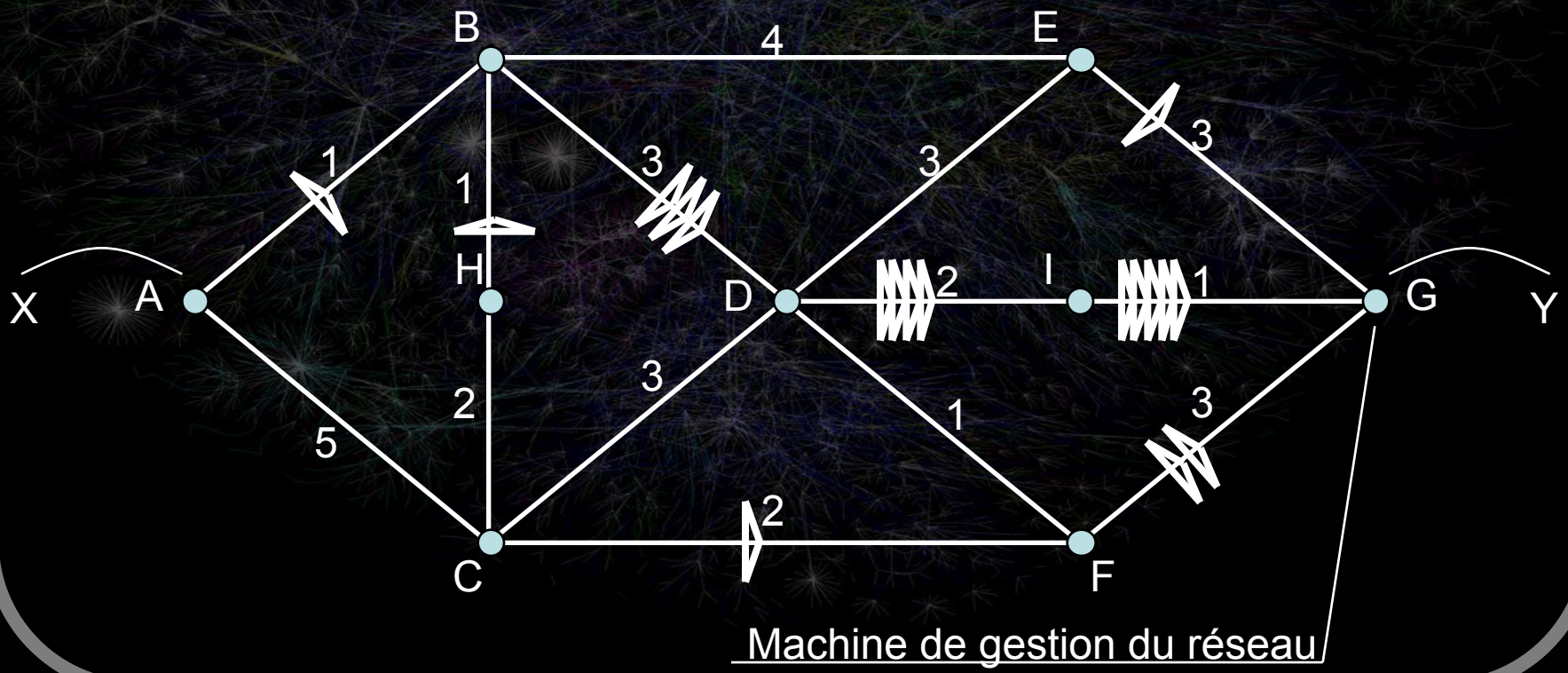
Arbre collecteur



Algorithmes centralisés

Maintenir l'état des tables de routages : Qui, Quand et Comment ?

- manuel, automatique centralisé, automatique décentralisé ;
- sur décision humaine, périodique, apériodique ;
- décision humaine, algorithmes centralisés, algorithmes distribués.



routage multi-chemin

Objectif : disposer de chemins de substitution,

- en cas de perturbation de certaines liaisons ;
- en cas de panne de certaines machines nœuds.

Stratégie : Dans les tables de routage de chaque machine nœud indiquer n liaisons possibles pour atteindre la machine cible. Chaque liaison pourra être pondérée avec un poids qui permet alors de définir la probabilité que les paquets soient routés par chacune des liaison possibles pour sa destination.

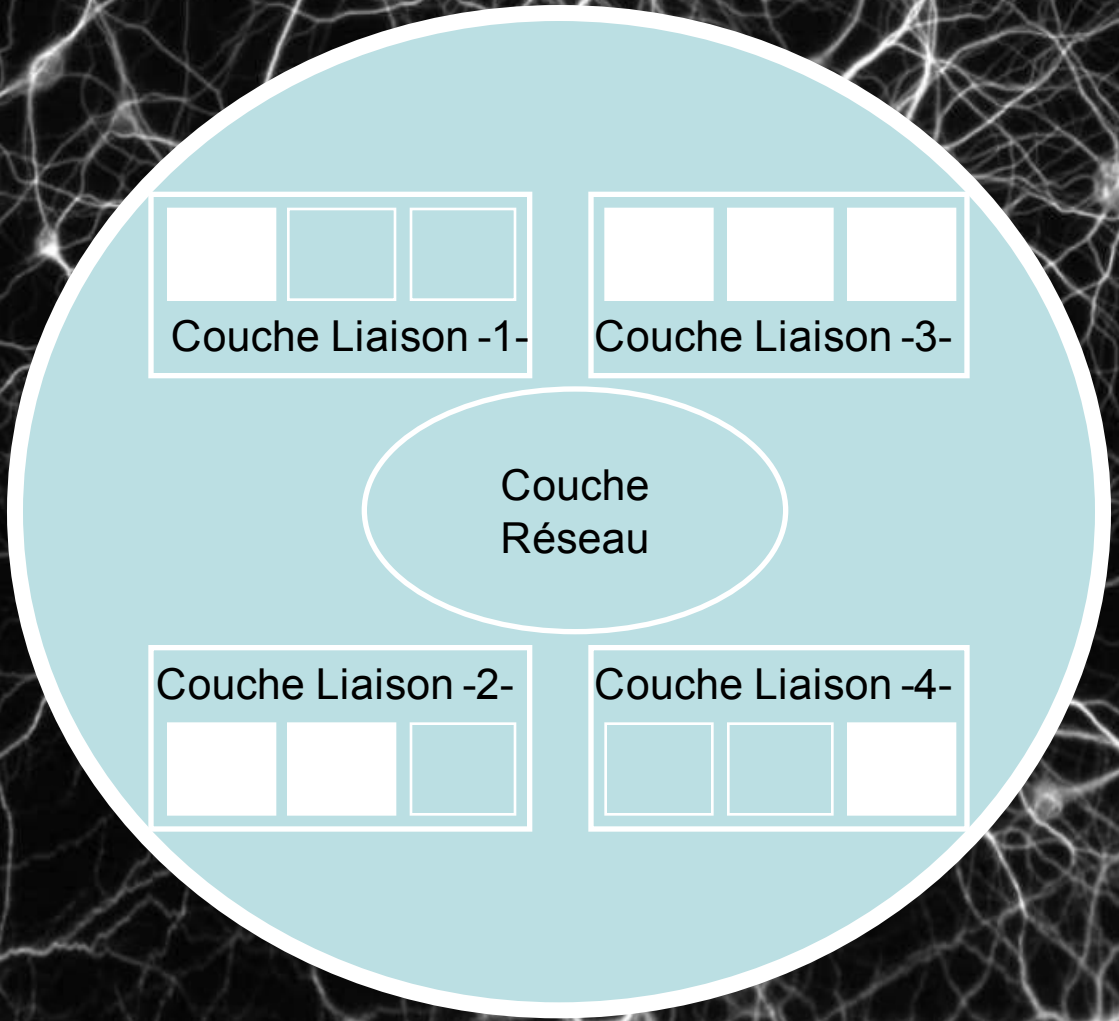
Exemple :

```
Table de routage de A :
@dest. : A liaisons : {(-,1)}
@dest. : B liaisons : {(→B;0,95), (→C;0,05)}
@dest. : C liaisons : {(→C;0,95), (→C;0,05)}
@dest. : * liaisons : {(→B;0,77), (→C;0,23)}
```

Id de liaison

Probabilité d'usage

Algorithmes de routage local



Algorithmes de routage par inondation

Tout paquet reçu est réémis sur toutes les liaisons.

➔ Génération d'un nombre infini de duplication de paquets!

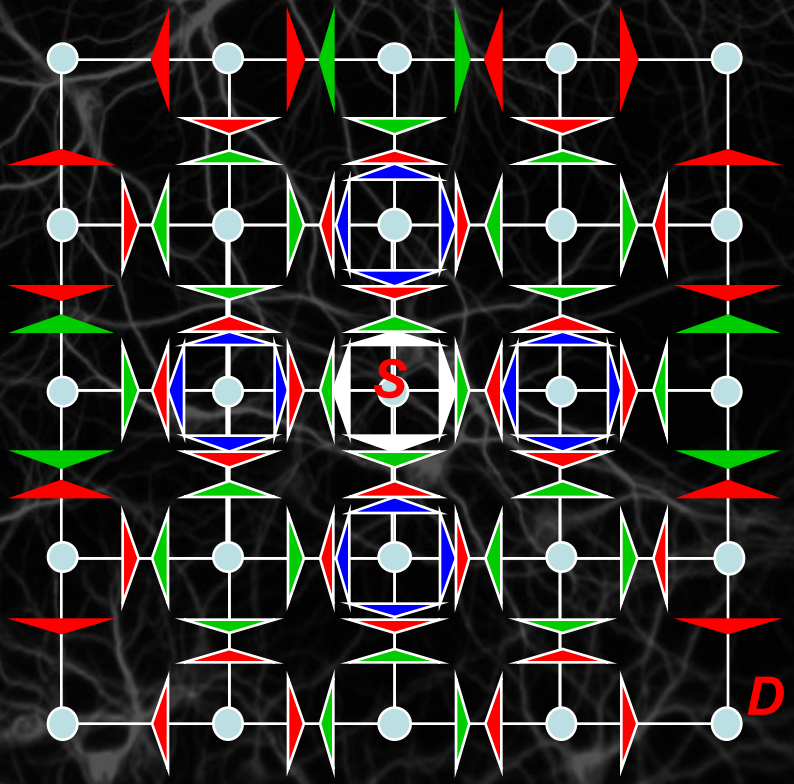
Limiter le processus en utilisant par exemple un compteur de durée de vie du paquet. Cet algorithme ne nécessite aucune connaissance minimale du réseau. (la distance max entre deux nœud).

Paquet initial Pack1
(transmit sur les 4 liaisons de S)

Paquets générés à l'étape 1
(1ere retransmission du Pack1)

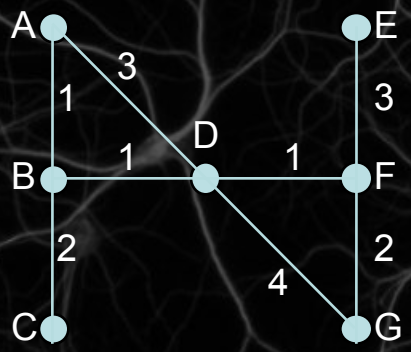
Paquets générés à l'étape 2
(2ere retransmission du Pack1)

Paquets générés à l'étape 3
(3ere retransmission du Pack1)



Algorithmes de routage distribué

Deux familles :



Routage à *Vecteur de Distances* :

(*algorithme de Bellman-Ford « distribué »*)

(1) échanger sa tables de routage *partielles* avec ses voisins :

(2) recalculer ses routes en fonction des tables reçues des voisins.

```
D:
Dst  Lien  cout
 A   ->A    3
 B   ->B    1
 C   -   ?
 D    -     0
 E   -   ?
 F   ->F    1
 G   ->G    4
```

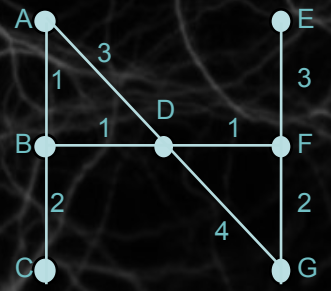
Routage à *Etats de lien* :

(1) échanger ses liens directs (*par inondation)*

But : Apprendre la topologie du réseau :

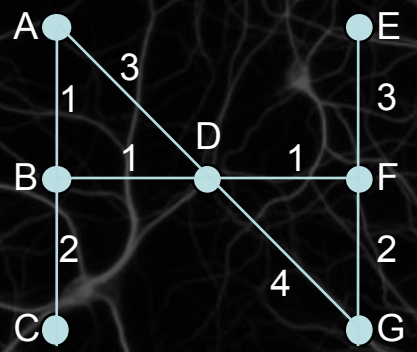
(2) algorithme de *Dijkstra « local ».*

D:



Algorithme de routage *vecteur de distances*

Etape #0



A:

Dst	Lien	cout
A	-	0
B		
C		
D		
E		
F		
G		

B:

Dst	Lien	cout
A		
B	-	0
C		
D		
E		
F		
G		

C:

Dst	Lien	cout
A		
B		
C	-	0
D		
E		
F		
G		

D:

Dst	Lien	cout
A		
B		
C		
D	-	0
E		
F		
G		

E:

Dst	Lien	cout
A		
B		
C		
D		
E	-	0
F		
G		

F:

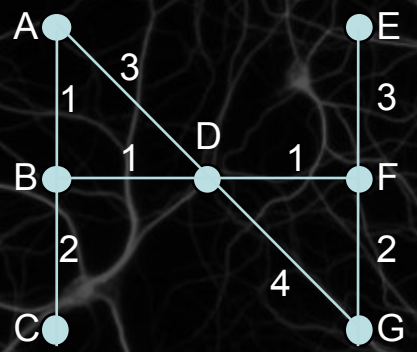
Dst	Lien	cout
A		
B		
C		
D		
E		
F	-	0
G		

G:

Dst	Lien	cout
A		
B		
C		
D		
E		
F		
G	-	0

Algorithme de routage *vecteur de distances*

Etape #1



A:

Dst	Lien	cout
A	-	0
B	->B	1
C		
D	->D	3
E		
F		
G		

B:

Dst	Lien	cout
A	->A	1
B	-	0
C	->C	2
D	->D	1
E		
F		
G		

C:

Dst	Lien	cout
A		
B	->B	1
C	-	0
D		
E		
F		
G		

D:

Dst	Lien	cout
A	->A	3
B	->B	1
C		
D	-	0
E		
F	->F	1
G	->G	4

E:

Dst	Lien	cout
A		
B		
C		
D		
E	-	0
F	->F	3
G		

F:

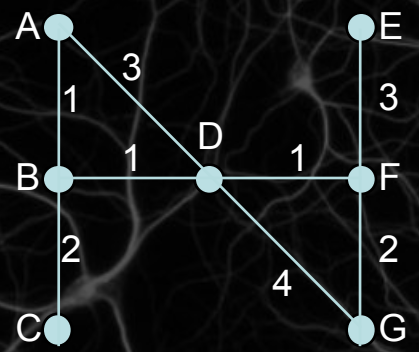
Dst	Lien	cout
A		
B		
C		
D	->D	1
E	->E	3
F	-	0
G	->G	2

G:

Dst	Lien	cout
A		
B		
C		
D	->D	4
E		
F	->F	2
G	-	0

Algorithme de routage *vecteur de distances*

Etape #2



A:

Dst	Lien	cout
A	-	0
B	->B	1
C	->B	3
D	->B	2
E		
F	->D	4
G	->D	7

B:

Dst	Lien	cout
A	->A	1
B	-	0
C	->C	2
D	->D	1
E		
F	->D	4
G	->D	6

C:

Dst	Lien	cout
A	->B	3
B	->B	1
C	-	0
D	->B	3
E		
F		
G		

D:

Dst	Lien	cout
A	->B	2
B	->B	1
C	->B	3
D	-	0
E	->F	4
F	->F	1
G	->F	3

E:

Dst	Lien	cout
A		
B		
C		
D	->F	4
E	-	0
F	->F	3
G	->F	5

F:

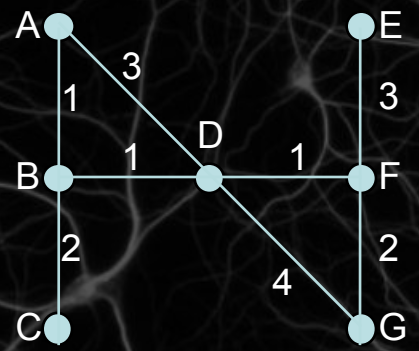
Dst	Lien	cout
A	->D	4
B	->D	2
C		
D	->D	1
E	->E	3
F	-	0
G	->G	2

G:

Dst	Lien	cout
A	->D	8
B	->D	5
C		
D	->D	3
E	->F	5
F	->F	2
G	-	0

Algorithme de routage *vecteur de distances*

Etape #3



A:

Dst	Lien	cout
A	-	0
B	->B	1
C	->B	3
D	->B	2
E	->D	7
F	->B	3
G	->B	6

B:

Dst	Lien	cout
A	->A	1
B	-	0
C	->C	2
D	->D	1
E	->D	5
F	->D	4
G	->D	4

C:

Dst	Lien	cout
A	->B	3
B	->B	1
C	-	0
D	->B	3
E		
F	->B	5
G	->B	7

D:

Dst	Lien	cout
A	->B	2
B	->B	1
C	->B	3
D	-	0
E	->F	4
F	->F	1
G	->F	3

E:

Dst	Lien	cout
A	->F	7
B	->F	5
C		
D	->F	4
E	-	0
F	->F	3
G	->F	5

F:

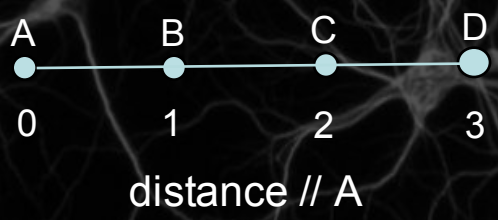
Dst	Lien	cout
A	->D	3
B	->D	2
C	->D	3
D	->D	1
E	->E	3
F	-	0
G	->G	2

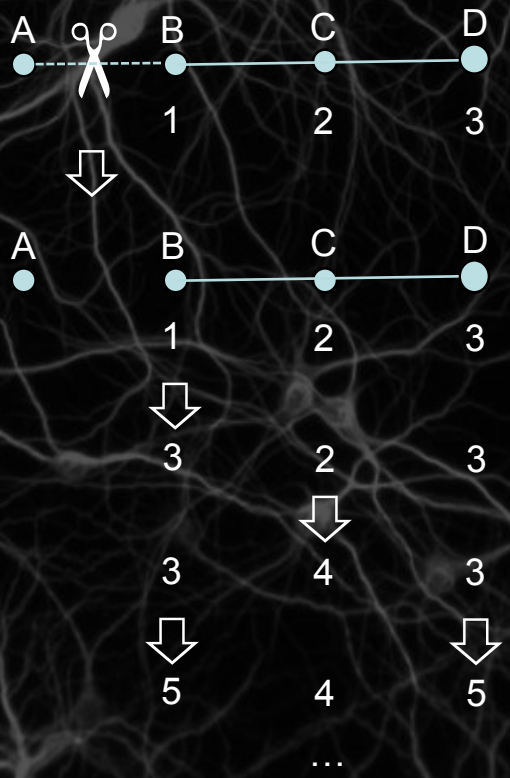
G:

Dst	Lien	cout
A	->F	5
B	->F	4
C	->D	7
D	->F	3
E	->F	5
F	->F	2
G	-	0

Algorithme de routage *vecteur de distances*

Problème : le *Compte-à-l'infini* …

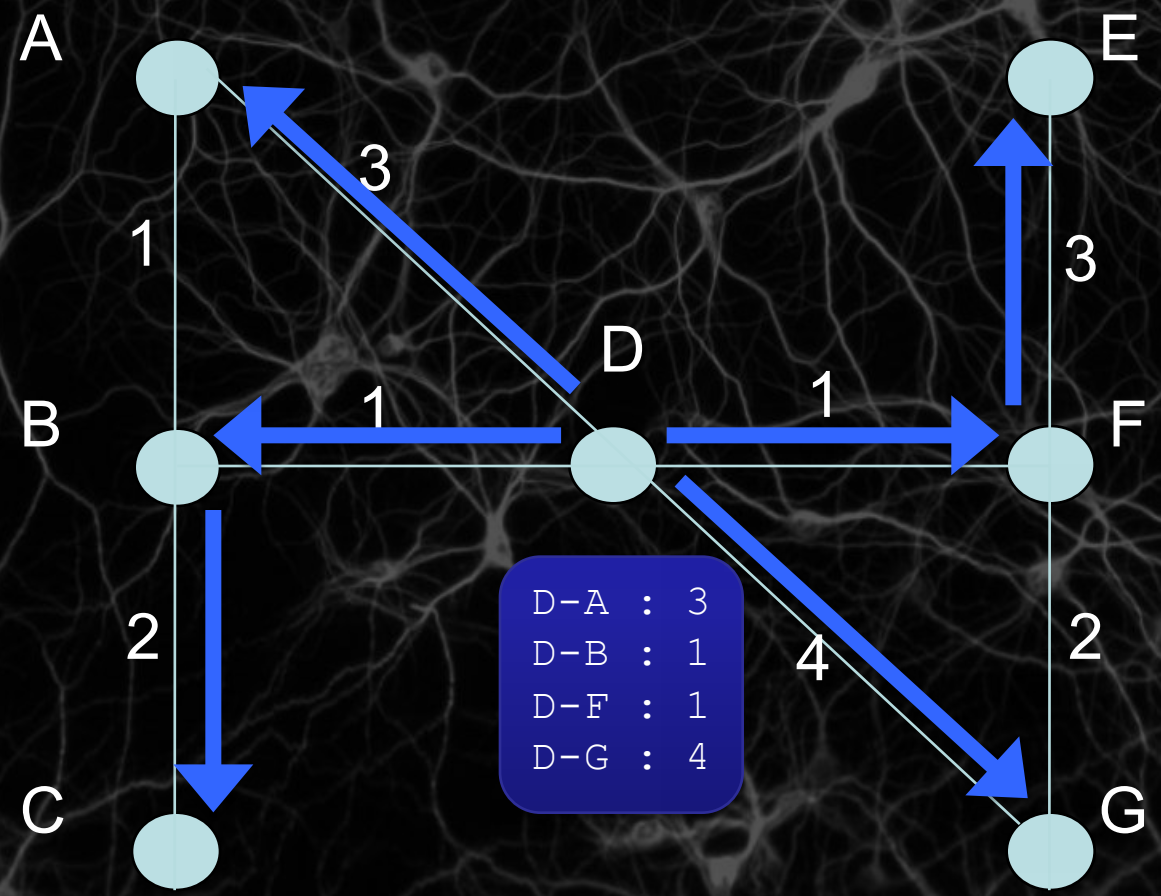




Extrait de ***Réseaux*** *de* ***Andrew Tanenbaum***

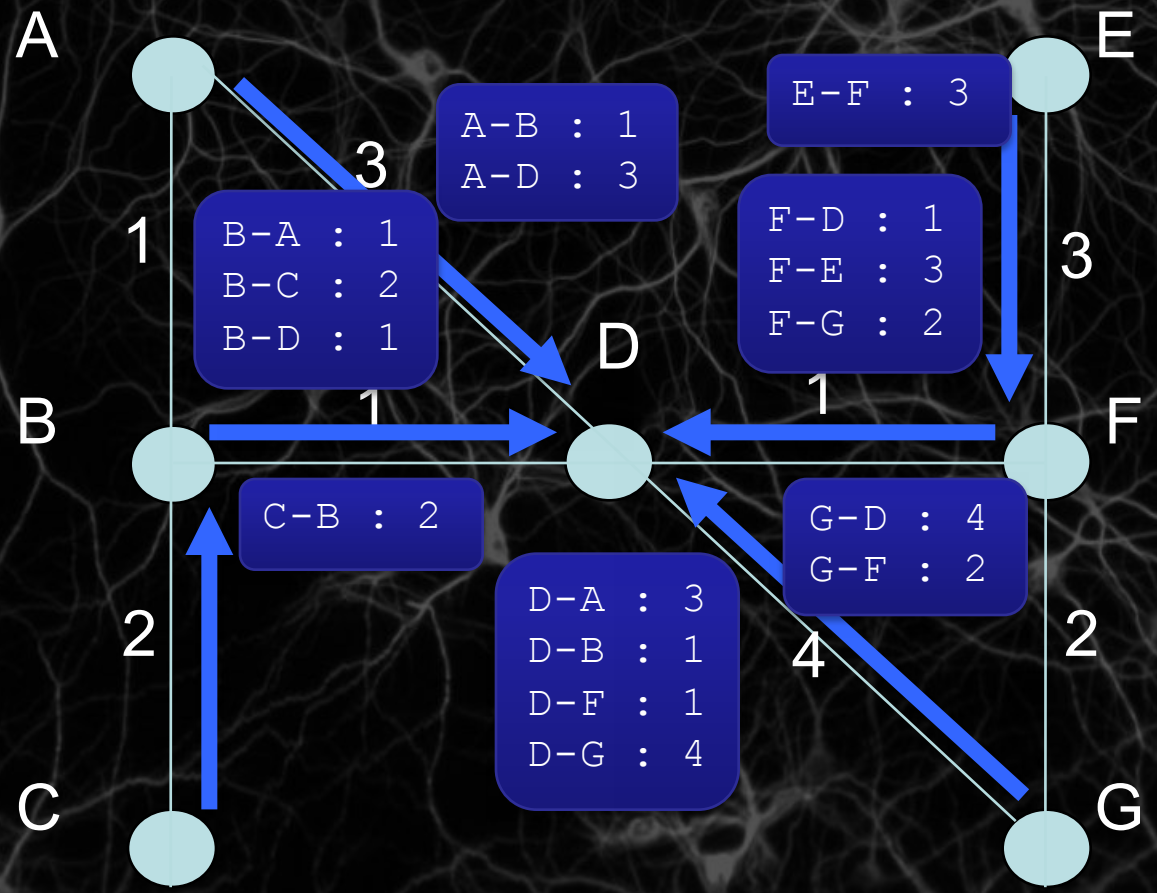
Algorithme de routage *à états de liens*

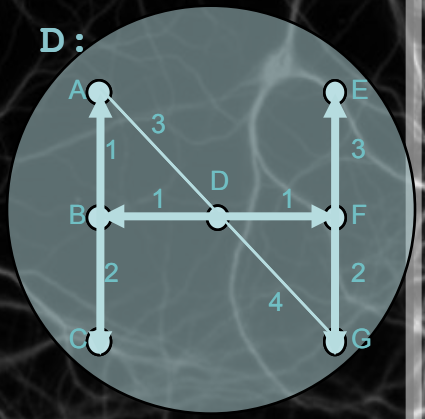
Echange des liens : *exemple pour les liens diffusés par D*



Algorithme de routage *à états de liens*

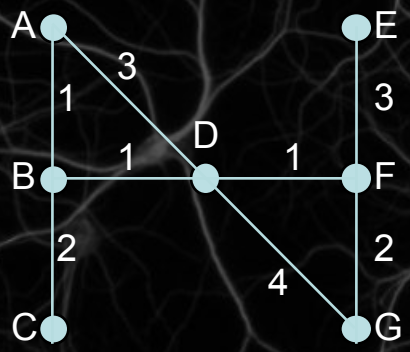
Echange des liens : *exemple pour les liens reçus par D*





Algorithmes de routage distribué

Deux familles :



Routage à *Vecteur de Distances* :

(*algorithme de Bellman-Ford « distribué »*)

(1) échanger sa tables de routage *partielles* avec ses voisins :

(2) recalculer ses routes en fonction des tables reçues des voisins.

D:

Dst	Lien	cout
A	->A	3
B	->B	1
C	- ?	?
D	-	0
E	- ?	?
F	->F	1
G	->G	4

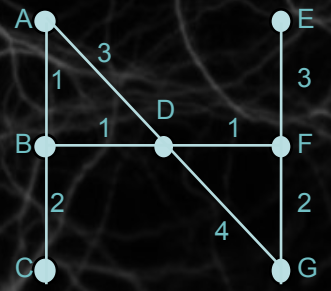
Routage à *Etats de liens* :

(1) échanger ses liens directs (*par inondation)*

But : Apprendre la topologie du réseau :

(2) algorithme de *Dijkstra « local ».*

D:



Les algorithmes de routage hiérarchiques

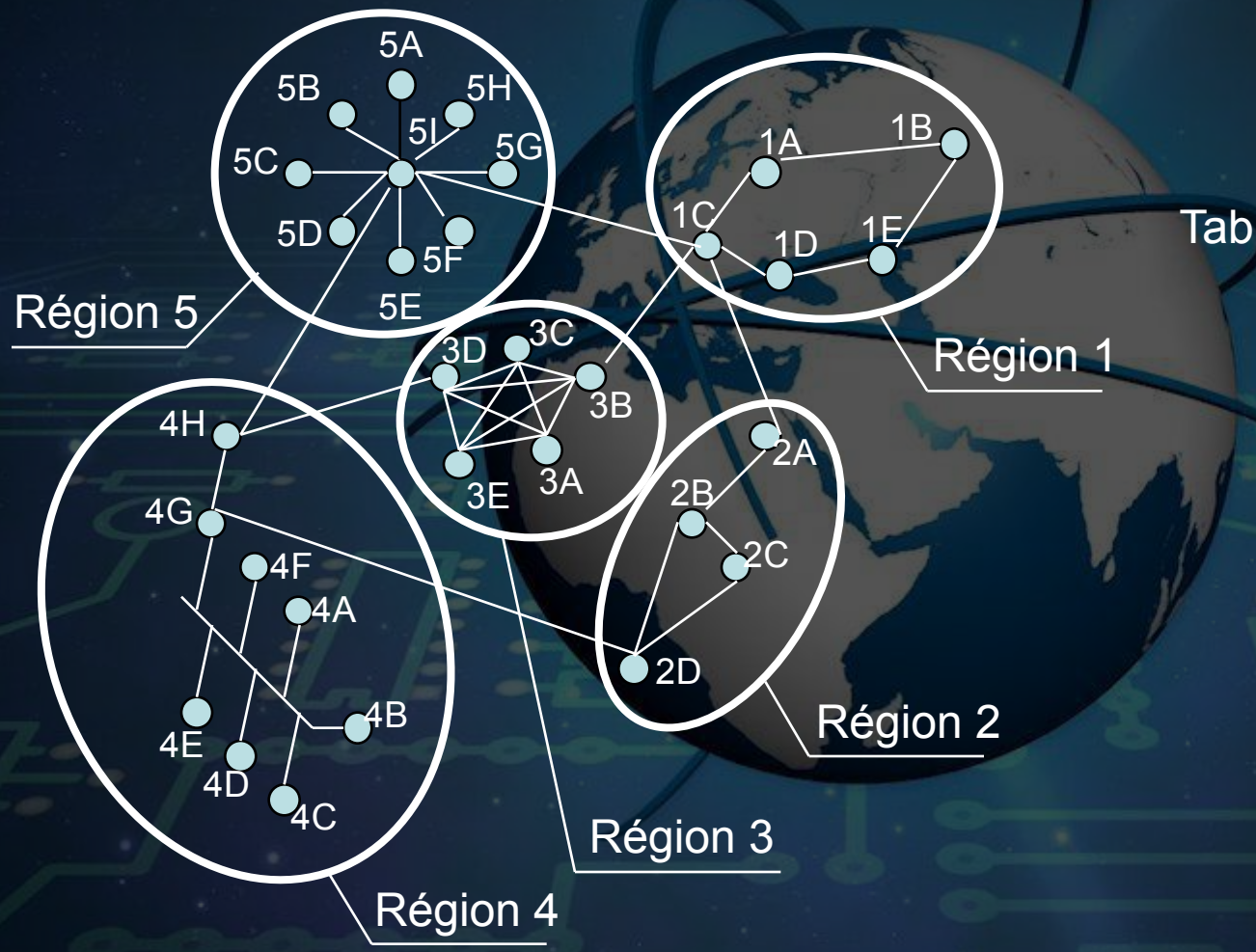


Table de routage de 2C :

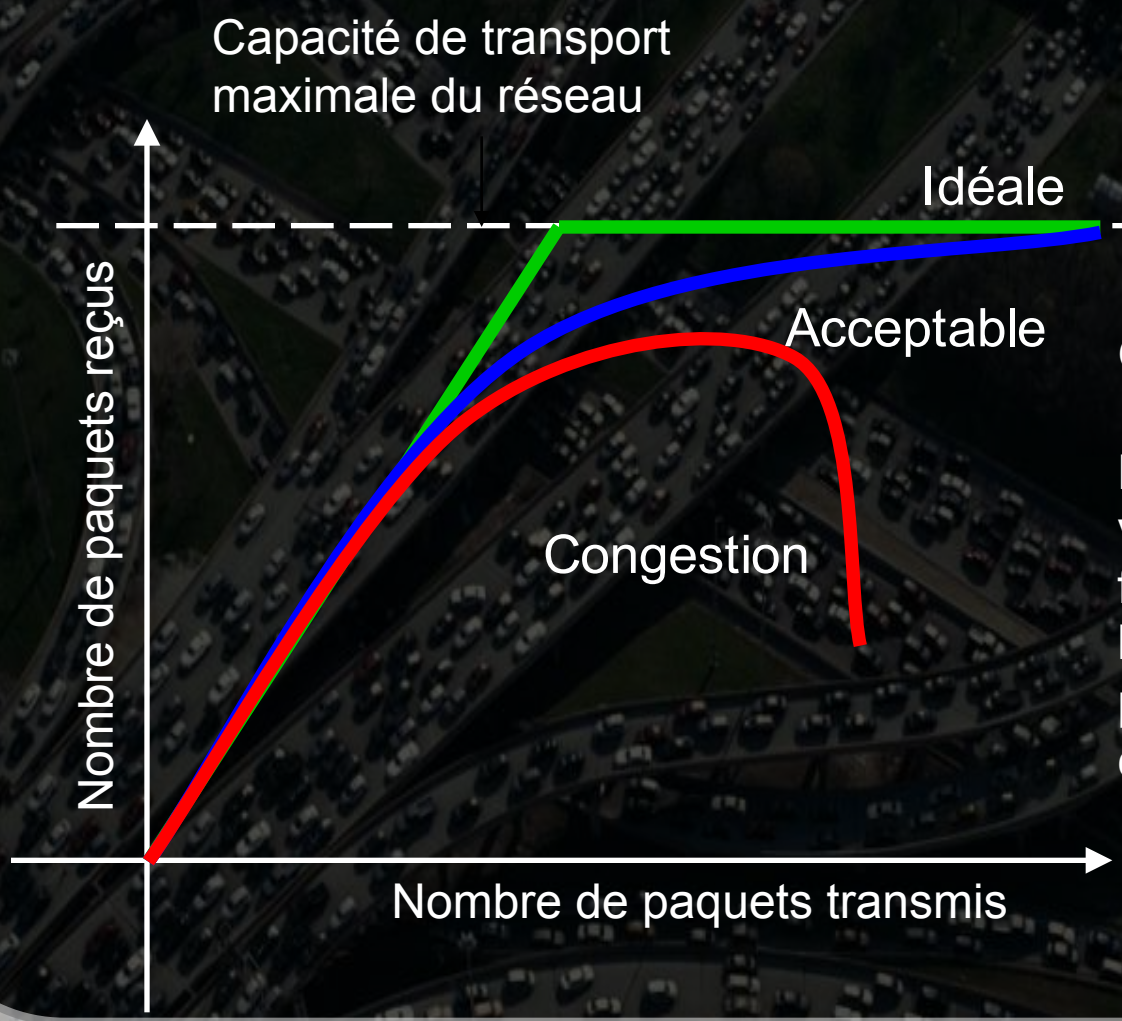
```
2A -> 2B
2B -> 2B
2C -> --
2D -> 2D
1* -> 2B
3* -> 2D
4* -> 2D
5* -> 2B
```

Algorithmes de gestion de la diffusion

Certains paquets sont destinés à plusieurs interlocuteurs présents sur le réseau global (multicast). Comment router ces paquets ?

1. des paquets (unicast) pour chaque destinataire :
2. un paquet (broadcast) réémis sur tous les liens : Inondation
3. Un paquet avec une liste d'adresse : Routage multi destination
   Le paquet est réémis sur les liaisons associées à au moins 1 adresse de la liste.
4. Routage du chemin inverse : optimal pour un broadcast…
   Le paquet est réémis sur les liaisons appartenant à l'arbre collecteur de la source.

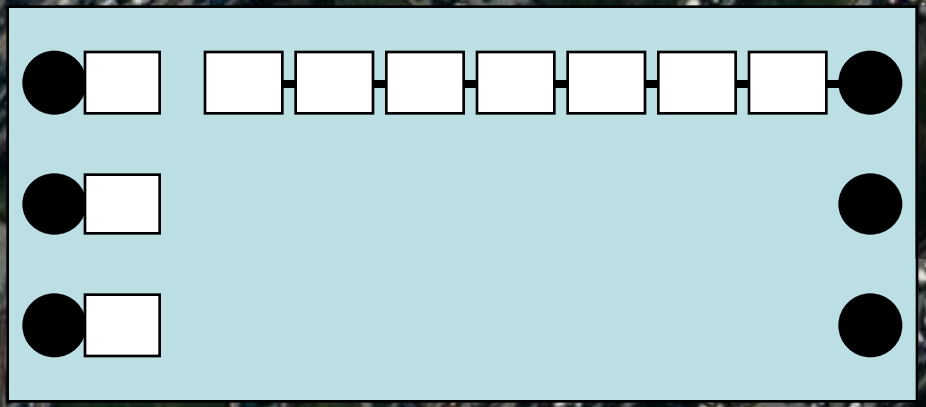
Congestion du réseau



Gérer la congestion :

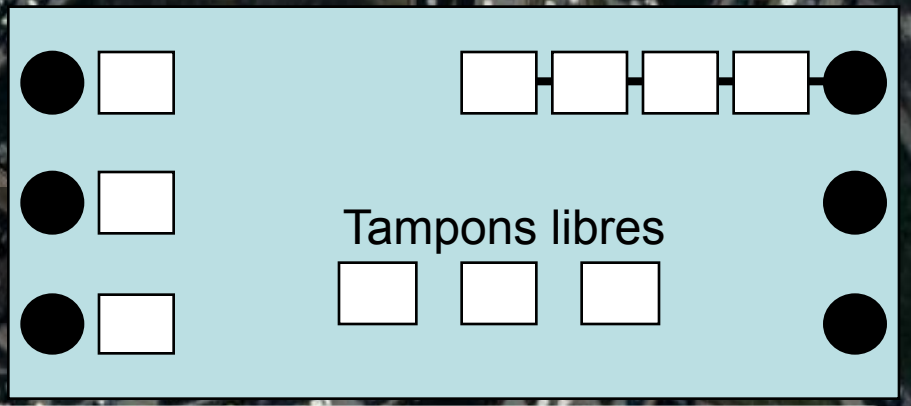
Proposer des mécanismes visant à garantir le bon fonctionnement du réseau lorsque le nombre de paquets entrant excède les capacités du réseau.

Gestion de la congestion par destruction des paquets



➔ Si une liaison monopolise l'ensemble des tampons disponibles, les paquets à destination d'autres liaisons sont perdus.

Il faut conserver un nombre minimum de tampons libres pour d'autres liaisons.



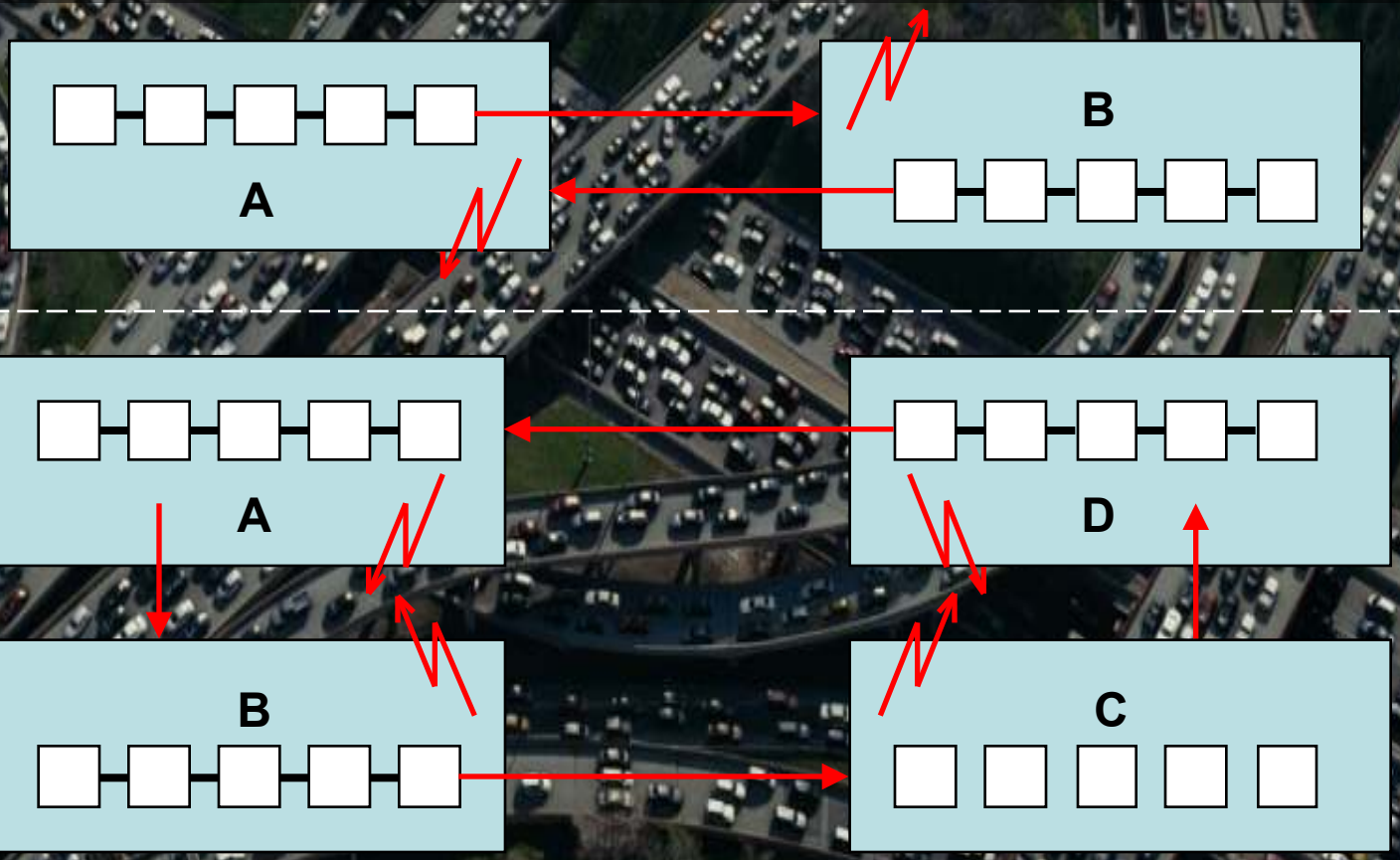
Selon les résultats d'Irland (78), un maximum simple d'est :

$$m = k / \sqrt{s}$$

avec m le nombre max. de tampon pour une file, k le nombre de tampon total, et s le nombre de liaisons de sortie.

Les étreintes fatales
(*DeadLock*)

e.g. Liaison en « fenêtre glissante »
La couche réseau est bloquée, le réseau se congestionne.



Gestion de la congestion par pré allocation des tampons

L'excès de paquet se manifeste par une saturation des tampons d'émission des machines qui forment les nœuds du réseau.

Première idée : Pré réserver les tampons pour chaque chemin initié dans le réseau.

- Connaître les chemins utilisés
  - Réseau en mode connecté
  - En cas de protocole « à fenêtre glissante » entre la source et la destination, il faut autant de tampons que le prévoit la fenêtre glissante dans chaque intermédiaire.

- Contrôle de congestion isarythmique -

Objectif :

interdire l'émission de paquets lorsque le réseau a atteint sa charge de travail maximale.

Proposition :

Chaque paquet représente un jeton. Chaque machine nœud du réseau dispose initialement d'un « certain » nombre de jetons. Lorsqu'une machine émet un paquet, elle perd un jeton. Lorsqu'elle reçoit un paquet, elle gagne un jeton.

Limite de la solution :

- Problème de perte de performance du réseau lorsque des paquets sont perdus entre leur émission et leur réception.
- n'empêche pas un nœud de recevoir plus de paquet qu'il ne peut en gérer (pas de garantie de flux).

Le contrôle de flux

Réduire la quantité de paquets échangés entre les machines nœuds chaque seconde lorsque le réseau se congestionne pour éviter la congestion.

Solution correcte pour éviter la surcharge :

- des liaisons physiques ;
- des capacités de traitement des machines nœuds ;

Solution médiocre pour répondre à une congestion.

- Inadaptée à un trafic irrégulier ;
- Sous-exploite la capacité de transport du réseau.

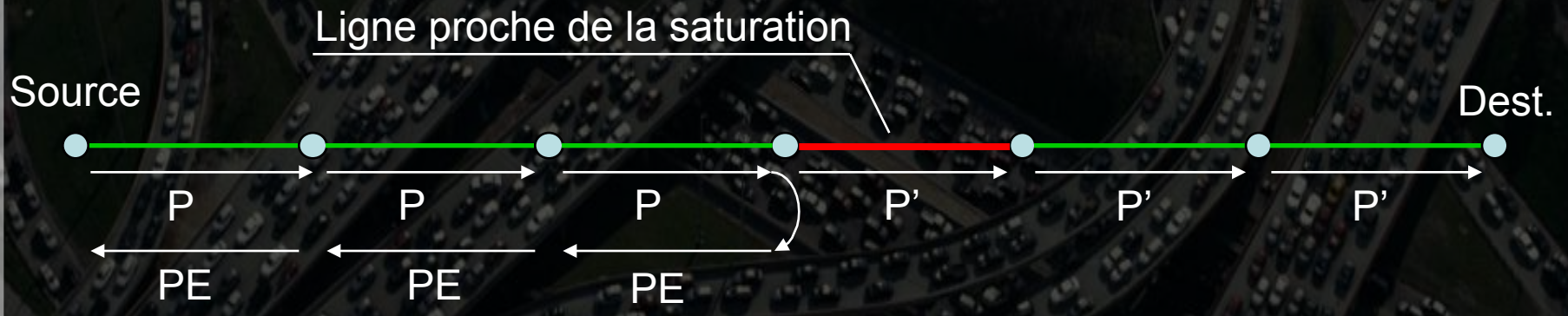
Paquets d'engorgement

Calculer un taux d'occupation (u) maximum et décider d'un seuil d'occupation acceptable.

On peut calculer u avec : $u_{nouveau} = a.u_{ancien} + (1-a)f$

Ou *f* est 0 ou 1 selon que la ligne est occupée lors de l'échantillonnage.

a est la « faculté d'oublier » les enchantions anciens.



P : Paquet émis par la Source (et réémis par les intermédiaires) ;

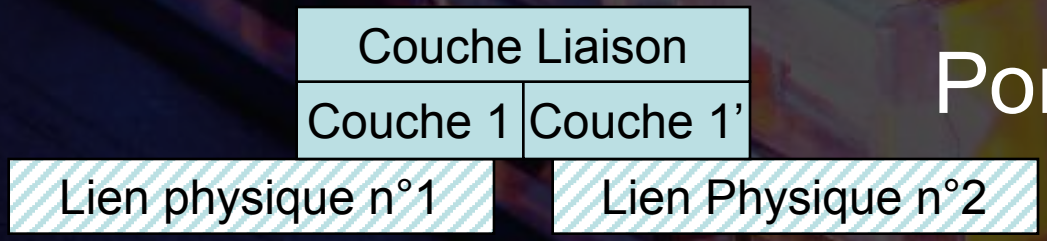
P' : Paquet émis par la Source avec un tag de saturation ;

PE : Paquet d'engorgement à destination de la source.
(pour que la source réduise son débit vers la Dest.)

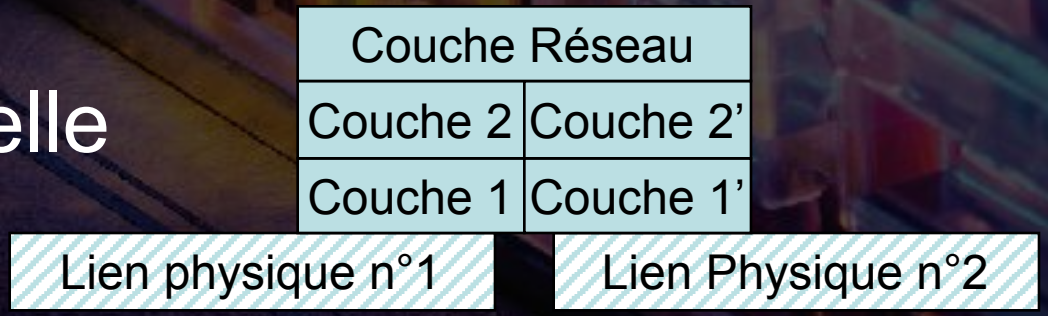
Interconnecter des réseaux

Répéteur

Pont

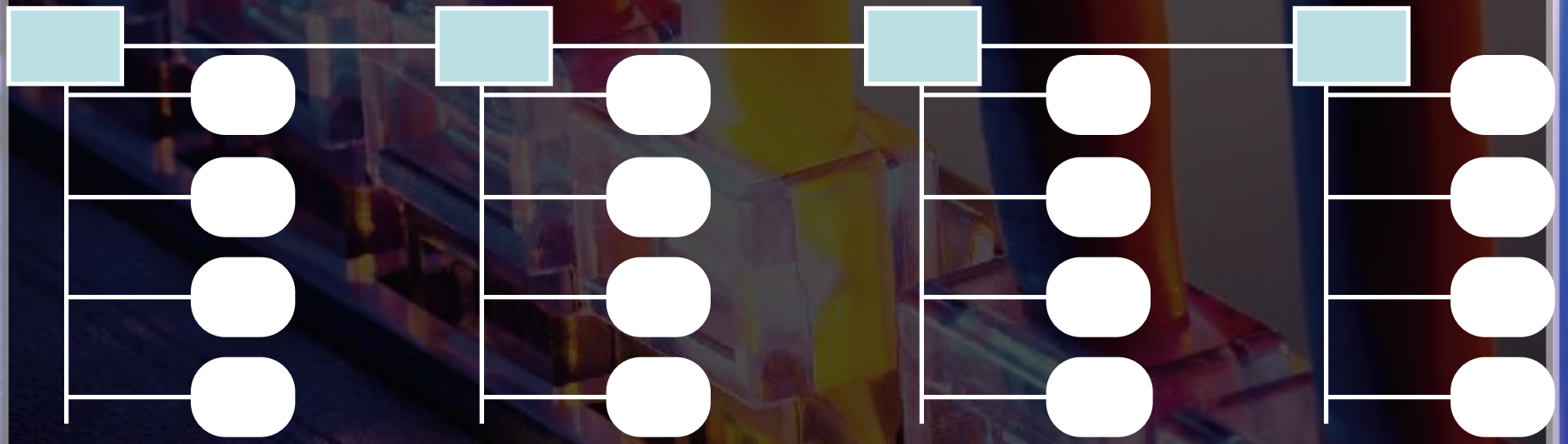


Passerelle



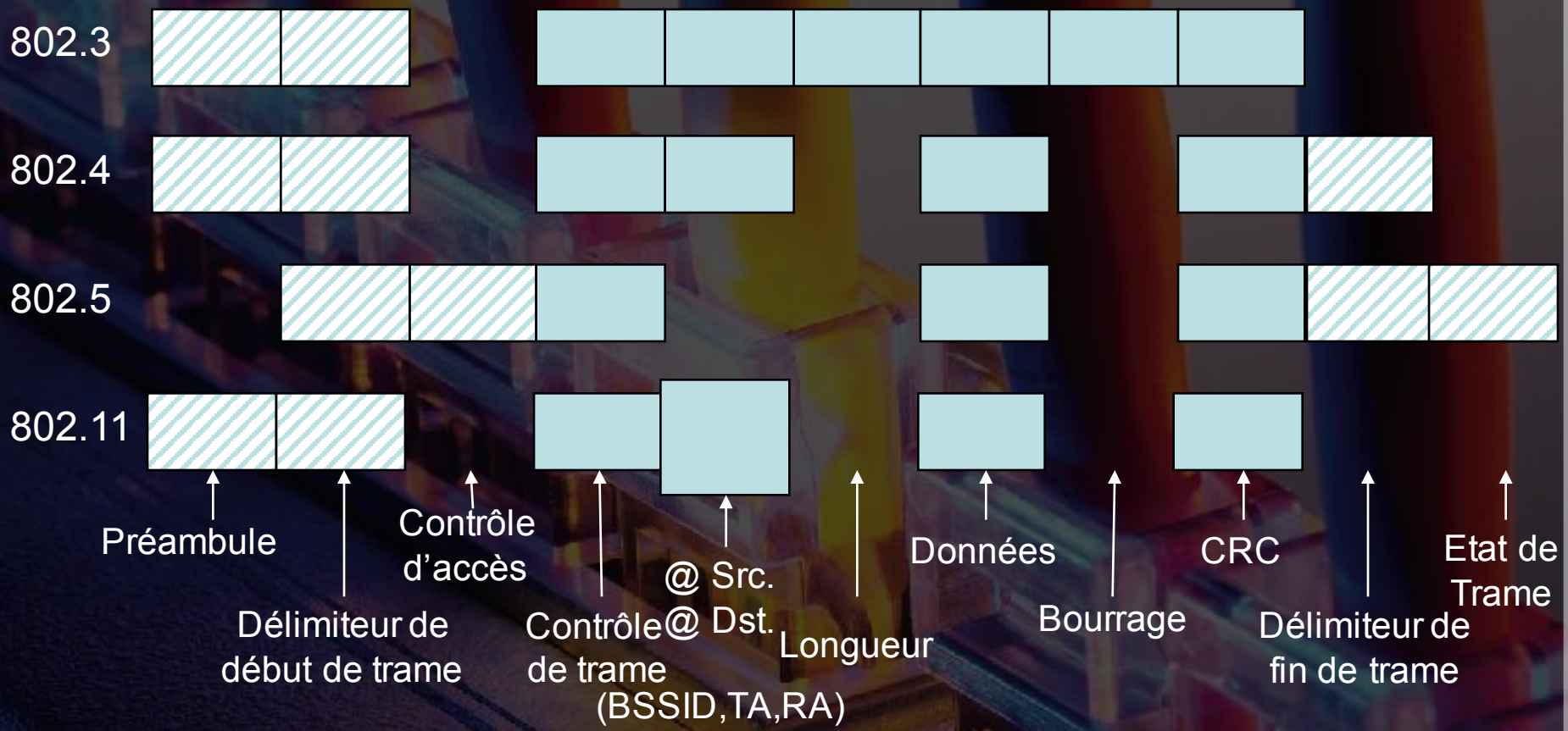
Interconnexion par **répéteur**

Nécessite l'homogénéité du support de communication physique.



Sur l'exemple d'une liaison de type Ethernet (802.3) les répéteurs entraînent + d'émetteurs potentiels sur le même support, donc plus de collisions et un risque d'effondrement des performance au-delà d'un certain seul.

Interconnexion par **pont**



Suppose une certaine « équivalence » entre le contenu des trames ci-dessus des trames IEEE.

Interconnexion par **pont**

Exemples des traitements à réaliser pour chaque type de conversion.

	802.3 « Ethernet »	802.4 « Token bus »	802.5 « Token ring »	802.11 « Wifi »
802.3		1,4	1,2,4,8	1,8
802.4	1,5,8,9,10	9	1,2,3,8,9,10	1,5,8,9,10
802.5	1,2,5,6,7,10	1,2,3,6,7	6,7	1,2,5,6,7,8,10
802.11	1,10	1,4	1,2,4,8	1

Actions :

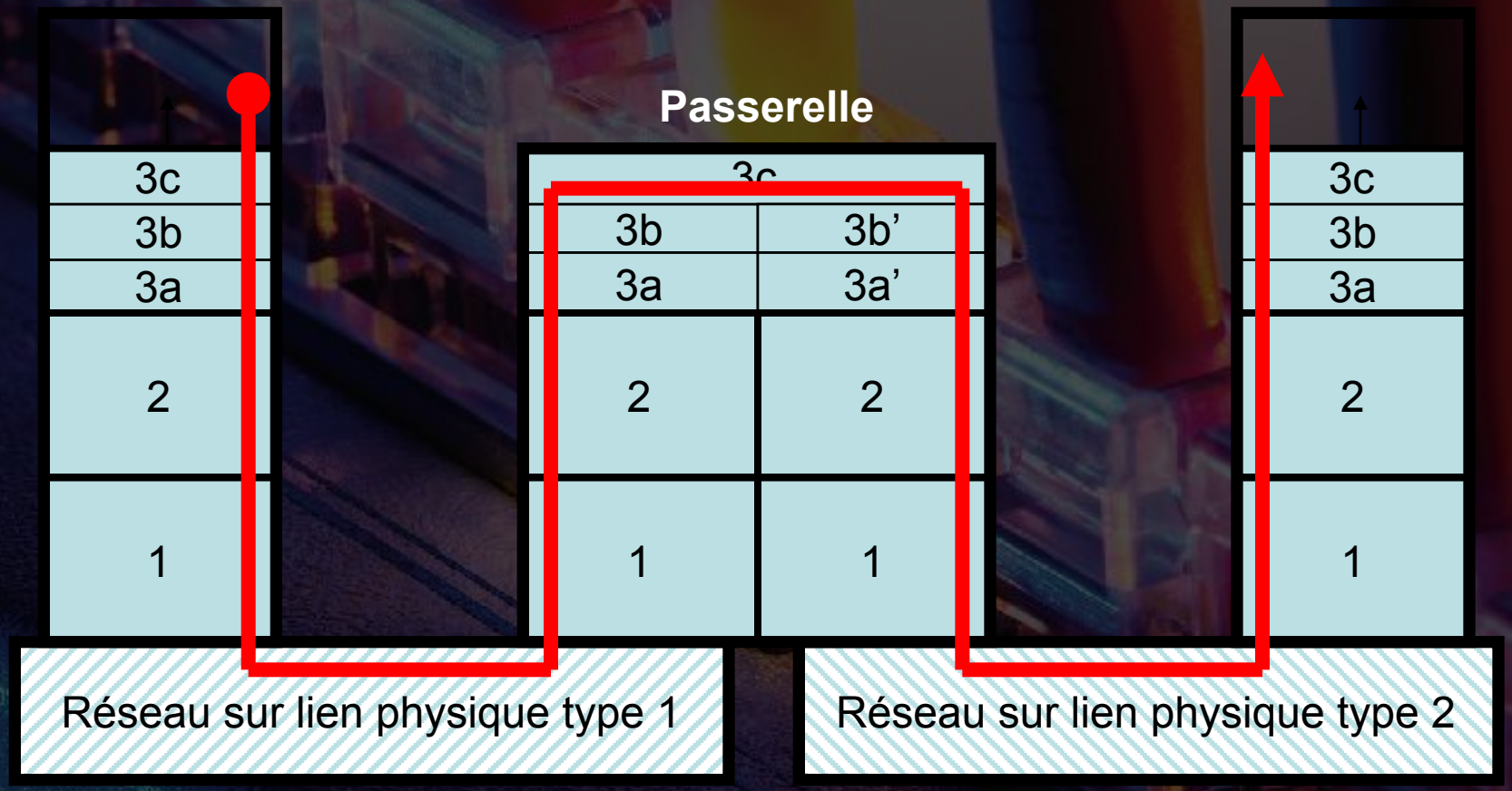
1. Reformater la trame et recalculer le CRC
2. Inverser l'ordre des bits
3. Copier la priorité qu'elle ait ou non un sens
4. Générer une priorité fictive
5. Annuler la priorité
6. Vider l'anneau
7. Positionner les bits A et C
8. Problème de congestion Lien rapide vers Lien Lent
9. Problème du jeton de transfert, ACK reporté ou annulé
10. Panique lorsque la trame est trop longue!

Annexes :

802.3	1518 octets	10Mbits/s
802.4	8191 octets	10Mbits/s
802.5	5000 octets	4Mbits/s
802.11	2312 octets	1Mbits/s

Interconnexion par **passerelle**

Une passerelle est une « machine nœud » exclusivement dédiée à l'aiguillage de paquet. Dans un réseau conventionnel une passerelle est le premier objet à pouvoir faire un routage « intelligent » parce que reposant sur la couche 3.



Interconnexion par **passerelle** - fragmentation -