

---

# Rapport projet industriel : Blockchains

---



## UE : PROJET INDUSTRIEL MAIN

*Étudiants :*

Tahar AMAIRI  
Hamza RAIS  
Benjamin DESTAL  
Ghassen HACHANI

*Encadrant :*

Maria POTOP-BUTUCARU

*Responsable UE :*

Hacène OUZIA

*Groupe :*

Blockchains

7 mai 2023

# Table des matières

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Motivation</b>	<b>3</b>
<b>3</b>	<b>État de l'art</b>	<b>4</b>
<b>4</b>	<b>Concept</b>	<b>5</b>
4.1	Présentation . . . . .	5
4.1.1	Les entités du réseau . . . . .	5
4.1.2	Les rôles . . . . .	5
4.1.3	L'infrastructure du réseau . . . . .	6
4.1.4	Le portefeuille . . . . .	7
4.2	Fonctionnement . . . . .	8
4.3	Consensus . . . . .	13
4.4	Hypothèses . . . . .	14
<b>5</b>	<b>Implémentation</b>	<b>14</b>
5.1	OMNeT++ . . . . .	14
5.2	Tendermint . . . . .	15
5.3	Jericho . . . . .	18
<b>6</b>	<b>Business Plan</b>	<b>20</b>
6.1	Jericho tokens . . . . .	20
6.2	Initial Coin Offering . . . . .	20
6.3	Coin Pricing . . . . .	21
6.4	L'importance d'une communauté . . . . .	21
<b>7</b>	<b>Site Web</b>	<b>22</b>
7.1	Motivation . . . . .	22
7.2	Page d'accueil . . . . .	22
7.3	Squelette du site . . . . .	24
<b>8</b>	<b>Impacts</b>	<b>26</b>
<b>9</b>	<b>Conclusion</b>	<b>28</b>
<b>10</b>	<b>Bibliographie</b>	<b>29</b>

# 1 Introduction

Dans le cadre de notre projet industriel, nous avons travaillé sur le sujet des crypto-monnaies et plus précisément sur celles adaptées au marché de l'Internet of Things (IoT).

Une crypto-monnaie est une monnaie virtuelle qui s'échange de pair en pair sans l'intervention de banques ou d'autres acteurs intermédiaires selon un protocole spécifique dont le plus populaire est appelé chaîne de blocs, ou blockchain. Une blockchain est une technologie de stockage et de transmission d'informations. En soit, il sagit d'une base de données distribuée dont la gestion est traitée par un réseau de noeuds inter-connectés, où chaque transaction va être validée par un ou des utilisateur.s du système avant d'être enregistrée dans la chaîne contre une commission.

La technologie blockchain permet la fiabilité et la traçabilité des transactions. Les crypto-monnaies sont apparues en 2009 avec le Bitcoin, créé par un programmeur sous le pseudonyme de Satoshi Nakamoto. Cependant, aujourd'hui de nombreuses crypto-monnaies sont apparues sur le marché ayant chacune un protocole unique. Mais beaucoup d'entre elles, et notamment les plus populaires, utilisent des protocoles blockchains très similaires.

Ces crypto-monnaies présentent plusieurs avantages majeurs par rapport à des monnaies traditionnelles. Premièrement, elles sont très sécurisées grâce à la blockchain qui est un système presque inviolable puisqu'il faudrait pirater de nombreuses bases de données différentes, chacune doublement sécurisée. Elles sont aussi transparentes, c'est à dire que les membres du réseau valident et vérifient les transactions, les fraudes peuvent être aisément repérées. Enfin, les crypto-monnaies sont autonomes puisque les utilisateurs se vérifient entre eux et donc suppriment l'intermédiaire humain qui peut être source de méfiance mais aussi de perte d'argent.

## 2 Motivation

Malgré de nombreux avantages, les crypto-monnaies présentent aussi de nombreux défauts. Le défaut majeur du système de validation du BitCoin est qu'il très inégalitaire puisqu'il se base sur la résolution de problèmes cryptographiques et privilégie ainsi les utilisateurs ayant le plus de puissance de calcul d'où une consommation énergétique élevée et des frais pouvant excéder le montant d'une transaction.

Ainsi, ce désavantage à lui seul peut freiner la démocratisation de ces crypto-monnaies dans certains domaines tels que l'IoT. C'est donc dans l'optique de l'effacer que des crypto-monnaies comme IOTA ont émergé.

IOTA est une crypto-monnaie introduite en 2017 dont le but est de fournir un protocole permettant un paiement sécurisé entre les appareils de l'IoT. Et comme toute crypto-monnaie, IOTA a besoin d'un système de stockage et de validation des transactions. Comme dit précédemment, la grande majorité des crypto-monnaies actuelles utilisent le protocole blockchain mais l'intérêt majeur d'IOTA est de ne pas utiliser cette architecture. Ainsi, cela lui permet d'éviter les désavantages inhérents à celle-ci et d'utiliser plutôt une architecture moins conventionnelle appelée Tangle.

Néanmoins, IOTA est toujours dans une version expérimentale et utilise donc un module appelé le "Coordinateur" qui est en charge de la validation des transactions. En d'autres termes, la version actuelle d'IOTA est centralisée, ce qui est à l'opposé de ce que recherchent les crypto-monnaies.

C'est pourquoi nous allons durant ce projet, proposer un concept de blockchain adapté au marché de l'IoT, qui devra donc satisfaire certaines propriétés :

- qu'il soit extensible, vu le nombre grandissant de machines connectées.
- qu'il soit sécurisé contre toute forme d'attaques.
- qu'il ait des frais limités pour éviter la perte dans la valeur échangée.  
Pour cela, il faudrait qu'on évite d'utiliser toute forme de Preuve de Travail intensive.

Nous verrons donc à travers ce rapport le concept initial que nous proposons, des hypothèses l'accompagnant et certains de ses résultats. Nous essayerons aussi de donner une dimension entrepreneuriale à notre projet ;

### 3 État de l'art

Notre projet concerne les crypto-monnaies. Ce concept n'est pas nouveau et une formidable quantité de concepts existent déjà. La spécificité de Iota, et par conséquent de notre projet, est de proposer une crypto-monnaie adaptée au marché de l'internet des objets. C'est cette spécifité qui a guidé nos recherches et réflexions. Actuellement, aucune solution décentralisée n'est adaptée à ce marché.

Nous avons passé de nombreuses semaines en début de projet à nous renseigner sur l'état de l'art dans le domaine des blockchains. Nous ne savions pas encore lesquels de ces papiers allaient nous être utiles. Plus globalement, nous nous sommes aussi renseignés sur le fonctionnement général d'IOTA, avec lequel nous n'étions pas tous familiers. Nous avons donc lu les papiers que certains de nos camarades avaient écrits, décrivant certains résultats invalidant leur fonctionnement de manière décentralisée.

L'un des problèmes d'IOTA étant son système de consensus, nous avons donc exploré de nouvelles pistes, par exemple les consensus TenderMint [2] ou encore Spectre [5]. Cette recherche est d'un réel intérêt puisque notre infrastructure nécessite effectivement un consensus efficace et dont on soit sûr de la correction. Finalement, notre choix s'est porté sur Tendermint car il était efficace et simple à implémenter. Il ne demandait pas de forme de réseau particulière, ce qui s'accordait avec nos concepts.

Enfin, nous nous sommes aussi renseignés sur différentes structures de réseaux. Celle qui a particulièrement retenu notre attention s'appelle CAN [4]. Cette structure n'est pas particulièrement récente, mais elle semblait convenir à notre idée de l'époque pour le réseau. Nous avons continué à réfléchir en gardant en tête nos priorités : vitesse et sécurité du réseau. Finalement, nous avons choisi d'ignorer les structures que nous avons découvertes et avons choisi une idée beaucoup plus simple, qui nous semblait plus naturelle, hiérarchiser le réseau en arbre.

Les recherches ont été importantes pour nous donner des pistes de réflexion et nous familiariser avec les concepts. Nos idées ont beaucoup évolué au cours du projet et avoir toutes ces références nous a permis de rebondir lors de nos remises en question.

## 4 Concept

### 4.1 Présentation

#### 4.1.1 Les entités du réseau

Nous allons distinguer deux types de noeuds dans le réseau :

- les noeuds fixes (NF) : ils forment des regroupements, ou *clusters* et ils ont un unique objectif : valider les transactions, mettre à jour les portefeuilles locaux et finalement diffuser sa nouvelle version. Les NF ne peuvent pas se déplacer, cependant, ils ont différents rôles à remplir qu'on détaillera par la suite.
- les noeuds mobiles (NM) : ce sont tout simplement les utilisateurs du réseau et l'unique action qu'ils peuvent effectuer est d'émettre des transactions vers les NF. Bien plus, de par leur caractère, ils peuvent se connecter et se déconnecter à volonté d'un cluster à un autre. Ces noeuds représentent en d'autres termes tout objet connecté, c'est à dire : des téléphones, des montres connectées, des ordinateurs, des bornes de paiement etc...

Ainsi, si Bob veut effectuer une transaction avec Alice alors les deux doivent nécessairement se connecter à un NF d'un cluster.

#### 4.1.2 Les rôles

Chaque cluster de NF joue un rôle bien particulier au sein du réseau. En effet, on distingue :

- les NF **diffuseurs** : ces noeuds jouent le rôle d'intermédiaire entre le réseau et les NM. Ils reçoivent les demandes de transactions de la part des NM puis les diffusent vers leurs NF **pères**. Finalement, lorsqu'une transaction est validée par le réseau, ils la communiquent aux NM concernés.
- les NF **responsables** : ces noeuds n'ont aucune communication avec les NM et sont en contact uniquement avec des NF. Ils ont pour rôle de valider les transactions à leur échelle (i.e de vérifier dans leurs portefeuilles locaux si les dites transactions ne présentent aucun conflit) et de diffuser les transactions validées à tout le réseau afin que les NF/NM puissent mettre à jour leurs portefeuilles locaux.

#### 4.1.3 L'infrastructure du réseau

La structuration du réseau se fait via un arbre n-aire où les feuilles représentent uniquement des NF **diffuseurs** et le reste des noeuds des NF **responsables**. La notion importante derrière cette structuration c'est qu'elle permet une décomposition géographique par différents niveaux : par ville, par département, par région, par pays puis par continent. Le nombre de niveau n'a pas de limite et dépend de l'implémentation cherchée. En effet, prenons l'exemple d'une ville comme Paris : celle-ci peut-être décomposée par arrondissement mais chacun de ces arrondissements peut-être par la suite décomposés par quartiers :

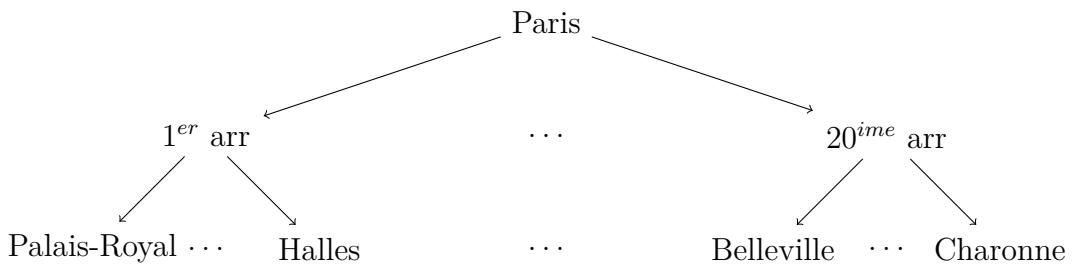


FIGURE 1 – Exemple de décomposition avec la ville de Paris

Nous pouvons continuer à décomposer au niveau des quartiers et ainsi de suite d'où un nombre de niveau illimité. Par analogie, l'infrastructure du réseau s'articulera comme ceci pour cet exemple :

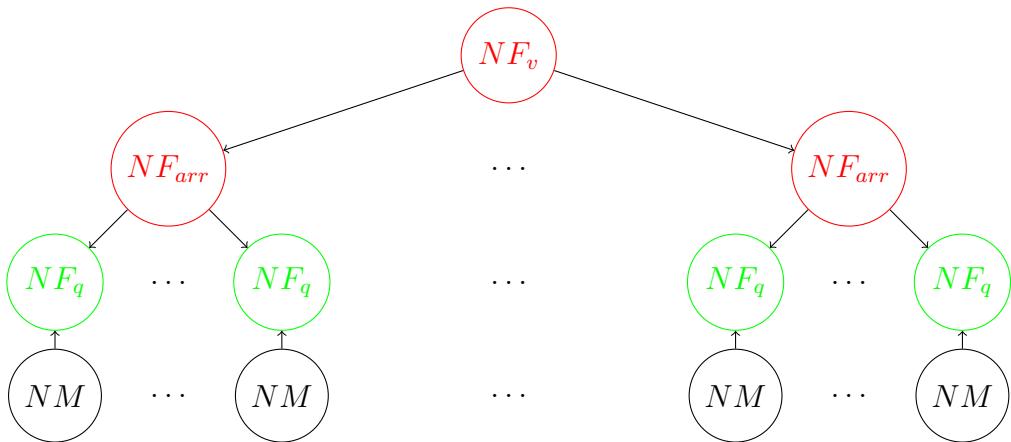


FIGURE 2 – Structure du réseau pour la ville de Paris

### **Légende :**

- $NF_v$  : cluster de NF représentant Paris
- $NF_{arr}$  : cluster de NF représentant un arrondissement de Paris
- $NF_q$  : cluster de NF représentant un quartier d'un arrondissement
- $NM$  : cluster de NM situés géographiquement dans un quartier
-  : NF responsables
-  : NF diffuseurs

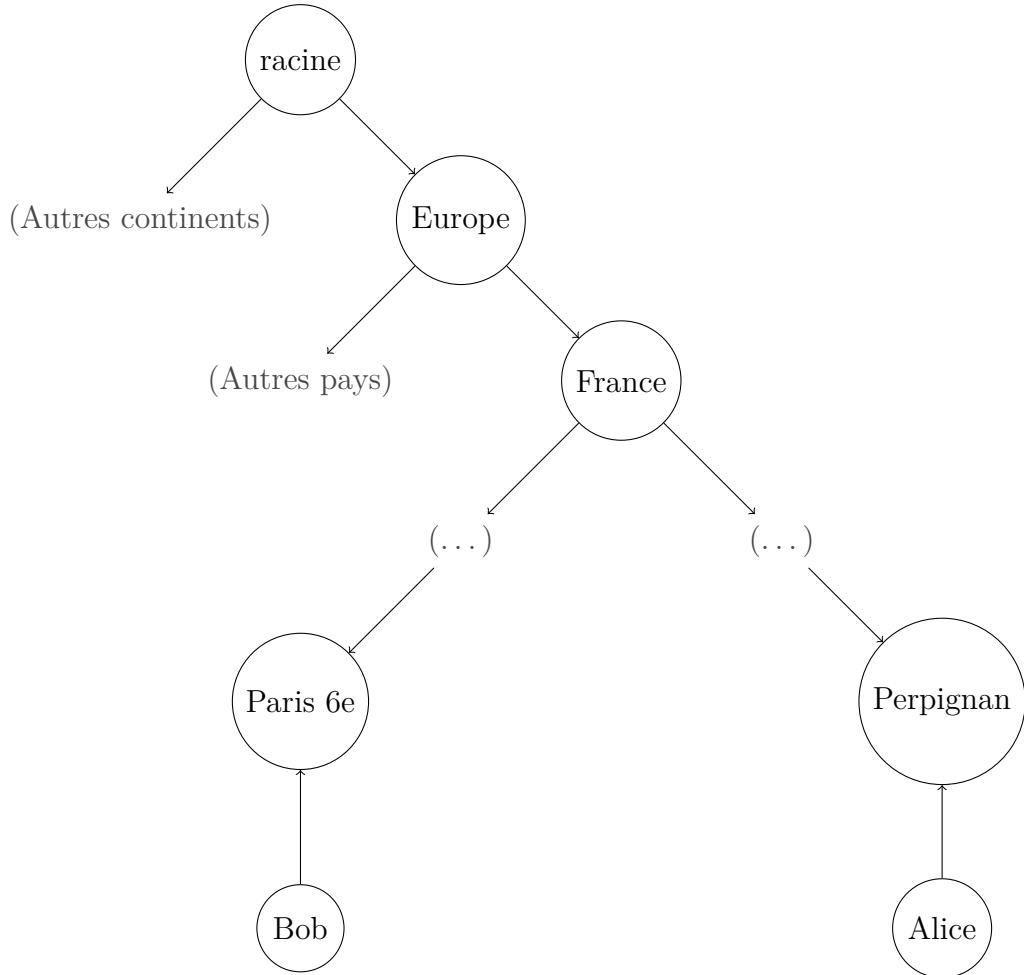
Comme dit auparavant, les feuilles du graphe représentent les NF diffuseurs et donc le dernier niveau de décomposition (ici, le niveau quartier) d'où, une connexion des NM avec les  $NF_q$ . Le reste des noeuds des niveaux supérieurs sont des NF responsables.

#### **4.1.4 Le portefeuille**

Afin de stocker l'historique de transactions, chaque NF aura un portefeuille local en forme d'une succession de blockchain. Celle-ci sera répliquée et propre pour chacun des utilisateurs (i.e des NM). En effet, on utilisera un système clé-valeur où chaque clé (e.g l'ID unique d'un utilisateur par exemple) sera assignée à sa blockchain. Ainsi, pour connaître l'état de Bob il suffira de connaître son ID pour obtenir directement son historique en forme de blockchain. Chaque bloc représentera une transaction effectuée par Bob et contiendra donc : l'ID de la transaction, le montant, la date, le lieu, l'ID du deuxième participant etc... Par ailleurs, le premier bloc, qu'on appellera "Genesis", sera le bloc créé lors de l'inscription de Bob sur le réseau.

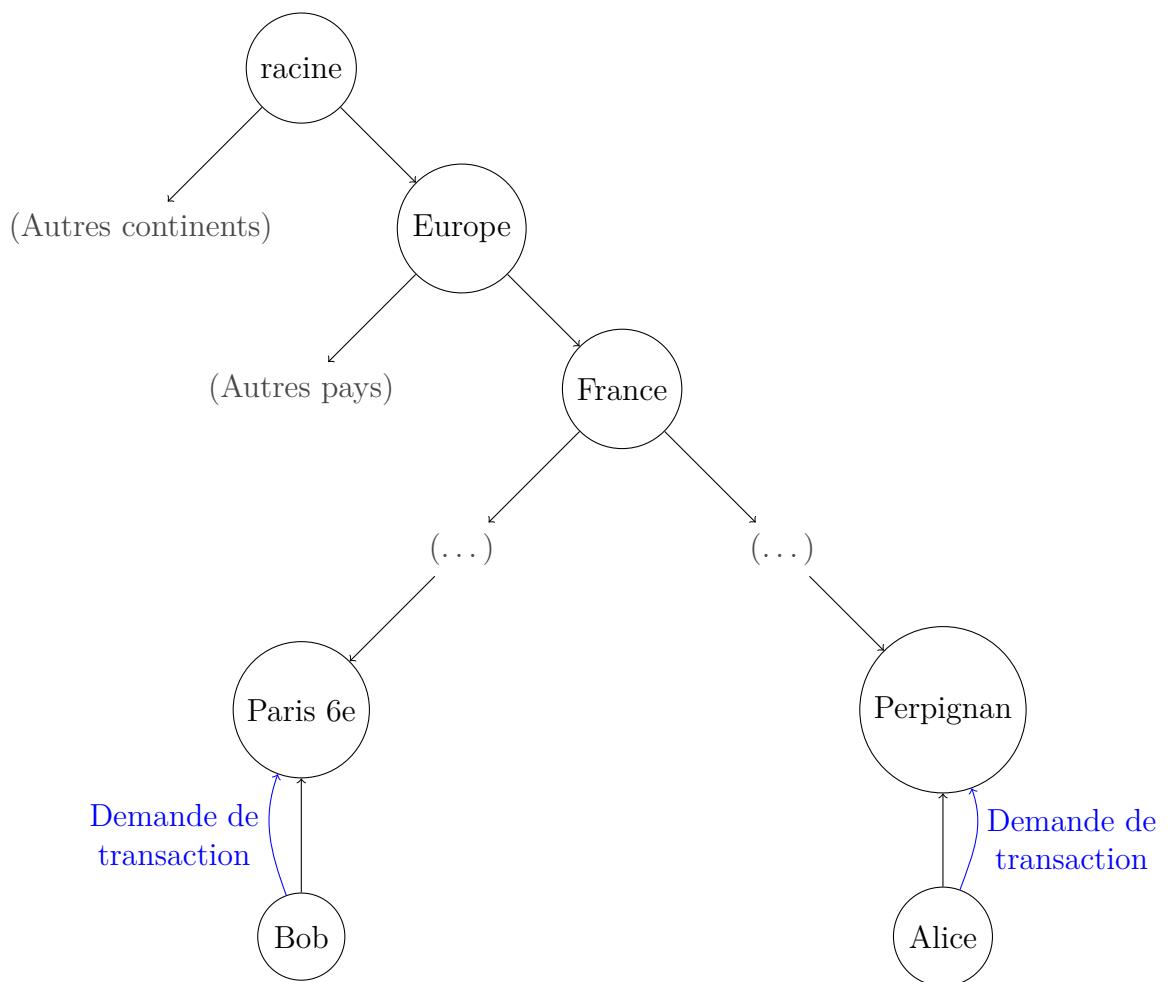
## 4.2 Fonctionnement

Considérons deux utilisateurs, Alice et Bob. Ceux-ci veulent pouvoir effectuer des transactions, et sont donc des noeuds mobiles. Chacun d'entre eux va se connecter au NF correspondant à sa région géographique :

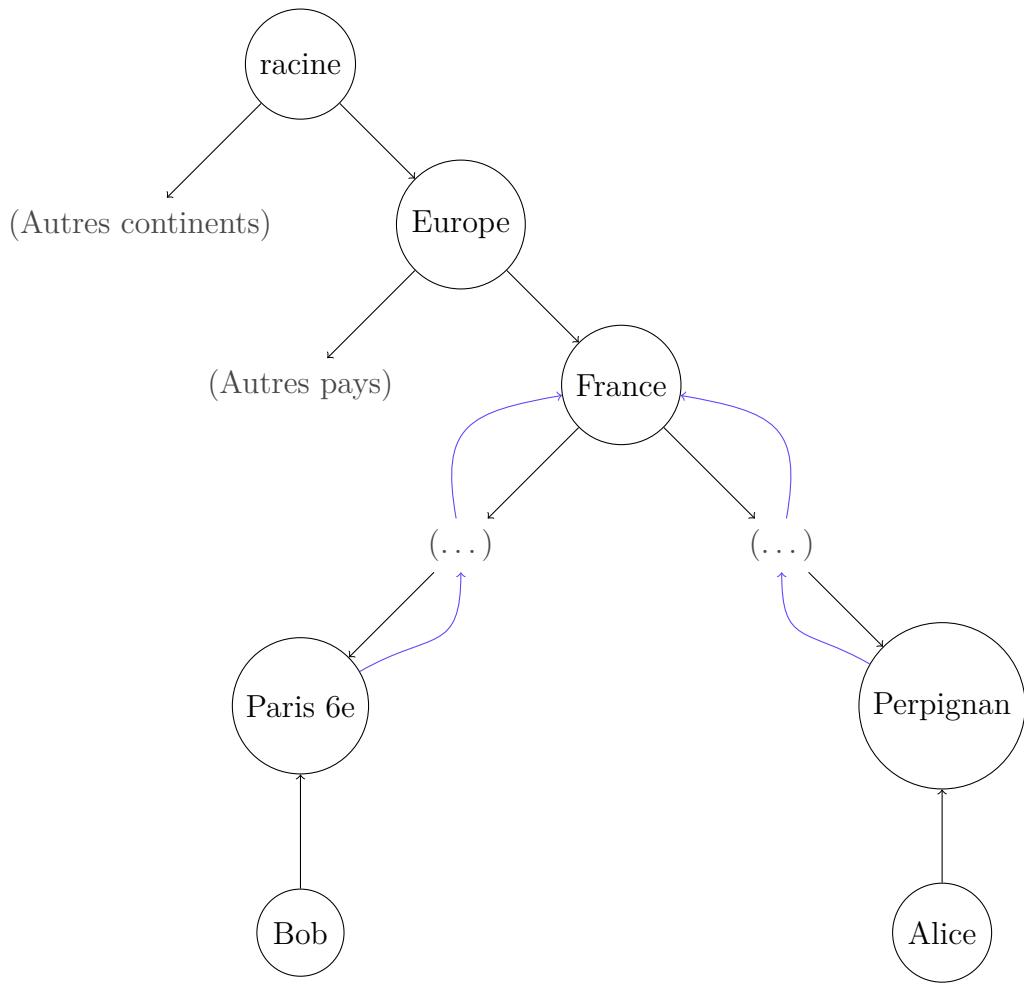


Dans cet exemple, les NF "Paris 6e" et "Perpignan" sont des NF diffuseurs et le reste des NF sont responsables.

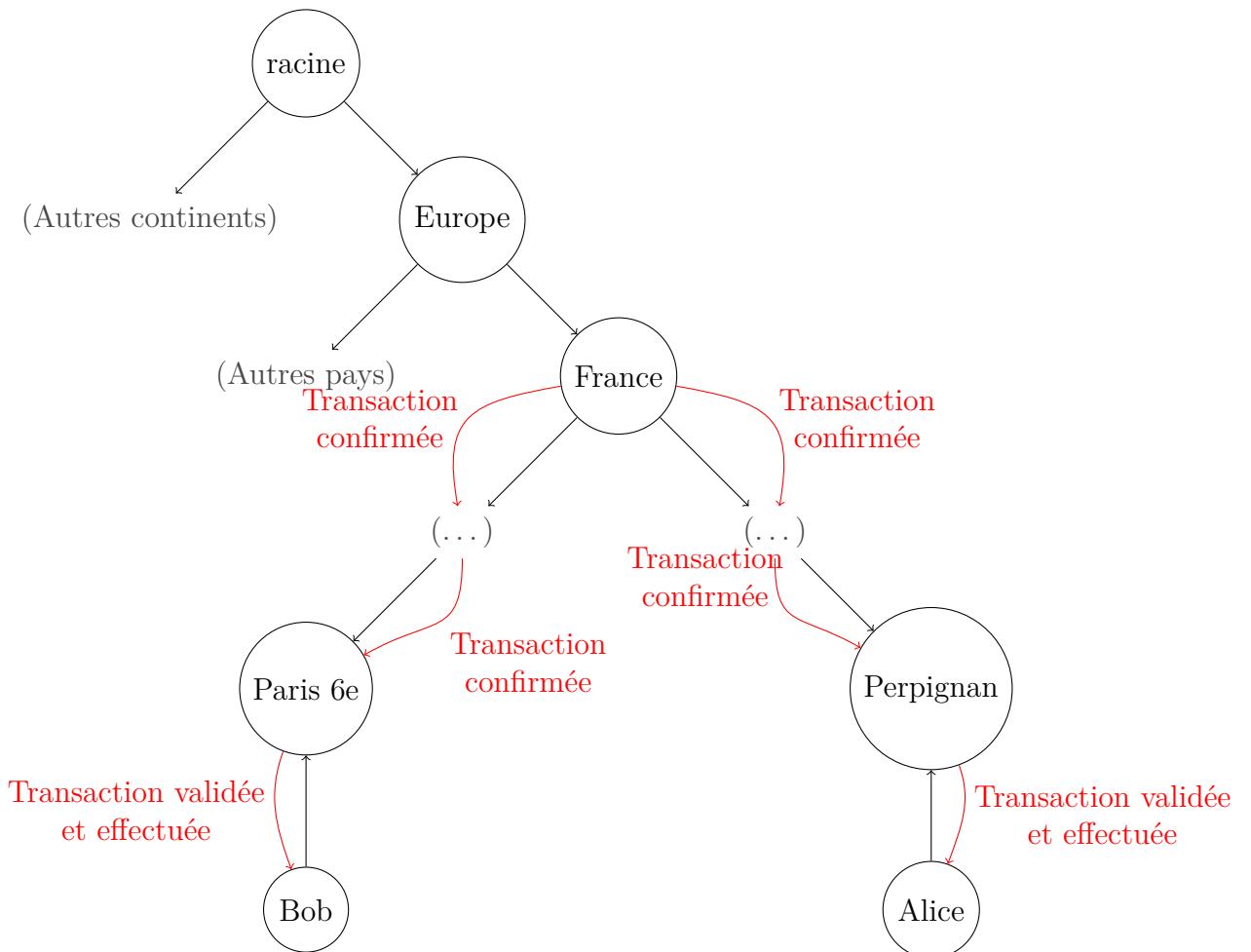
Alice et Bob décident d'effectuer une transaction. Après que la demande de transaction ait été acceptée par les deux, l'information est transmise à leurs NF respectifs. Le NF de Bob possède dans son portefeuille local toutes les informations relatives à son compte, notamment son solde. Ce NF (qui est, on le rappelle, un regroupement d'entités), doit parvenir à un consensus et autoriser, ou non, la transaction. Par ailleurs, la même chose se produit pour Alice :



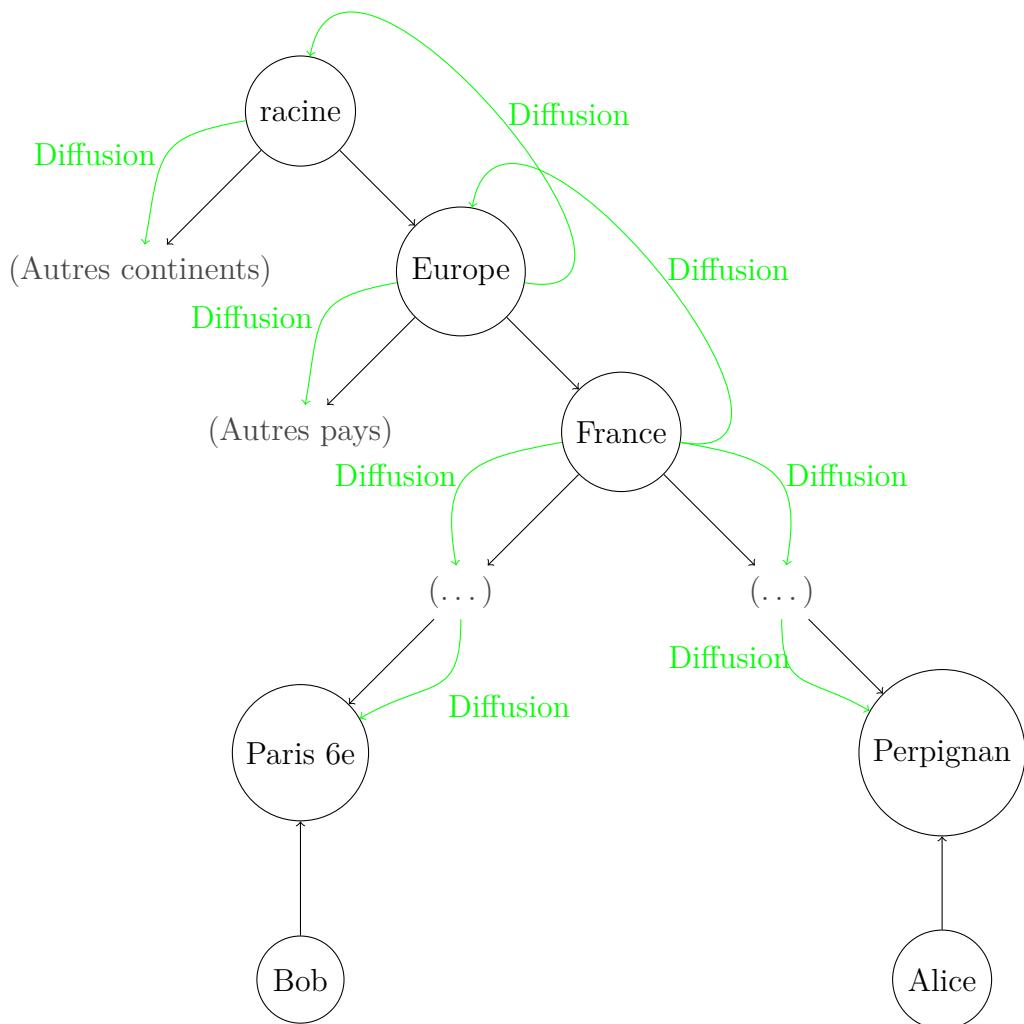
Pour chacun de ces deux noeuds, lorsqu'un consensus est établi, il fait "remonter l'information". C'est à dire qu'il fait remonter les informations de la transaction jusqu'au premier parent commun. De cette manière, celui-ci aura les résultats provenant des deux consensus des NF de Bob et Alice :



Lorsque ce parent reçoit les deux informations provenant des deux noeuds impliqués dans la transaction, il renvoie un message jusqu'à ces deux noeuds, indiquant que la transaction est bien validée, et celle-ci est donc définitivement effectuée. Pour cela, il suffit de vérifier le résultat des deux consensus : si les deux sont positifs alors la transaction est validée :



Au même moment, la nouvelle transaction validée est diffusée tout au sein du réseau pour que chaque noeud fixe puisse mettre à jour son portefeuille local. Pour cela, le parent commun des deux NF de Bob et Alice enverra la nouvelle transaction à ses fils et à ses parents qui feront de même :



## 4.3 Consensus

Le consensus est la brique fondamentale de notre projet : c'est ce concept qui permet d'avoir une structure effectivement décentralisée. C'est le consensus qui permet aux transactions d'être réalisées de manière sécurisée et empêche les transactions frauduleuses d'être effectuées. Pour faire simple, le consensus est simplement la manière dont les entités du réseau se mettent d'accord sur la validité d'une transaction.

Le système de consensus a des problèmes inhérents :

- On constate premièrement que si le réseau est majoritairement "corrompu", alors les transactions frauduleuses n'auront aucune difficulté à être acceptée.
- On remarque aussi que, si l'on a un consensus qui implique l'accord de l'intégralité des entités du réseau, alors dès que le réseau grossit, le consensus devient plus long à se faire.
- Enfin, les deux points précédents permettent de se rendre compte que l'on doit négocier entre la vitesse du consensus et sa sécurité : un consensus qui demande l'avis de beaucoup d'entités est lent, mais il est plus difficile de corrompre un grand nombre d'entités un consensus qui se restreint à une petite population devient donc rapide mais aussi moins sécurisé.

Le premier point ne peut pas être corrigé, par essence du consensus et de la décentralisation. On cherche donc un compromis entre vitesse et sécurité. Ici, notre solution ne provenait pas tant de l'algorithme de consensus utilisé, mais de la manière dont on l'utilise. Nous avons divisé notre réseau en noeuds, et ainsi, lors d'une transaction, le consensus n'aura lieu que dans les noeuds de plus bas niveau, directement concernées par cette transaction.

Une autre idée, évoquée mais non implémentée, est d'ajouter d'autres arbres, en parallèle de l'arbre principal. Le consensus se déroulera aussi dans ces arbres. Nous aurons ainsi plusieurs architectures, basées sur différents éléments, que ce soit géographique, au mérite etc. En ayant ces structures supplémentaires, il devient beaucoup plus difficile d'infiltrer les regroupements responsables du consensus. Et la vitesse finale du consensus, ne s'en retrouve que peu diminuée.

## 4.4 Hypothèses

Une des hypothèses les plus importantes qu'on admet dans le réseau est le délai pour pouvoir se connecter suite à un changement de localisation. Soit  $t_c$  ce délai et  $t_d$  le délai nécessaire pour diffuser une transaction depuis la dernière localisation connue jusqu'à la nouvelle. Dans ce cas,  $t_c$  doit être fixé tel que  $t_c > t_d$  de manière à laisser du temps au réseau de diffuser correctement la nouvelle transaction jusqu'au nouveau point de connexion. Cette contrainte est très importante car elle permet de contrer l'utilisation malveillante d'un VPN et n'impacte que très peu les utilisateurs honnêtes.

De ce premier point, découle la deuxième hypothèse : lorsqu'un NM Bob se connecte à un NF, alors ce dernier a forcément la dernière version vérifiée du portefeuille de Bob. En d'autres termes, celui-ci connaît le dernier état de l'historique de Bob dans tout le réseau ou du moins se rapproche le plus de celui-ci (d'où l'utilisation d'un consensus dans la partie fonctionnement). De cette façon, la vérification d'un NF se basera toujours sur une version authentique et vérifiée du portefeuille de Bob évitant ainsi les doubles dépenses.

## 5 Implémentation

Dans cette section nous allons discuter de l'implémentation de notre concept et des différentes composantes le formant. L'entièreté du code se trouve dans le git suivant [1].

### 5.1 OMNeT++

Pour simuler notre réseau nous allons utiliser un logiciel nommé **OM-NeT++** qui est une bibliothèque et un cadre de simulation réseau écrit en C++ à événements discrets.

Tout projet utilisant OMNeT++ doit contenir trois types de fichiers :

- Un fichier **NED** permettant de définir les composantes du réseau et les paramètres de la simulation. Dans notre cas c'est le fichier `jericho.ned`, définissant tous les types de noeuds dont on a discuté auparavant : NF, NM etc...

- Des fichiers sources **C++** qui contiendront le code à exécuter par les modules. Pour notre simulation, chaque noeud aura son code source qui lui sera associé avec son header.
- Et finalement un fichier **INI** permettant d'initialiser la simulation et sa configuration (nombre de fois qu'on veut simuler, le temps d'exécution, l'interface de simulation (Tkenv (GUI), Qtenv (GUI) ou bien Cmdenv (console)). Dans notre cas c'est le fichier `jericho.ini`.

## 5.2 Tendermint

Comme dit auparavant nous avons choisi le consensus de Tendermint et nous nous sommes basés sur le papier suivant concernant l'implémentation [2]. Cependant, nous avons effectué plusieurs modifications à la version synchronisée. Le choix de cette version par rapport à celle asynchrone n'est pas anodin, car comme le consensus se fait au niveau d'un cluster, les noeuds sont très proches localement et donc le temps de latence est très faible. Par ailleurs, à cause de notre infrastructure, nous avons aussi besoin d'obtenir un consensus très rapidement d'où le choix porté sur la version synchronisée.

Tout d'abord, nous avons implémenté en amont l'entièreté des algorithmes de la version synchronisée sur OMNeT++ (l'implémentation se trouve [ici](#)). Cet algorithme de consensus fonctionne par **tour** et chaque tour contient plusieurs phases :

- Phase de **préproposition** : le noeud "proposeur" envoie aux autres noeuds le block (i.e la transaction) qu'il veut ajouter à la blockchain.
- Phase de **proposition** : les noeuds "validateurs" ayant reçu à leur tour la préposition du proposeur, vérifient le block proposé par rapport à la version local de leur portefeuille et émettent leur réponse à tous les noeuds du cluster. Chaque noeud collecte ainsi les propositions des autres noeuds.
- Phase de **vote** : à ce stade, chaque noeud, validateur comme proposeur, possède un ensemble de propositions. S'il y a *assez* de propositions favorables au block, alors le noeud vote en sa faveur et émet son vote.
- Phase de **synchronisation** : la dernière phase permet d'obtenir la décision portant sur le block, i.e, s'il doit être rejeté ou ajouté à la blockchain. Pour cela, chaque noeud regarde s'il possède *assez* de votes favorables au block. Si c'est le cas, alors celui-ci est ajouté à la blockchain. Cette phase permet aussi de synchroniser tous les noeuds afin de débuter un nouveau tour.

Cette brève explication ne mentionne pas les **epochs** : une epoch représente un petit laps de temps dans lequel un noeud peut proposer plusieurs blocks à valider. Par conséquent, durant un tour il peut y avoir plusieurs epochs permettant ainsi au noeud proposeur de faire valider de nouveaux blocks ou bien de retenter la validation d'un ancien block s'il a été refusé auparavant. Nous avons bien implémenté cette fonctionnalité lors de la programmation du consensus mais celle-ci n'est pas présente dans la simulation complète de Jericho (même s'il est très simple de l'incorporer).

Quand on parle d'*assez* de propositions ou de votes, il est en réalité question d'un seuil que le cluster doit fixer via un paramètre  $f$  représentant l'hypothétique nombre possible de noeuds byzantins présents dans le cluster. Ainsi, pour fixer un vote (resp. décision) concernant un block il faut au moins  $2f+1$  propositions (resp. votes). Ce seuil est très important car il permet de fixer la facilité à laquelle on peut valider un block : si  $f$  est très petit, la validation d'un block ne requiert que très peu de propositions/votes. Dans un monde parfait, cela ne posera aucun problème mais si le réseau est composé d'un nombre important de noeuds byzantins, le consensus devient vulnérable. Il est donc très important de fixer judicieusement cette valeur.

La version synchronisée codée a nécessité certaines modifications afin d'être implémentée sous OMNeT++. En effet, les plus importantes sont l'ajout d'une quatrième phase (celle de la synchronisation) et le fait que le noeud proposeur est le **leader**, i.e, que c'est lui qui amorce chaque début de phase. En réalité, le consensus n'est pas totalement synchronisé car le leader est toujours d'une phase en avance et il n'y que les noeuds validateurs qui sont parfaitement synchronisés. Au cas où le leader ne répond plus, il est toujours possible de continuer le consensus en initiant un nouveau tour : cela peut se faire si les noeuds validateurs ne reçoivent pas de message au bout d'un certain temps.

Finalement, voici les algorithmes modifiés qu'on utilisera dans notre implémentation finale (une démo est disponible [ici](#)) :

---

**Algorithm 1 : Phase PRE-PROPOSE & PROPOSE**

---

**Input** :- ID (ID du noeud)  
- currentProposer (ID du noeud proposeur)  
- v = null (valeur du block pré proposé)  
- proposal (la proposition)  
- timeOut (temps pour délivrer le block)

1 **Phase PRE-PROPOSE :**  
2 if *ID* = *currentProposer* then  
3     proposal  $\leftarrow$  *getValue()* // get a valid block to issue  
4     v  $\leftarrow$  *proposal*  
5     Broadcast(*proposal*)  
6 **Phase PROPOSE :**  
7 if *ID*  $\neq$  *currentProposer* then  
8     v  $\leftarrow$  *Receive()* // receive the preproposal block  
9     // check if the block is valid w.r.t the blockchain  
10    if *isValid(v)* then  
11      proposal  $\leftarrow$  v  
12      Broadcast(*proposal*)  
13    else  
14      proposal  $\leftarrow$  null  
15    Collect the proposals from the other nodes

---

---

**Algorithm 2 : Phase VOTE & SYNCHRONISATION**

---

```
Input  :- vote = null (valeur du vote)
        - f (nombre hypothétique de byzantins)
        - decision (état final du block)
        - nodeNumber (nombre de noeuds dans le cluster)

1 Phase VOTE :
2 if  $\exists 2f + 1$  unique proposals of prop  $\in$  collected proposals then
3   |   vote  $\leftarrow$  prop
4   |   Broadcast(vote)
5 while timeOut do
6   |   Collect the votes from the other nodes
7 Phase SYNCHRONISATION :
8 if  $\exists 2f + 1$  unique votes of vo  $\in$  collected votes then
9   |   decision  $\leftarrow$  vo
10 currentProposer  $\leftarrow$  (currentProposer + 1) mod nodeNumber
11 if ID = currentProposer then
12   |   Launch phase PRE-PROPOSE
```

---

### 5.3 Jericho

Tout comme l'algorithme de consensus de Tendermint, l'implémentation de Jericho se fera sous OMNeT++. Premièrement, pour construire le réseau en forme d'arbre n-aire nous allons utiliser un script python qui écrira en forme de fichiers CSV :

- les **connexions** entre les noeuds
- les **coordonnées** de chaque noeud dans l'arbre afin de permettre un affichage clair à l'aide de Qtenv
- et finalement, les **plus petits ancêtres communs** pour chaque feuille du réseau (i.e les noeuds NF diffuseurs).

Nous allons construire de cette manière le réseau car la syntaxe du langage NED d'OMNeT++ est très limitée et ne permet pas de construire des topologies complexes. Les fichiers CSV seront lus en amont par un module spécial qui n'aura qu'un rôle dans l'implémentation : construire l'arbre n-aire. Ce module est le **ConfiguratorModule**.

Les entités du réseau décrites au chapitre 4 seront quant à elles modélisées par les modules suivants :

- **NFs** : pour les noeuds diffuseurs ("spreader")
- **NFr** : pour les noeuds responsables ("responsible")
- **NM** : pour les noeuds mobiles

Contrairement à notre concept initial, nous implémenterons qu'un module NM qui s'occupera d'envoyer de façon aléatoire les transactions aux NFs (pour cela il utilisera une loi exponentielle). Bien plus, les clusters ne seront présents qu'au niveau des feuilles :

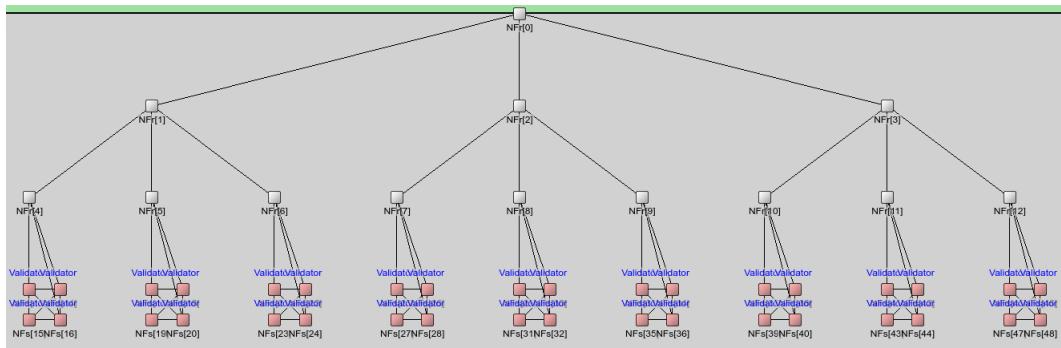


FIGURE 3 – Réseau de Jericho

Détaillons maintenant comment le réseau fonctionne :

1. Tout d'abord, le noeud NM génère une transaction et l'envoie vers deux feuilles pour être validée.
2. Les noeuds NFs ayant reçu la transaction initient un consensus au sein de leur cluster respectivement. S'ils ne peuvent initier un consensus car ce n'est pas encore leur tour, la transaction est cachée dans une file d'attente (avec la structure `std::queue` pour être traitée ultérieurement).
3. Lorsque le résultat du consensus est obtenu, il est envoyé vers le plus petit ancêtre commun entre les deux NFs traitant la demande. Lors de l'acheminement de la demande vers le NFr, le chemin est enregistré avec la transaction (i.e quels noeuds ont fait transiter la demande).
4. Lorsque la transaction est reçue par le NFr, celui-ci vérifie le résultat des consensus des deux NFs. Si l'un d'eux rejette la transaction, la transaction l'est aussi. La réponse est par la suite envoyée vers les deux NFs en utilisant les chemins enregistrés.

Tout comme pour les NFs, un NFr peut cacher une demande le temps de recevoir le résultat du second consensus.

5. Finalement, le résultat est aussi diffusé dans tout le réseau et les NFs renvoient le résultat de la transaction vers le noeud NM.

Une démonstration complète du fonctionnement du réseau se trouve [ici](#).

## 6 Business Plan

Le lancement de notre concept dans le marché des cryptomonnaies nécessite la mise en place d'un business plan prévoyant les étapes de la création et de l'évolution de la "startup" Jericho. On a choisi dans ce cadre de concevoir des jetons (tokens) associés à notre cryptomonnaie

### 6.1 Jericho tokens

Ce sont une forme moderne de jetons qui donnent à celui qui les détient le droit de les échanger contre des services. Les avantages de ces jetons c'est qu'ils assurent la sécurité et la confidentialité des transactions tout en étant échangeables directement et sans l'intervention d'un tier. Grâce à leur adaptabilité sur les systèmes IoT et leurs faibles frais de transactions, ces tokens faciliteront les échanges des biens et des services à l'échelle internationale.

### 6.2 Initial Coin Offering

Une ICO (Initial Coin Offering) est une méthode de levée de fonds, fonctionnant via l'émission d'actifs numériques échangeables contre des cryptomonnaies durant la phase de démarrage d'un projet.

Pour faire une ICO, il faut commencer par définir le "white paper" associé à Jericho. Le white paper est un document similaire au document d'investissement classique pour le lancement d'une entreprise en bourse. Ce document précise comment seront distribués les fonds levées et combien les investisseurs peuvent gagner de jetons. Il décrit également nos services et comment leur associer des jetons ainsi que les paramètres de vente de ces jetons (dates du début et de la fin de la vente, prix initial, agenda évolutive, nombre total des jetons qui seront disponibles, le nombre créé à chaque période si la vente s'étale sur plusieurs périodes).

## **6.3 Coin Pricing**

L'objectif de notre projet c'est de proposer un jeton qui tend vers un prix stable d'équilibre à partir d'un prix initial. Le prix de lancement initial est conditionné par le succès de la levée de fonds qui doit être réalisée en amont. Ce prix doit anticiper entre autres les frais des transactions tout en reflétant le service fourni. Son évolution sera guidée par les spéculations sur le marché et l'activité de la communauté Jericho.

Dans l'optique de concrétiser notre projet en une startup, notre professeur référente nous a mis en relation avec PEPITE (Pôle Etudiant Pour l'Innovation, le Transfert et l'Entrepreneuriat) qui a pour mission d'aider les étudiants de la Sorbonne Université à concrétiser leurs projets et à trouver des investisseurs.

## **6.4 L'importance d'une communauté**

Sans une communauté qui supporte une cryptomonnaie, l'ICO n'aura pas de participants lors de la mise en vente des jetons. C'est donc un indicateur majeur pour le succès du projet. La création de l'intérêt se fait généralement en ayant recours à l'intégration d'une équipe spécialisée en marketing "digital". Ceci assurera un style unique de marketing et organisera la présence sur les différents réseaux sociaux.

L'une des plateformes principales pour organiser les activités et les partages au sein de la communauté est Telegram Cryptocurrency Community. En effet, l'application Telegram représente un outil de communication et de partage sécurisé et largement répandu dans les cercles de passionnés de blockchain.

Les communautés blockchains sont un moteur indispensable. Quelles financent ou partagent des idées de développement, les communautés participent à sa démocratisation. Avoir un site web ouvert pour le public contenant les détails techniques et commerciaux de Jericho et qui permet aux investisseurs et aux curieux d'apprendre plus sur nos projets et technologies semble être indispensable.

# 7 Site Web

## 7.1 Motivation

Dans le cadre de notre projet, notre enseignante référente nous a demandé d'aller plus loin : préparer un site web afin de communiquer les informations sous-jacentes au projet, comme si on allait directement former une start-up souhaitant une levée de fonds de la part d'investisseurs.

Au début de ce projet, nos compétences en programmation web n'étaient pas développées. Par conséquent, on a dû apprendre à utiliser de manière efficiente HTML et CSS. HTML est le langage qui permet d'afficher du texte sur un site et CSS permet de magnifier le texte.

## 7.2 Page d'accueil

Étant donné que le site web est destiné au domaine de la blockchain, nous avons décidé de lui donner une page d'accueil claire séparée en plusieurs onglets de navigation à savoir :

- l'onglet **notre équipe** : il permet de présenter nos motivations, nos expériences professionnelles et notre bagage technique.
- l'onglet **Jericho** : il permet d'introduire le consensus d'abord imaginé puis construit au cours de ce projet. C'est un onglet fondamental pour créer une relation de confiance entre les noeuds du réseau intéressé par le projet car il présente les technologies utilisées.
- l'onglet **réseau** : il permet d'expliquer la structure du réseau c'est-à-dire de la disposition géographique des noeuds et de leurs comportements respectifs.
- l'onglet **investisseurs** : il présente la stratégie marketing et d'Initial Coin Offering (ICO) mais également d'évoquer la vision long-terme du projet.
- l'onglet **découverte** permet à quiconque intéressé par le projet n'ayant pas l'habitude d'entendre certains termes techniques utilisés, de pouvoir avoir accès à une explication brève et vulgarisée sur quelques notions fondamentales.
- enfin, l'onglet **contact** regroupe uniquement les adresses mails des parties prenantes du projet. En effet, dans celui-ci, il figure les liens LinkedIn de l'ensemble des membres de l'équipe.

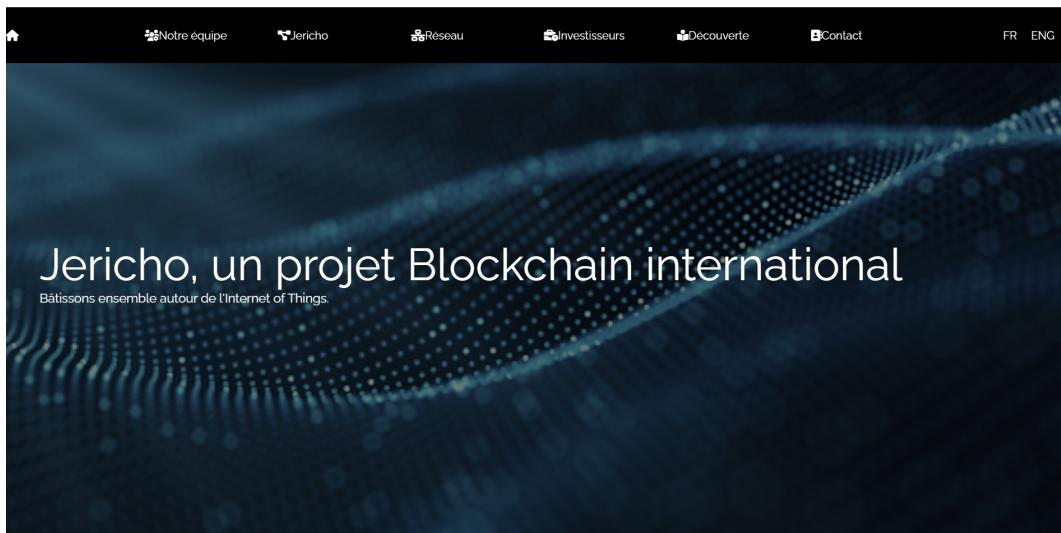


FIGURE 4 – Page d'accueil en français

Afin d'être adapté au marché des cryptomonnaies, le site Web peut également être lu en anglais :

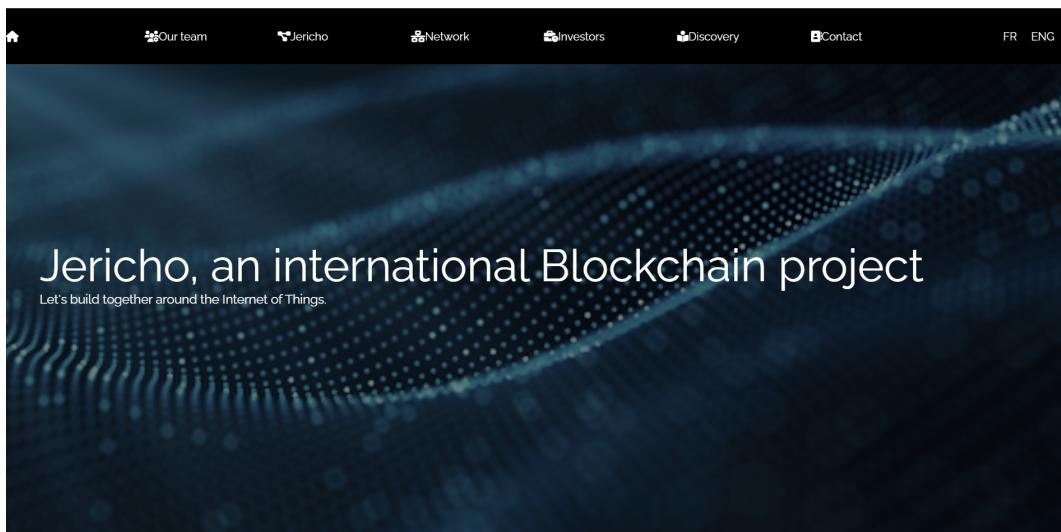


FIGURE 5 – Page d'accueil en anglais

Pour que le site soit lisible en anglais, il n'y a pas de traduction automatique de la part d'un tiers comme Google Traduction. Par contre, nous avons créé un dossier supplémentaire nommé **eng** afin de mettre du code HTML contenant le texte traduit au sein des balises.

### 7.3 Squelette du site

Cette sous-section tentera de résumer la structure du site sans énumérer l'entièreté du code. Tout d'abord, le site est composé d'une barre de navigation qui est la première `<div>` du site avec une classe nommée `top` afin de pouvoir écrire du code CSS permettant de laisser la barre de navigation immobile lors du défilement.

Chacun de ses onglets est accompagné par des icônes choisies du site de Font Awesome. De plus lorsque l'on défile les onglets, une couleur bleue montre quel onglet on survole avec la souris, ceci est rendu possible avec CSS avec `:hover`. Avec la même philosophie, CSS souligne les onglets de langue lorsqu'on les survole. Ensuite, le dessous de la barre de navigation dans la page d'accueil est un header au sens de HTML et il est accompagné d'un code CSS qui permet d'obtenir une image de fond.

Plus bas, nous nous dirigeons vers une autre `<div>`, celle de l'onglet notre équipe, qui est également une `<div>` dont la particularité est l'ensemble des `<div>` imbriquées qui permettent de représenter l'équipe par un ensemble de cinq cartes individuelles. Le reste du site est un ensemble de grandes `<div>` enveloppant chacune un axe représenté par un onglet dans la barre de navigation.

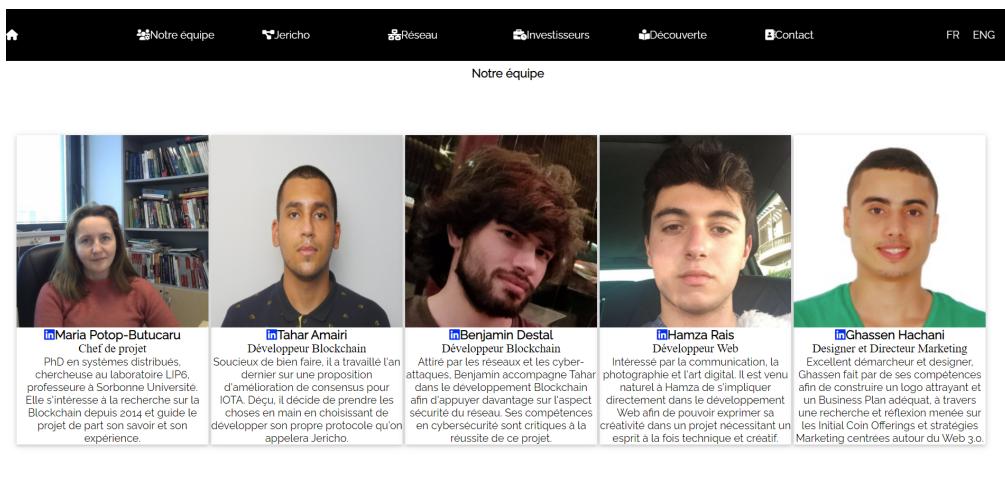


FIGURE 6 – Cartes individuelles

Ces cartes sont accompagnées d'un logo LinkedIn procuré par Font Awesome à l'aide d'un code Javascript. Lorsque l'on clique dessus, cela nous mène vers le LinkedIn de la personne souhaitée, cela est rendu possible avec HTML par une balise `<a>` avec l'attribut `href` contenant le lien du LinkedIn. Ensuite, on ferme la `<div>` représentant l'ongle "notre équipe" donnant place à la `<div>` du consensus Jericho. Sa particularité réside dans le choix de souligner à gauche chacun des paragraphes à l'aide de la sélection `.jericho-mission .text` et de l'instruction `border-left`. Le titre "nouveau consensus" est rendu possible par le code CSS attribué à la classe `centered-paragraph`.



FIGURE 7 – Paragraphes espacés et bordurés

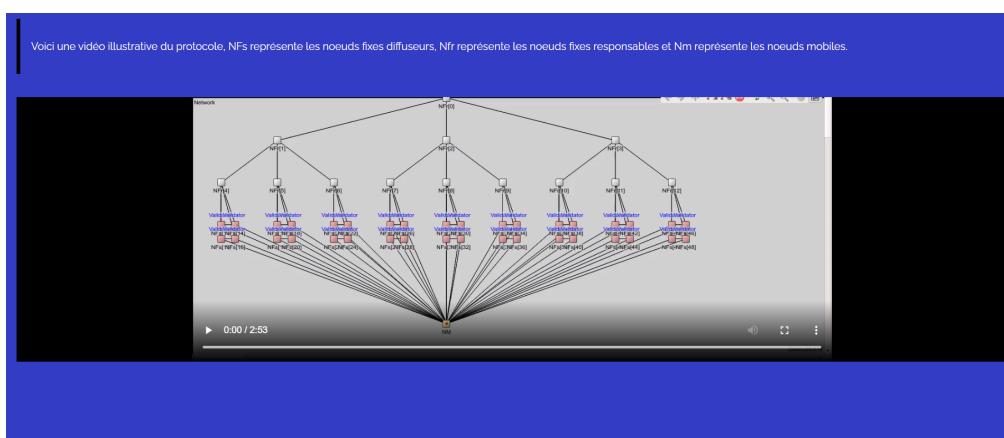


FIGURE 8 – Vidéo explicative

Également, l'onglet "réseau" décrit le rôle et comportement des différents noeuds du réseau Jericho dans un fond de couleur noir.

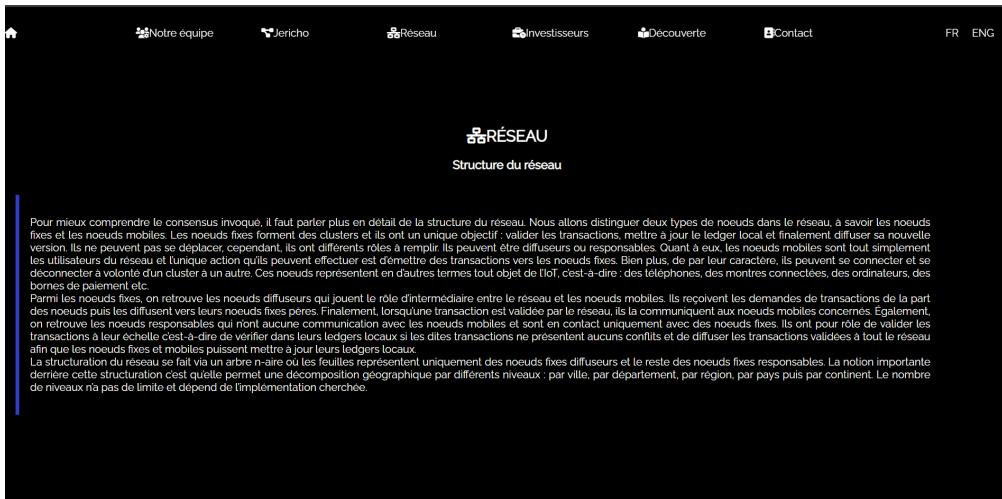


FIGURE 9 – Axe Réseau

Le reste des <div> sont tirés d'un raisonnement analogue en attribuant des couleurs différentes et choisies pour donner un effet dynamique au site. Finalement, le site peut-être accessible via ce [lien](#).

## 8 Impacts

Les cryptomonnaies sont d'actualité et ont eu déjà de nombreux impacts aussi bien environnementaux, éthiques que sociétaux. Concernant les impacts environnementaux, ils sont négatifs puisque la majorité des crypto-monnaies utilisent la technologie de blockchain et ont donc besoin de serveurs/mineurs pour fonctionner qui sont très coûteux en énergie. Par exemple, on est de l'ordre de **147 TWh par an** pour le Bitcoin, ce qui le placerait en **27ème pays** le plus consommateur d'énergie entre l'Egypte et l'Ukraine :



FIGURE 10 – Consommation annuelle du BTC, source : [3]

Ainsi, la popularisation des cryptomonnaies pose donc un sérieux problème énergétique et donc environnemental.

D'un point de vue sociétal, les crypto-monnaies peuvent et ont déjà modifié les moyens de paiements traditionnels et donc bousculé une grande partie de notre société (voir l'offensive du gouvernement chinois contre la production et le trading du Bitcoin par exemple). Bien plus, les applications relatives aux cryptomonnaies ne s'arrêtent pas au monde de la finance mais peuvent affecter comme nous l'avons vu avec notre concept des domaines comme de l'IoT, qui aujourd'hui est présent tout autour de nous. Il est donc très important que le monde de la cryptomonnaie s'adapte aux enjeux environnementaux et sociétaux et c'est ce que nous avons essayé de faire : de proposer un concept "écoresponsable".

## 9 Conclusion

Ce projet a été l'occasion pour nous de travailler sur différents domaines : cryptomonnaies, sécurité, algorithmique distribuée, développement web, l'initial coin offering... Par ailleurs, nous avons aussi pu découvrir légèrement le monde de l'entrepreneuriat. Malheureusement, nous n'avons pas pu fixer un rendez-vous avec la [SATT Lutec](#) et PEPITE pour découvrir les formations nécessaires pour lancer une start-up.

Notre premier objectif était de proposer un concept afin d'adapter les cryptomonnaies au monde de l'IoT, i.e, un réseau de noeud sécurisé, distribué et sans Preuve de Travail. A l'aide de notre première implémentation, nous avons réussi à montrer que notre réseau pouvait fonctionner et qu'il y a encore énormément à explorer. En effet, Jericho vise à être très flexible et nous avons énormément d'idée :

- pourquoi se limiter à un algorithme de consensus lorsqu'il est possible d'avoir plusieurs types de consensus pour chaque cluster ? On pourrait reléguer cette responsabilité aux noeuds NFs.
- afin d'améliorer la sécurité, un noeud NM peut être obligé d'envoyer sa transaction vers plus de deux noeuds NFs.
- on peut aussi ajouter un niveau supplémentaire dans l'arbre contenant des noeuds planificateurs qui vont aller distribuer selon la charge du réseau les transactions vers les clusters les plus libres. Cela sera aussi un moyen d'augmenter la sécurité car un attaquant aura beaucoup plus de difficulté à prédire les noeuds NFs qui recevront les transactions.
- chaque cluster pourra former une "société" dans laquelle chaque noeud peut voter pour le type de consensus qu'il souhaite utiliser, l'économie qu'il souhaite mettre en place (par exemple les frais d'utilisation) etc...
- on peut mettre en place un système de crédit "social" où les noeuds les plus honnêtes et performants pourront avoir accès aux niveaux supérieurs de l'arbre. En plus d'une topologie, l'arbre pourra représenter une hiérarchie sociale.
- pour les clients qui effectuent des transactions souvent à l'international, il sera possible d'ouvrir un lightning network.
- finalement, pour une meilleure communication, les NFs peuvent utiliser CAN.

## 10 Bibliographie

- [1] Tahar AMAIRI, Hamza RAIS, Benjamin DESTAL et Ghassen HACHANI. *Implementation of Jericho using OMNeT++*. URL : <https://github.com/T-amairi/Jericho>.
- [2] Yackolley AMOUSSOU-GUENOU, Antonella del POZZO, Maria POTOP-BUTUCARU et Sara TUCCI-PIERGIOVANNI. *Dissecting Tendermint*. URL : <https://arxiv.org/abs/1809.09858>.
- [3] University of CAMBRIDGE. *Cambridge Bitcoin Electricity Consumption Index*. URL : <https://cbeci.org/>.
- [4] Sylvia RATNASAMY, Paul FRANCIS, Mark HANDLEY, Richard KARP1 et Scott SHENKER. *A Scalable Content-Addressable Network*. URL : <https://conferences.sigcomm.org/sigcomm/2001/p13-ratnasamy.pdf>.
- [5] Yonatan SOMPOLINSKY, Yoad LEWENBERG et Aviv ZOHAR. *Serialization of Proof-of-work Events : Confirming Transactions via Recursive Elections*. URL : <https://eprint.iacr.org/2016/1159.pdf>.