Paul Alfred Briand

AirNet

Pour vraiment tout pouvoir partager

Sommaire :

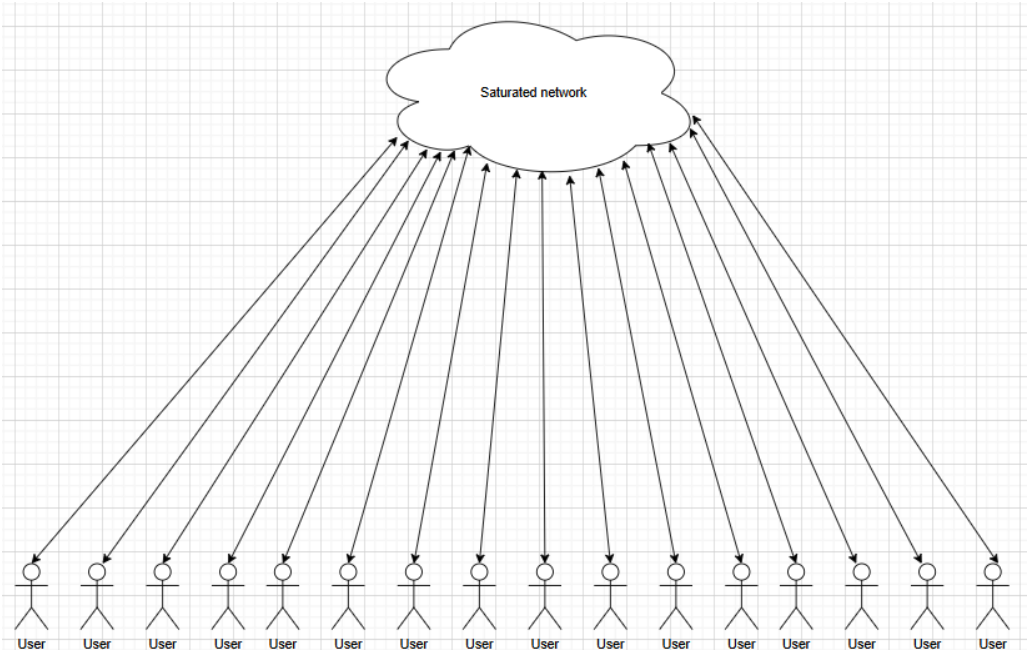
I) Description du projet

AirNet serait une application qui aurait pour but, de pouvoir mutualiser les connexions entre utilisateurs proches face à réseau qui serait surchargé. Ainsi, notre application formerait des groupes de 5 à 8 personnes (à définir), on peut choisir de se regrouper avec les contacts de son téléphone ou bien avec des inconnus, pour que les téléphones rassemblent les données entre-elles (A voir si les données à envoyer au réseau surchargé pourraient être formatées, comme un document Zip, pour économiser des la place, et optimiser le temps de trajet vers le réseau saturé).

A noter que ce projet n'a pas pour but de remplacer le partage de connexion, tous les utilisateurs de l'app utiliseraient leurs propres données mobiles, il n'y aura pas un téléphone qui servirait d'émetteur. Ce serait de la mutualisation intelligente.

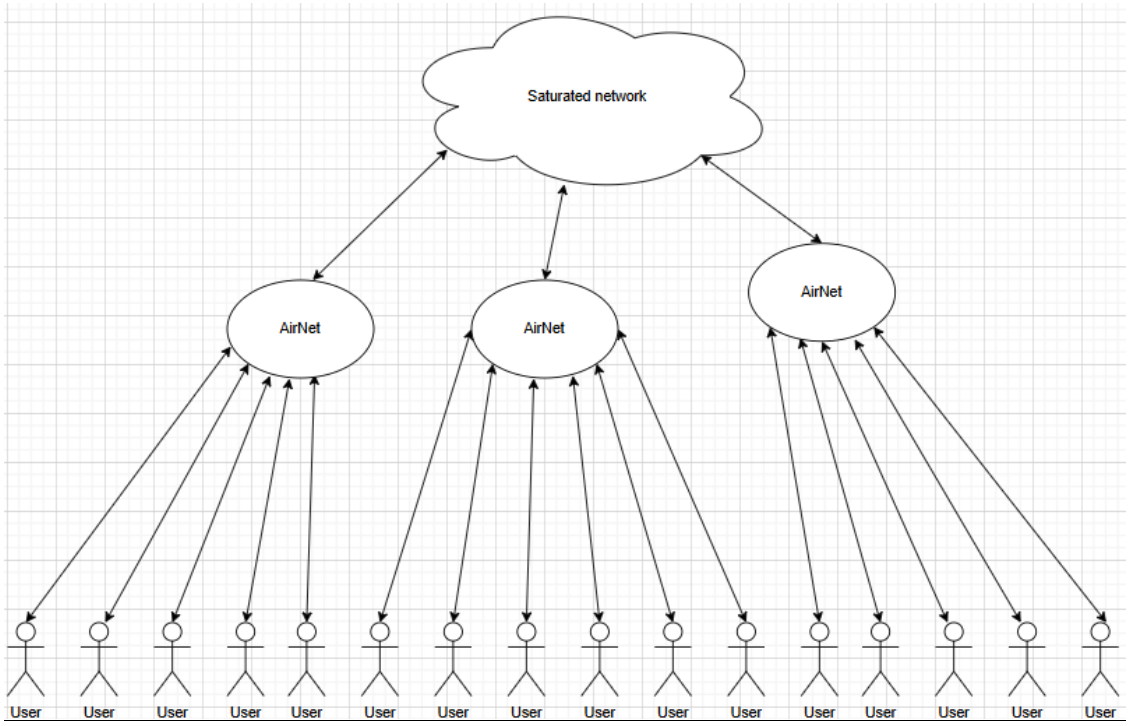
On pourrait imager ce projet de la sorte :

Cas initial :



Chaque utilisateur est son propre est connecté individuellement, le réseau est saturé et le débit est ralenti, la latence augmente et les requêtes échouent.

Avec notre solution :



Désormais, les utilisateurs se regroupent et mutualisent leur connexion. L'appareil avec le meilleur débit pourra servir d'hébergeur pour les autres (à noter que l'hébergement se fait dynamiquement, si l'hébergeur n'a plus de batterie ou si son débit descend, alors on changera d'hébergeur), comme une antenne avant l'antenne, pour envoyer toutes les requêtes des N téléphones d'un seul coup, et les redistribuer d'un seul coup. Les données mutualisées seront bien sûr chiffrées pour éviter toute écoute.

A NOTER QUE, ceci ne serait que la version alpha du projet AirNet. L'objectif final du projet étant de ne pas avoir 1 téléphone hébergeur pour tout le groupe, mais un serveur virtuel réparti entre les téléphones, ainsi, tous les utilisateurs formeraient un mesh réseau local (réseau maillé), les données seraient routées intelligemment au sein du serveur, le model aurait une meilleure scalabilité, serait plus intelligent et surtout, la charge serait répartie entre tous.

Réseau mesh : réseau où chaque appareil (nœud) peut communiquer directement avec les autres sans dépendre d'un point central (comme un routeur ou une antenne).

Dans notre cas, l'information va donc circuler va Bluetooth ou Airdrop entre les appareils en local avant d'être envoyée au réseau cellulaire.

II) Etat de l'art

Dans la limite des solutions actuelles, il est impossible de mutualiser comme on l'entend, la bande passante entre plusieurs appareils. Les seules solutions qui s'en approchent seraient le partage de connexion, qui suppose qu'une seule personne partage ses données mobiles, et qu'il soit le seul à être facturé. Le NetShare Wifi-Hotspot, une application sur Android qui permet de partager son réseau Wifi en utilisant le téléphone de l'utilisateur comme un routeur, c'est un cas de partage classique de données de 1 vers plusieurs. Une alternative est celle des réseaux maillés, où chaque téléphone peut communiquer directement avec les autres sans passer par une antenne. Chaque appareil relaie les données des autres, ce qui permet de partager la charge, d'éviter les points de congestion, et d'améliorer la résilience du réseau. Ce modèle, déjà utilisé en zones rurales ou lors de catastrophes, pourrait être adapté à un usage urbain temporaire comme celui d'AirNet. À terme, AirNet viserait à créer un réseau maillé mobile intelligent, où les connexions sont optimisées entre tous les utilisateurs sans point central.

III) Structures et algorithmes

Commençons par l'algorithme de la version Alpha du projet, le téléphone avec le meilleur débit devient l'émetteur.

Pseudo-code de comment choisir l'émetteur :

```
Importer bibliothèque de mesure de débit
Importer horloge
Importer check niveau batterie

Définir INTERVALLE_VERIFICATION = 20 secondes  # Tout les combien de temps est-ce qu on vérifie le débit
Définir SEUIL_BATTERIE = 20%                   # Le seuil de batterie, un émetteur ne dois pas pouvoir nous lacher
Définir FENETRE_GLISSANTE = 5                  # Moyenne sur 5 mesures

Initialiser une liste pour stocker les derniers débits de chaque téléphone

#On boucle pour vérifier que l utilisateur soit bien connecté à AirNet
while AirNet == True:
    Attendre INTERVALLE_VERIFICATION

    # Vérifier la batterie
    if batterie.niveau <= SEUIL_BATTERIE:
        Ce téléphone n'est pas éligible comme émetteur
    else:
        # Mesurer et stocker le débit
        debit_actuel = mesurer_debit()
        ajouter_debit_a_liste(debit_actuel)

        # Partager le débit avec les autres
        envoyer_debit_aux_autres(debit_actuel)

    # Recevoir les débits des autres téléphones
    pour chaque autre téléphone dans le groupe:
        debit_recu = recevoir_debit()
        ajouter_debit_a_liste_autre_telephone(debit_recu)

    # Calculer la médiane des débits pour chaque téléphone
    pour chaque téléphone dans le groupe:
        si téléphone.batterie > SEUIL_BATTERIE:
            mediane_debit = calculer_mediane(derniers_debits[telephone], FENETRE_GLISSANTE)
        sinon:
            mediane_debit = 0  # Non éligible

    # Choisir l'émetteur
    emetteur = telephone_avec_meilleure_mediane()
    if emetteur != emetteur_actuel:
        notifier_le_groupe(emetteur)
        emetteur actuel = emetteur
```

A noter que l'on regardera le débit en Uploading ET en Downloading.

De plus, il nous faut maintenant définir comment l'information circulera des téléphones à l'émetteur, et comment l'émetteur réussira à redistribuer les données.

Pour développer la version Bêta, on utilisera donc un réseau virtuel mesh comme alternative, pour éviter qu'un seul téléphone ne porte toute la charge de distribution et redistribution des données. Cela garantira que l'émetteur ne brule pas toute sa batterie avant les récepteurs.

Voici un pseudo-code explicatif (Partie gestion de données à améliorer) :

```
# Constantes
MAX_TELEPHONES = 8 ou 5 # Nombre maximum de téléphones dans le réseau mesh
PROTOCOLE_LOCAL = "Bluetooth"  # Protocole pour connecter les téléphones (ex. Wi-Fi Direct ou Bluetooth)

# Initialisation du réseau mesh
reseau_mesh = initialiser_reseau_mesh(PROTOCOLE_LOCAL)

# Boucle principale pendant que AirNet est actif
tant que AirNet est actif :

    # Gestion des nouveaux téléphones
    si un_nouveau_telephone_est_detecte() :
        si reseau_mesh.taille() < MAX_TELEPHONES :
            ajouter_telephone_au_reseau(nouveau_telephone)
            synchroniser_etat_reseau()
        sinon :
            rejeter_nouveau_telephone()  # Le réseau est plein

    # Gestion des données à envoyer depuis n importe quel téléphone
    pour chaque telephone dans reseau_mesh :
        si telephone.a_des_donnees_a_envoyer() :
            # Diviser les données en morceaux selon le nombre de téléphones
            morceaux_donnees = diviser_donnees(telephone.donnees, reseau_mesh.taille())

            # Distribuer chaque morceau à un téléphone du réseau
            pour chaque morceau, telephone_cible dans associer(morceaux_donnees, reseau_mesh.telephones) :
                envoyer_morceau_au_telephone(morceau, telephone_cible)

            # Chaque téléphone envoie son morceau au réseau cellulaire
            pour chaque telephone_cible dans reseau_mesh.telephones :
                si telephone_cible.a_un_morceau() :
                    reponse = telephone_cible.envoyer_au_reseau_cellulaire(telephone_cible.morceau)
                    telephone_cible.stocker_reponse(reponse)  # Stocker temporairement la réponse

            # Rassembler les réponses et reformer les données originales
            reponses = collecter_reponses_du_reseau_mesh()
            donnees_originales = reassembler_donnees(reponses)

         # Renvoyer les données au téléphone d origine (en dehors de la boucle, un téléphone peut recevoir des données sans en avoir émis)
         envoyer_au_telephone_original(donnees_originales, telephone)

    # Gestion des téléphones qui quittent le réseau
    si un_telephone_quitte_le_reseau() :
        retirer_telephone_du_reseau(telephone_partant)
        synchroniser_etat_reseau()

    # Maintenance périodique du réseau
    si temps_de_verifier_reseau() :
        verifier_connectivite_reseau()
        optimiser_chemins_routage()
```

Toute la complexité de se projet repose sur le fait de réussir à compacter les données, et les transmettre vers le réseau émetteur en toute sécurité, ainsi que la redistribution des données après avoir eu un retour du réseau.