LES RESEAUX SANS FIL

Wireless Network

Les Réseaux (1)

- 2 modèles différents

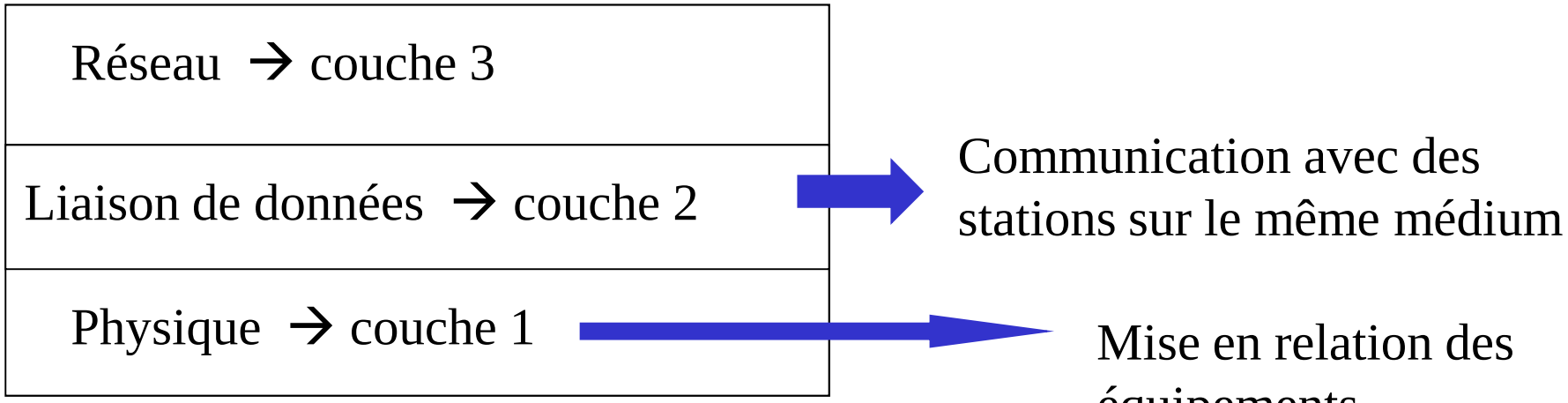
Modèle OSI

7	Application
6	Présentation
5	Session
4	Transport
3	Réseau
2	Liaison de données
1	Physique

Modèle TCP/IP

Application
Transport
Réseau
Accès réseau

Les Réseaux (2)



- Le rôle de la couche 3 est d'acheminer **des paquets** entre un système source et un système destination **ne se trouvant pas sur le même médium.**
  - Les couches MAC sont insuffisantes

- La couche *Réseau* rend des services à la couche transport et demande des services à des couches liaison de données.

Sans Fil (1)

- Aucun câble ne relie les équipements
  -> utilisation des ondes ….

- Utilisation de bandes de fréquence
  - 2 organismes en charge de la normalisation des fréquences
    - USA : IEEE (Institute of Electrical and Electronics Engineers)
    - Europe : ETSI (European Telecommunication Standards Institute)

Sans fil (2)

- But : être compatible avec les réseaux câblés
- Quelques exemples :
  - WLAN : le plus connu : Wi-Fi (Wireless-Fidelity)
    - différentes normes : 802.11a, 802.11b, 802.11g, 802.11n
  - WPAN : pour les courtes distances
    - différentes normes : bluetooth, zigbee, 802.15…., Ultra-Wideband (UWB (utilisé par W-USB))
  - WMAN ou WRAN : Wireless Regional Area Network pour les distances moyennes
    - différentes normes : WiMAX, 802.16…, 802.20 (mobilité), 802.22 (même longueur d'onde que la TV…)
  - WWAN:
    - les communications satellitaires
    - famille 3GPP : les normes GSM, GPRS, EDGE, UMTS , LTE, ..

La fonction routage pour les réseaux sans fils

- Quelques rappels sur le routage
- Différents types de routage
  - routage proactif
  - routage réactif
  - routage géographique

Quelques rappels

- La couche 3
  - Désignation des réseaux, des systèmes
    - ➔ adressage respectant **unicité** (exemple : IPv4, IPv6,…)
  - Calcul de routes, gestion du routage
    - Utilisation des tables de routage

Si l'adresse-IP destination	193.54.51.0	192.168.51.0	193.54.49.0	default
avec le masque	255.255.255.0	255.255.255.0	255.255.255.0	0.0.0.0
expédier à	193.54.51.1	193.54.51.8	193.54.51.241	193.54.51.254
par l'interface MAC	eth0	eth1	eth0	eth0

- **Problème : créer les tables de routage !!!**
- 2 possibilités :
  - Routage statique
  - Routage dynamique

Routage

Routage statique

- convient uniquement pour des sites de taille modeste.
- généralement, le routage est modifié après la découverte du problème.
- architecture géographique très peu flexible.

Routage dynamique

- *indispensable dès que la topologie est complexe ou qu'elle change au cours du temps*
- algorithme en général auto-cicatrisant
  (si lien coupé, on recherche un autre passage)
- permet de maintenir des tables de routage cohérente

Routage dynamique

- But : mettre a jour dynamiquement les tables de routage lorsque :
  - La topologie interne de « l'AS » est modifiée (câble coupé, routeur ajouté, modification de l'adressage, …)

- Deux classes d'algorithmes existent :

les algorithmes ***Vector-Distance (à vecteurs de distance)***

- chaque routeur ne connaît que ses voisins
- échange des tables de routage en utilisant un vecteur
- Utilisation et sélection des routes ayant le coût minimal
- Exemple : RIP, IGRP, EIGRP

les algorithmes ***Link-State (à état de liens)***

- Chaque routeur reconstruit une carte complète de l'AS
- Chaque routeur recherche la meilleure route vers les autres routeurs
- Exemple : OSPF, IS-IS

Routage à vecteurs de distance (1)

- Basé sur l'algorithme de Bellman-Ford
  (nommé DBF : Distributed Bellman-Ford)
  - Les routeurs ne connaissent que leurs voisins (directement "connecté")

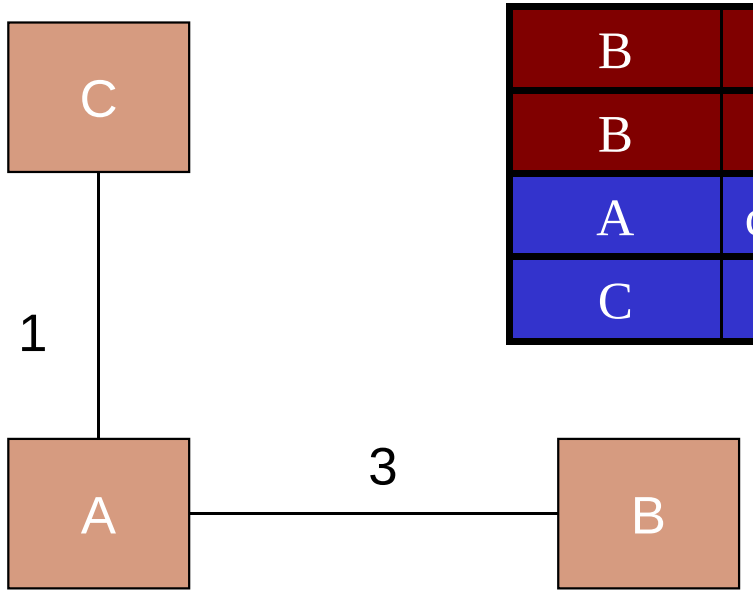
- Fonctionnement :
  - On échange entre nœuds voisins **un vecteur donnant les coûts (vecteur de distance)** permettant d'atteindre toutes les destinations connues par le routeur et stockées dans la table de routage
  - Un routeur qui reçoit ce vecteur, le compare avec ses propres coûts connus et met à jour sa propre table de routage :
    - si une route reçue comprend un plus court chemin
    - si une route reçue est inconnue
  - Si la table d'un routeur est modifiée, le routeur enverra son vecteur de distance à tous ces voisins. Si plus aucune table n'est modifiée, l'algorithme se termine, on dit qu'il y a convergence.

Routage à vecteurs de distance (2)

C	P	coût
C	/	/
B	A	4
A	connect	1

B	P	coût
B	/	/
A	connect	3
C	A	4

A	P	coût
A	/	/
B	connect	3
C	connect	1



B envoie sa table a A Vecteur = (B=0)

C envoie sa table a A Vecteur = (C=0)

A envoie sa table a B et C Vecteur = (A=0, B=3, C=1)

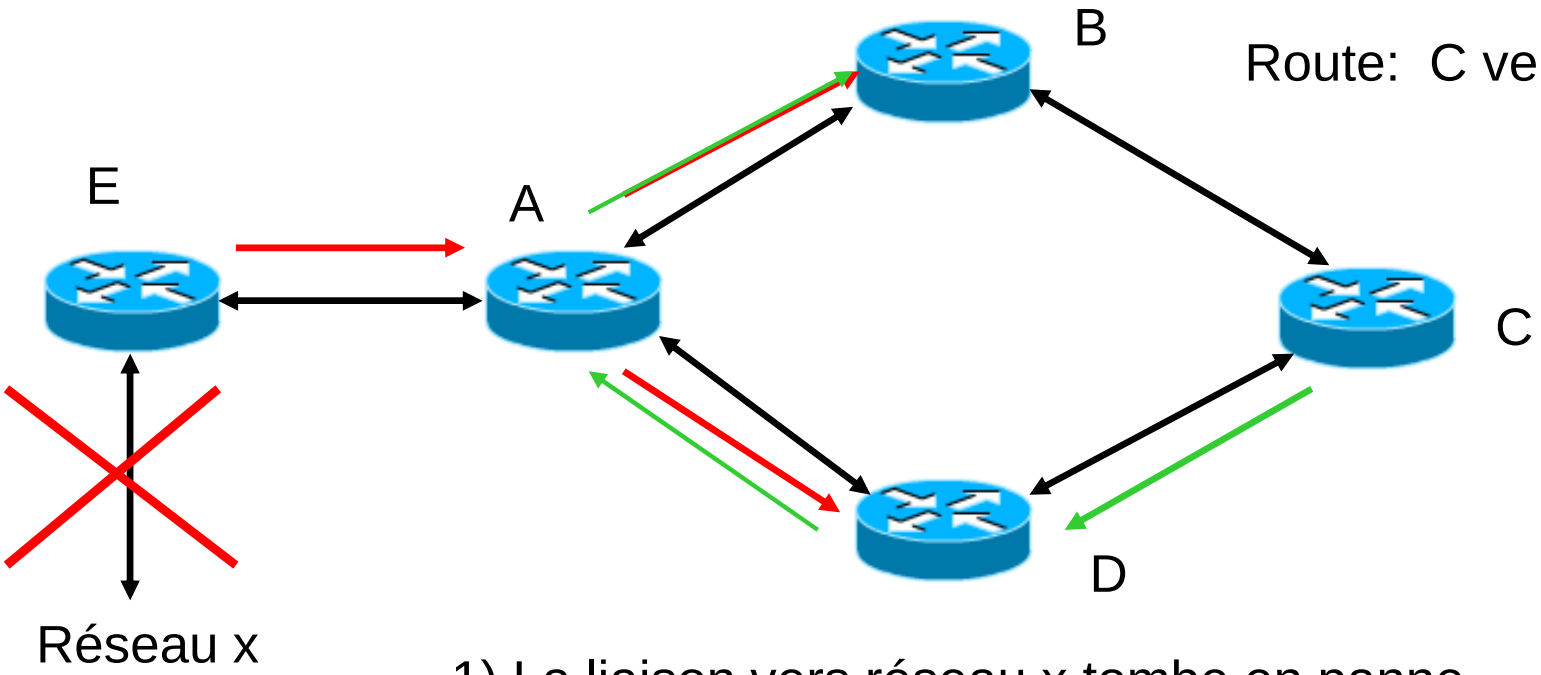
Routage à vecteurs de distance (3)

- Avantage :

  Facile à mettre en œuvre

- Inconvénients :
  - La taille des informations de routage est proportionnelle au nombre de routeurs du domaine
    - peut donner de gros vecteurs de distance ➔ lourd
  - Coût de routage uniquement défini selon la distance entre 2 destinations
  - ***Problème avec la création de boucles***

Exemple de boucle



1) La liaison vers réseau x tombe en panne

2) E avertit le routeur A → plus de route vers x

3) A avertit les routeurs B et D → plus de route vers x

4) C envoie une mise à jour de sa table vers D
route vers x en passant par B

5) D envoie à A route vers x en passant par C, puis
A envoie à B ….

PB

Contre-mesure

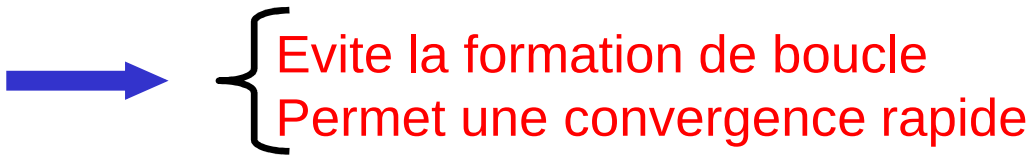
- Solutions :
  - le nombre de saut maximum
  - la règle du split horizon
    - interdiction de renvoyer une information de routage sur l'interface qui a reçu cette information
  - le poison reverse
    - route mise à la métrique maximum
  - les compteurs de retenue
    - mise en place d'un timer permettant d'accepter ou non une nouvelle route suivant sa valeur
  - la mise à jour déclenchée

Algorithme à état de liens (1)

- Les routeurs maintiennent une carte complète du réseau et calculent les meilleurs chemins localement.
  - Utilisation d'une base de données (BD topologiques)
  - Utilisation d'un algorithme pour créer un arbre de plus court chemin (SPF)
  - Utilisation d'une table de routage

- Un routeur teste périodiquement l'état des liens qui le relie à ses routeurs voisins (envoie d'une trame HELLO) pour savoir s'ils sont vivants.

- Un routeur « multicast » périodiquement ces états *(Link-State Advertisement-LSA-)* à tous les autres routeurs du domaine *(pas uniquement aux voisins)* ou lorsqu'il y a un changement de topologie.
  - utilisation de la technique du split horizon



Algorithme à état de liens (2)

- Les routeurs ne communiquent pas la liste de toutes les destinations connues, mais seulement les informations ayant changées
  - Mise à jour partielle, évite de surcharger le réseau

- Lorsqu'un message parvient à un routeur, celui-ci met à jour sa BD topologique et recalcule localement pour chaque lien modifié, la nouvelle route selon l'algorithme de Dijkstra (***Shortest Path First algorithm)*** qui détermine le plus court chemin pour toutes les destinations à partir d'une même source.

Avantages	**Inconvénients**
Convergence rapide sans boucle	Capacité de calcul important
Métrique précise et couvrant plusieurs besoins	Plus de mémoire pour stocker les informations
Taille des messages échangés entre routeurs petite	Fort débit lors de la convergence de l'algorithme
Calcule des routes sur chaque routeur	

Et pour

les réseaux sans fil ?

Les réseaux mobiles Ad Hoc (1)

Définition:

Un réseau mobile ad hoc, appelé généralement **MANET** (Mobile Ad hoc NETwork), consiste en une grande population, relativement dense, d'unités mobiles qui se déplacent dans un territoire quelconque et dont le seul moyen de communication est l'utilisation des interfaces sans fil, sans l'aide d'une infrastructure préexistante ou administration centralisée.

Un réseau ad hoc peut être modéliser par un graphe G = (V,E)

où - V représente l'ensemble des nœuds et

- E l'ensemble des connections qui existent entre ces nœuds.

But : Faire communiquer deux nœuds distants en passant par un ou plusieurs intermédiaires

Application : aéroport, sécurité routière, randonnée …

Les réseaux mobiles Ad Hoc (2)

- Caractéristiques :
  - Topologie dynamique
    - c'est le but de ces réseaux, pouvoir bouger, donc changement continu de la topologie.
    - impossible de savoir si un ordinateur sera encore joignable dans les minutes suivantes.
  - Bande passante limitée /débit limité
    - du fait du partage de la bande passante
  - Contraintes énergétiques
    - mouvement ➔ utilisation batterie…
  - Absence d'infrastructure
  - Sécurisation des données
    - problème au niveau physique (inondation, dispersion,…)

Routage dans réseaux MANET

Objectifs :

- Minimiser la charge réseau
  - Eviter les boucles
  - Eviter la concentration du trafic en un point
  - Eviter de surcharger les entêtes des trames ou des paquets
- Effectuer des communications multi-points fiables
- Assurer un routage optimal
- Optimiser le temps de latence
  - De nombreux protocoles utilisent un timer pour gérer leurs envois de données → problème si latence trop longue

Réseaux MANET

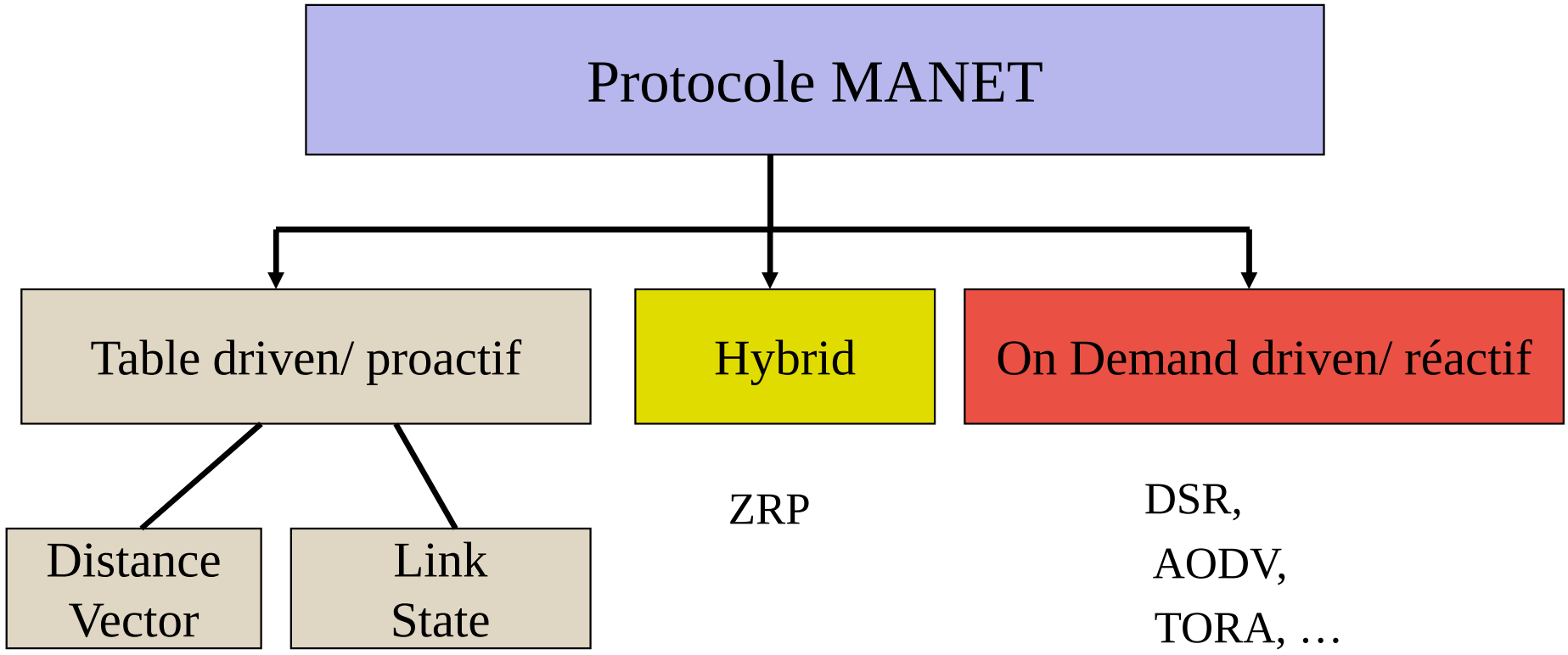
Classification

- Mode infrastructure
  - Routage classique filaire (protocole statique/ dynamique)
- Réseau "mesh" sans fil (Réseau maillé sans fil)
  - Routage **dynamique classique** (OSPF, RIP,…)
  - Les nœuds servant de routeurs ne bougent pas
- Réseau Ad hoc
  - Création de nouveaux protocoles de routages dynamiques
  - Chaque station est un routeur potentiel
  - Protocole proactifs, réactifs, mixtes

Routage, classification

- Protocoles proactifs
  - échange périodique de paquets de contrôle
  - toutes les routes sont disponibles
    → mise à jour périodique des tables de routage
  - exemple : DSDV, OLSR, FSR, WRP,…
- Protocoles réactifs
  - les routes sont créées à la demande
  - minimisation du trafic, mais augmentation du temps de latence
  - ouverture des routes par inondation
  - exemple : DSR, AODV, TORA,…
- Protocoles mixtes
  - proactif local, réactif éloigné
  - exemple : ZRP,…

Routage, classification (2)



DSDV (1)

- Destination-Sequenced Distance Vector protocol
  (Vecteur de Distance à Destination Dynamique Séquencée)

- But : - Garder la simplicité des algorithmes à vecteurs de distance
  - Eviter le problème de la formation des boucles
  - Autoriser les stations (nœuds) à ne pas bouger

- Implantation:
  - Basé sur l'algorithme de Bellman-Ford
    → (amélioration du RIP)
  - Ajout d'un nouvel attribut dans la table de routage :
    le Numéro de Séquence (SN : Sequence Number)
  - Chaque nœud joue le rôle d'un routeur (si nécessaire)

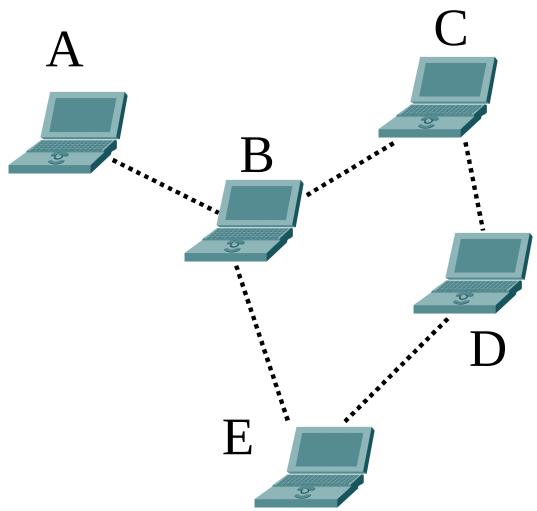
DSDV (2)

Chaque nœud a une table de routage qui contient :

- Pour toutes les destinations possibles
  - Le nombre de sauts (de nœuds) nécessaire pour atteindre la destination
  - Le prochain saut pour chaque destination
  - Le numéro de séquence qui correspond à un nœud destination

Exemple : table de routage de B

Dest	Saut	Metric	N° seq
A	A	1	A-212
B	/	0	B-24
C	C	1	C-145
D	C	2	D-378
E	E	1	E-156



DSDV (3)

Mise à jour

- Périodique (comme dans RIP → 15s)
- Déclenchée
  - ➡ lorsqu'il y a de forts changements dans la topologie

2 façons de réaliser cette mise à jour

- soit mise à jour complète
  - on envoie la totalité de la table de routage
- soit mise à jour partiel ou incrémental
  - on envoie que les informations ayant été modifiées depuis la dernière mise à jour complète

Si mise à jour incrémental ne tient pas dans une NPDU, alors on fait une mise à jour **complète.**

DSDV (4)

- Changement dans la table de routage
  - Toute route avec un numéro de séquence supérieure à celui existant est mise à jour
  - Si même numéro de séquence, alors on compare la métrique :
    - si métrique plus petite, mise à jour, sinon rien
      (minimisation du chemin parcouru)
  - Si route inexistante, alors rajout.

⇒ Les changements sont diffusés aux autres nœuds.

DSDV (4bis)

Dest	Saut	Metric	N° seq
A	A	0	A-130
B	B	1	B-56
C	E	2	C-211
D	E	3	D-156
E	E	1	E-39

paquet de maj venant de A

+

Dest	Saut	Metric	N° seq
A	A	1	A-126
B	/	0	B-56
C	A	4	C-210
D	E	2	D-156
E	E	1	E-39

Table de routage de B

=

Table de routage de B

Dest	Saut	Metric	N° seq
A	A	1	A-130
B	/	0	B-56
C	A	3	C-211
D	E	2	D-156
E	E	1	E-39

DSDV (5)

- Perte d'un lien
  - Incrémentation du numéro de séquence et diffusion
  - Métrique mise à l'infini
  - On évite la création de boucle grâce au numéro de séquence

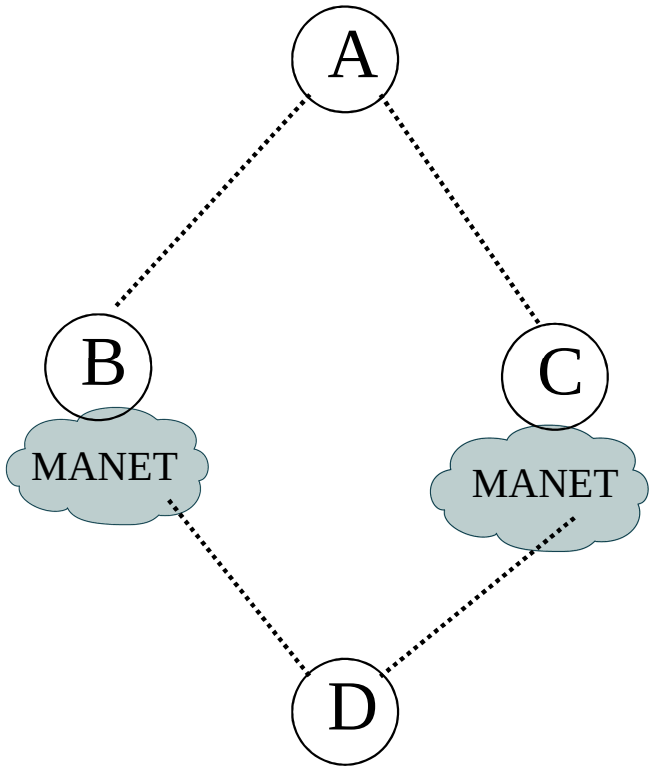
- Nouveau lien
  - apparition ou ré-apparition d'une station
  - Incrémentation du numéro de séquence et diffusion

Si déplacement d'un nœud, on a perte d'un lien et ensuite création d'un nouveau lien, d'où même numéro de séquence

→ mise à jour correct de la table de routage

DSDV (6)

- Quelques problèmes :
  - **La fluctuation**
  - route mise à jour : si n° séquence reçu > n° séquence actuel
    ou si même numéro de séquence, et métrique inférieure



On suppose :

- B a un chemin vers D en 10 sauts
- C a un chemin vers D en 8 sauts

Si D met un jour sa route vers A avec un nouveau numéro de séquence alors…

B envoie maj vers A → A change de route

C envoie maj vers A → A rechange de route

la route de A vers D va fluctuer entre B et C

DSDV (7)

Raisons

- irrégularité dans les mises à jour,
- trafic asynchrone,
- différentes vitesses de propagation,…

Solutions

- **Maintenir deux tables**
  - une table contenant la table de routage
  - une table contenant le temps à attendre avant de diffuser un changement dans la table de routage

DSDV (8)

Conclusion

- DSDV est *assez lent à converger*
  (problème des algorithmes à vecteurs de distance)
- DSDV ne peut converger si les nœuds ne s'arrête pas de bouger pendant 300 secondes
- Le taux de transmission réussit est de 70% à 90% lors de mouvement non rapide
- DSDV reste assez instable lors de changement assez brusque de topologie
- DSDV nécessite de temps en temps des mises à jour complète pour un bon fonctionnement
- Utilisation de deux tables : une de routage, une pour les "updates"

OLSR (1)

- Optimized Link State Routing
  - Basé sur un algorithme d'état de lien "optimisé"
    - Gestion des relais multipoints (MPR : MultiPoint Relay)
  - Protocole en cours de normalisation par l'IETF
    RFC 3626
  - Utilise plusieurs tables :
    - Table de voisinage
    - Table de topologie du réseau
    - Table de routage

OLSR (2)

Découverte du voisinage

- Envoie de trames HELLO périodique
- 4 sortes de lien
  - lien symétrique
  - lien asymétrique
  - lien MPR
  - lien perdu → plusieurs trames "hello" sans réponse

→ Un lien sera considéré comme fiable s'il est symétrique

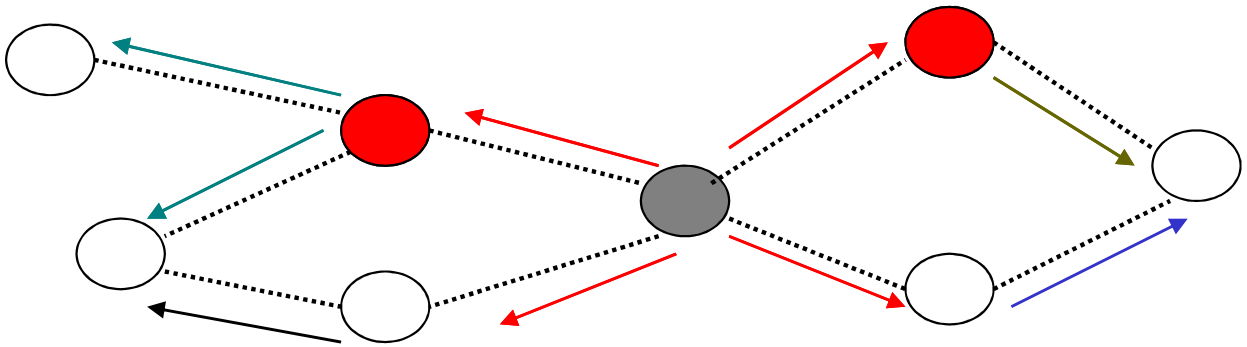
4 échanges pour avoir lien symétrique :

- A envoie une trame Hello vide (on découvre)
- B répond avec une trame Hello asymétrique (B entend A)
- A répond avec une trame Hello symétrique
- B répond avec une trame Hello symétrique

OLSR (3)

- Les relais multipoints (MPR)
  Chaque nœud envoie à tous ces voisins sa table de voisinage.
  - Permet de remplir la table de topologie
  - Permet à un nœud de connaître la liste des stations se trouvant à 2 sauts

  - <u>But des relais multipoints</u> : minimiser le nombre de transmission pour que tous les voisins de second niveau reçoivent l'information.
    (une station peut-être visible par plusieurs nœuds)



lien symétrique

SANS MPR → 5 retransmissions

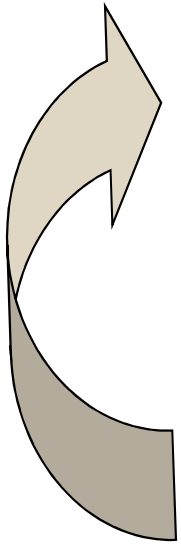
AVEC MPR → 3 retransmissions

OLSR (4)

Election des MPR

- Algorithme np-complets ➔ impossible car trop lent
- utilisation d'un heuristique

1. On recherche les nœuds du second degré isolé
   (lien unique avec un nœud du 1er degré)
   ➡ on étiquette alors les nœuds du 1er degré les desservant



2. On élimine tous les nœuds accessibles par les MPRs trouvés

3. On recherche le nœud du 1er degré permettant d'accéder aux plus grands nombres de nœuds du second degré

4. On étiquette ce nœud alors comme MPR, et on revient à l'étape 2 tant qu'il reste des nœuds du second degré.

OLSR (5)

Base de données topologique

- Pour chaque destination, on garde :
  - l'adresse de destination
  - le dernier point relais (MPR) avant l'adresse de destination
  - un numéro de séquence
  - le temps avant destruction si aucune remise à jour

Pour construire cette base, les MPRs diffusent de **temps en temps** leur table de voisinage (Topology Control –TC-) qui sont relayés par les autres MPRs.

Si la trame est trop grande, plusieurs trames sont envoyées.

Chaque TC a un numéro de séquence (cf DSDV)

Attention : un nœud non MPR n'envoie que des trames "hello" à ces voisins et donc ne diffuse rien

OLSR (6)

Table de routage

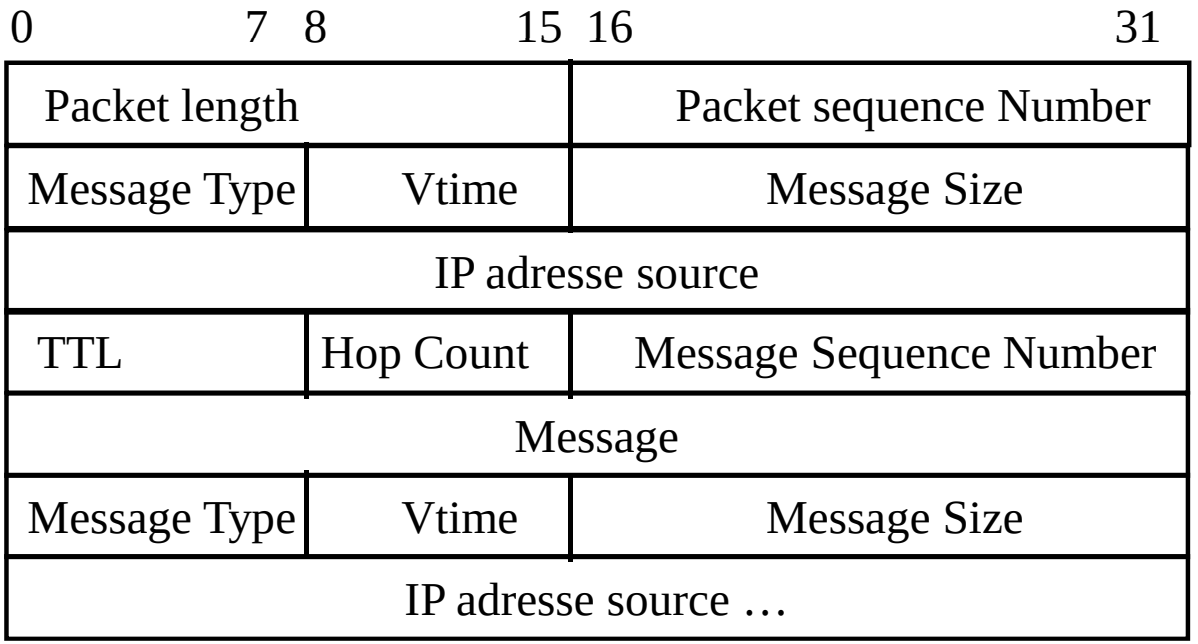
- création identique à un algorithme d'état de lien
  - utilisation table de voisinage, bd topologique et SPF.
  - utilisation des liens stables

Autres

- 4 sortes de messages de contrôle / UDP port 698
  - trame Hello
  - trame TC
  - trame MID
    → message contenant les différentes interfaces d'un nœud
  - trame HNA
    → message permettant de déclarer des adresses réseaux en dehors du MANET (utilisé par les passerelles)

OLSR (6.1)

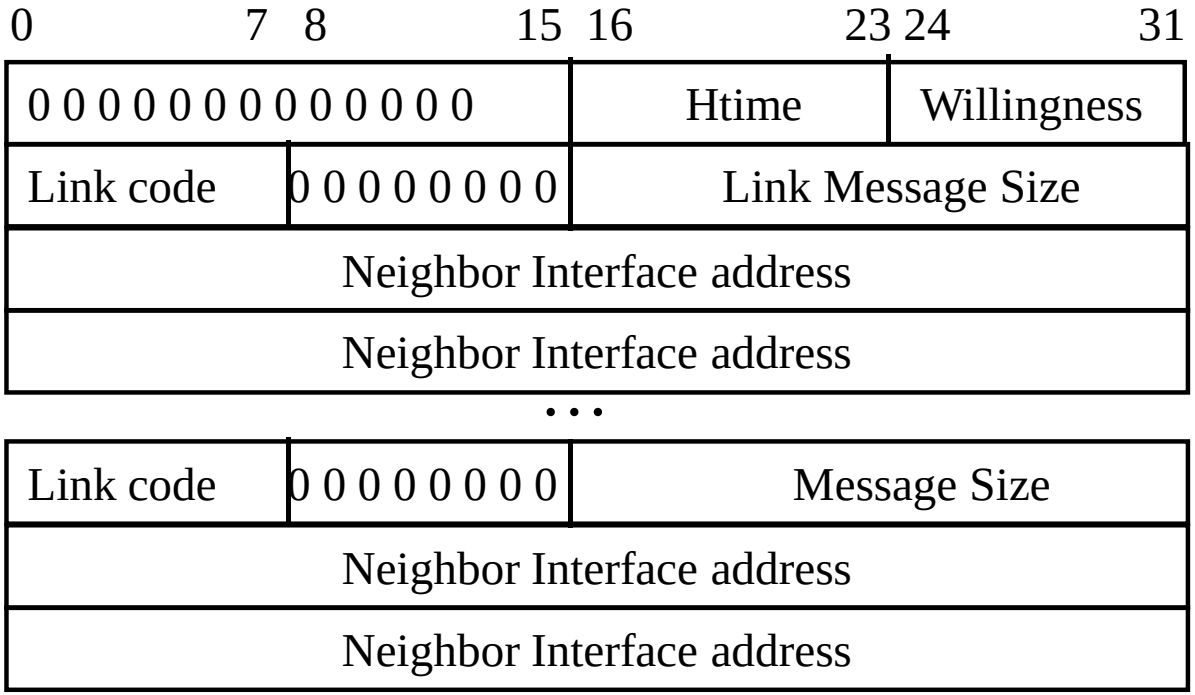
- Format de la couche applicative OLSR



Packet sequence number : augmente de 1 à chaque envoi
Message type : 1 = hello, 2 = TC, 3 = MID, 4=HNA
Vtime : temps de validité du contenu du message
Hop Count : nb de saut d'un message . on incrémente de 1 à chaque réémission
Message : contenu du segment

OLSR (6.2)

- Format du message hello



Htime : temps entre 2 messages hello

Willingness : volontariat pour s'occuper du routage :

0 = Will_never → jamais MPR, 7 = will_always →toujours MPR

Link code :link type (symmétric, lost, asymetric,) + neighbor type (sym,mpr,..)

OLSR (7)

- Conservation de l'énergie
  - Une station mobile peut se mettre en attente
    Si nœud normal, aucun problème
    Si MPR ➔ problème la diffusion des informations

Solution :

- le nœud ne répond plus aux messages "hello", mais continue de diffuser les messages TC
- le nœud va être considéré comme perdu, donc modification de la topologie
- recalcule pour un nœud lointain d'un nouveau MPR, le nœud redevient un nœud normal.

OLSR (8)

Conclusion

- Protocole avec la technique à état de liens
  - Pas de boucle

- Minimisation de la bande passante
  - en minimisant le nombre d'envoie
  - en utilisant la périodicité

- Beaucoup de travail sur ce protocole

  exemple : OOLSR, NOA-OLSR, SMOLSR,…

Protocole réactif

Protocole réactif

- apprentissage des routes à la demande
- permet d'éviter le stockage de routes inutiles
- diminution des messages de contrôle si la topologie ne bouge pas

- ouverture des routes par inondation
  - Sélection de la route la plus courte renvoyée
  - en cas de rupture, récupération de la route par inondation

- autorise
  - lien symétrique
  - lien asymétrique

DSR (1)

- DSR (Dynamic Source Routing)
  Routage à source dynamique

  (RFC 4728)

- 2 fonctions principales :
  - Découverte des routes
  - Maintenance des routes

- Quelques hypothèses
  - Diamètre du réseau < 10
  - lien symétrique/asymétrique

DSR (2)

La découverte d'une route

- une station source S veut envoyer un paquet vers D

  Après consultation de la table de routage, aucune route présente

  → obligation de la créer

- Broadcast d'un paquet ROUTE REQUEST *(requête de route)* avec la destination D

  - Ce paquet contient :

    - l'adresse de la source (initiateur de la demande)
    - l'adresse destination
    - un numéro de requête
    - une liste de tous les nœuds traversés

      (fonction enregistrement de la route)

DSR (3)

- Réception d'un paquet Route Request par le nœud p
  - Si le couple <adresse initiateur, n°requête> existe déjà, ou que p est déjà dans la liste des nœuds traversés, le paquet est ignoré
  - Sinon, p est rajoutée à la liste des nœuds traversés et le paquet est rediffusé.
  - Si p est l'adresse de destination, on renvoie dans un paquet **ROUTE REPLY** l'enregistrement de route qui contient le chemin pour aller de S à D.
    - ➡ Les chemins peuvent être asymétrique

DSR (4)

- Il est nécessaire de trouver un chemin de retour
  - table de routage ?

On inclue dans ROUTE REQUEST de D vers S l'enregistrement de route précédent
(permet d'éviter les boucles infinis)
(fonction de piggybacking)

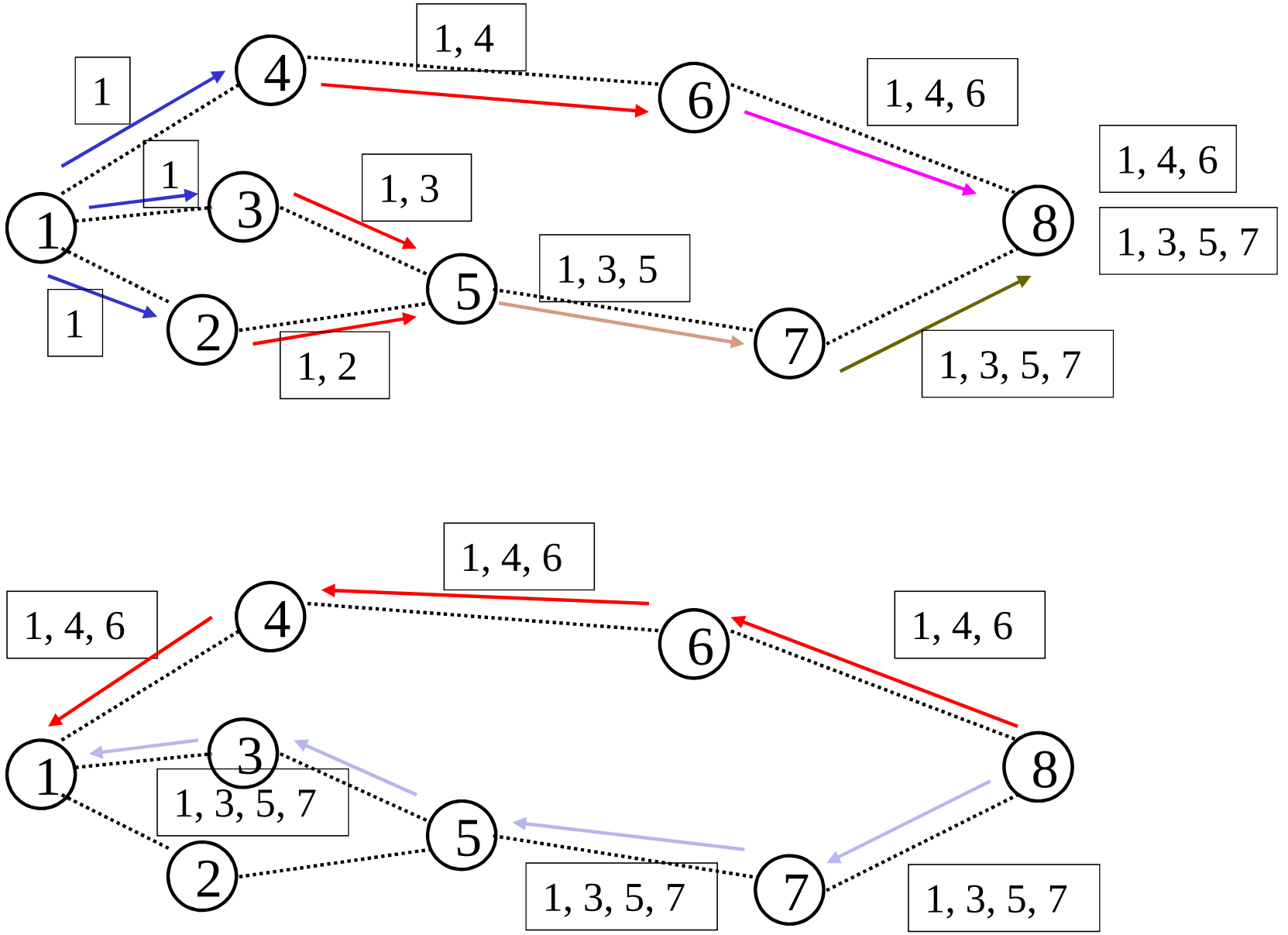
Lorsque le *route reply* revient à la source, on ajoute ce chemin à la table de routage

→ le paquet peut alors être renvoyé

Si plusieurs retours, on choisit le plus courts chemins.

- stockage des autres routes (en cas de problème)
- sans boucle car on a le chemin entier

DSR 4 (bis)


1
4
1, 4
6
1, 4, 6
1, 4, 6
1
1
3
1, 3
1, 3, 5, 7
8
1
5
1, 3, 5
1
2
7
1, 2
1, 3, 5, 7
1, 4, 6
1, 4, 6
4
6
1, 4, 6
1
3
8
1, 3, 5, 7
5
2
7
1, 3, 5, 7
1, 3, 5, 7

DSR (5)

Envoie d'un paquet

- un paquet contient dans son entête tous les nœuds à traverser
- Lorsqu'un nœud reçoit un paquet, il enlève son nom de l'entête et transmet le paquet au nœud suivant

Problème création de route

- Met un certain temps ➔ incompatible avec les timers
- Stockage des différents paquets dans un buffer

  ➡ Send Buffer

  utilisation d'un timer spécifique pour l'attente de ces paquets
- On ne peut pas initialiser trop de route à la fois, car explosion du send buffer

DSR (6)

Optimisation

- Un nœud recevant une route reply peut stocker en cache la route décrite

  ⇒ nécessité d'avoir des liens symétriques

- Un nœud recevant une Route Request regarde dans son cache s'il n'a pas déjà la route pour la destination
  - Si oui, retour d'une route reply avec l'information
    - en faisant attention à ne pas introduire de boucle
  - Si non, rajout dans liste d'enregistrement puis broadcast

Il est possible de imiter le nombre de saut d'un paquet en introduisant un "limitateur de saut" (ex : 6)
Celui-ci est décrémenté à chaque nœud → si 0, le paquet n'est plus broadcasté

DSR (7)

La maintenance d'une route

- La couche 2 permet de savoir si un nœud est toujours accessible et en fonction
  (exemple : en WIFI, émission d'un ack après une trame)

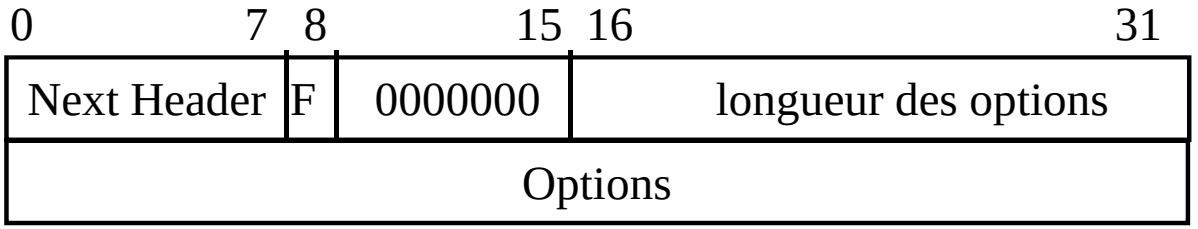
- Si un nœud tombe, envoie d'un paquet ROUTE ERROR au nœud source
  - possible car route connu entre le nœud qui voit l'erreur et le nœud source (du fait du route reply)
  - Le nœud source peut :
    - Soit choisir une autre route s'il en a une en cache
    - Relancer une découverte de route

DSR (8)

- Sauvetage
  - En cas de route error, le nœud ayant découvert le problème peut regarder dans son cache s'il existe une autre route
    - si oui, envoie de route error, puis modification de l'entête du paquet par le nouveau chemin, et mise ne place d'un bit : salvage
      (pas plus d'un sauvetage → boucle)
    - si non, procédure normal

DSR (9)

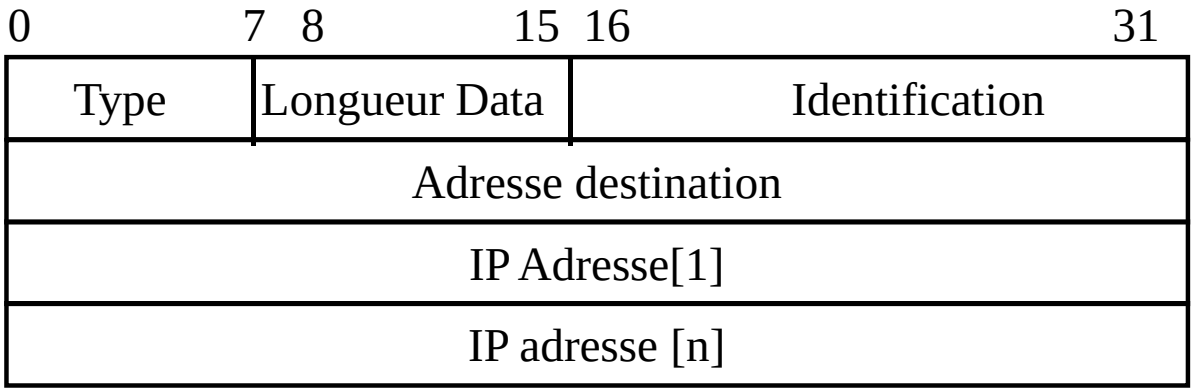
- Format de la couche réseau DSR
  - Ajout au niveau des options de la couche 3



Next Header : protocole encapsulé de niveau 4
bit F : Flow State header : 0
Option : messages du protocole DSR (Route Request, Route Reply, …)

Exemple : Option Route request :

Type=2



➡ A chaque nœud, on rajoute une adresse IP

DSR (10)

- Conclusion
  - protocole réactif introduisant un faible surcoût au niveau message
  - Mise en place d'un buffer d'émission le temps de trouver une route
  - Pas de multicast possible

AODV (1)

- Ad Hoc On –Demand Distance Vector
  - Basé sur l'algorithme de Bellman-Ford
    - vecteur de distance, mais en évitant les boucles
    - assez proche de DSDV et de DSR.
  - Protocole en cours de normalisation par l'IETF
    RFC 3561
  - Protocole réactif
    - supporte l'unicast et le multicast
    - Plutôt développé pour les liens symétriques
    - 2 fonctions : la découverte des routes
      la maintenance des routes

AODV (2)

Table de routage

- Elle contient :
  - l'adresse de destination
  - Le nombre de sauts (de nœuds) nécessaire pour atteindre la destination
  - Le prochain saut
  - Le numéro de séquence qui correspond à cette destination
  - Le temps d'expiration de l'entrée de la table
  - La liste des précurseurs

Une destination D est rajoutée dans la table de routage seulement lorsque la station veut envoyer un message à D

➡ temps d'expiration arrive à 0 → élimination de la destination

AODV (3)

Découverte d'une route

- Broadcast d'un paquet ROUTE REQUEST *(requête de route)* avec la destination D
  - Ce paquet contient :
    - un numéro de broadcast (RREQ ID)
    - l'adresse de la source (initiateur de la demande)
    - l'adresse destination
    - le n° séquence correspondant à la dernière fois que cette destination était dans la table de routage (si jamais, n° séquence = 0 et flag = U)
    - le n° séquence à utiliser actuellement pour le retour

Si pas de retour avant NET_TRAVERSAL_TIME, n° de broadcast est incrémenté de 1, et une RREQ est relancée

un seul n°broadcast par station (pour toutes les destinations)

AODV (4)

Réception d'une RREQ

- Si destination,
  - Vérification si (n° RREQ, id_source) n'a pas été déjà reçu
  - mise à jour de sa table de routage
    - insertion du chemin vers le destinataire
    - mise à jour du n° séquence,
    - mise à jour du temps d'expiration, …
  - Renvoie d'une Route Reply (RREP) en unicast

- Si nœud intermédiaire,
  - vérification si (n° RREQ, id_source) n'a pas été déjà reçu
  - cas 1 : Pas de route disponible vers destination D
    - Mise à jour de la table de routage pour le retour (on suppose les liens symétriques)
    - Diffusion de la demande en ajoutant 1 au nombre de saut

AODV (5)

- cas 2: route disponible vers la destination D
  - Vérification du n°séquence de l'ancienne route (stocké dans le paquet RREQ) < n° séquence de la route trouvée dans la table de routage
  - Renvoie d'une Route Reply en unicast avec le nombre de saut correspondant à la distance jusqu'à destination
  - Mise à jour de la liste des prédécesseurs pour cette destination

➡ **Mais, il faut avertir la destination qu'un nœud cherche à le joindre (car autrement table de routage incomplète)**

Emission d'une RREQ "gratuite"
vers la destination (flag G)

(champ de la trame RREQ)

AODV (6)

Mise à jour table de routage

- Lors de la réception d'une RREQ
- Lors de la réception d'une RREP
  - lorsqu'un n° séquence est plus grand que celui de la table
  - Même numéro de séquence, mais nombre de saut plus petit
- A chaque réception d'un paquet, remise à jour du délai 'expiration de la route (temps restant + ACTIVE_ROUTE_TIMEOUT)

- Lors de la réception d'une RERR
- Lorsqu'un timeout de validité de la route expire
  - La route est éliminée ou marquée inutilisable

Un chemin figurant dans la table de routage est considéré comme :

un chemin actif

AODV (7)

Route reply

- Lorsqu'une station Z reçoit une route reply, on renvoie l'information à un nœud stocké dans la table de routage

  (du fait du passage précédent de RREQ)

  - Mais si lien unidirectionnel ➡ ECHEC

Problème :

- Si le nœud source renvoie une RREQ, forte probabilité de se retrouver dans le même cas

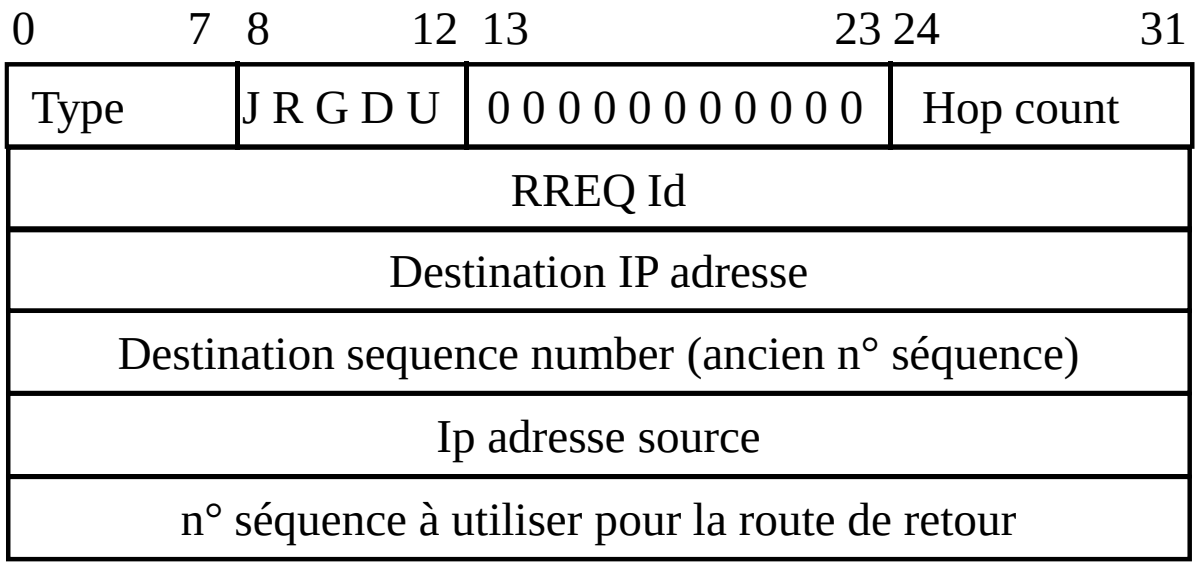
- Si le nœud Z reçoit après le lien unidirectionnel une autre route valide → information peut être effacée si chemin plus long

Solution : Si un nœud s'aperçoit d'un lien unidirectionnel, ce lien est mis en blacklist, c'est à dire que l'on ignorera par la suite toutes les RREQ qui en proviennent.

AODV 7.1

- Trames de contrôle RREQ

type = 1
car RREQ

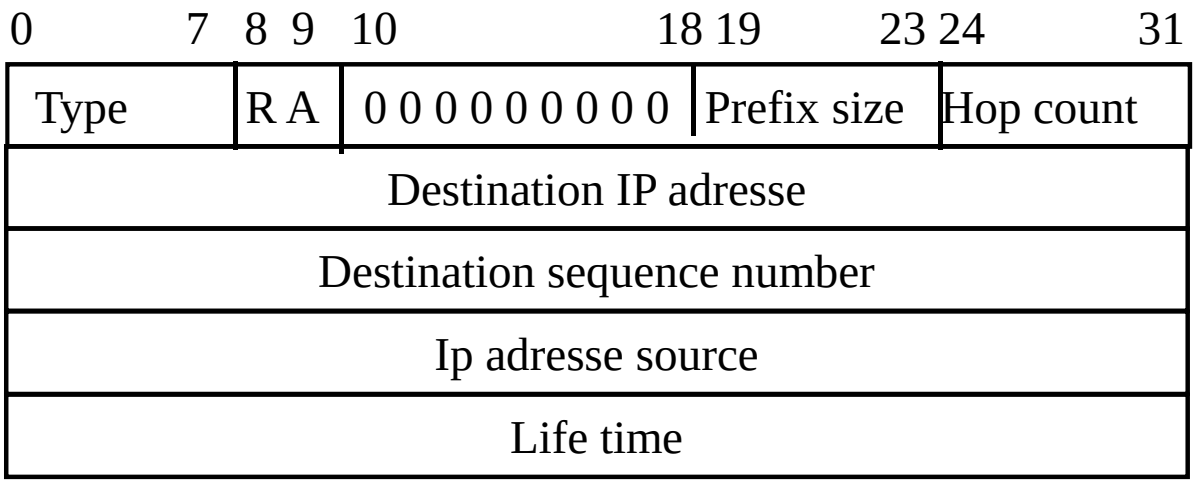


bit J : join flag : réservé multicast
bit R : repair flag : réservé multicast
bit G : gratuité du RREQ
bit D : seulement le destinataire peut renvoyer un RREP
bit U : Pas de destination sequence number valide → 0

AODV 7.2

- Trames de contrôle RREP

type = 2
car RREP

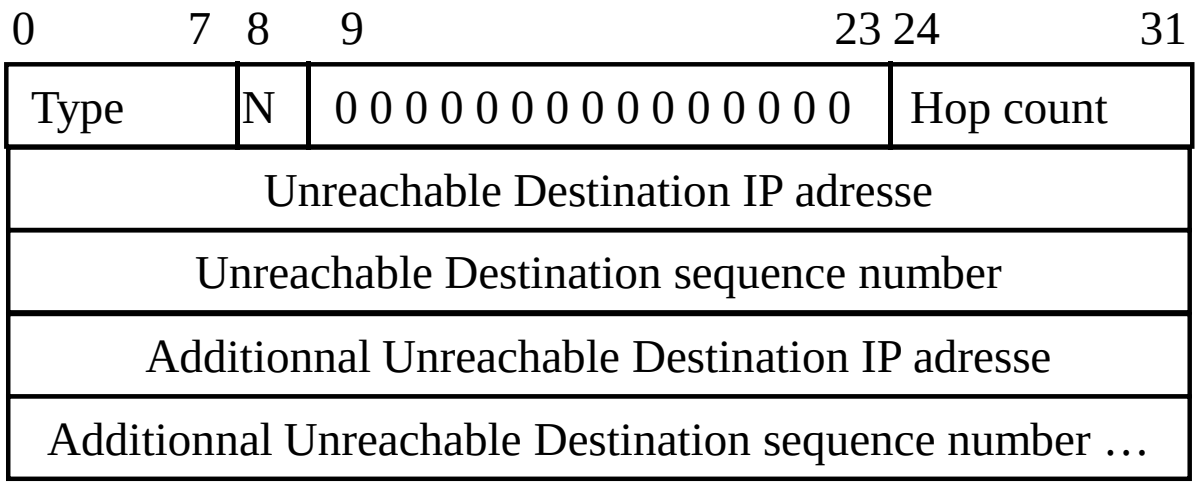


bit R : repair flag : réservé multicast
bit A : demande d'une confirmation d'acquittement
prefix size : pour englober les sous-réseaux (si nécessaire)
destination IP adresse : l'adresse de la destination pour la route
adresse source : adresse IP de la station ayant fait le RREQ
life time : durée de vie de la route

AODV 7.3

- Trames de contrôle RERR

type = 3
car RERR



bit N : route réparée

AODV (8)

Limitation :

- Chercher une route prend du temps
  - maximum RREQ_RATELIMIT début de connexion par secondes
  - Tous les paquets sont bufferisés
  - Lorsqu'une route est trouvée, tous les paquets sont envoyés
    sinon, tous les paquets sont effacés et la couche transport est prévenu

  - Par défaut, RREQ_RATELIMIT = 10

AODV (9)

Optimisation

- Minimisation du chemin parcouru par les paquets RREQ
  - le TTL au niveau IP de la 1er RREQ est mis à TTL_START
    (généralement TTL_START = 1)
  - Si pas de réponse RREP, alors TTL = TTL + TTL_INCREMENT
    (généralement TTL_INCREMENT = 2)
  - Le TTL augmente jusqu'à TTL_THRESHOLD (7)
  - 2 essais maximum avec le TTL max avant de déclarer la route inaccessible
  - Si perte d'un chemin dans la table de routage, alors
    TTL = distance connu dans la table +TTL_INCREMENT
    (on autorise une recherche de chemin plus long)

AODV (10)

La maintenance d'une route

- Emission
  - Pas de communication réelle : envoie de paquets "hello" vers les nœuds appartenant à des chemins actifs
    (TTL =1 pour ne pas aller loin)

- Réception
  - Remise à jour du timer d'expiration quand :
    - réception d'un paquet hello
    - réception d'un paquet de données venant d'un nœud en direction d'un autre

    (temps restant + ACTIVE_ROUTE_TIMEOUT)

AODV (11)

- Route Error
  - En cas de disparition d'un nœud,
    - on recherche l'ensemble des destinations qui étaient accessibles par ce nœud
    - on incrémente le n° séquence de chaque destination devenue invalide
    - on envoie une RERR à tous les prédécesseurs de ces destinations
    - Mise à jour de la table de routage
  - Réception d'une RERR
    - on met à jour la table de routage
    - on envoie l'information aux prédécesseurs

On met en place un timer DELETE_PERIOD. Si RAS, alors effacement

AODV (12)

- Sauvetage !!
  - Le nœud le plus proche du nœud qui disparaît peut essayer de trouver une nouvelle route
    → émission d'une RREQ
    - Mise en attente des paquets dans un buffer

Si une réponse valide, envoie d'une RERR avec le bit N pour indiquer un changement dans la route, mais que celle-ci reste valide

→ remise à jour des routes et du nombre de saut pour toutes les stations

AODV (13)

- Conclusion
  - protocole réactif utilisant les vecteurs de distance
    - Convergence assez rapide
  - n° séquence permet d'éviter les boucles
  - un mécanisme existe pour traiter le multicast
  - Prêt à intégrer l'IPv6

  - Routage utilise trames UDP sur le port 654

TORA (1)

- Temporally-Ordered Routing Algorithm
  - Basé sur la notion de **DAG**
    (Directed Acyclic Graph)
  - et sur l'inversion de lien ("Link Reversal")

- But : minimiser les effets de changement de topologie

- Plusieurs chemins possibles pour une destination

- Algorithme réactif
  - → création de routes à la demande
  - → liens symétriques
  - → une horloge globale (tous les nœuds ont le même temps)

TORA (2)

- 3 fonctions principales
  - Création d'une route
  - Maintenance de la route
  - Elimination des routes

- Structure associée à un nœud : *un quintuplet*
  - *t : temps logique de défaillance*
  - *oid : identificateur du nœud de référence*
  - *r : bit de réflexion (0 = normal, 1= réflexion)*
  - *d : entier indiquant le niveau de référence*
  - *i : identificateur unique du nœud*

  (t, oid,r) est appelé : niveau de référence
  (d,i) : valeur dans ce niveau de référence ou hauteur

TORA (3)

- Quelques remarques :
  - Chaque nœud a un identificateur unique
  - Les liens sont symétriques
    - nœud i orienté vers le nœud j

      alors i a un lien descendant vers j (downstream link)
      j a un lien ascendant vers i (upstream link)

  - Chaque nœud connaît ses voisins et garde un vecteur des niveaux de référence de chacun d'eux.

TORA (4)

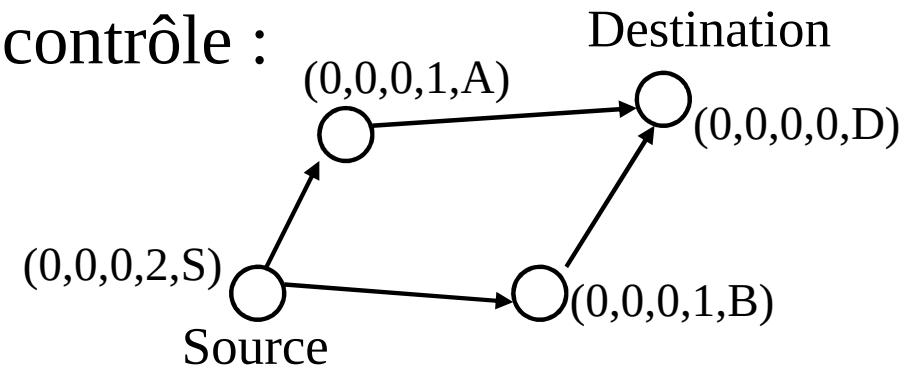
Création d'une route

- **But : créer un DAG** (DAG orienté destination)
  - utilisation de la valeur du niveau de référence (hauteur)
  - utilisation de l'ordre lexicographique

Le DAG sera orienté de la plus grande hauteur vers la plus petite. Si même hauteur, alors ordre lexicographique

On cherche un DAG entre une source et une destination
→ la destination doit avoir la hauteur la plus petite

Utilisation de deux trames de contrôle :
- Query (QRY)
- Update (UPD)



Au départ, les nœuds sont à : (-,-,-,-,x)

TORA (5)

- *Création d'une route* : envoi d'une requête QRY (diffusion)
  et mise à 1 du drapeau : route-required

- Algorithme lors de la réception d'une requête
  - Si le drapeau route-required est déjà à 1, abandon car route en cours de recherche
  - Si pas de chemin sortant vers la destination, alors mise à 1 de route-required et diffusion d'une query
  - Si un chemin sortant vers la destination existe et que la valeur est nulle, la hauteur de référence est mis au minimum des hauteurs de référence des voisins +1, puis on envoie un UPD.
  - Si un chemin sortant existe vers la destination et que la valeur est non nulle, on regarde si on a déjà renvoyé un UPD depuis que la valeur est non nulle. Si oui, abandon, si non, envoie d'un UPD.

TORA (6)

Algorithme lors de la réception d'un Update

- Si le drapeau route-required est à 1, la hauteur du niveau de référence est mis au minimum des hauteurs des niveaux de référence des voisins +1 et on renvoie un UPD.

- Autrement, on met à jour la hauteur des voisins ainsi que le niveau de référence si nécessaire et on regarde si le DAG reste inchangé.

  ➡ on vérifie qu'il existe toujours une sortie possible (downstream link)

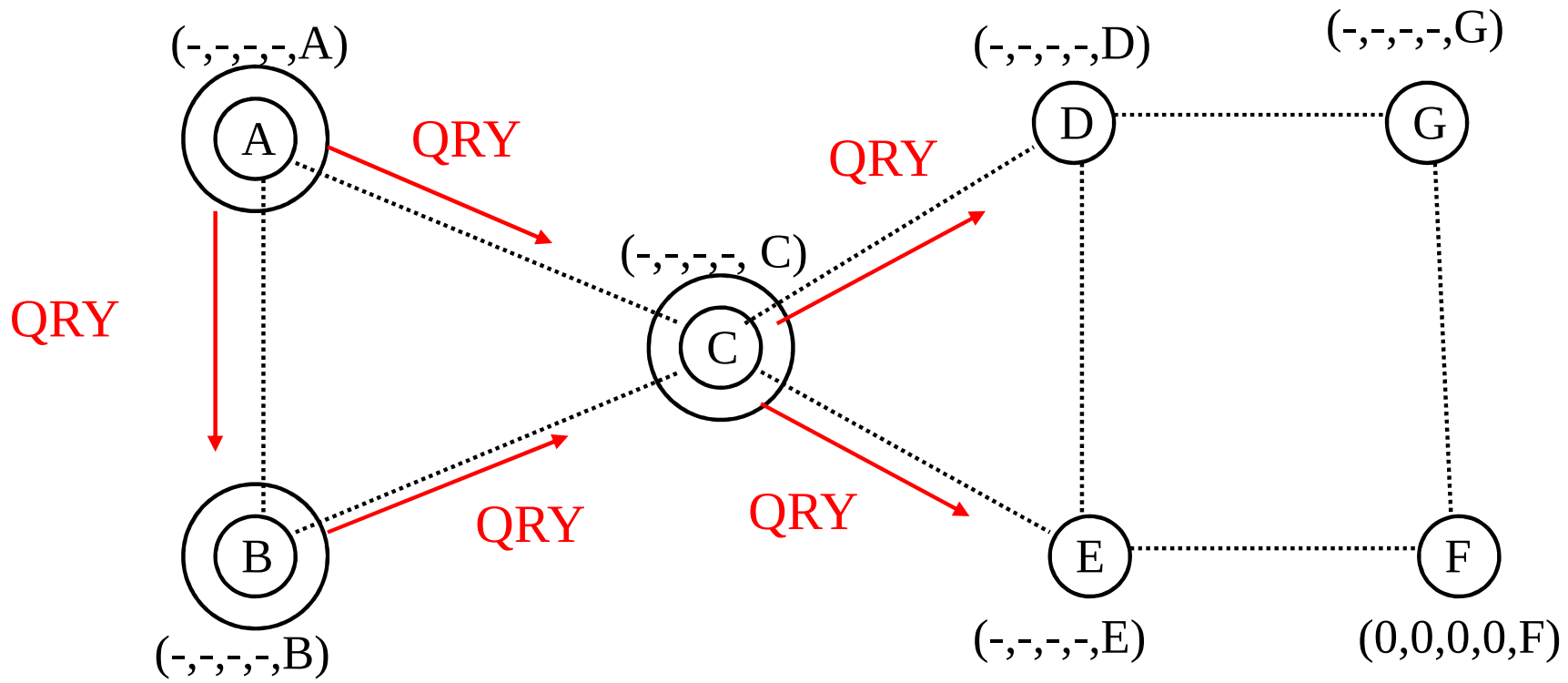
- Si oui, on ne fait rien
- Si non, mise à jour du DAG → maintenance de routes

TORA (6 bis)

Ordre lexicographique : A > B> C > D>…

But : chemin entre A et F

destination F d'où quintuplet : (0,0,0,0,F)

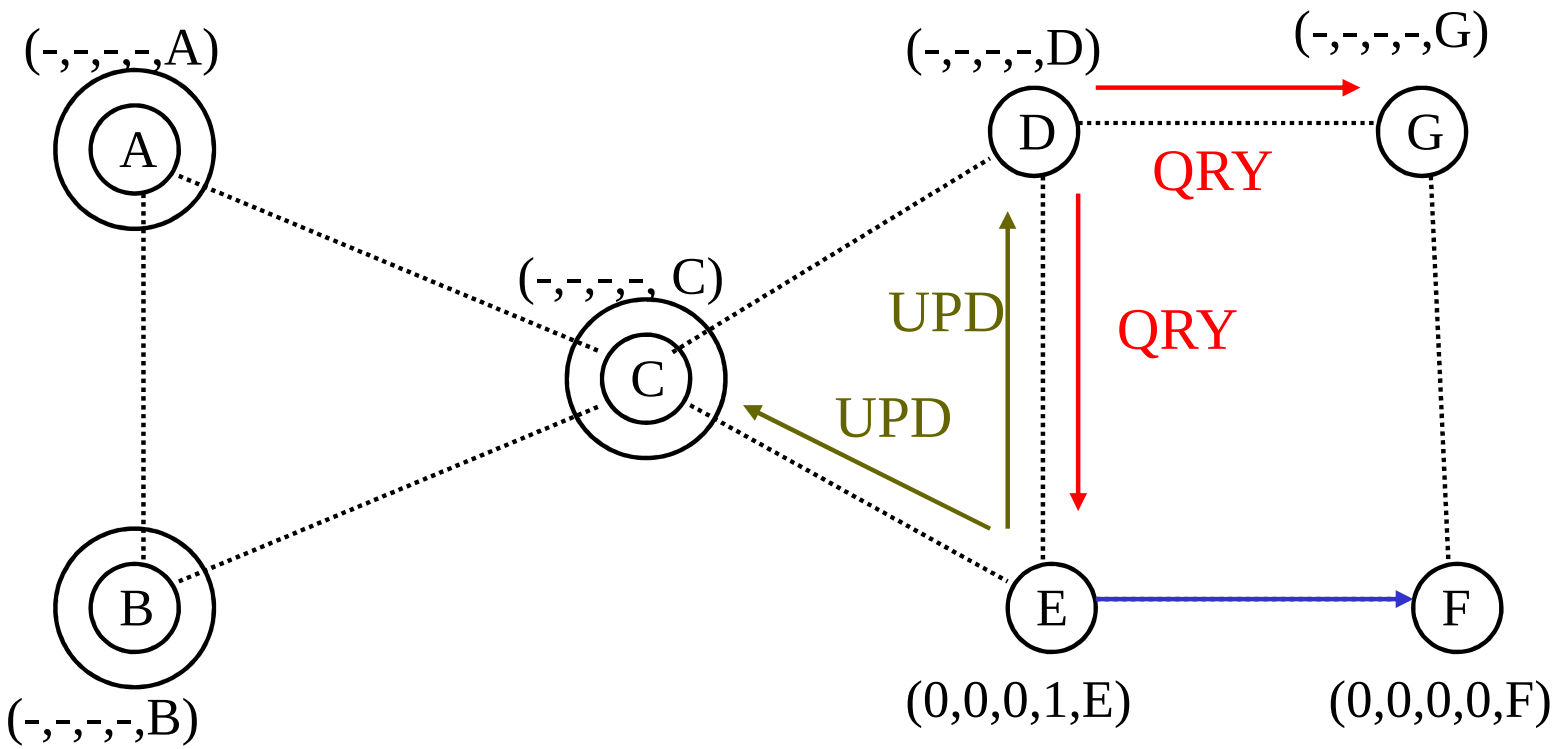


TORA (6 bis)

Ordre lexicographique : A > B> C > D>…

But : chemin entre A et F

destination F d'où quintuplet : (0,0,0,0,F)

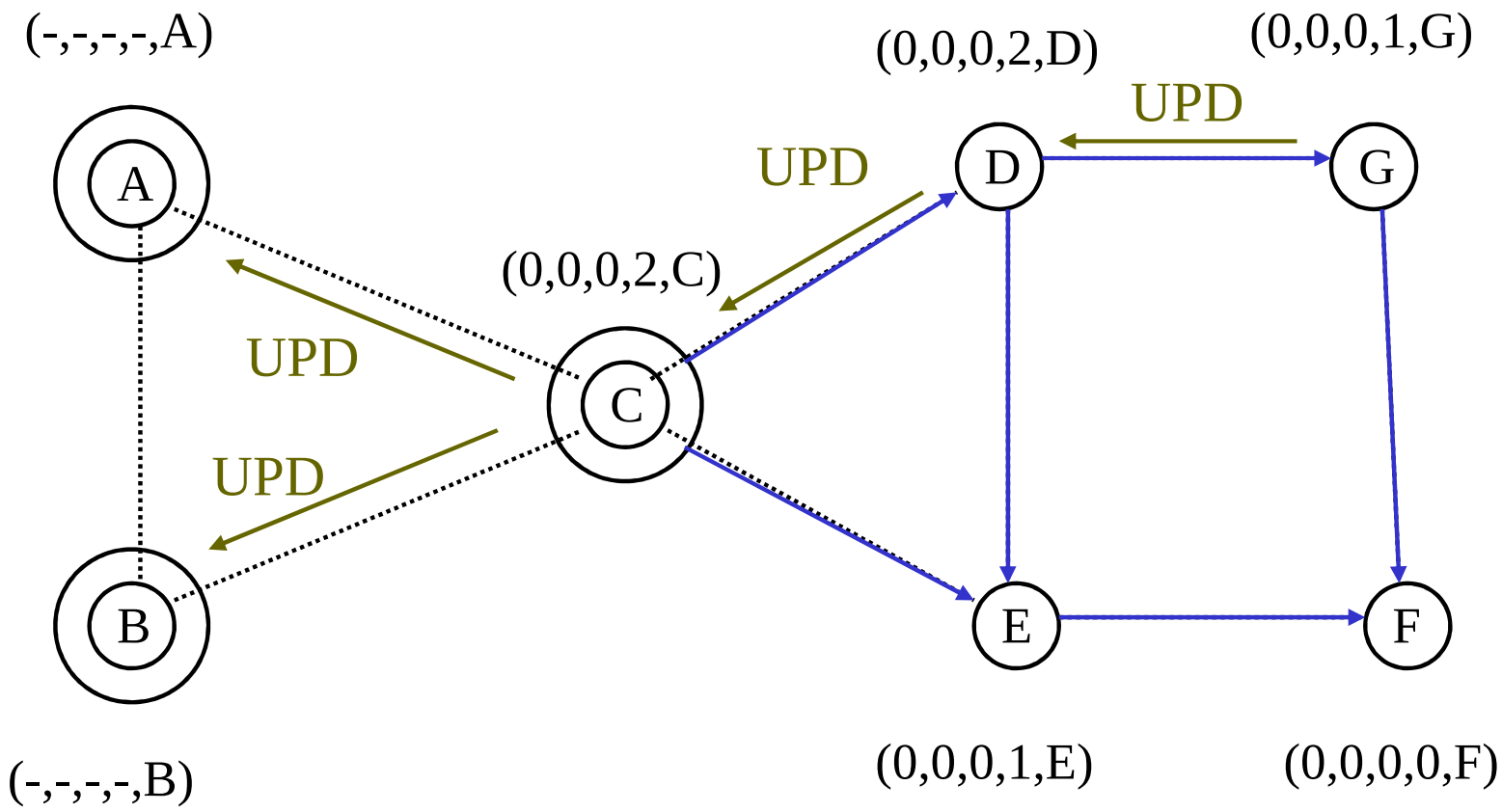


TORA (6 bis)

Ordre lexicographique : A > B> C > D>…

But : chemin entre A et F

destination F d'où quintuplet : (0,0,0,0,F)

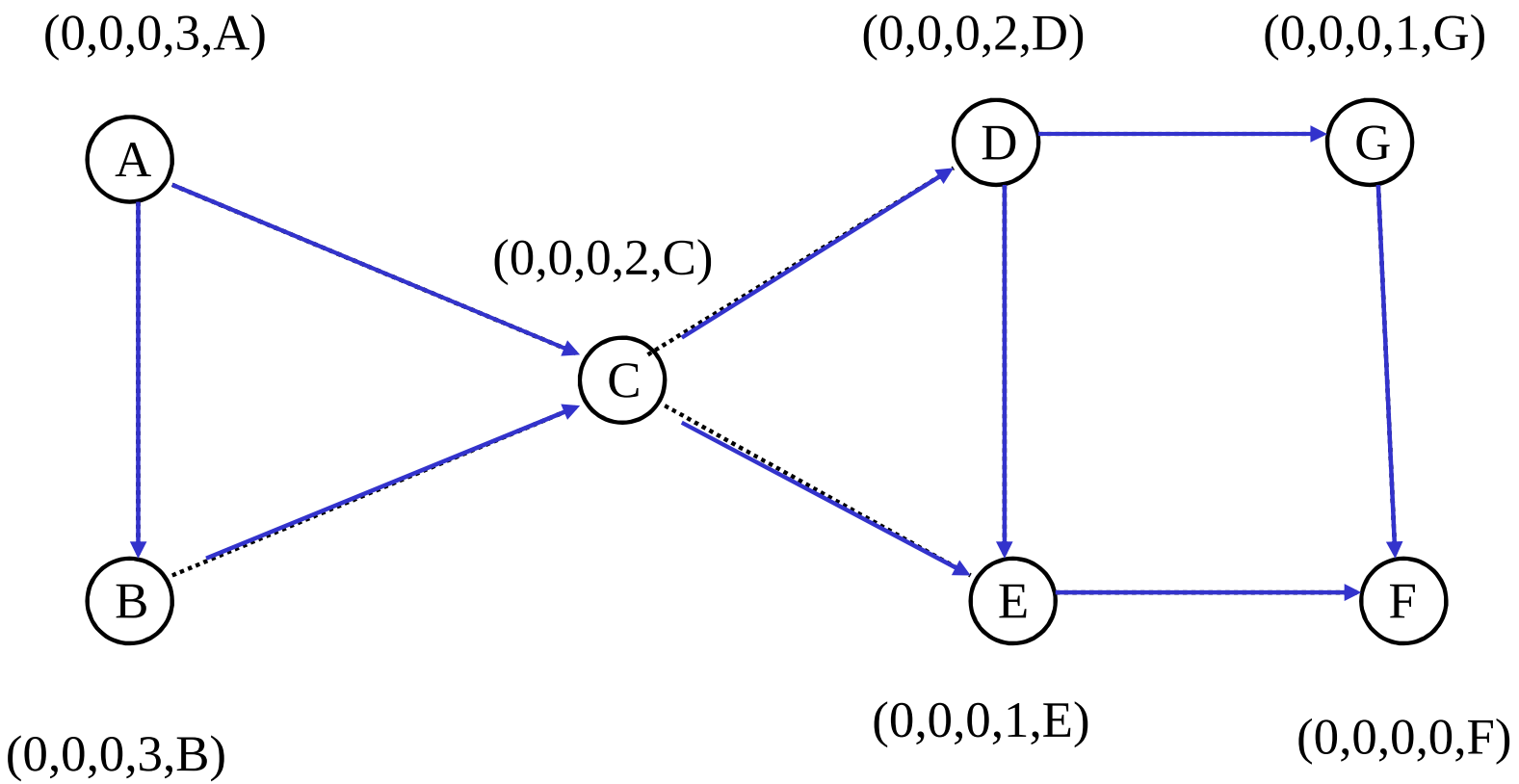


TORA (6 bis)

Ordre lexicographique : A > B> C > D>…

But : chemin entre A et F

destination F d'où quintuplet : (0,0,0,0,F)



TORA (7)

Maintenance d'une route

→ valable seulement pour les nœuds ayant une hauteur (niveau de référence) non nulle

- Perte du dernier lien descendant (downstream link)
  (plus de lien de sortie …)
- 5 cas différents pour la mise à jour du quintuplet

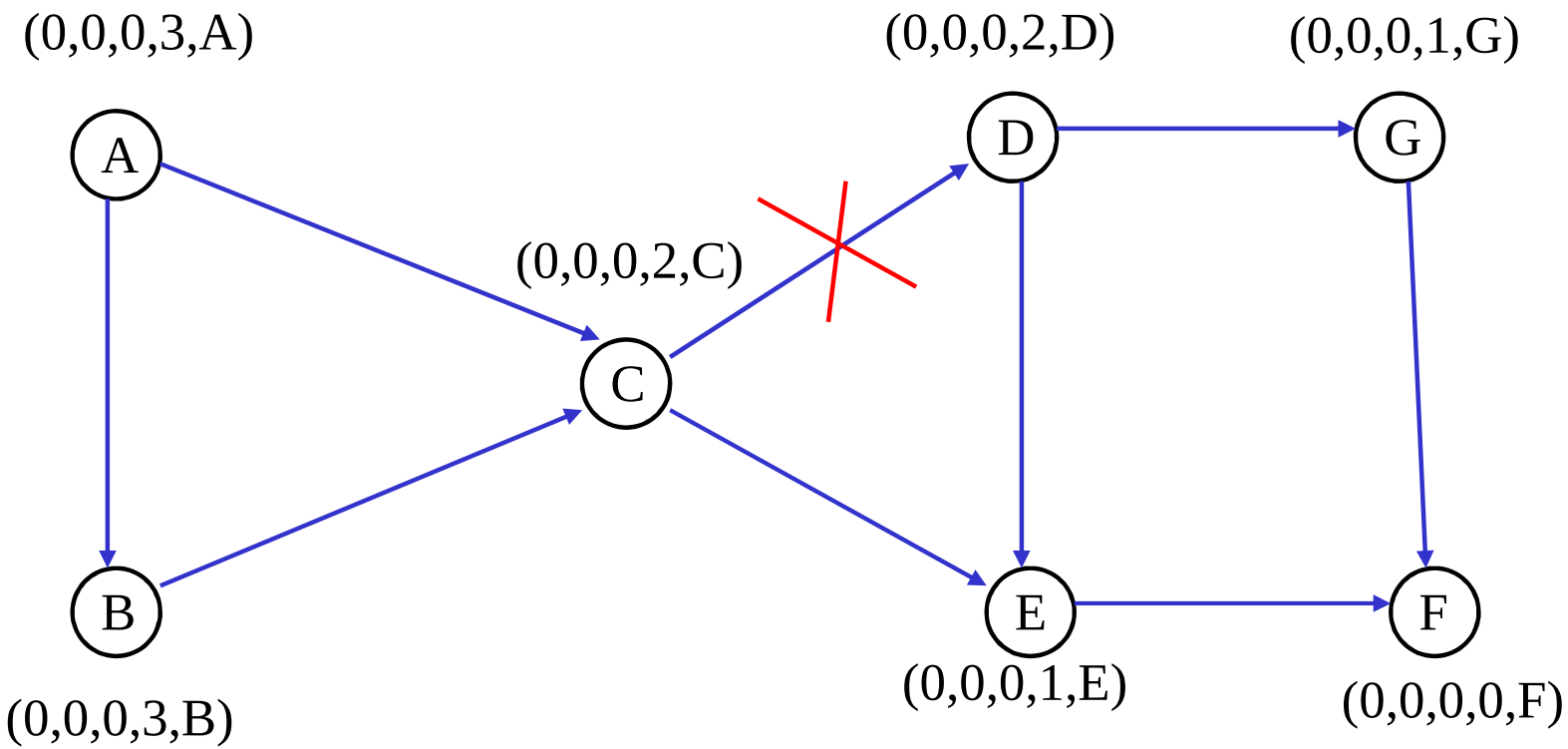
1. Plus de lien dû à une cassure, création d'un nouveau niveau de référence : (t,i,0, 0, i) avec t , temps de la cassure
   (sous condition qu'il existe encore un lien ascendant, autrement nœud mis à (-,-,-,-,i))
2. Plus de lien descendant suite à une mise à jour venant d'un UPD et un niveau de référence de ses voisins plus grand , on recopie le niveau de référence et la hauteur du niveau de référence est mis au minimum des hauteurs des niveaux de référence des voisins -1 (on obtient la plus petite hauteur du niveau de référence, elle peut être négative)

TORA (8)

3. Plus de lien descendant suite à une mise à jour venant d'un UPD et le niveau de référence de ses voisins est égal avec bit de réflexion (r) =0, on met r =1 et la hauteur du niveau de référence à 0.
4. Plus de lien descendant suite à une mise à jour venant d'un UPD et le niveau de référence de ses voisins est égal avec bit de réflexion (r) =1 et oid=i, début d'effacement de la route, nœud=(-,-,-,-,i)
5. Plus de lien descendant suite à une mise à jour venant d'un UPD et le niveau de référence de ses voisins est égal avec bit de réflexion (r) =1 et oid !=i, création d'un nouveau temps de référence (t,i,0,0,i)

Après la mise à jour 1,2, 3 et 5, diffusion d'un UPD

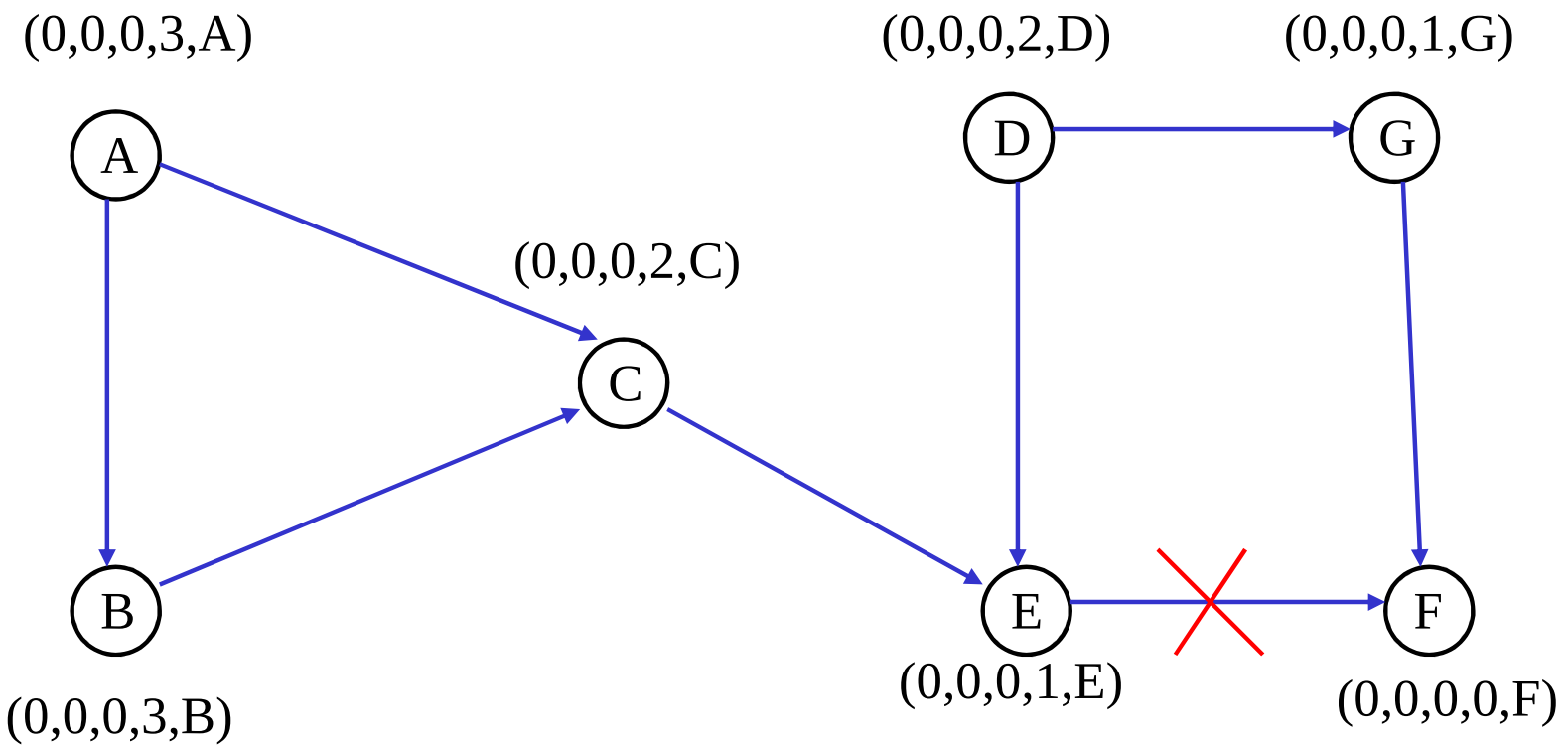
TORA (8.1)



Rien ne se passe car C a toujours un lien descendant

Intérêt de TORA dans un milieu dense : beaucoup de connexions descendantes et ascendantes…donc peu de changement

TORA (8.2)



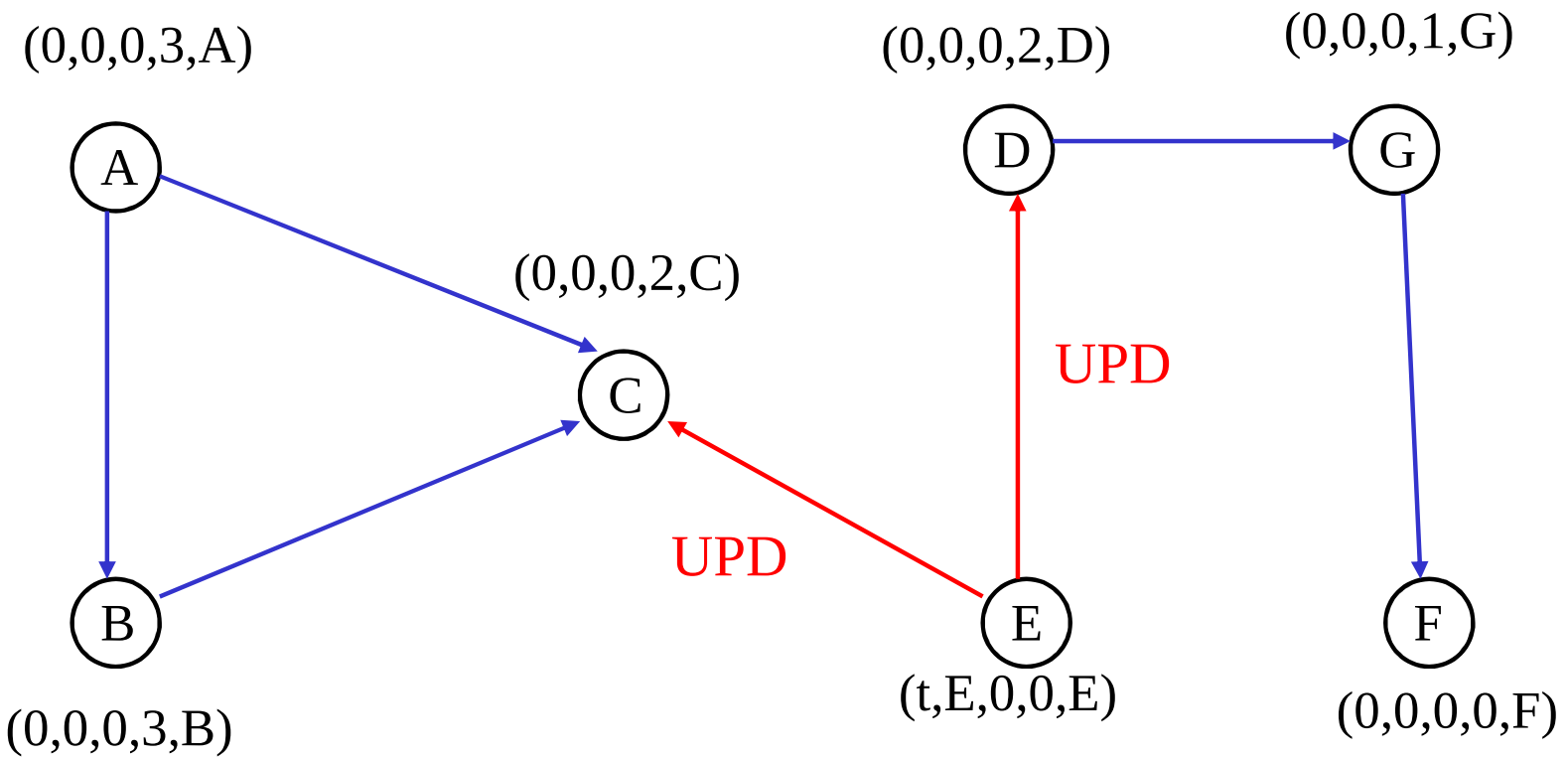
La liaison E-F casse…

Problème : E n'a plus de lien descendant

➡ Mise en route de la maintenance de route (lien renversé)

règle 1 appliquée

TORA (8.3)

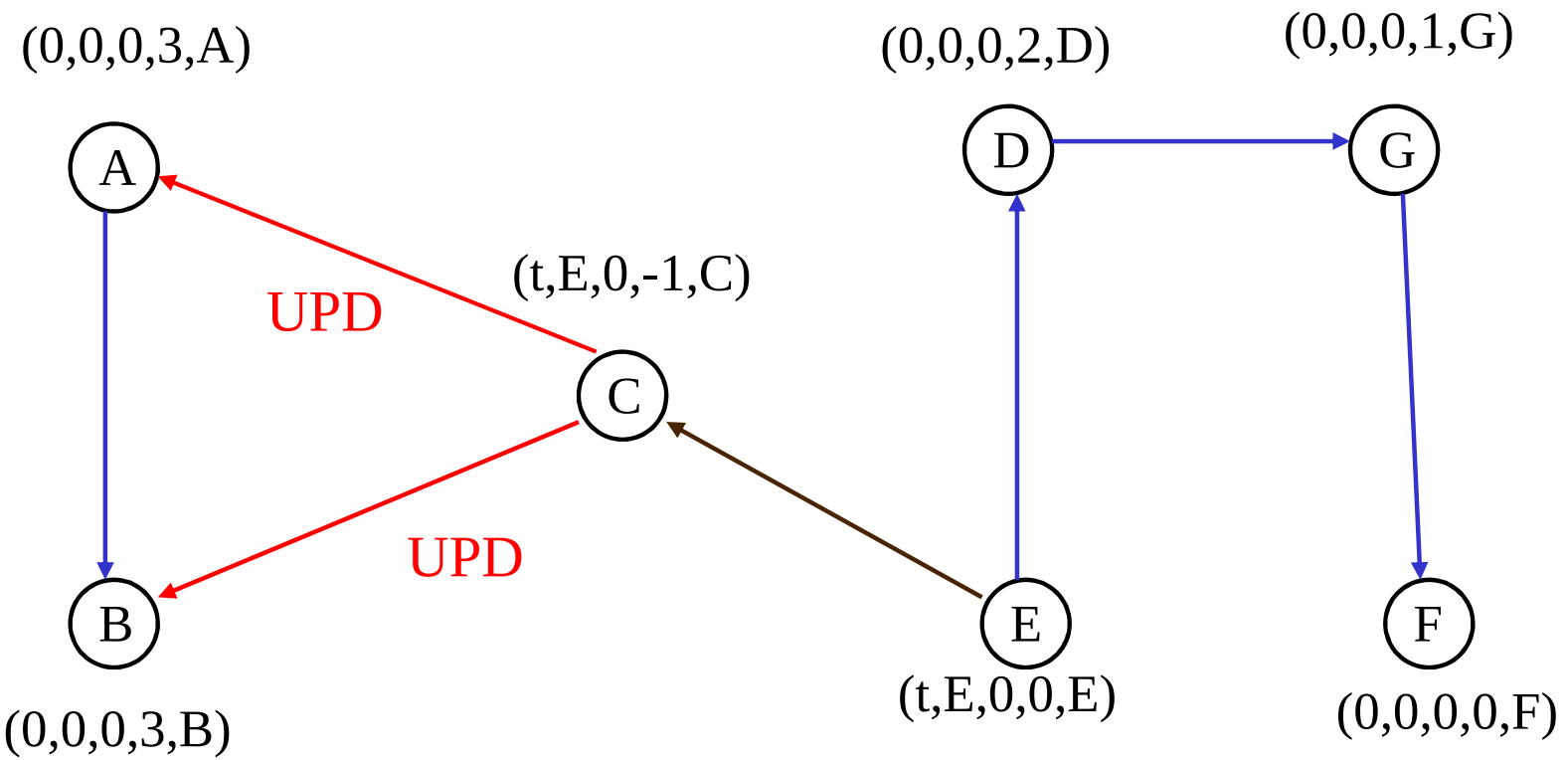


D a un lien descendant,
rien à faire

C n'a plus de lien descendant

→ Règle 2

TORA (8.4)

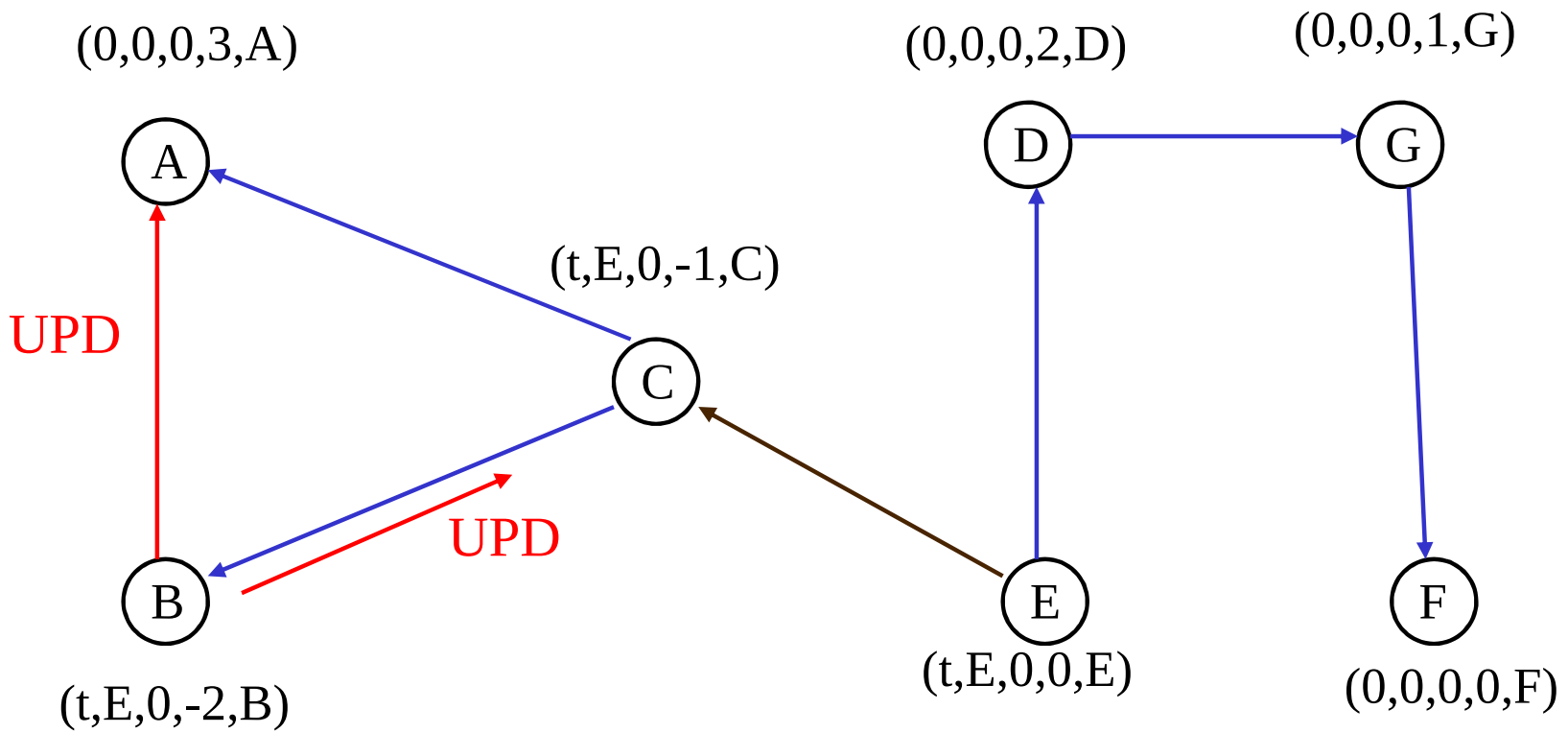


A a un lien descendant,
rien à faire

B n'a plus de lien descendant

→ Règle 2

TORA (8.5)

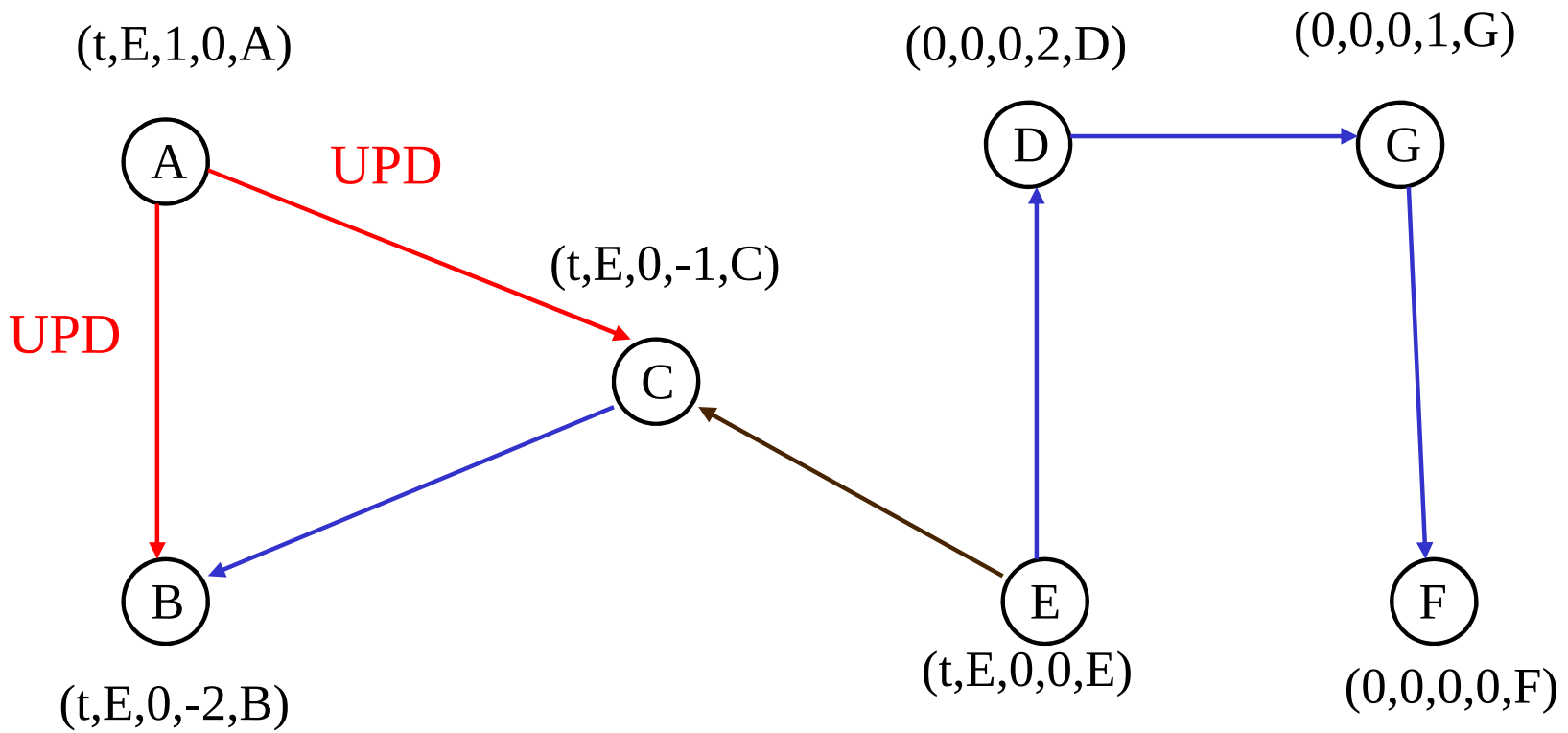


C a un lien descendant, rien à faire

A n'a plus de lien descendant

➡ Règle 3

TORA (8.6)

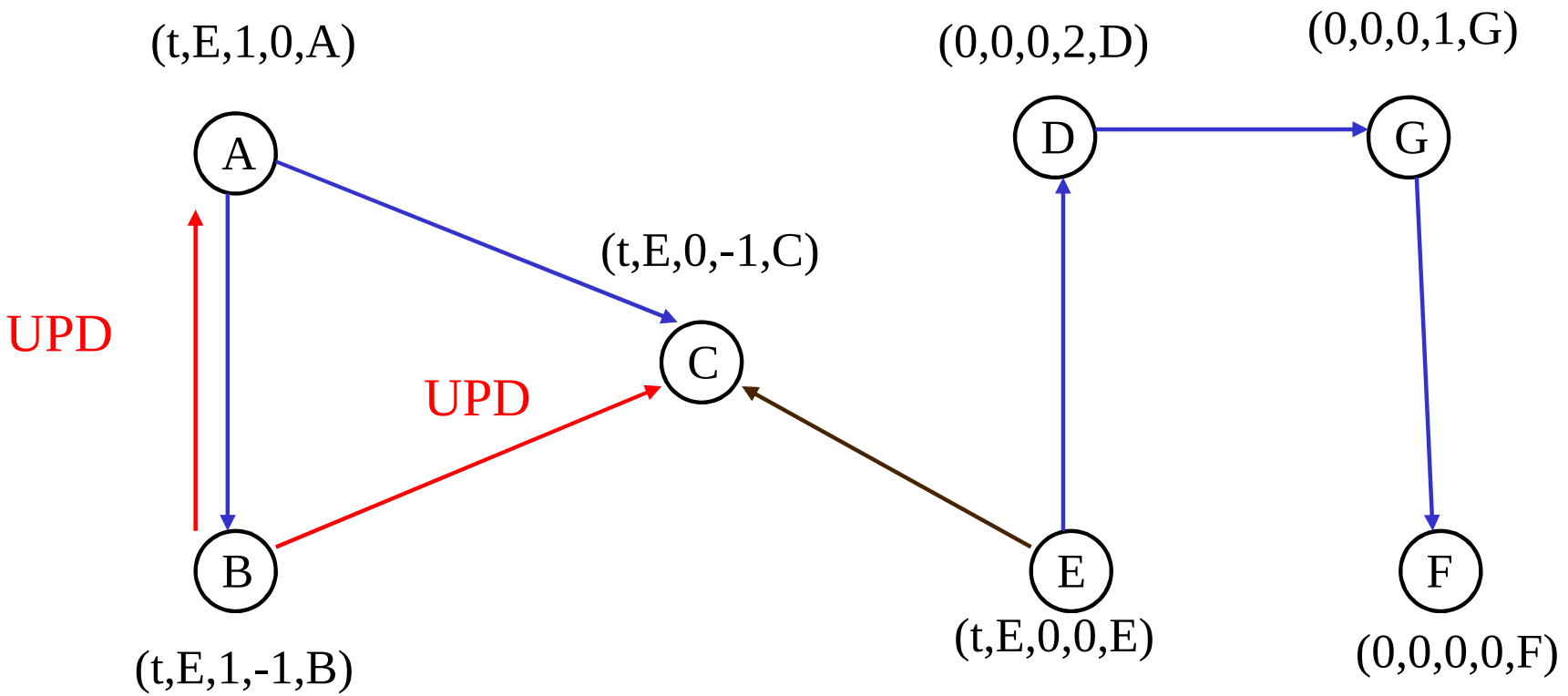


C a un lien descendant, rien à faire

B n'a plus de lien descendant

➡ Règle 2

TORA (8.7)

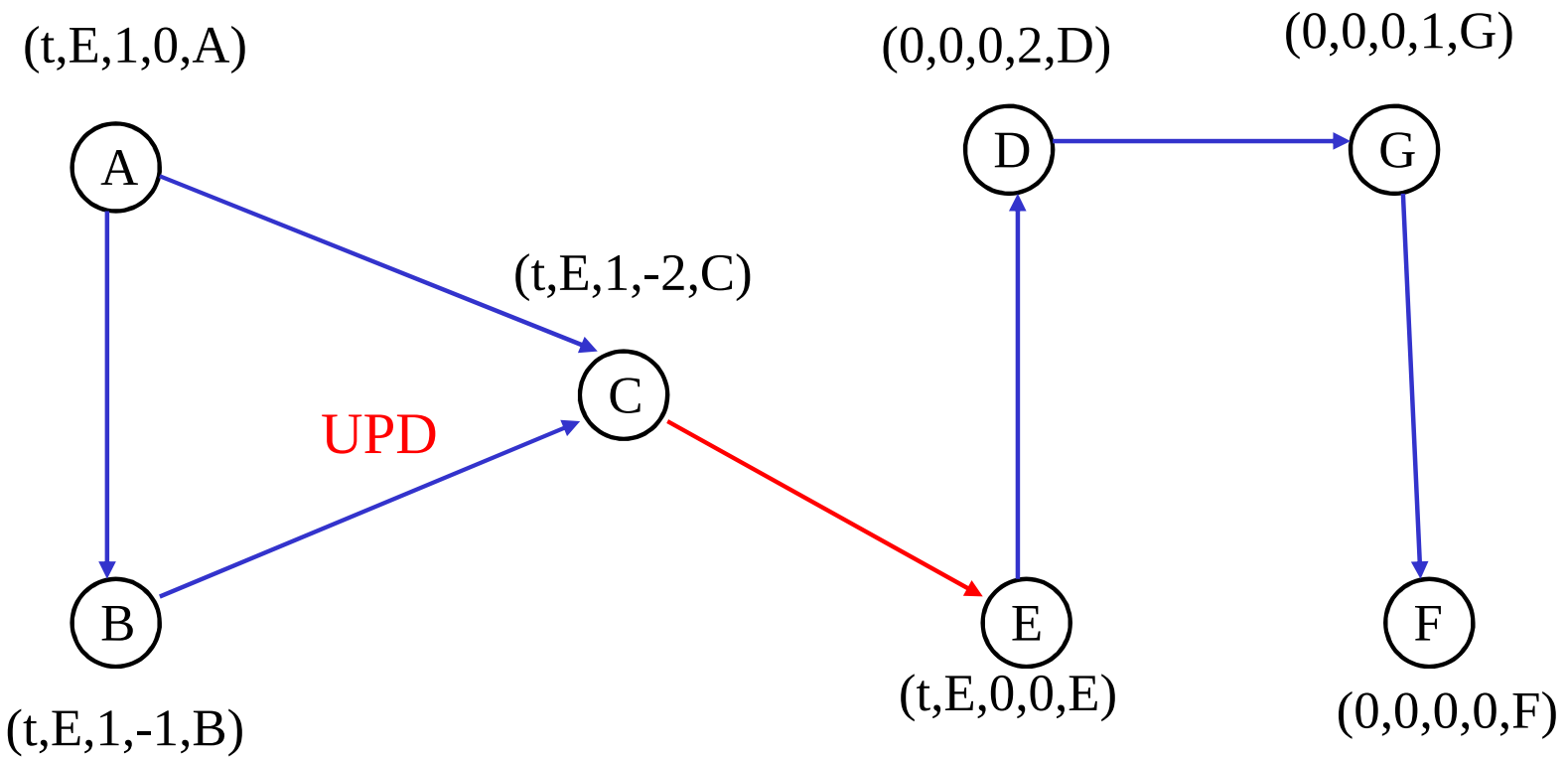


A a un lien descendant, rien à faire

C n'a plus de lien descendant

➡ Règle 2

TORA (8.8)



A a un lien descendant, rien à faire

C n'a plus de lien descendant

➡ Règle 2

TORA (9)

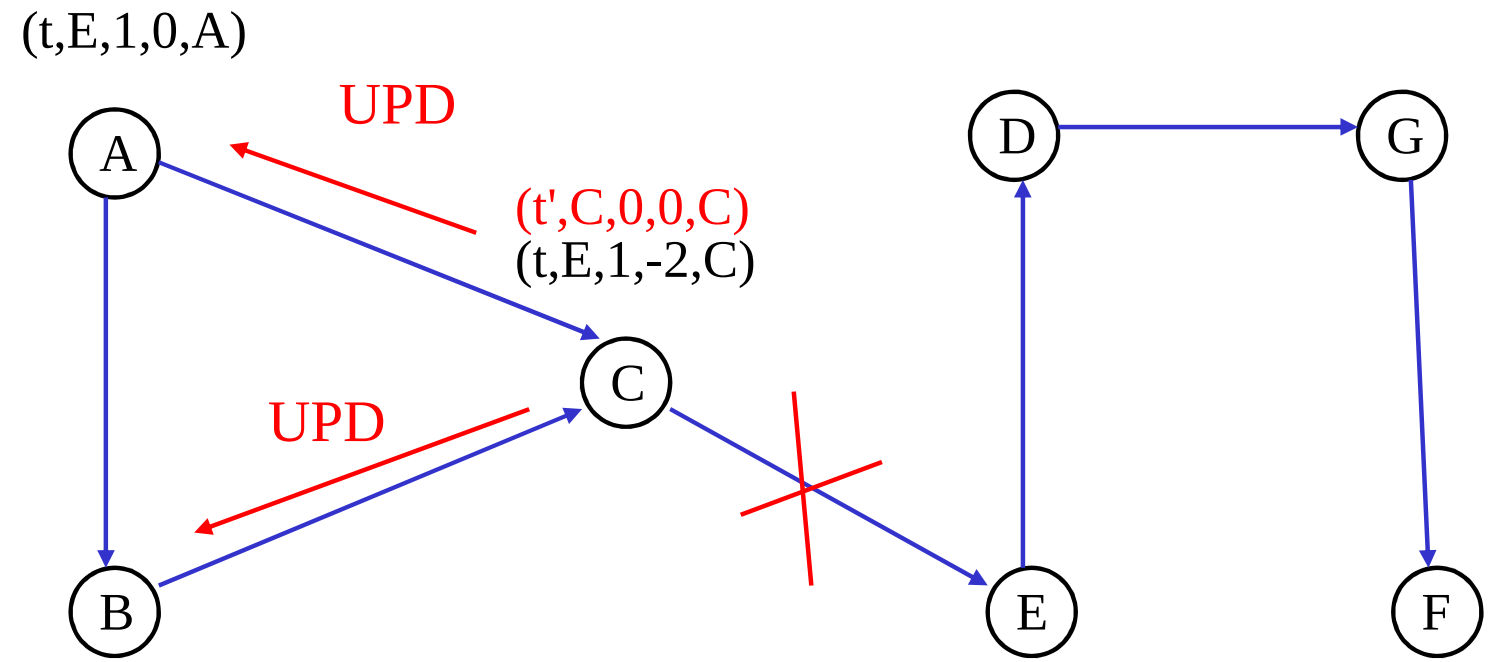
- Effacement d'une route
  - Lorsqu'on utilise la règle 4 de la mise à jour,

    ➡ apparition de partitions

    (une destination n'est plus atteignable)
  - Envoie de la trame de contrôle CLR (clear) contenant le nom de la destination et le niveau de référence
  - Lors de la réception d'un paquet CLR, quintuplet mis à :

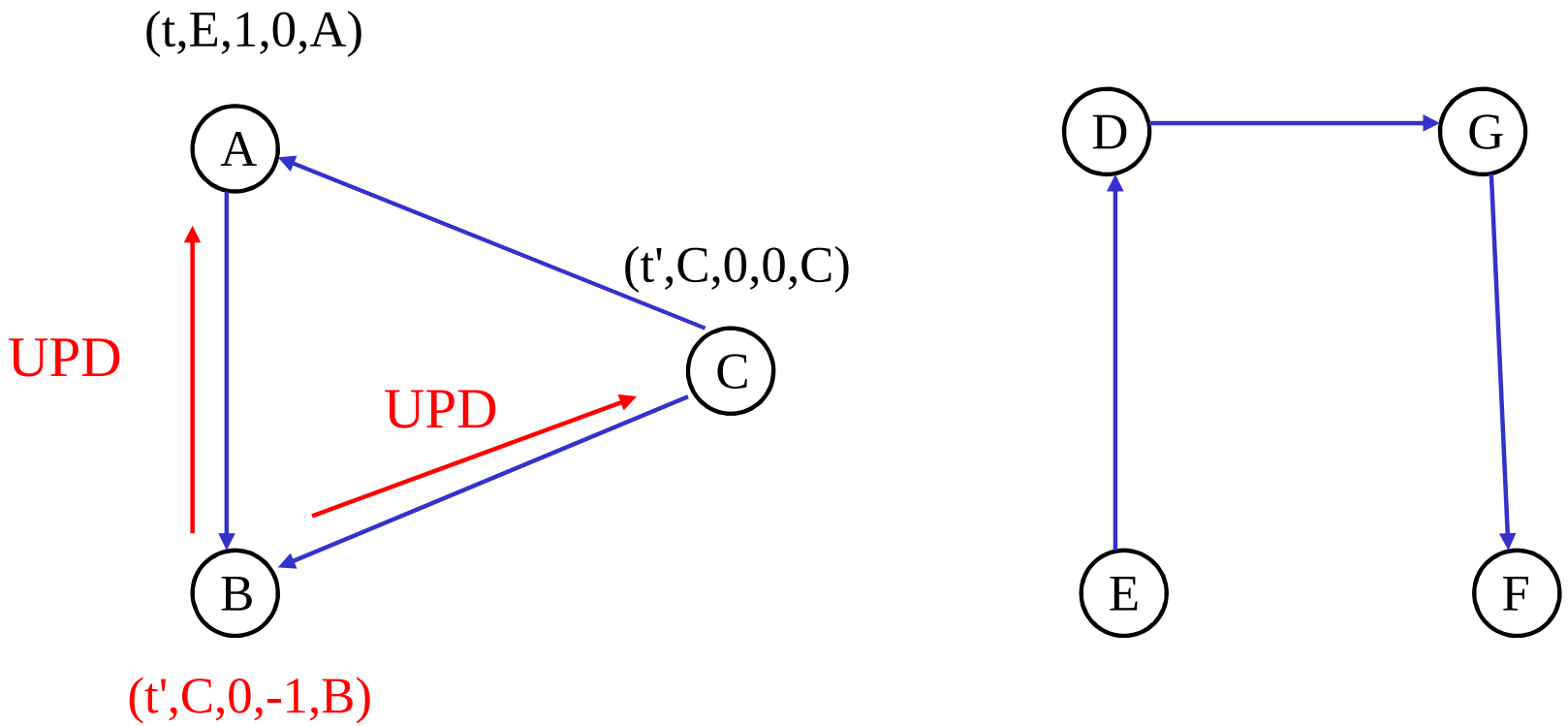
    (-,-,-,-,i)

TORA (9.1)



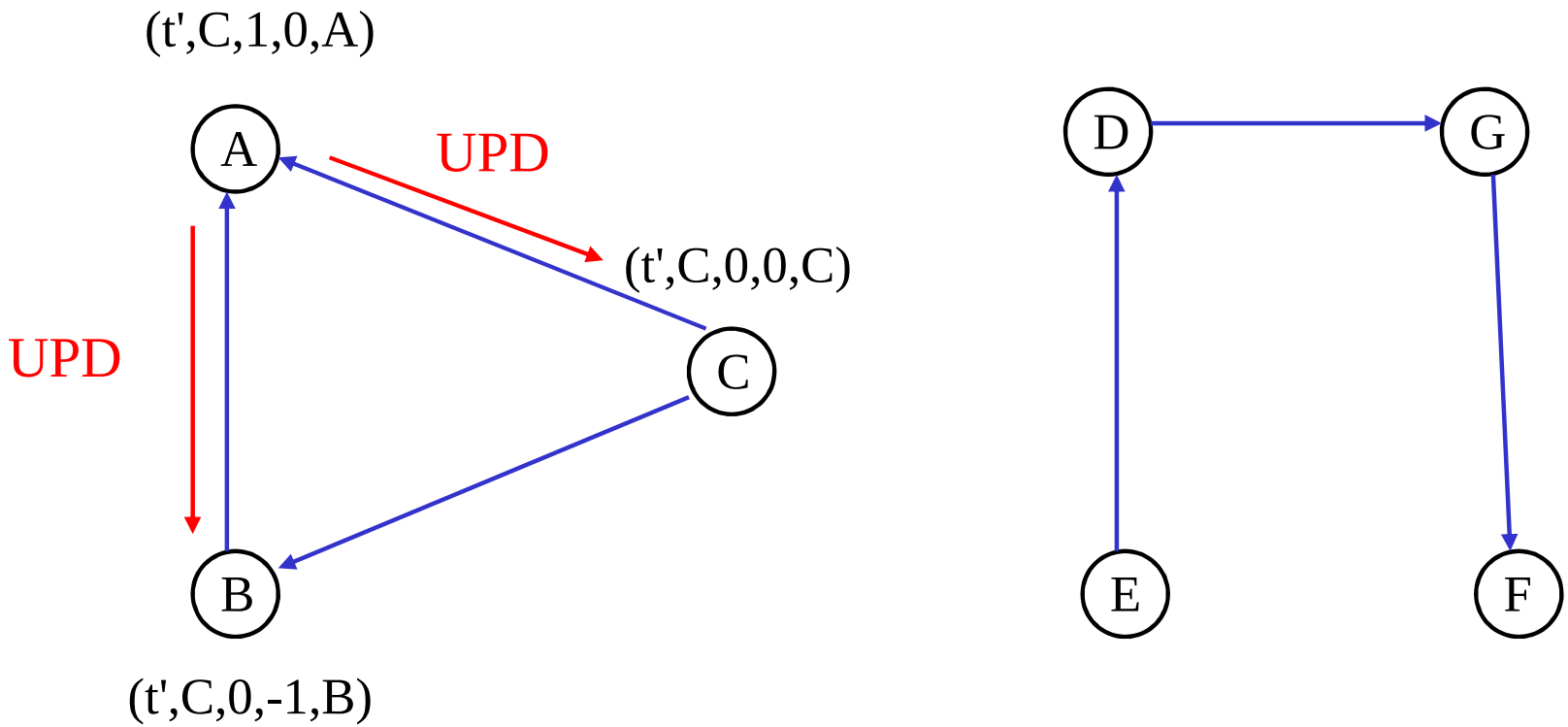
C n'a plus de lien descendant, envoie d'un update

TORA (9.2)



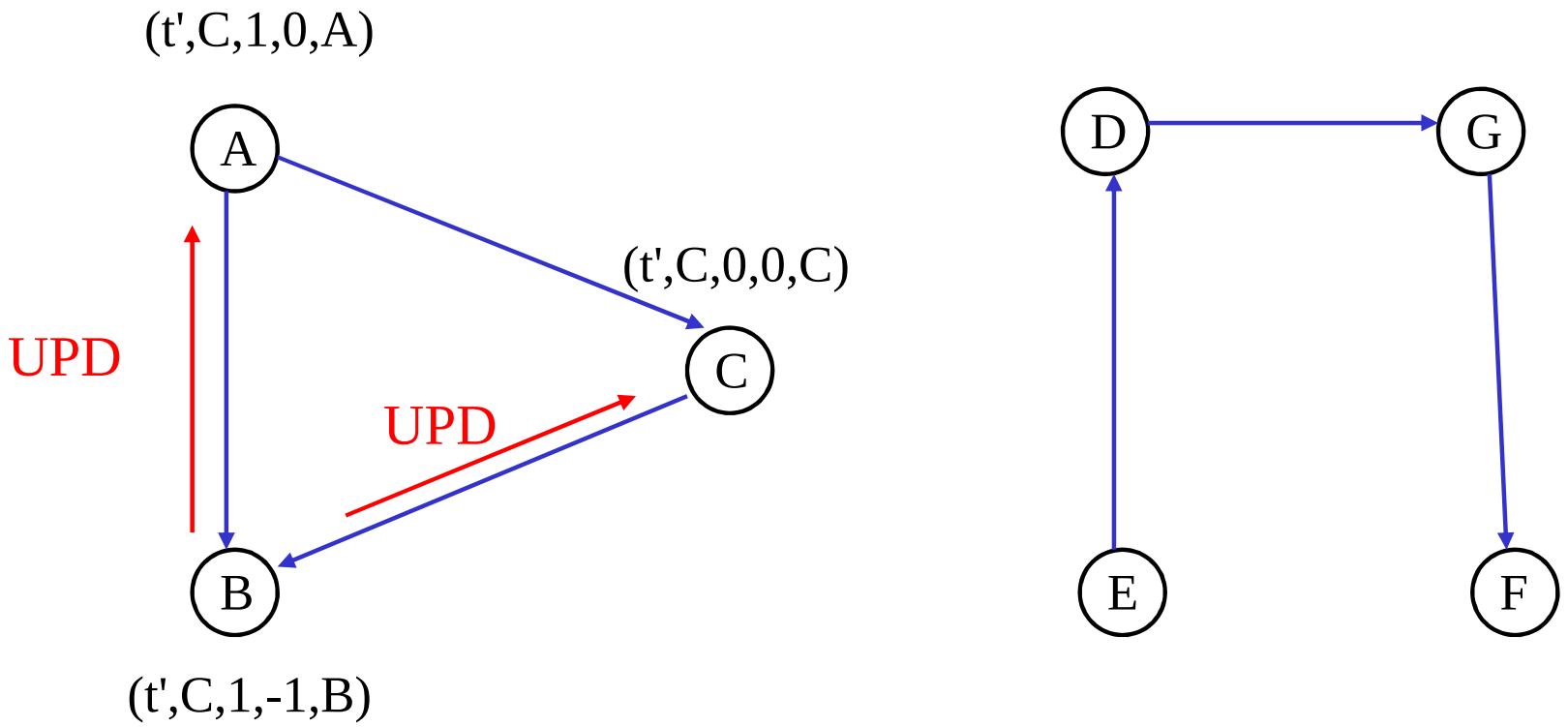
B utilise la règle 2 car plus de lien sortant

TORA (9.3)



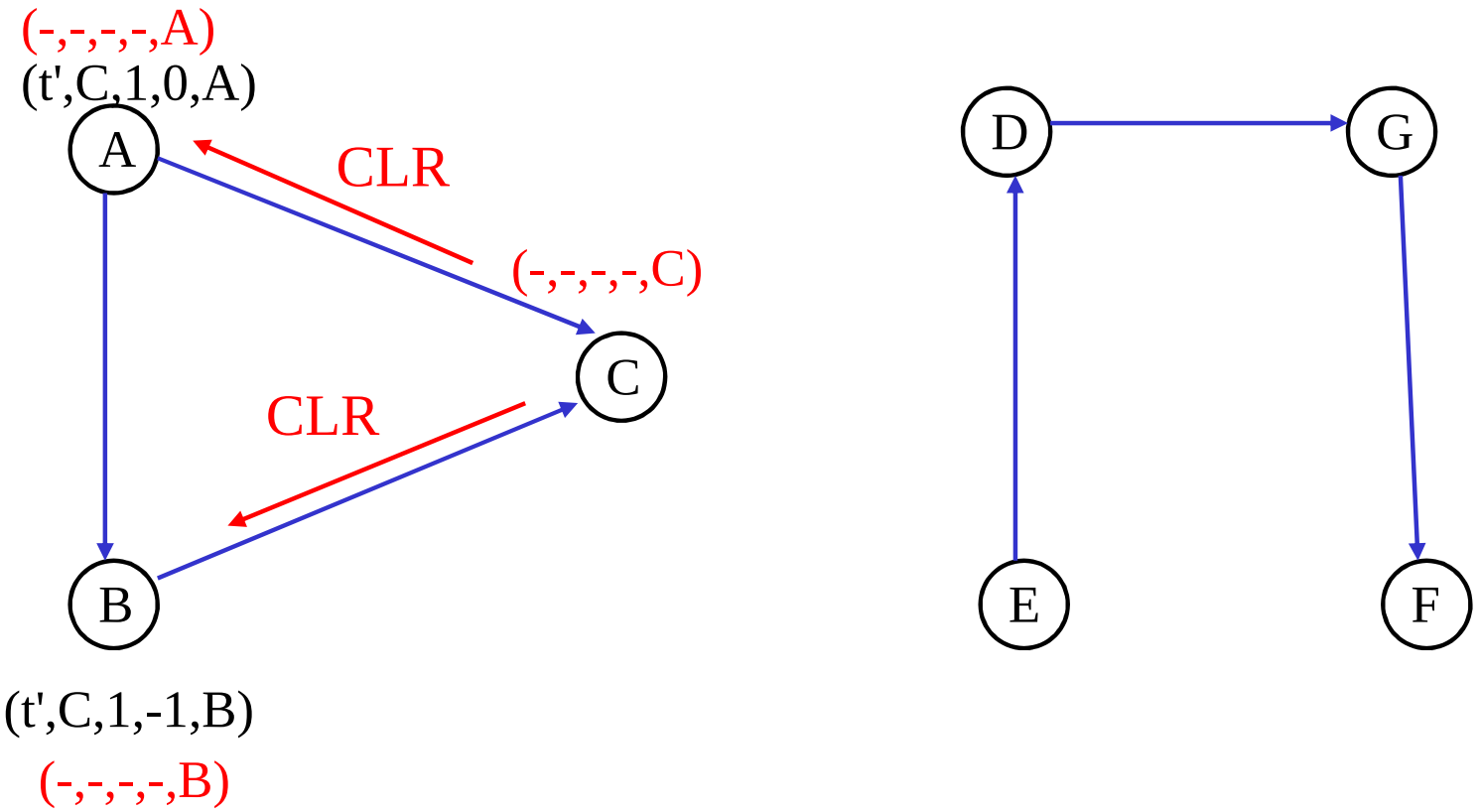
A utilise la règle 3 car plus de lien sortant

TORA (9.4)



B utilise la règle 2 car plus de lien sortant

TORA (9.5)



C utilise la règle 4 car plus de lien sortant

Effacement de la route

TORA (10)

- Conclusion
  - Une conception nouvelle -> utilisation de DAG
  - Très efficace dans un milieu dense, car beaucoup de liens
  - Utilise que des liens symétriques
  - Routage garanti sans boucle

    (hauteur dans le quintuplet permet de connaître le nombre de saut minimal pour atteindre la destination lors de la création de route…)

  - Nécessité d'un temps global

    Temporally-Ordered est le principe de base de ce protocole

Protocole hybride

- 2 façon de gérer dynamiquement les routes
  - soit d'une manière proactive
    - routes sont disponibles immédiatement
    - mais beaucoup de trafics de mises à jour
  - soit d'une manière réactive
    - routes sont longues à être trouvées
    - beaucoup de trafic pour trouver la route (inondation)
    - autrement, peu de trafic

- Hybride
  - mélange de réactif et proactif
  - Pour les nœuds proches, connaissance des routes immédiates
    → proactif
  - Pour des nœuds lointains, utilisation d'un protocole réactif

ZRP (1)

- Zone Routing Protocol
  - Comme son nom l'indique, basé sur des zones
    - Une zone contient l'ensemble des stations autour d'un nœud de référence dont la distance (nombre de sauts) est inférieure ou égale à x.
    - La distance x est appelé "zone radius"
    - Chaque nœud a sa zone individuelle. Les stations situées par le chemin le plus court à la distance x du nœud de référence sont appelées les nœuds frontaliers (peripheral ou bordercast node)
    - A l'intérieur de la zone, utilisation d'un protocole proactif, à l'extérieur utilisation d'un protocole réactif

ZRP (2)

Le protocole IARP

(Intrazone Routing Protocol)

- Routage à l'intérieure de la Zone
- Utilisation d'un protocole proactif (choix libre ex :OLSR, DSDV)

- Quelques spécificités :
  - Chaque nœud envoie sa table de voisinage à tous ces voisins
    (utilisation de trame hello)
  - Détection de la perte de voisinage
  - Une table de routage semi-complète…
    (seule les stations appartenant à la zone sont répertoriées)

ZRP (3)

Le protocole IERP

(Interzone Routing Protocol)

- Routage à l'extérieure de la Zone
- Utilisation d'un protocole réactif (choix libre ex :AODV)

- Quelques spécificités :
  - Désactivation de la table de voisinage
  - Utilisation d'un numéro de séquence pour éviter les boucles
  - Utilisation du protocole BRP

BRP : Border Resolution Protocol

ZRP (4)

BRP (à l'intérieur de IERP)

- Lorsqu'un nœud frontalier reçoit une demande pour une destination, *d'après les protocoles réactifs* → *diffusion*

- Mais, il a la connaissance de toutes les destinations dans sa zone,
  - si destination existe, retour de l'information
  - sinon diffusion que vers ses nœuds frontaliers.

  - Minimisation du nombres de requêtes envoyées.

Conclusion :

- Protocole essayant de prendre le meilleure des 2 grandes catégories
- Chemin proche dans la table de routage
- minimisation de l'inondation

LAR (1)

LAR : Location-Aided Routing

- Protocole réactif (basé sur DSR ou AODV)
- Minimisation de la diffusion par la connaissance géographique des nœuds
- Chaque nœud doit connaître sa position
  - utilisation du GPS (considéré sans erreur)
  - optionnel : connaître sa vitesse
- Emission d'une Route Request lorsque la destination n'appartient pas à la table de routage.

LAR (2)

Zone possible

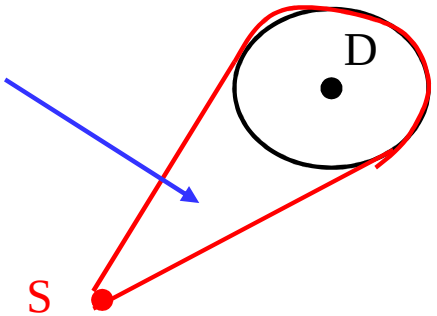
(zone probable où peut se trouver la destination D)

- Si connaissance de la dernière position de D,de sa vitesse et du temps passé alors
  → définition d'un cercle centré sur sa dernière position

- Si aucune information, alors zone possible = réseau ad hoc

Zone de recherche

- zone de recherche = zone possible + autre
  - si S n'appartient pas à la zone, il faut le rajouter

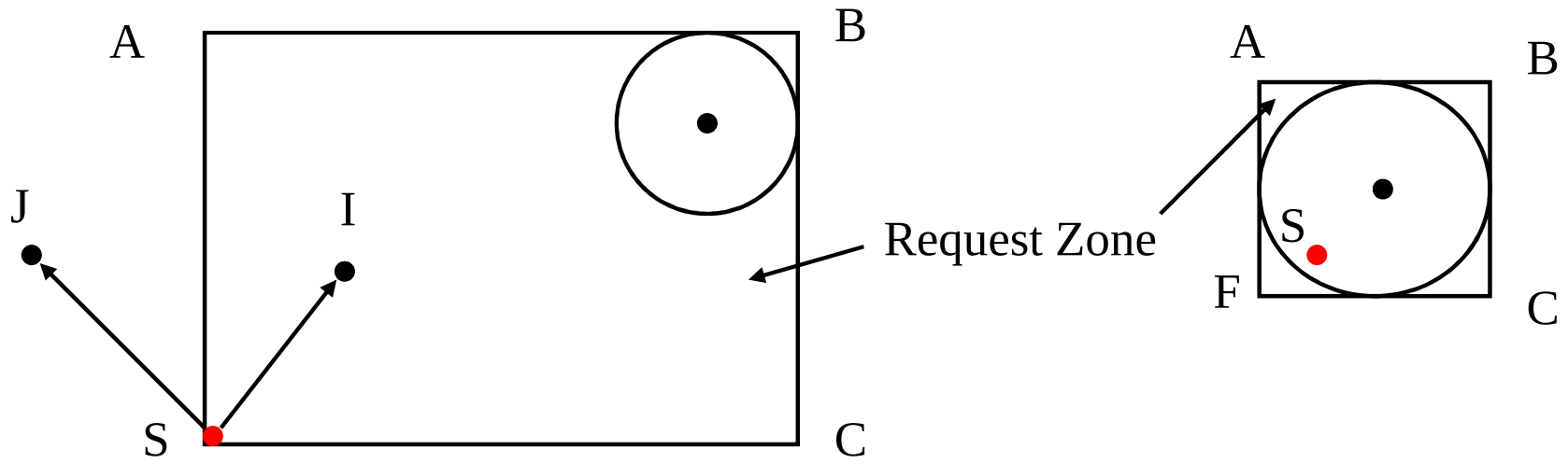


LAR (3)

- Zone de recherche
  - Nœud hors de la zone de recherche pour atteindre D
    - ➡ Agrandissement de la zone obligatoire
- Pratiquement
  - zone de recherche = rectangle englobant la source et la zone possible
    - Si un nœud reçoit une Route Request et qu'il n'appartient pas à la zone de recherche, arrêt de la diffusion
      - ➡ permet de minimiser la diffusion
    - si pas de résultat après un temps d'attente, on renvoie une Route Request sans délimitation
      - on se retrouve avec DSR ou AODV…

LAR (4)

- Trame envoyé
  - On passe comme information les coordonnées des 4 points du rectangle



  - *Autre possibilité* : calcul de la distance théorique entre S et D. Si la distance diminue, on diffuse, sinon, on ne fait rien car on s'est éloigné

  - **Amélioration :** utilisation de la vitesse

Autres protocoles possibles ?

Mobile IP …

Mobile IP (1)

- Mobile IP
  - Protocole utilisé pour changer de réseau **sans changer d'adresse IP**
    - Avantage : pour le correspondant, pas besoin de savoir où se situe le destinataire
    - 2 normes : Mobile IPv4 → RFC 3344
      Mobile IPv6 → RFC 3775
    - Problème au niveau 3 et au niveau 2 du réseau
      → couches supérieures = transparence

Mobile IP (2)

- Vocabulaire:
  - Adresse permanente : adresse IP par laquelle le système peut toujours être joint.
  - Réseau principal (mère) : réseau dans lequel se situe l'adresse permanente du système.
  - Agent mère : serveur statique se trouvant dans le réseau principal et stockant en mémoire la localisation du système mobile et pouvant en permanence le joindre
  - Réseau visité : Réseau dans lequel se situe le système
  - Adresse mobile : adresse IP du système dans le réseau visité (care-of-address)
  - Agent relais : serveur dans le réseau visté permettant de relayer des paquets vers un système mobile

Mobile IP (3)

Fonctionnement:

- Le système indique à l'agent mère qu'il part

  → agent mère va intercepter tous les paquets à destination du système en connaissant son adresse permanente

- Le système quitte le réseau principal
  - 2 possibilités :
    - existence *d'un agent relais* : le système va s'enregistrer sur cet agent relais

      → création d'un tunnel entre l'agent relais et l'agent mère pour faire passer les paquets à destination du système

      → entre l'agent relais et le système : même réseau

      - en utilisant les @MAC
      - en ayant récupéré une adresse mobile via l'agent relais

      → retour des paquets : soit en utilisant l'agent relais, puis l'agent mère
      soit en renvoyant avec son @IP permanente
      (PB : Pare-feu)

Mobile IP (4)

- Autre possibilité
  - *Pas d'agent relais*
    - Nécessité de récupérer une adresse mobile
    - Envoie de cette information à l'agent mère
    - création d'un tunnel entre l'agent mère et le système

- Divers
  - Protocole de routage non optimisé
    (routage triangulaire si tout va bien)
  - IPv6 permet le source routing, évite de passer par l'agent mère une fois la position du système connu
  - Problème d'authentification et de sécurité

  - Que faire si on arrive dans un réseau privé !!!