Matière 3 : Réseaux sans fil

Chapitre 1 : WiFi

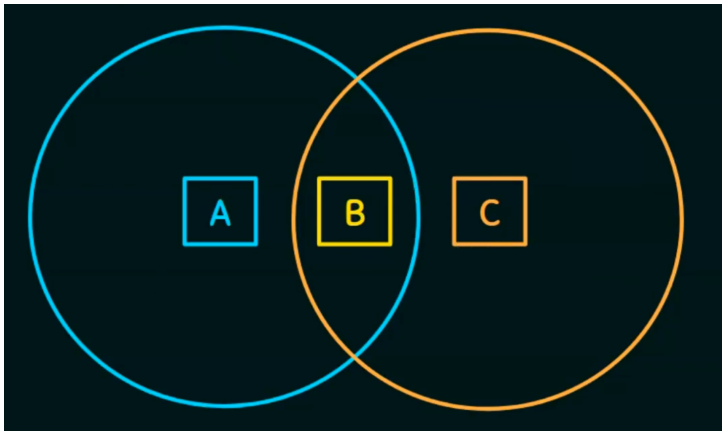
1 - Introduction

Aussi connu sous le nom de IEEE 802.11.
Il peut fonctionner selon deux architectures :

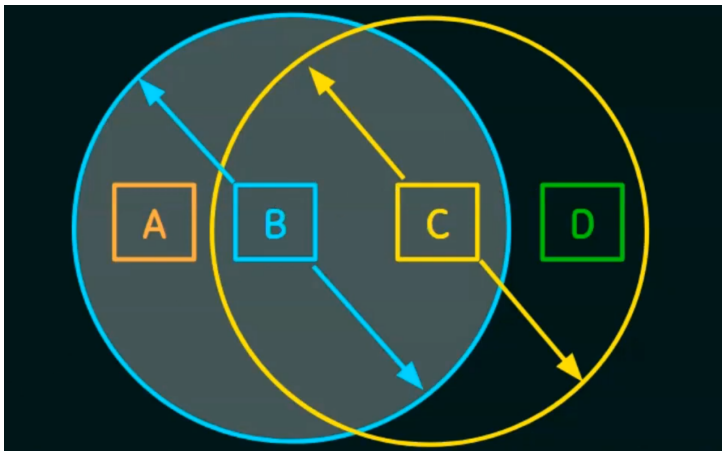
- WLAN : La plus connu, architecture centralisée, les appareils se connectent à un point d'accès (la Box) : permet d'accéder à L'Internet et de communiquer avec les autres appareils du réseaux
- Ad-Hoc : Architecture distribuée : pas de Point d'accès, permet la communication entre deux machines sans l'aide d'une infrastructure (un peu similaire à ce que propose le Bluetooth)

Dans les archi WLAN : **Problème majeur : Accès au Point d'accès** (deux stations ne peuvent transmettre simultanéments au PA)

2 - Le problème d'accès au support



- **Hidden terminal problem** : A et C sont des terminaux qui veulent émettre et B est le Point d'accès. A ne voit pas C et C ne voit pas A. Ainsi, si les deux émettent en même temps, ils ne vont pas le savoir et il va ya voir collision. Il va falloir trouver une solution pour régler ce problème.



- **Exposed terminal problem** : Cette fois on regarde les terminaux B et C, qui sont bien à la porté de l'un et de l'autre. Supposons que B emet à A. Ainsi, C va supposer qu'il ne peut pas émettre étant donné que B est déjà en train d'émettre. Or c'est une erreur car C peut émettre à D en même temps que B émet à A.
On va également regarder plus loin comment résoudre ce problème.

Les protocoles MAC existants ne permettaient pas de résoudre ces différents problèmes (CSMA/CD par exemple).
-> Emergence de **MACA :** Multiple Access with Collision Avoidance.

Introduction du protocole RTS/CTS pour résoudre les deux problèmes ci dessus :

- RTS/CTS pour **résoudre Hidden terminal** :

On reprends l'exemple ci dessus :

1) si C veut transmettre, il envoi RTS '(request to send) à B en indiquant la durée de la transmission à venir
2) B (Access Point) reponds à C avec un CTS (clear to send)
3) A va également recevoir le CTS et va déduire qu'il ne peut pas émettre pendant un certain temps (indiqué par C)
4) C transfère le paquet à B

- RTS/CTS pour **résoudre exposed terminal :** Il faut retenir qu'avant ça permettait de résoudre ce problème, mais que les nouvelles versions de RTS/CTS ne le permettent plus à cause de la mise en place d'un ACK.

Concercnant **CSMA/CA :** Mecanisme pour l'accès au support. On l'utilise avant d'envoyer RTS. En plus de CSMA/CA, on met en place niveau MAC un mécanisme ARQ (le destinateur des data va envoyer des ACK à l'envoyeur) et comme Ehternet la présence d'un Backoff pour une éventuelle retransmission.

3 - Multicast/Broadcast :

Le multicast/Broadcast peut poser problème au sein des réseaux sans fil, surtout un problème de collision. Pour l'éviter, les trames sont envoyé au plus bas débit possible. Ce n'est pas optimal, alors une des autres solutions, va être de transmorfer le multicast en pleins d'Unicast.

4 - QoS sur WiFi ?

Introduction de QoS dans Wifi en jouant surtout sur la couche MAC. Développement du protocole EDCA (Enhanced distributed channel access) qui permet de faire de la priorisation de traffic et donc est un outil de QoS.

5- Couche PHY

Depuis le développement de WiFi, on a amélioré les débits. Mais il reste un invariant : impossible de garantir un débit donné. Plusieurs raisons à cela : L'air est un environnement très compliqué par rapport aux réseaux filaires. De plus, la distance au Point d'Accès, le Mouvement des terminaux pendant la transmissions etc… font qu'on ne peut pas garantir un débit constant.

Principaux problèmes du canal radio :

- Perte du Chemin
- Interference avec des signaux de même fréquence
- Shadowing (atténuation du signal qui traverse un obstacle)
- Reflection (réflexion sur un obstacle : le signal ne passe pas)
- Multipath (le terminal reçoit plusieurs fois le signal)

Ces derniers problèmes expliquent donc également pourquoi on a une telle différence de débit en WiFi et pourquoi on ne peut garantir tel débit.

Pour palier à cela, plusieurs algortithmes ont été mis en place :

1) **ARF (Auto Rate Fallback).** C'est un algorithme qui va permettre de dire au PA à quel débit il faut envoyer. Au début, l'algo dit au PA d'émettre au débit maximum qu'il a identifié, puis ensuite il fonctionne un peu comme TCP :

- Si on a pas d'ACK (paquet perdu) : On baisse le débit
- Si on a 10 ACK successif : On augmente le débit
  -> **Faiblesse :** Ne prends pas en compte les réémissions de paquet

2) **Onoe :** Essaye de trouver (sur le long terme) le débit qui a moins de 50% de perte. Va fonctionner un peu comme ARF, mais va regarder si les paquets ont été beaucoup retransmis ou pas pour augmenter ou réduitre de débit :

- Au début : débit max, compteur de crédit à 0
  Ensuite, 1 fois/sec :
  - Augmente le débit si aucun paquet n'a été envoyé avec succès ou 10 paquets ou plus ont été envoyé et que la moyenne de réémission par paquet est supérieure à 1
  - Si le débit a plus de 10 crédit, on augmente le crédit (les crédit sont distribués comme suit : si moins de 10% des paquets ont nécéssité une réémision, on incrémente le crédit, sinon on le décrément)

  -> **Conclusion :** Onoe est conservatif et peut prendre du temps pour se stabiliser

3) **Recerver Based Auto Rate (RBAR) :** En gros, ce mécanisme va utiliser les messages RTS/CTS pour demander au destinataire à quel débit il devrait transmettre.

4) **Opportunistic Auto-Rate (OAR)** : Va également utiliser RTS/CTS pour contrôler le débit. (j'ai rien compris à ce protocole, c'est un truc sombre sur lequel y'a eu un papier de recherche sur les 20 dernières années.

5) **SampleRate :** En gros, va essayer de trouver le meilleur bit-rate possible, en ayant au préalable de la transmission une table avec les différents bit-rate et leur taux de paquets bien envoyés. Ensuite :
   - Au début, il sélectionne le débit le plus haut possible
   - Il change de débit binaire apès 4 drops successifs
   - tous les 10 paquets, il prends un bir rate random qui devrait faire mieux que le bit-rate actuel. (ces bits-rates sélectionnés dans la table qu'il a au préalable fait).

A retenir : Tous les algos de contrôle de débit sont pour des traffics unicast et ils ont tous besoin d'un feedback du déstinataire

Chapitre 2 : Rate control + Ad-hoc networks

1 - Réseaux Ad Hoc

Réseaux créés dynamiquement sans coordination préalable des stations
Chaque station est un routeur
Avantages :

- Déploiement facile et rapide
- Réseau moins dépendant d'une infrastructure

⇒ Interêt :

- militaire → communication sans opérateur téléphonique (DARPA packet radio network 1973-1987)
- civil → communications pour communautés isolées, réseaux maillés

Environnement symétrique → tous les noeuds ont des **capacités** et des **responsabilités identiques**
Capacités asymétriques → différences de :

- portées de transmission
- autonomie
- capacité de calcul
- vitesse de déplacement

Responsabilités asymétriques :

- seuls certains noeuds sont routeurs
- certains noeuds sont chefs d'un groupe de noeuds

Les réseaux ad hoc peuvent être amenés à coexister avec des réseaux en mode infrastructure

2 - Routage Unicast dans les MANET

MANET = mobile ad hoc network
Problèmes :

- mobilité des noeuds → taux d'erreur sur les liens ↗ si les noeuds bougent rapidement
- nouveaux critères de performance : stabilité des routes malgré la mobilité, consommation d'énergie

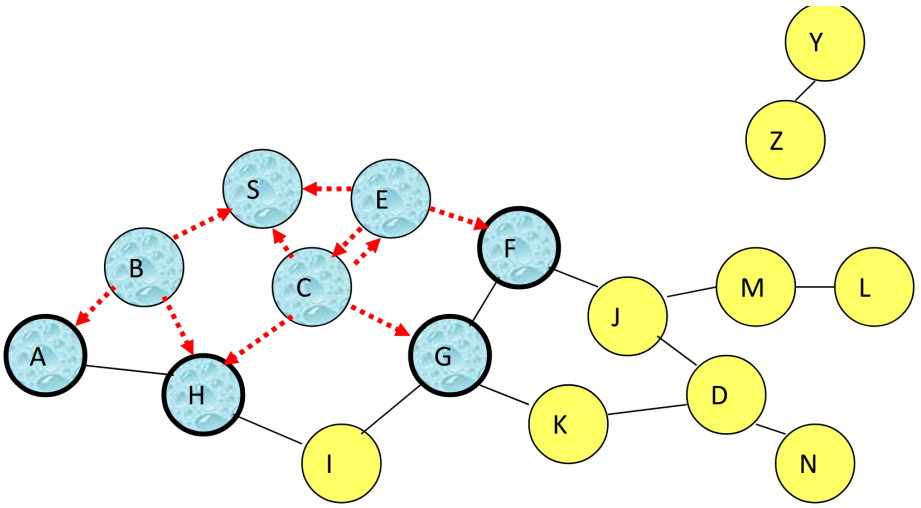
Types de protocoles de routage :

- Proactifs → déterminent les routes indépendamment du trafic (traditionnel)
  Overhead +,Latence - (les routes existent déjà quand on veut transmettre)
- Réactifs → calculent seulement les routes quand elles sont nécessaires
  Overhead -, latence + (car les routes sont calculées au moment de transmettre)
- Il existe aussi des protocoles hybrides

3 - Méthodes de routage

3.1 - Flooding for data delivery

Le nœud source S broadcast son paquet à tous ses voisins, qui le transmettent à tous leurs voisins, etc… Jusqu'à ce que la destination D reçoive le paquet (D ne retransmet pas le paquet)



Remarques :

- Numéros de séquence utilisés pour ne pas transmettre le même paquet 2 fois
- Collisions potentielles lors de 2 transmissions simultanées venant de stations cachées l'une pour l'autre → des noeuds **peuvent ne pas recevoir** le paquet (D par exemple)
- Pire cas : tous les noeuds atteignables depuis S reçoivent le paquet

Avantages :

- Simple
- Plus efficace que d'autres protocoles quand il ya peu d'infos transmises et que l'overhead d'autres protocoles est élevé (quand de petits paquets sont transmis peu souvent et que la topologie réseau change souvent)
- Fiabilité potentiellement + importante (plusieurs chemins pour 1 paquet)

Inconvénients :

- TRÈS GROS overhead potentiel (beaucoup de noeuds reçoivent des paquets dont ils ne sont pas destinataires)
- Fiabilité potentiellement - importante car broadcast peu fiable sans overhead et les collisions peuvent empêcher la réception

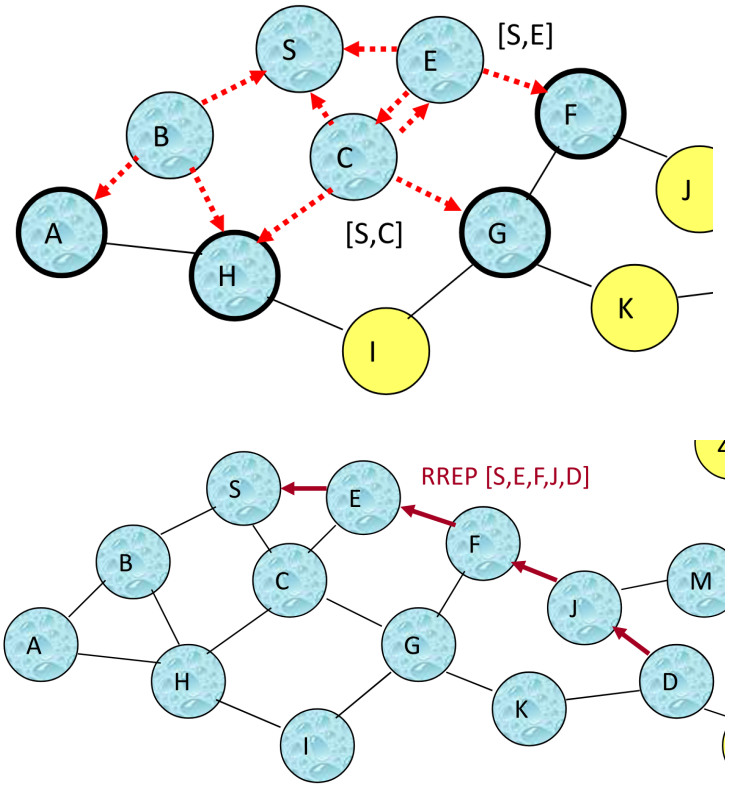
3.2 - Flooding of control packets

Inondation limitée aux trames de contrôle, utilisées pour découvrir des routes → ces routes seront utilisées plus tard pour acheminer des données → overhead amorti par la quantité de données transmises sur une route

3.3 - Dynamic source routing (DSR)

Si S ne connaît pas de route vers D, il commence un *route discovery* :

- S inonde le réseau de *route request* (RREQ)
- Chaque noeud forward un seul RREQ en y ajoutant son identifiant
- D ne forward pas de RREQ, mais envoie une *route reply* (RREP) → la RREP est envoyée sur une route obtenue en inversant la route contenue dans la RREQ (en supposant que les liens sont **bidirectionnels,**sinon D fait un *route discovery* vers S avec un piggybacking du $RREP_{S \to D}$ sur le $RREQ_{D \to S}$)
- La RREP contient la route de S à D par laquelle est passée la RREQ reçue par D



Lors de l'envoi d'un paquet de données, la route complète est incluse dans l'en-tête du paquet.

Avantages :

- Routes maintenues seulement entre les noeuds qui communiquent, et mises en cache → overhead réduit
- Une seule *route discovery* donne plusieurs routes vers la destination

Inconvénients :

- Routage par la source → taille des en-tête ↗
- les RREQ atteignent potentiellement tous les noeuds
- Il faut éviter les collisions entre RREQs
- Congestion si trop de noeuds répondent avec une route qu'ils ont en cache → *Route Reply Storm*, qui peut être évité si un noeud qui entend un RREP plus court que le sien s'abstient de répondre

3.4 - Ad Hoc On-Demand Distance Vector Routing (AODV)

Tentative d'améliorer DSR en stockant les routes sur les nœuds et non plus dans les paquets.

Quand un nœud reçoit un RREQ, il enregistre une route (*reverse path*) menant vers la source.

Les noeuds ne retiennent les *reverse path* que des routes actuellement utilisées (expiration après timeout)

Détection de pannes de liens par envoi régulier de messages Hello entre noeuds voisins

Utilisation de messages *Route Error* (RRER) vers la source pour signaler les pannes de liens. S reçoit le RRER → nouvelle *route discovery* lancée

Optimisation possible : les RREQ sont d'abord envoyés avec un TTL faible pour limiter leur propagation, puis de + en + grand si aucune RREP n'est reçue.

3.5 - Link state routing (LSR)

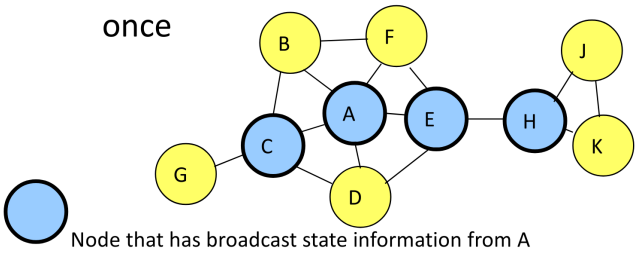
Routage proactif :

- Chaque noeud envoie périodiquement les informations de ses liens
- Chaque noeud retransmet les informations de ses voisins
- Chaque noeud utilise toutes ces informations pour calculer des routes

3.6 - Optimized LSR (OLSR)

Moins d'overhead que LSR car moins de nœuds retransmettent les informations car seuls les nœuds dits **multipoint relays** d'un nœud X retransmettent.
Les nœuds *multipoint relay* d'un nœud X sont les voisins de X tels que chaque voisin de X à distance 2 est voisin à distance 1 d'un *multipoint relay de X.*

- Nodes E and K are multipoint relays for node H
- Node K forwards information received from H
  - E has already forwarded the same information once



Chapitre 3 : WiFi-based Indoor Localization

Le GPS marche pas bien à l'intérieur des bâtiments, on peut utiliser le WiFi pour le remplacer.

1 - Localisation basée sur le fingerprinting (RADAR)

RADAR est un système basé sur les radiofréquences pour localiser et suivre les utilisateurs à l'intérieur des bâtiments, il permet d'utiliser l'infrastructure existante du réseau local sans fil. Il se base sur le fait qu'il y a une corrélation entre l'intensité du signal reçu et la distance à la station de base.

1.1 - Phase d'apprentissage (Off-Line)

On commence par créer une carte radio, on peut utiliser deux méthodes :

1.1.1 - Méthode empirique

- Les stations de base émettent des balises (beacons) périodiquement : on va enregistrer le champ SS (Signal Strength) à plein d'endroits
- Enregistrer les SS des stations à portée et les coordonnées correspondantes (l'orientation de l'utilisateur doit également être prise en compte, tuples de la forme (x,y,d,ss1,...,ssn) pour n stations de base)
- Précis mais contraignant (on doit prendre les mesures manuellement, et si on bouge une station de base il faut reconstruire la carte radio)

1.1.2 - Méthode mathématique

- Calculer le SS à l'aide d'un modèle de propagation simple (prise en compte de l'affaiblissement sur le trajet et de l'atténuation des parois)
- Plus pratique mais moins précise

1.2 - Phase de fonctionnement

- Extraire le SS des trames de balise (beacons) de la station de base
- Transmettre une demande de localisation à la station de base avec SS en entrée
- Trouver l'entrée de la carte radio qui correspond le mieux au SS mesuré (on minimise la distance euclidienne entre la mesure (ss1, ss2, ss3) et les données de la carte radio (ss'1,ss'2,ss'3) pour obtenir une position (x,y,d)).

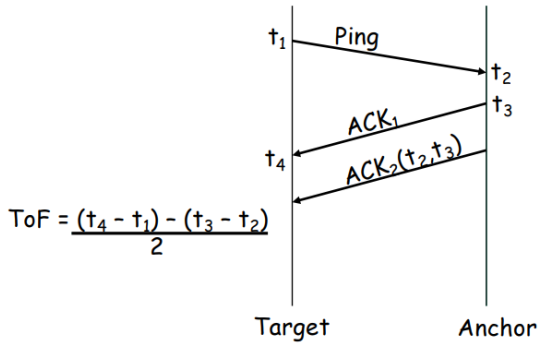
Pour évaluer la performance de cette méthode, on peut enlever un point connu de la carte radio, puis tenter d'estimer sa position à partir du reste des données. Plus la carte radio contient de points, plus on sera précis. En moyenne on obtient une précision de 3m sur la position. On peut descendre à 2m si on moyenne sur plusieurs voisins.

Pour un utilisateur mobile qui se déplace c'est un peu plus complexe, on utilise une fenêtre glissante de 10 échantillons de SS pour calculer l'intensité moyenne du signal de façon continue, la précision est un peu moins bonne que pour un utilisateur stationnaire (~3,5 m).

2 - Localisation basée sur le temps de vol (Time of Flight)

Le problème de localisation est réduit à un problème de calcul de la distance entre la cible et un ensemble de points dont les coordonnées sont connues. Calculer la distance entre deux appareils revient à calculer le temps nécessaire à un signal pour parcourir la distance entre les deux (ToF).

Two-way ranging (TWR) :



ACK_2 nécessaire pour éviter d'avoir à synchroniser les horloges des deux machines.

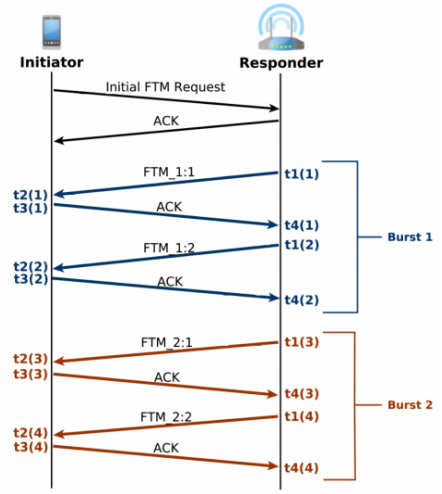
3 - WIFI FTM (IEEE 802.11AC)

FTM = Fine Time Measurement, il permet à une station WiFi de calculer sa distance à un point d'accès à portée sans avoir à s'associer avec lui (précision de l'ordre du mètre). Il est supporté par les principaux fabricants de WiFi et Android.

Fonctionnement :

- Le processus commence par une station WiFi (appelée *initiateur*) qui recherche les points d'accès supportant FTM.
- Si un point d'accès supportant FTM est détecté, l'*initiateur* envoie à ce dernier une trame de demande FTM.
- A la réception de cette demande, le point d'accès peut choisir de l'ignorer ou de devenir un *répondeur*.
- Les deux stations commencent une série de burst (envoi par le *répondeur* de paquets FTM puis d'un ACK par l'*initiateur*), permettant à l'*initiateur* d'estimer le temps d'aller-retour (RTT) avec le *répondeur*.

$$RTT = \frac{1}{N}\sum_{i=1}^{N}\left(t_4(i) - t_1(i)\right)$$



Problème : lorsqu'il y a un obstacle entre le point d'accès et la station, on mesure un signal qui a potentiellement rebondi sur un mur -> la distance mesurée sera celle parcourue par le signal, donc plus grande que la distance en ligne droite.

Chapitre 4 : Internet of Things (IoT)

Définition de IoT : Interconnexion via Internet d'appareils informatiques embarqués dans objets de tous les jours, leur permettant de transmettre et de recevoir des données

Exemples d'application de l'IoT : les objets de santé connectés, les smart cities, transports…

Caractéristiques fondamentales de l'IoT :
Hétérogénéité : capteurs, réseau (portée, capacité, consommation électrique, infrastructure), applications
Echelle : en nombre d'appareils plutôt qu'en quantité de donnée (des milliards d'appareils sont attendus => IPv6 obligatoire)

BLUETOOTH

En fait ce cours ne parle presque pas d'IOT mais surtout de Bluetooth du coup jsp pourquoi il s'appelle comme ça

1 - Introduction

Historique
Norme créée en 1994 par le fabricant **Ericsson**.
Objectifs : **faible consommation, pas cher, remplacement des câbles courts** (portée de 0-**10m**, faible débit de donnée 19.2-100kbps)
En 1998, les fabricants Ericsson, Nokia, IBM, Toshiba, Intel… forment un groupe d'intérêt pour développer une solution respectant ces objectifs => spécification sortie en **1999**

=> **IEEE 802.15** = **WPAN (Wireless Personal Area Network)** = Bluetooth => définition de 4 groupes de travail :

- 802.15.1 = spécification des couches physiques et MAC pour la connection sans fil d'appareils dans un champs de 10m (aussi appelé POS = personal operating space, un POS peut être statique ou en mouvement)
- 802.15.2 = coexistence et interopérabilité entre WLAN (=WiFi) et WPAN (=Bluetooth)
- 802.15.3 = hauts débits (jusqu'à 20Mbps) pour multimédia
- 802.15.4 = appareil à bas débits et faible consommation et complexité

Standard Bluetooth : spécifie le système entier, du niveau radio au niveau application. La **pile de protocole se situe en partie au niveau hardware et en partie au niveau software**.

Quelques notions sur le Bluetooth :

- Il opère sur la même bande de fréquence **2.4 GHz** que le Wifi.
- Il s'appuie sur le **frequency hopping spread spectrum (=étalement de spectre par saut de fréquence)** : cf. caractéristiques pour le Bluetooth ci-dessous
- 2 appareils se trouvant à moins de 10m peuvent échanger jusqu'à 720kbps de capacité

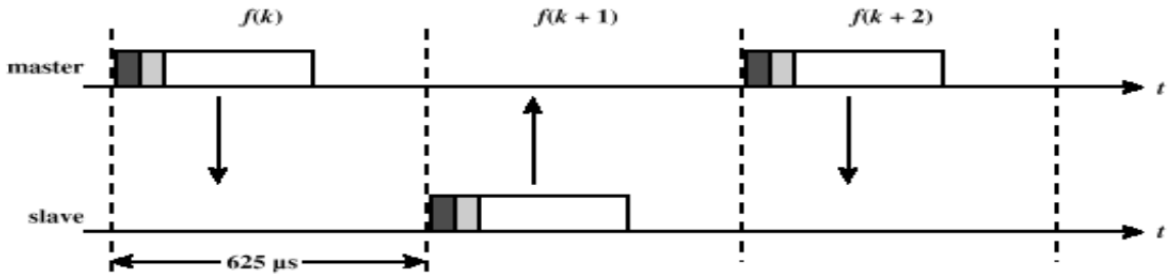
2 - Frequency Hopping Spread Spectrum

Définition : **Frequency hopping spread spectrum** (= étalement de spectre par saut de fréquence) = méthode de transmission de signaux par ondes radio qui **utilise**

alternativement plusieurs canaux (sous-porteuses) répartis dans une bande de fréquence selon une séquence pseudo-aléatoire connue de l'émetteur et du récepteur.

Caractéristiques du Frequency hopping spread spectrum **pour le Bluetooth** :

- **canaux de 1 MHz** de large (=> **79 canaux au total**)
- la fréquence/le **canal change à chaque paquet**
- chaque **slot dure 625µs (=> 1600 hoping/s)**
- la transmission d'un paquet dure entre 1, 3 ou 5 slots (slot ≠ canal = fréquence)
- **la séquence des sauts de fréquence est transmis est transmise au sein de tout le piconet (voir définition juste en dessous), le maître transmet aux esclaves**

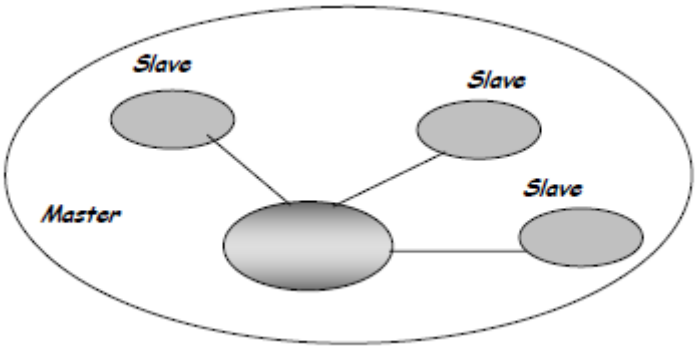


Accès à un canal / une fréquence (TDD)

- **Le master est toujours celui qui initie l'échange de données**
- Le **slave répond au master en suivant la séquence** de sauts de fréquence spécifiée par ce dernier
- Un **slave transmet UNIQUEMENT en réponse à un master**
- Le master transmet sur les **slots impairs** tandis que le slave transmet sur les **slots pairs**

3 - Piconet

Piconet = une architecture Bluetooth = unité de base d'un réseau Bluetooth



On a un **Maître qui peut avoir jusqu'à 7 Esclaves**. Il contrôle également les transmissions à l'intérieur de son Piconet. Il n'y a **pas de conflit** à l'intérieur d'un Piconet.

Autre architecture Bluetooth possible (mais **non vue**) : **Scatternet** (un appareil peut appartenir à plusieurs piconets mais un appareil ne peut être maître que d'un seul piconet)

Création d'un Piconet

Un Piconet se compose d'un ensemble de canaux de communications que les membres du Piconet vont utiliser. Un canal de communication consiste en une séquence de sauts de fréquence bien définie.

Comment est créé un Piconet ?
=> 2 étapes : **Inquiry** & **Paging** (= **Enquête/Interrogation** et **Pagination**)

4 - Inquiry Procedure

Inquiry Procedure

Envoi d'une **enquête/interrogation qui est en fait une demande des appareils proches** (<10m). Les appareils qui sont en "mode visible" délivrent une **inquiry response** (cette procédure peut prendre **jusqu'à 10.24 secondes**), après cela l'appareil doit connaître tout le monde dans les 10 mètres autour de lui.

→ **S'il y a un problème avec les inquire messages** ça peut venir de 2 possibilités : l'émetteur et le receveur ne sont pas encore connectés et sont donc sur des séquences de sauts de fréquence différentes et donc sur des canaux / fréquences différents ; s'ils sont sur le même canal celui-ci est peut-être "encombré" de bruit

3.1 - Transmission des **Inquiry Messages**

L'appareil qui émet l'**interrogation** envoie cette **interrogation sur 16 fréquences différentes**. Ces 16 fréquences forment la **inquiry hop sequence**, appelée un **train**. Il y a alors 2 types de trains : génération en utilisant les 28 bits les moins significatifs du GIAC & DIAC (pour les imprimantes). La transmission se fait sur chaque alternate slot tandis que les intermediate slot sont utilisés pour écouter les éventuelles réponses.

Gestion du bruit

- Les appareils répondent toujours à un inquiry message avec un inquiry response. **Un appareil interrogateur est autorisé à recevoir plusieurs réponses d'un seul appareil**.
- Pour prendre en compte le fait qu'un canal peut être "bruyant" et que les transmissions peuvent être perdues, **un train est répété jusqu'à 4 fois**.

3.2 - Inquiry Scan

Un appareil écoute périodiquement sur une seule fréquence (parmi 16 fréquences) pour chercher des inquiry messages. Il reste dans cet état assez longtemps pour qu'un appareil interrogateur puisse couvrir les 16 fréquences.

3.3 - Inquiry Response

Quand un inquiry message est reçu durant l'état de inquiry scan, l'appareil envoie alors un paquet de réponse contenant son adresse. Ce paquet n'est pas envoyé directement après la réception de la demande pour éviter les collisions. **L'appareil attend un nombre aléatoire de inquiry scans complets (entre 0 et 127)** puis envoie son **paquet FHS** à l'interrogateur, ce paquet FHS contient **son adresse, son horloge et des informations indiquant quand l'appareil entre dans son état de scan de pages**. Un appareil interrogateur qui reçoit une réponse **n'acquitte pas directement** cette dernière

mais continue sa inquiry procedure. **Il utilisera cette réponse seulement quand il voudra paginer l'appareil**.

5 - Paging Procedure

Paging Procedure (procédure très similaire à l'interrogations)
Si une connection est désirée après inquiry procedure => paging

Cette procédure n'a besoin que de l'adresse de l'appareil pour paginer ce dernier mais son **horloge** (contenue aussi dans le paquet FHS) **permet d'accélérer le processus**. **L'appareil qui démarre la procédure de paging sera le master du piconet** composé de lui-même et de l'appareil paginé si ce dernier accepte la connexion.

Procédure en 6 étapes :

1. Page scanning : l'appareil envoie un **page message** à l'appareil avec lequel il souhaite se connecter (similaire aux inquire messages, sur 2 trains de fréquences de 16 fréquences chacun). Une fois une page response reçue, il va arrêter et passer à l'étape 2.
2. Dans la **page response**, **un acquittement contenant l'ID du slave est envoyé au master**
3. Le **Master** envoie un **paquet FHS** au **slave** pour lui **donner son horloge**
4. En utilisant les données du paquet FHS, le **slave adopte la séquence de saut de fréquence du master et se synchronise à son horloge**. Le slave émet une dernière réponse qui est alignée sur l'horloge native du slave.
5. Quand le master reçoit un paquet, il retourne à son pattern de saut de fréquence et assigne une **Active Membre Address (AMA)** au slave dans le piconet. Le master sonde également (via un paquet spécifique) le slave pour s'assurer qu'il est bien sur son pattern de saut de fréquences.
6. Le slave répond à ce sondage avec **n'importe quel paquet pour prouver qu'il est bien sur le bon canal**. L'acquittement doit être reçu par le master dans le temps imparti. A la fin de cette étape, une nouvelle connexion synchronisée est établie entre le master et le slave.

6 - Link Manager

Il s'occupe de toutes les **créations de liens, management et fin d'opérations**. Responsable des toutes les **ressources de lien physique** dans le système (contrôle et négociation de la taille des paquets lors de la transmission de données). Il contrôle les modes d'opération des appareils dans le piconet. Met en place, met fin et gère les connexions en bande de base entre les services : **établit différents types de liens en fonction des requêtes venant de la couche L2CAP (SCO ou ACL)**.

6.1 - Liens ACL (Asynchronous Connection-Less)

Designé pour la **transmission de données**. Routage de **paquets**. Intégrité des paquets gérée via une vérification d'erreurs et de la retransmission. Un lien ACL entre un master et un slave.

6.2 - Liens SCO (Synchronous Connection Oriented)

Designé pour les informations **temps réel** comme l'audio ou la vidéo. Connection **circuit** ou les données sont transmises régulièrement. Retransmission non nécessaire. Jusqu'à 3 liens SCO par piconet.

6.3 - Link Manager Operation

Les appareils sont en **mode Standby** par défaut jusqu'à ce qu'ils soient connectés au piconet.

On a **4 modes de connexion**, qui permettent à l'appareil d'ajuster la consommation d'énergie, la performance, le rôle et le nombre de participants dans un piconet :

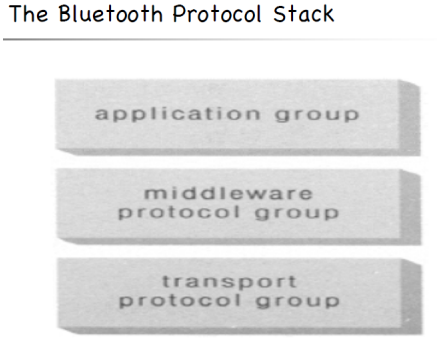
- **Active mode** : jusqu'à 7 slaves actifs pour chaque master. Adresses en 3-bits (AM_ADDR) données à chaque slave actif. Une unité participe activement sur un canal et reçoit des communications sur une trame donnée. Une unité consomme beaucoup d'énergie.
- **Hold mode** : Libère un slave pour participer sur un autre piconet, faire du scanning, paging ou du inquiry, passer en mode économie d'énergie. L'unité garde les adresses des membres actifs. L'unité ne supporte plus les paquets ACL mais les paquets SCO oui. Le master et le slave s'accordent sur un durée d'attente au bout de laquelle le slave reviendra se synchroniser avec le trafic du canal. Faible consommation d'énergie.
- **Sniff mode** : Très similaire au hold mode. Le slave est libre de revenir à des intervalles de temps fixés. Le master ne peut communiquer qu'à des times slots de "sniff" arrangés
- **Park mode** : L'unité parkée abandonne les adresses des membres actifs. Pour gérer la réadmission rapide de l'unité parkée, le master lui attribue 2 adresses temporaires 8-bit. L'unité parkée reste synchronisée au canal et opère un repos à faible consommation d'énergie. Elle se réveille régulièrement pour maintenir la synchronisation. Ce mode permet de connecter plus de 7 appareils à un master. Les appareils actifs et parkés peuvent être échangés pour permettre plusieurs connexions à un seul piconet.

6.4 - Segmentation et Réassemblage

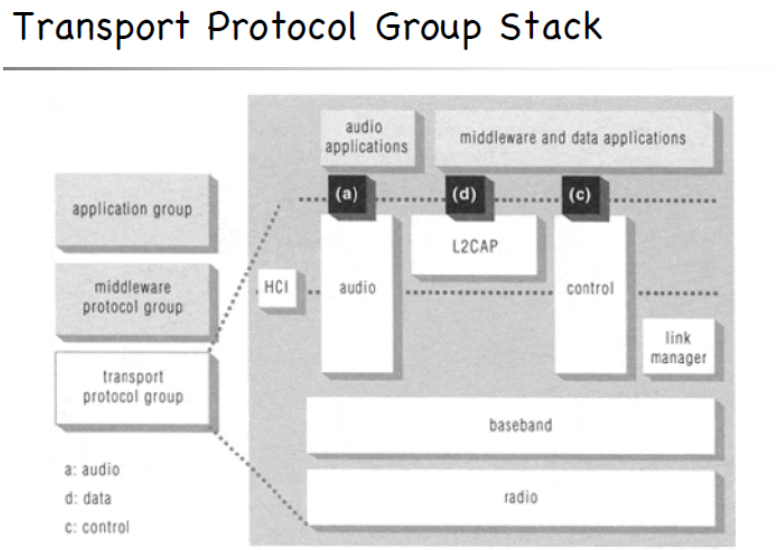
Les paquets bande de base sont limités en taille (charge utile max = 2745 bits). **L2CAP** est protocole utilisé par le Bluetooth qui s'occupe de faire de l'adaptation entre les couches hautes et les couches physiques (cf. pile de protocoles du groupe de transport plus bas). L2CAP accepte des paquets de taille allant jusqu'à 64kb. L2CAP segmente des gros paquets en plus petits paquets en bande de base plus facile à gérer. Ces paquets sont ensuite réassemblés.

Qualité de service

Certaines applications demandent des paramètres spécifiques de qualité de service : pic de bande passante, lance, variation de délai… **L2CAP fourni la QoS demandée si possible et notifie l'application si le lien ne peut pas supporter la demande.**



Transport Protocol Group = permet aux appareils Bluetooth de se localiser les uns les autres, de créer, configurer et gérer leurs liens physiques et logiques
Middleware Protocol Group = nécessaire pour que les applications nouvelles et existantes opèrent sur des liens Bluetooth => PPP, IP, TCP, RFCOMMM



6.5 - BILAN

Avantages du Bluetooth

- Faible consommation d'énergie
- Faible prix des composants bluetooth

Inconvénients du Bluetooth

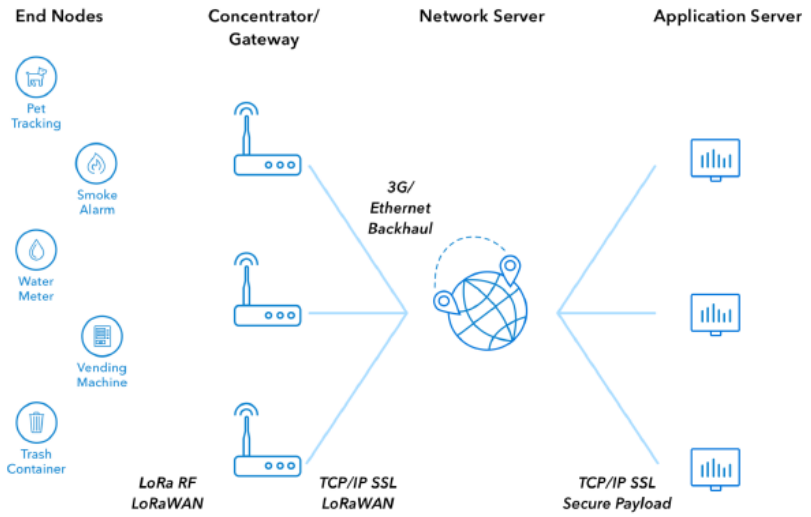
- Les réseaux LAN sans-fil offrent des débits plus élevés et des distance de communication plus grandes
- Possibilité d'interférences sur les bandes de fréquences de 2.4GHz

6.6 - Pour aller plus loin

LoRaWAN = protocole de télécommunication permettant la communication à bas débit, par radio, d'objets à faible consommation électrique communiquant selon la technologie LoRa et connectés à l'Internet via des passerelles, participant ainsi à l'IoT.

Specification	LoRa Technology Support
Standard	LoRa Alliance
Operational Frequencies	Unlicensed ISM band 868, 915 MHz
Modulation	Chirp spread spectrum (CSS)
Coverage Range (Km)	2 - 5 (urban) / 15 (rural)
Data Rate (kbps)	0.3 - 50 (EU) / 0.9 - 100 (US)
Topology	Star

6.6.1 - LoRaWAN architecture



Composants de l'architecture LoRaWAN :

- **End Device (ED)** : les capteurs, peuvent transmettre sur n'importe quel canal, à tout moment en utilisant un débit disponible
- **GW** : fournit la connexion à Internet et évite les collisions
- **Network server** : surveille les GWs et EDs, rassemble les données en évitant les doublons, choisi par quel GW communiquer avec un ED
- **Application server** : l'application IoT en question

End device classes