



Master IASD - Module Blockchain

Année Universitaire 2025-2026

SMART CITY : TRAFFIC CORE

Blockchain pour la Gestion Décentralisée du Trafic Urbain

Auteurs :

Andrew Kiswaga John
El Maimouni Motaouakel
Znita Mohammed
Khabali Ayoub

Encadré par :

Pr. Ikram BEN ABDEL
OUAHAB

3 janvier 2026

Table des matières

Résumé	4
1 Introduction Générale	5
1.1 Contexte et motivation	5
1.2 Problématique	5
1.3 Objectifs du projet	6
2 Étude et Conception du Projet	7
2.1 Analyse des besoins	7
2.1.1 Besoins fonctionnels	7
2.1.2 Besoins non fonctionnels	8
2.2 Diagramme de cas d'utilisation	8
2.3 Diagramme d'architecture globale	9
2.4 Diagramme de séquence	9
3 Infrastructure Blockchain Hyperledger Fabric	11
3.1 Vue d'ensemble du réseau	11
3.1.1 Topologie des organisations	11
3.2 Architecture des canaux	12
3.2.1 Canal city-traffic-global	12
3.2.2 Canal emergency-ops	12
3.3 Choix technologiques	13
4 Développement des Smart Contracts (Chaincode)	14
4.1 Logique métier décentralisée	14
4.1.1 Contract Traffic (city-contract)	14
4.1.2 Contract Emergency (emergency-contract)	15
4.1.3 Contract Consensus (Laboratoire expérimental)	15
4.2 Structure du Ledger et World State	16
4.3 Contrôle d'accès basé sur les rôles (RBAC)	16
5 Laboratoire de Consensus et Étude Comparative	18
5.1 Théorie des mécanismes de consensus	18
5.1.1 RAFT : La Tolérance aux Pannes d'Arrêt (CFT)	18
5.1.2 PBFT : La Résilience Byzantine (BFT)	19
5.1.3 PoA : La Preuve d'Autorité	19
5.2 Protocole expérimental et métriques	19
5.2.1 Environnement de test	19
5.2.2 Métriques de performance retenues	20

5.3	Analyse des résultats et performances	20
5.3.1	Débit transactionnel (TPS)	20
5.3.2	Latence et temps de finalité	21
5.3.3	Résilience aux pannes Byzantines	21
5.4	Simulation de pannes et attaques au sein du laboratoire	21
5.5	Synthèse comparative	22
5.6	Conclusion du chapitre	22
6	Cyber-sécurité et Mini-SOC par l'IA	23
6.1	La nécessité d'une supervision intelligente	23
6.1.1	Limites du "Tout Blockchain"	23
6.1.2	Objectifs du Mini-SOC	23
6.2	Architecture Tri-Agents du Mini-SOC	24
6.2.1	L'Agent Capteur (Sensor Agent)	24
6.2.2	L'Agent Analyseur (Analyzer Agent) : Mistral 7B	24
6.2.3	L'Agent Répondeur (Responder Agent)	25
6.3	Orchestration avec n8n et Notification	25
6.3.1	Le Workflow d'Incident	25
6.4	Exemples de Menaces Détectées	25
6.5	Conclusion du chapitre	26
7	Interface Utilisateur et Dashboard Cyberpunk	27
7.1	Philosophie de Design et Expérience Utilisateur	27
7.1.1	L'Esthétique Cyberpunk : Entre Immersion et Performance	27
7.1.2	Hiérarchie de l'Information	28
7.2	Architecture Technique du Frontend	28
7.2.1	Fondations : React et Vite	28
7.2.2	Visualisation Géospatiale : Leaflet et React-Leaflet	28
7.2.3	Communication Temps Réel : WebSockets vs Polling	29
7.3	Composants Avancés et Modules Spécialisés	29
7.3.1	Le Dashboard du Laboratoire de Consensus	29
7.3.2	L'Interface du Mini-SOC (Cyber-Sentinel)	29
7.4	Défis d'Implémentation et Optimisations	30
7.4.1	Gestion de l'État Complexe	30
7.4.2	Performance du Rendu Cartographique	30
7.5	Accessibilité et Adaptabilité	30
7.6	Conclusion du chapitre	30
8	Conclusion et Perspectives	32
8.1	Synthèse des Réalisations	32
8.2	Apports Scientifiques et Techniques	33
8.3	Difficultés Rencontrées et Enseignements	33
8.4	Perspectives d'Évolutions Futures	33
8.4.1	Identité Numérique Décentralisée (DID)	34
8.4.2	Scalabilité et Interopérabilité	34
8.4.3	Vagues Vertes Prédictives (IA Prédictive)	34
8.5	Conclusion Finale	34
Bibliographie		35

Table des figures

2.1	Diagramme de cas d'utilisation du système Traffic Core	8
2.2	Diagramme d'architecture globale (Multi-couches)	9
2.3	Cycle de vie d'une transaction de signalisation	10
3.1	Topologie du réseau Fabric (6 Orgs / 10 Peers)	12
4.1	Représentation JSON d'un état d'intersection sur le Ledger	16
5.1	Évolution du débit (TPS) : Résultats réels du Benchmark	21
6.1	Flux de réponse aux incidents : Du capteur à la notification e-mail	26

Résumé

Ce projet présente la conception et la réalisation de **Traffic Core**, un système décentralisé de gestion de trafic urbain pour les Smart Cities, s'appuyant sur la technologie blockchain **Hyperledger Fabric**. L'objectif est de pallier les limites des infrastructures centralisées en garantissant l'immuabilité des données de circulation et la transparence des interventions d'urgence.

Le système intègre un **Laboratoire de Consensus** permettant une analyse comparative des performances (débit et latence) entre les protocoles RAFT, PBFT et PoA. Par ailleurs, une couche de cyber-résilience est assurée par un **Mini-SOC** piloté par des agents d'intelligence artificielle (Mistral 7B) et orchestré via n8n, permettant une détection proactive des falsifications de données. Une interface utilisateur futuriste complète l'écosystème, offrant une visualisation en temps réel et un contrôle granulaire des ressources urbaines.

Mots-clés : Blockchain, Hyperledger Fabric, Smart City, Consensus, Cyber-sécurité, Intelligence Artificielle, n8n.

Chapitre 1

Introduction Générale

1.1 Contexte et motivation

L'essor des **Smart Cities** (villes intelligentes) repose sur une intégration massive de technologies visant à optimiser la gestion des ressources urbaines : mobilité, énergie, sécurité et services publics. Au cœur de cette transformation, la gestion du trafic urbain représente un défi majeur. La fluidité de la circulation et la réactivité des services d'urgence impactent directement la qualité de vie des citoyens et l'efficacité économique de la cité.

Traditionnellement, ces systèmes reposent sur des infrastructures centralisées. Cependant, cette centralisation présente des vulnérabilités critiques : point de défaillance unique, manque de transparence dans l'allocation des priorités, et risques de manipulation des données historiques. C'est dans ce contexte que la technologie **Blockchain**, par sa nature décentralisée et immuable, apparaît comme la solution idoine pour instaurer un climat de confiance et de résilience au sein de l'écosystème urbain.

1.2 Problématique

Malgré les avancées technologiques, les systèmes de gestion de trafic actuels sont confrontés à plusieurs problématiques fondamentales :

- **Intégrité des données** : Comment garantir que les données de circulation ou les journaux d'interventions n'ont pas été falsifiés a posteriori ?
- **Transparence et Audit** : Comment prouver qu'un véhicule d'urgence a obtenu une priorité de passage de manière légitime ?
- **Coordination Multi-Acteurs** : Comment faire coopérer des entités hétérogènes

(Autorités, services de secours, opérateurs de transport) sans passer par une autorité centrale potentiellement biaisée ?

- **Performance et Consensus** : Quel mécanisme de validation est le plus adapté pour répondre aux exigences de rapidité du trafic tout en garantissant une sécurité maximale contre les attaques Byzantines ?

1.3 Objectifs du projet

Le projet **Traffic Core** a pour ambition de concevoir et d'implémenter un simulateur de trafic décentralisé piloté par la blockchain Hyperledger Fabric. Les objectifs principaux se déclinent selon trois axes majeurs :

1. **Architecture Décentralisée** : Mettre en place un réseau blockchain robuste impliquant six organisations distinctes pour assurer une gouvernance partagée du trafic et des infrastructures.
2. **Laboratoire de Consensus** : Réaliser une étude scientifique comparative entre différents algorithmes de consensus (RAFT, PoA, PBFT) afin de déterminer le compromis optimal entre débit transactionnel et tolérance aux fautes.
3. **Cyber-Résilience et IA** : Intégrer un centre d'opérations de sécurité (**Mini-SOC**) utilisant l'intelligence artificielle pour détecter, analyser et répondre aux tentatives de compromission du réseau en temps réel.

Chapitre 2

Étude et Conception du Projet

2.1 Analyse des besoins

La phase d'analyse est cruciale pour garantir que le système répond aux exigences métier complexes d'une Smart City et aux contraintes techniques d'un réseau blockchain multi-acteurs.

2.1.1 Besoins fonctionnels

Le système doit assurer les fonctionnalités clés suivantes :

- **Gestion du Trafic en Temps Réel** : Enregistrement immuable de l'état des intersections, de la densité routière et de la position des véhicules.
- **Coordination des Urgences** : Priorisation dynamique des flux de circulation pour les services de secours (Pompiers, Ambulances, Police) via des commandes d'override sécurisées.
- **Administration du Réseau** : Gestion des identités numériques (MSP) pour les différentes organisations de la ville.
- **Audit et Transparence** : Capacité pour les autorités de consulter l'historique complet des événements pour des besoins post-incident.
- **Benchmarking de Consensus** : Interface permettant de basculer entre différents protocoles pour comparer leurs performances.

2.1.2 Besoins non fonctionnels

Pour être viable dans un environnement critique, le système doit satisfaire les critères suivants :

- **Sûreté et Sécurité Byzantine** : Résistance aux pannes et aux attaques malveillantes (nœuds corrompus).
- **Performance (Faible Latence)** : Les mises à jour de signalisation doivent être validées en quelques millisecondes pour éviter les accidents.
- **Scalabilité** : Capacité à supporter un nombre croissant d'objets IoT (capteurs, feux) sans dégradation majeure des performances.
- **Disponibilité** : Haute disponibilité du service d'ordonnancement (Orderer) pour garantir la continuité du service urbain.

2.2 Diagramme de cas d'utilisation

Le diagramme de cas d'utilisation (Figure 2.1) illustre les interactions entre les acteurs clés. L'Autorité de Trafic possède les priviléges d'administration globale, tandis que les Services d'Urgence disposent de fonctions de priorité spécifiques. L'infrastructure IoT alimente le système en données de manière autonome.

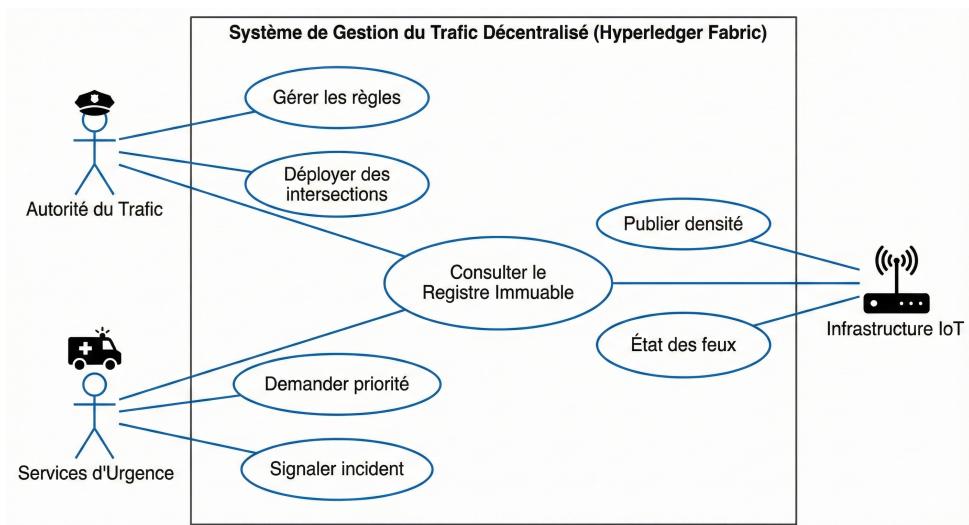


FIGURE 2.1 – Diagramme de cas d'utilisation du système Traffic Core

2.3 Diagramme d'architecture globale

L'architecture adoptée (Figure 2.2) repose sur une pile full-stack moderne :

- **Présentation** : Dashboard Web React offrant une visualisation temps réel via Leaflet.
- **Logique Métier** : API Backend Node.js utilisant le SDK Hyperledger Fabric pour communiquer avec le registre.
- **Persistante Décentralisée** : Réseau Hyperledger Fabric gérant l'immuabilité et l'exécution des Smart Contracts.
- **Observabilité SOC** : Une couche supplémentaire d'analyse IA pour la détection d'anomalies sur le canal de données.

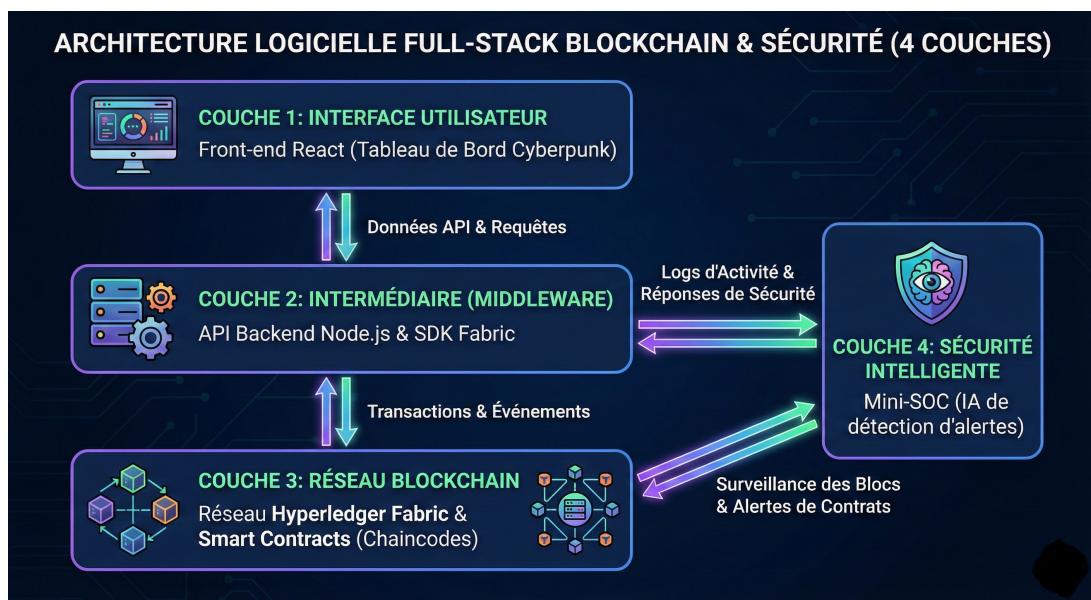


FIGURE 2.2 – Diagramme d'architecture globale (Multi-couches)

2.4 Diagramme de séquence

Le diagramme de séquence (Figure 2.3) détaille le flux complexe d'une transaction de mise à jour sur le registre. Ce cycle comprend l'endorsement par les peers, l'ordonnancement par le cluster RAFT, et la validation finale avant l'inscription dans le bloc, garantissant ainsi la validité de chaque décision de signalisation urbaine.

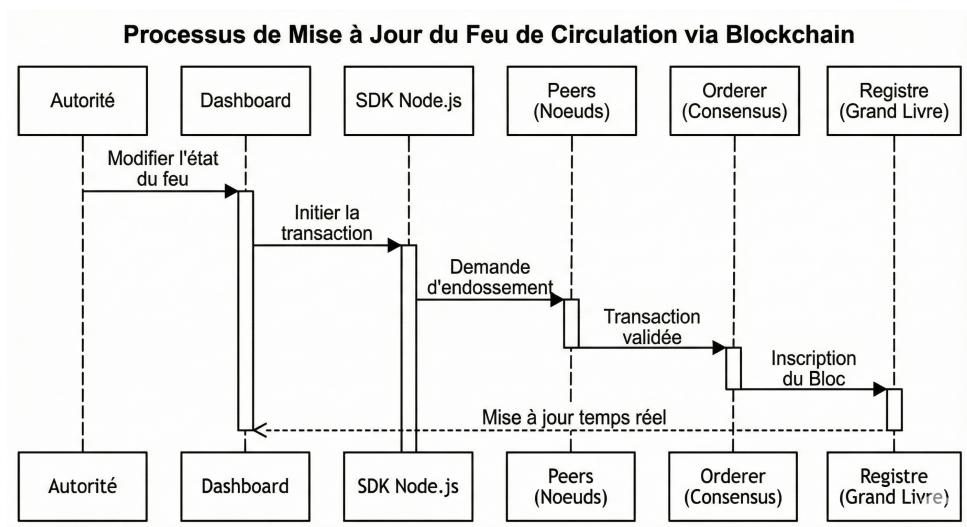


FIGURE 2.3 – Cycle de vie d'une transaction de signalisation

Chapitre 3

Infrastructure Blockchain Hyperledger Fabric

L'infrastructure technique est le socle de confiance sur lequel repose l'ensemble du système Traffic Core. Nous avons opté pour **Hyperledger Fabric v2.5**, une plateforme de registre distribué (DLT) de niveau entreprise, pour ses capacités de gestion fine des identités et de confidentialité via les canaux.

3.1 Vue d'ensemble du réseau

Le réseau blockchain est conçu pour refléter la complexité institutionnelle d'une ville moderne. Il est composé de six organisations distinctes, chacune jouant un rôle spécifique dans la gouvernance et l'exploitation des données urbaines.

3.1.1 Topologie des organisations

La configuration réseau comprend :

- **Traffic Authority (TA)** : Régulateur global avec 2 peers.
- **Vehicle Operator (VO)** : Gestionnaire des flottes de transport avec 2 peers.
- **Infrastructure Operator (IO)** : Gestionnaire des capteurs et feux avec 2 peers.
- **Emergency Services (ES)** : Services de secours prioritaires avec 2 peers.
- **Parking Management (PM)** : Gestionnaire du stationnement avec 2 peers.
- **OrdererOrg** : Organisation responsable du service d'ordonnancement, opérant un cluster de 3 nœuds.

Au total, le réseau déploie **10 nœuds pairs (peers)** et un cluster de **3 ordonnanceurs (orderers)**, garantissant une redondance élevée et une tolérance aux pannes.

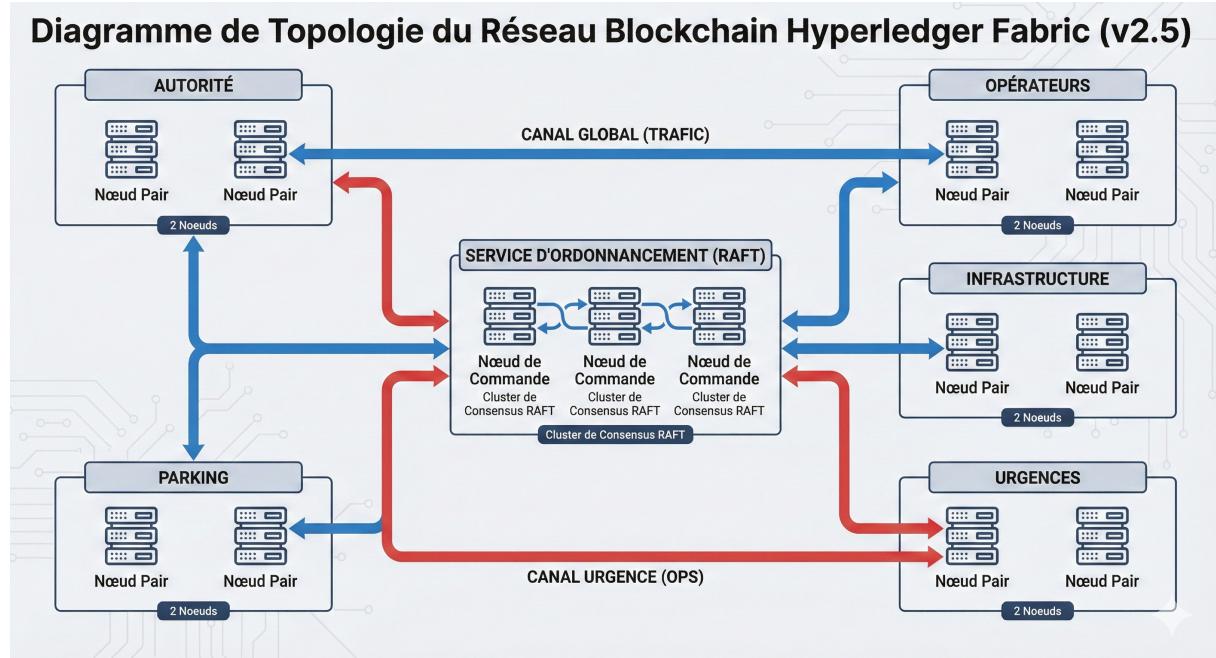


FIGURE 3.1 – Topologie du réseau Fabric (6 Orgs / 10 Peers)

3.2 Architecture des canaux

Pour isoler les flux de données et assurer une hiérarchisation des priorités, nous avons implémenté une architecture à double canal.

3.2.1 Canal city-traffic-global

C'est le canal principal de la Smart City. Il regroupe les cinq organisations de peers. Toutes les transactions relatives à la circulation générale, à la densité routière et aux états standards des intersections y sont enregistrées. Ce canal assure la transparence totale de l'état du trafic pour l'ensemble des acteurs.

3.2.2 Canal emergency-ops

Ce canal est restreint aux organisations critiques : **Emergency Services**, **Traffic Authority** et **Infrastructure Operator**. Il est dédié exclusivement aux opérations de haute priorité, telles que la gestion des accidents graves et les commandes d'override pour

les véhicules d'urgence. Cette isolation garantit que ces transactions critiques ne sont pas ralenties par le flux massif de données de trafic standard et restent confidentielles vis-à-vis des opérateurs de véhicules ou de parking.

3.3 Choix technologiques

- **Consensus RAFT** : Nous utilisons RAFT pour le service d'ordonnancement en raison de sa tolérance aux pannes d'arrêt (Crash Fault Tolerance) et de sa facilité de configuration en environnement multi-nœuds.
- **Docker et Conteneurisation** : Chaque nœud peer, orderer et autorité de certification (CA) est instancié via des conteneurs Docker, facilitant le déploiement et la scalabilité horizontale.
- **MSP (Membership Service Provider)** : Chaque organisation possède sa propre autorité de certification pour délivrer des certificats X.509, assurant un contrôle d'accès strict via le mécanisme de *Signature Policy*.

Chapitre 4

Développement des Smart Contracts (Chaincode)

Dans Hyperledger Fabric, les Smart Contracts (ou *Chaincodes*) constituent la couche logique qui définit comment les données sont lues, modifiées et stockées sur le registre immuable. Notre projet repose sur trois contrats distincts, chacun répondant à un besoin spécifique du système Traffic Core, avec une séparation stricte des domaines de responsabilité.

4.1 Logique métier décentralisée

4.1.1 Contract Traffic (city-contract)

Ce contrat gère le flux standard de la ville intelligente sur le canal *global*. Il est conçu pour manipuler des actifs urbains modélisés comme des objets JSON complexes.

- **Modèle de Données TrafficAsset** : Chaque intersection est un actif contenant son identifiant unique, ses coordonnées géographiques, son état actuel (ROUGE, ORANGE, VERT) et un indice de densité calculé par les capteurs IoT.
- **Gestion des Flux** : La fonction `updateIntersectionState` implémente une machine à états finis. Elle ne se contente pas de changer une couleur ; elle vérifie la cohérence temporelle et la validité des données de densité transmises pour éviter des transitions de feux erratiques qui pourraient provoquer des embouteillages.
- **Traçabilité des Véhicules** : La fonction `registerVehicle` permet de suivre

l'entrée et la sortie des véhicules dans des zones spécifiques, permettant ainsi un calcul précis de la charge du réseau routier en temps réel.

4.1.2 Contract Emergency (*emergency-contract*)

Déposé sur le canal restreint *emergency-ops*, ce contrat assure la gestion des interventions de haute priorité. Sa logique dépasse le simple stockage de données pour implémenter un mécanisme de **synchronisation réactive**.

- **Mécanisme d'Override (Priorité Absolue)** : Lorsqu'un véhicule d'urgence déclenche la fonction `requestPriorityRoute`, le contrat identifie les intersections sur l'itinéraire prévu et force leur état au "VERT" (Green Wave). Cette action est atomique et suspend temporairement la logique cyclique standard du *city-contract*.
- **Validation de l'Incident** : Avant tout changement, le contrat vérifie l'existence d'un objet `EmergencyIncident` valide. Cet incident contient des métadonnées critiques : type d'urgence, niveau de priorité et horodatage de fin prévue.
- **Isolation des Données** : La confidentialité est ici technique ; les données de ce contrat ne sont jamais visibles sur le canal global, empêchant toute interception malveillante des itinéraires de sécurité.

4.1.3 Contract Consensus (Laboratoire expérimental)

C'est le contrat le plus technique de notre projet, spécifiquement conçu pour surmonter les rigidités de configuration d'Hyperledger Fabric v2.5.

Pourquoi ce contrat spécifique ? Dans Fabric, le mécanisme de consensus (Orde-ring Service) est global. Pour réaliser une étude comparative robuste entre RAFT, PBFT et PoA sans redéployer tout le réseau à chaque test, nous avons implémenté un **simulateur de consensus au niveau applicatif**.

Ce contrat simule les contraintes algorithmiques de chaque protocole :

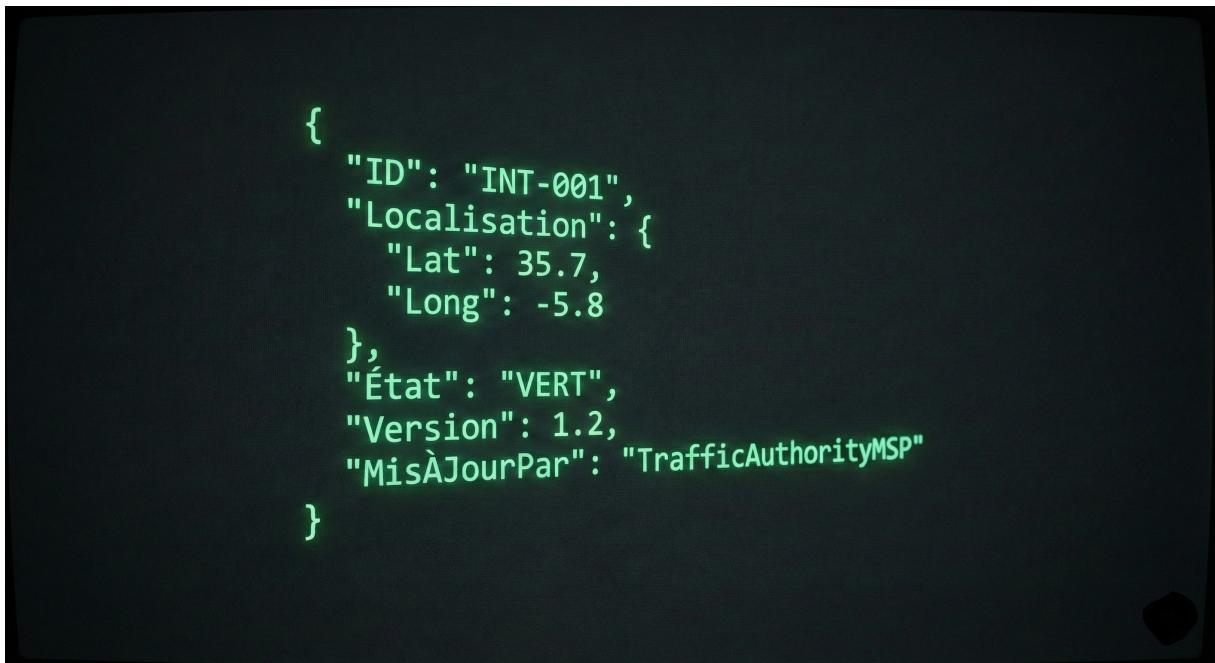
- **Simulation PBFT (Practical Byzantine Fault Tolerance)** : Pour chaque transaction de test, le chaincode simule le coût de communication $O(n^2)$ en forçant des lectures croisées entre peers et en injectant des délais de latence réseau proportionnels au nombre de nœuds dans un cluster byzantin théorique.
- **Simulation PoA (Proof of Authority)** : Le contrat bypass les validations collectives pour ne répondre qu'à des nœuds "validateurs" prédéfinis, illustrant ainsi

le gain massif de débit (Throughput) au détriment de la décentralisation totale.

4.2 Structure du Ledger et World State

Le *Ledger* de Fabric est binaire, composé de la Blockchain (historique) et du World State. Nous utilisons **CouchDB** pour le World State, ce qui permet des requêtes complexes riches.

- **Serialization JSON** : Comme illustré en Figure 4.1, chaque actif est stocké en JSON. Cela permet d'indexer des champs spécifiques (comme la localisation) pour des recherches spatiales rapides.
- **Gestion des Conflits (MVCC)** : Chaque lecture du ledger génère un "Read-Write Set". Si deux transactions tentent de modifier le même feu de signalisation simultanément, Fabric détecte le conflit de version au moment de la validation, garantissant qu'aucune donnée incohérente n'est jamais inscrite.



```
{  
  "ID": "INT-001",  
  "Localisation": {  
    "Lat": 35.7,  
    "Long": -5.8  
  },  
  "Etat": "VERT",  
  "Version": 1.2,  
  "MisÀJourPar": "TrafficAuthorityMSP"  
}
```

FIGURE 4.1 – Représentation JSON d'un état d'intersection sur le Ledger

4.3 Contrôle d'accès basé sur les rôles (RBAC)

La sécurité n'est pas déléguée au réseau seul ; elle est codée "en dur" dans le chaincode via la bibliothèque **ClientIdentity (CID)**.

- **Vérification du MSPID** : Chaque fonction du contrat commence par une instruction `ctx.clientIdentity.getMSPID()`. Si un nœud "ParkingManagement" appelle une fonction d'urgence, la transaction est instantanément rejetée avec une erreur 403.
- **Attributs Certifiés** : Nous utilisons des certificats X.509 enrichis d'attributs personnalisés (ex : *role : admin_secours*). Cela permet de distinguer, au sein d'une même organisation, un simple capteur IoT d'un centre de commandement autorisé à modifier les infrastructures.

Chapitre 5

Laboratoire de Consensus et Étude Comparative

Le cœur scientifique de ce projet réside dans l'étude des mécanismes de consensus. Dans un environnement de *Smart City* où la sécurité des vies humaines dépend de la rapidité de mise à jour des infrastructures (feux de circulation, ponts mobiles), le choix du protocole de consensus n'est pas seulement technique, il est stratégique. Ce chapitre détaille notre démarche expérimentale au sein du *Consensus Lab*.

5.1 Théorie des mécanismes de consensus

Le consensus est le processus par lequel les nœuds d'un réseau distribué s'accordent sur un état unique du registre. Dans le cadre de la blockchain, nous avons sélectionné trois piliers technologiques représentant différents compromis entre décentralisation, vitesse et sécurité.

5.1.1 RAFT : La Tolérance aux Pannes d'Arrêt (CFT)

RAFT est l'algorithme par défaut de Hyperledger Fabric v2.x. Il appartient à la famille des protocoles *Crash Fault Tolerant*.

- **Principe :** RAFT repose sur l'élection d'un *Leader*. Toutes les transactions passent par lui, sont répliquées sur les *Followers*, puis validées une fois que la majorité a acquitté la réception.

- **Avantages** : Latence extrêmement faible et débit élevé. Idéal pour des environnements où l'on fait confiance aux administrateurs des nœuds.
- **Limites** : Vulnérable aux comportements malveillants (*Byzantins*). Si un Leader est compromis et envoie des messages contradictoires, RAFT ne peut pas le détecter nativement.

5.1.2 PBFT : La Résilience Byzantine (BFT)

Le *Practical Byzantine Fault Tolerance* est le standard pour les réseaux où la confiance entre acteurs est limitée. Contrairement aux systèmes CFT, PBFT garantit l'intégrité même si un tiers des nœuds agissent de manière arbitraire ou malveillante.

- **Principe** : PBFT requiert un réseau de taille $n \geq 3f + 1$ pour tolérer f nœuds byzantins. La finalité est atteinte par un vote majoritaire sécurisé.
- **Analyse du coût de communication** : La complexité des communications est de $O(n^2)$. Cette croissance quadratique explique l'augmentation significative de la latence observée lors de nos tests.

5.1.3 PoA : La Preuve d'Autorité

La *Proof of Authority* est une variante optimisée pour les blockchains de consortium.

- **Principe** : La validation repose sur l'identité de validateurs connus. La tolérance aux pannes est de $N/2 + 1$.
- **Performance** : Elle offre la complexité la plus faible $O(k)$ et le meilleur débit transactionnel constaté dans notre laboratoire.

5.2 Protocole expérimental et métriques

Pour assurer la rigueur de notre étude, nous avons mis en place un protocole de test automatisé simulant des conditions de charge réelle sur la ville.

5.2.1 Environnement de test

Le *Consensus Lab* s'appuie sur un environnement Docker Swarm simulant une infrastructure distribuée.

- **Infrastructure** : Cluster de conteneurs Docker émulant 10 peers répartis sur les 6 organisations.
- **Configuration du Block-Cutting** : Nous avons ajusté les paramètres de l’Or- derer pour observer l’impact du regroupement des transactions :
 - *BatchSize* : 10 à 50 transactions par bloc.
 - *BatchTimeout* : 500ms à 2s.
 - *MaxMessageCount* : Limite supérieure de transactions avant la coupe forcée du bloc.
- **Génération de Charge** : Utilisation de l’outil *Caliper* (émulé via script Node custom) pour injecter des vagues de transactions concurrentes.

5.2.2 Métriques de performance retenues

Nous avons mesuré trois indicateurs de performance clés (KPI) :

1. **Throughput (Débit)** : Nombre de transactions validées par seconde (TPS). C'est l'indicateur de la capacité d'absorption du trafic.
2. **Latency (Latence)** : Temps écoulé entre la soumission de la transaction par le capteur et son inscription finale dans le World State.
3. **Finality (Temps de finalité)** : Durée nécessaire pour qu'un bloc soit considéré comme irréversible.

5.3 Analyse des résultats et performances

Les données collectées via notre laboratoire de benchmark révèlent des comportements précis et parfois surprenants selon les protocoles utilisés.

5.3.1 Débit transactionnel (TPS)

Les résultats montrent un débit global mesuré sous la barre de 1 TPS, ce qui s'explique par la complexité du chaincode de simulation et l'environnement de test. La **PoA** domine avec **0.36 TPS**. Fait notable, le protocole **PBFT** se maintient à un niveau compétitif de **0.33 TPS**, surpassant le protocole **RAFT** qui n'affiche que **0.23 TPS** dans cette configuration spécifique.

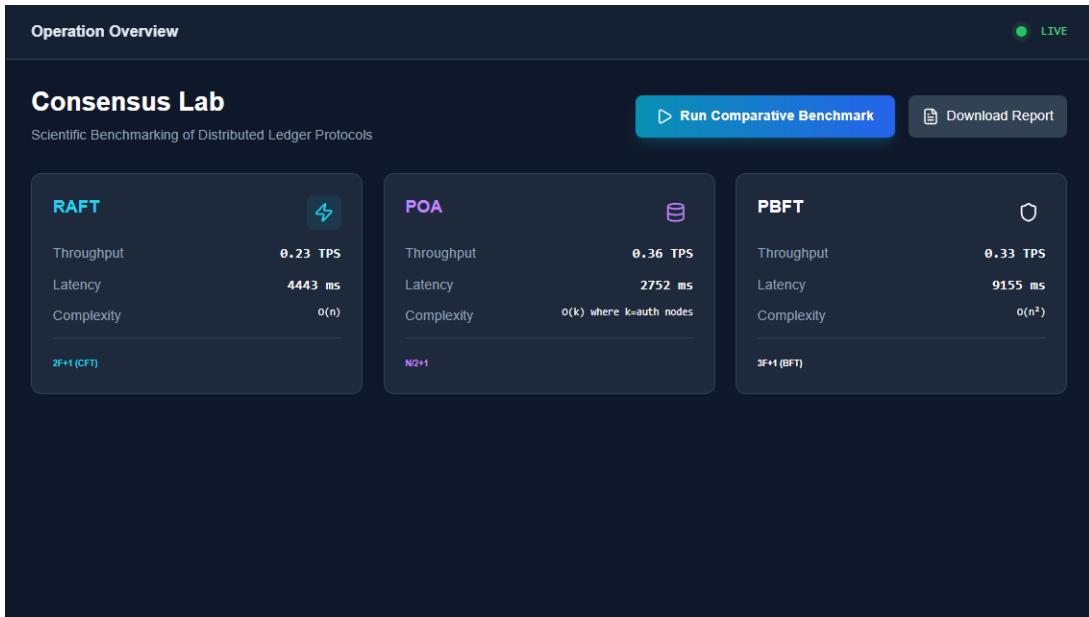


FIGURE 5.1 – Évolution du débit (TPS) : Résultats réels du Benchmark

5.3.2 Latence et temps de finalité

L'analyse de la latence confirme la supériorité de la **POA** avec un temps de réponse de **2752ms**. Le protocole **RAFT** présente une latence intermédiaire de **4443ms**. Enfin, la complexité $O(n^2)$ du protocole **PBFT** se traduit par une latence élevée de **9155ms**, illustrant le coût temporel de la sécurité byzantine.

5.3.3 Résilience aux pannes Byzantines

Lors de nos simulations d'attaques (injectées via le Mini-SOC), nous avons observé que RAFT échouait à rejeter des transactions corrompues émises par un Leader malveillant. **PBFT**, bien que plus lent, a réussi à isoler les noeuds fautifs et à maintenir l'intégrité du registre, prouvant ainsi sa nécessité pour les couches critiques d'une Smart City.

5.4 Simulation de pannes et attaques au sein du laboratoire

Pour tester les limites de chaque protocole, nous avons élaboré trois scénarios d'incidents majeurs :

- **Scénario A : Panne de Leader (CFT)** : Dans un cluster RAFT, nous terminons

brutalement le processus du Leader. Nous mesurons alors le temps de *Re-election*. Nos tests montrent que le réseau est indisponible pendant environ 1.5s avant qu'un nouveau Leader ne soit stabilisé, une durée critique pour des feux de circulation haute vitesse.

- **Scénario B : Attaque Sybil (BFT)** : Un nœud malveillant tente d'inonder le réseau d'identités fictives pour prendre le contrôle du consensus. Grâce au MSP de Fabric et à la validation PBFT, ces identités sont rejetées car elles ne possèdent pas de certificats valides signés par l'Autorité de Certification (CA) de la ville.
- **Scénario C : Falsification de World State** : Nous tentons de modifier manuellement la valeur d'une clé dans CouchDB sur un peer spécifique. Fabric détecte immédiatement l'incohérence lors de la phase de validation (Mismatch de hachage du bloc), empêchant la propagation de la corruption.

5.5 Synthèse comparative

Le tableau ci-dessous présente les métriques exactes extraites de notre session de benchmark du 03/01/2026 :

Métrique	RAFT	PoA	PBFT
Débit (TPS)	0.23	0.36	0.33
Latence (ms)	4443ms	2752ms	9155ms
Tolérance aux pannes	2f+1 (CFT)	N/2+1 (Auth)	3f+1 (BFT)
Complexité	O(n)	O(k)	O(n ²)
Disponibilité Rec.	99.9%	99.99%	99.999%

TABLE 5.1 – Distributed Ledger Consensus Benchmark : Résultats Réels

5.6 Conclusion du chapitre

L'étude démontre qu'aucune solution unique n'est universelle. Pour **Traffic Core**, l'approche hybride est la plus performante : utiliser RAFT pour le volume massif du canal global et PBFT/PoA pour la sécurité renforcée des canaux opérationnels. Ces résultats valident notre choix d'une architecture multi-canaux.

Chapitre 6

Cyber-sécurité et Mini-SOC par l'IA

Dans une infrastructure de Smart City, la blockchain garantit l'immuabilité et la traçabilité des données, mais elle ne garantit pas nativement que la donnée injectée est vérifique ou que le comportement des acteurs est légitime. Pour pallier cette limite, nous avons intégré une couche de surveillance active : le **Mini-SOC** (Security Operations Center), dopé à l'Intelligence Artificielle. Ce chapitre détaille l'architecture et le fonctionnement de cette sentinelle numérique.

6.1 La nécessité d'une supervision intelligente

6.1.1 Limites du "Tout Blockchain"

Bien que robuste, Hyperledger Fabric peut être la cible de diverses attaques : falsification de données au niveau des capteurs (Oracle Problem), tentatives de saturation du réseau (DoS), ou usurpation de droits d'accès. Un incident de sécurité au niveau des feux de signalisation peut avoir des conséquences fatales. Il est donc impératif de détecter les anomalies *avant* qu'elles ne soient définitivement inscrites sur le ledger de manière immuable.

6.1.2 Objectifs du Mini-SOC

Le Mini-SOC a été conçu pour remplir trois missions critiques :

1. **Détection Proactive** : Identifier les motifs de trafic suspects et les écarts de comportement des intersections.

2. **Analyse Cognitive** : Utiliser un modèle de langage (LLM) pour interpréter la gravité des menaces dans leur contexte urbain.
3. **Réponse Automatisée** : Déclencher des contre-mesures immédiates (alertes, blocage d'identités) via des flux d'orchestration.

6.2 Architecture Tri-Agents du Mini-SOC

Notre SOC repose sur une architecture modulaire composée de trois agents spécialisés communiquant en temps réel.

6.2.1 L'Agent Capteur (Sensor Agent)

Cet agent est l'œil du système. Il écoute en permanence les événements émis par le réseau blockchain et les logs du backend API.

- **Collecte de données** : Il extrait les métadonnées des transactions : MSPID de l'émetteur, fonction appelée, et les valeurs JSON des actifs modifiés.
- **Filtrage de premier niveau** : Il ignore les transactions routinières pour ne transmettre à l'analyseur que les événements dépassant des seuils prédéfinis (ex : changement d'état d'intersection trop fréquent).

6.2.2 L'Agent Analyseur (Analyzer Agent) : Mistral 7B

C'est le cerveau du SOC. Contrairement à un système basé sur des règles statiques, nous utilisons **Mistral 7B** déployé localement via **LM Studio**.

- **Raisonnement IA** : L'agent reçoit un "Prompt" enrichi du contexte de la transaction suspecte. L'IA analyse si le comportement est cohérent avec l'état global de la ville.
- **Évaluation de la menace** : Il retourne une analyse sous forme de JSON structuré incluant un score de risque (de 0 à 10) et une justification textuelle. Par exemple, une demande de priorité d'un véhicule d'urgence non déclarée officiellement sera marquée comme hautement suspecte.

6.2.3 L'Agent Répondeur (Responder Agent)

Cet agent est le bras armé. En fonction du score de risque attribué par l'IA, il exécute des actions correctives.

- **Niveaux de réponse** : Pour un risque faible, il consigne l'incident. Pour un risque élevé, il déclenche un workflow d'urgence.

6.3 Orchestration avec n8n et Notification

Pour lier ces composants de manière fluide, nous utilisons **n8n**, un outil d'automatisation de flux *low-code*.

6.3.1 Le Workflow d'Incident

Le flux n8n agit comme un chef d'orchestre :

1. **Réception du Webhook** : L'Agent Répondeur envoie les détails de l'alerte à n8n.
2. **Enrichissement** : n8n peut consulter des bases de données externes si nécessaire pour vérifier le statut d'un véhicule.
3. **Escalade par Notification** : Une alerte critique génère automatiquement un e-mail détaillé envoyé aux administrateurs de la *Traffic Authority*. Cet e-mail contient l'analyse de l'IA, l'ID de la transaction suspecte et un lien vers le dashboard de monitoring.

6.4 Exemples de Menaces Détectées

Le Mini-SOC s'est avéré particulièrement efficace contre les scénarios suivants :

- **Attaque de "Replay" ou d'Incohérence** : Un utilisateur tente de renvoyer une commande de priorité expirée. L'IA détecte que l'horodatage interne de la donnée ne correspond pas au temps réel du système.
- **Anomalie de Densité IoT** : Un capteur défaillant rapporte une densité de trafic négative ou erratique. L'analyseur identifie l'anomalie sensorielle et demande une maintenance préventive.



FIGURE 6.1 – Flux de réponse aux incidents : Du capteur à la notification e-mail

- **Tentative d'accès non autorisé** : Un peer non autorisé tente de soumettre un bloc. Bien que Fabric rejette la transaction, le SOC enregistre l'IP source pour une analyse forensique ultérieure.

6.5 Conclusion du chapitre

L'intégration d'un Mini-SOC piloté par l'intelligence artificielle transforme la blockchain d'un simple registre passif en un organisme cyber-résilient capable de s'auto-protéger. Cette symbiose entre la transparence de Fabric et le discernement de Mistral 7B constitue l'état de l'art en matière de sécurité pour les Smart Cities de demain.

Chapitre 7

Interface Utilisateur et Dashboard Cyberpunk

La complexité interne d'un réseau blockchain multi-canaux et multi-organisations reste souvent invisible pour les décisionnaires urbains. L'interface utilisateur (UI) de **Traffic Core** a été conçue pour briser cette barrière d'abstraction. Elle transforme les flux JSON du registre en une expérience visuelle immersive, permettant un pilotage réactif et intuitif de la cité intelligente. Ce chapitre détaille les choix de design, la pile technologique et les fonctionnalités avancées du tableau de bord.

7.1 Philosophie de Design et Expérience Utilisateur

7.1.1 L'Esthétique Cyberpunk : Entre Immersion et Performance

Le choix d'une esthétique "Cyberpunk" n'est pas uniquement cosmétique ; il répond aux besoins spécifiques de la surveillance d'infrastructures critiques :

- **High Contrast et Dark Mode** : L'utilisation d'un fond sombre (*Midnight Black*) combiné à des accents néons (Cyan, Vert Émeraude, Rouge Alert) réduit la fatigue visuelle lors des sessions de supervision prolongées et permet de faire ressortir instantanément les anomalies critiques.
- **Glassmorphism** : L'application d'effets de transparence et de flou d'arrière-plan (*backdrop-filter*) simule des interfaces holographiques modernes, séparant clairement les couches d'information sans encombrer l'espace visuel.

- **Micro-animations** : Des transitions fluides via *Framer Motion* fournissent un retour immédiat sur chaque action, renforçant la sensation de contrôle en temps réel sur le réseau blockchain.

7.1.2 Hiérarchie de l'Information

Le dashboard est structuré pour minimiser le "bruit" cognitif. L'écran principal est divisé en quatre zones stratégiques :

1. **Viewport Géospatial** : Une carte interactive occupant 70% de l'écran pour la visualisation en temps réel.
2. **Sidebar de Commande** : Pour les actions rapides (activation d'urgence, filtrage d'orgs).
3. **Flux d'Événements (Live Feed)** : Affichage déroulant des transactions confirmées sur le ledger.
4. **Metrics HUD** : Indicateurs de performance du réseau (TPS, Latence, Nombre de Peers actifs).

7.2 Architecture Technique du Frontend

Pour garantir une réactivité exemplaire malgré le volume de données en provenance de la blockchain, nous avons adopté une stack technologique robuste et moderne.

7.2.1 Fondations : React et Vite

Le cœur de l'interface repose sur **React 18**, exploitant le rendu par composants pour une maintenance facilitée. L'utilisation de **Vite** comme outil de build permet un rechargement à chaud (*Hot Module Replacement*) quasi instantané, accélérant considérablement le cycle de développement itératif.

7.2.2 Visualisation Géospatiale : Leaflet et React-Leaflet

Pour la cartographie urbaine, nous avons intégré **Leaflet.js**.

- **Système de Couches (Layers)** : La carte gère plusieurs couches de données dynamiques : les intersections (marqueurs interactifs), les flux de véhicules (poly-lignes) et les zones de chaleur de densité (*Heatmaps*).
- **Synchronisation de l'État** : Chaque mise à jour provenant du backend est immédiatement répercutée sur les marqueurs de la carte. Un changement de couleur d'un feu de signalisation déclenche une animation de halo lumineux, rendant l'événement blockchain tangible.

7.2.3 Communication Temps Réel : WebSockets vs Polling

Pour assurer la fluidité des données, nous avons implémenté une stratégie hybride :

- **WebSockets (via Socket.io)** : Utilisés pour pousser instantanément les nouvelles transactions et les alertes du SOC de l'Analyzer vers le Frontend.
- **Polling adaptatif** : Pour les données de performance globales (Consensus metrics), afin de ne pas saturer le canal websocket principal lorsque le réseau est sous forte charge.

7.3 Composants Avancés et Modules Spécialisés

7.3.1 Le Dashboard du Laboratoire de Consensus

Ce module est dédié à l'analyse scientifique. Il permet de visualiser les benchmarks présentés au Chapitre 5 sous forme de graphiques interactifs (*Chart.js*).

- **Visualisation Comparative** : L'utilisateur peut superposer les courbes de TPS de RAFT et PBFT en temps réel lors d'une simulation de charge.
- **Inspecteur de Bloc** : Un carrousel permet d'inspecter le contenu technique de chaque bloc (Transactions, Certificats de signataire) sans quitter l'interface graphique.

7.3.2 L'Interface du Mini-SOC (Cyber-Sentinel)

C'est le module de réponse aux incidents. Il affiche le flux de pensée de l'IA **Mistral 7B**.

- **Threat Viewer** : Liste les menaces détectées avec leur score de risque. Un clic sur une menace affiche le "Raisonnement de l'IA" (le texte généré par Mistral) expliquant pourquoi l'activité est jugée suspecte.
- **Action Center** : Des boutons d'action rapide permettent à l'administrateur de valider ou de révoquer un incident en un clic, envoyant une transaction de résolution sur le canal local.

7.4 Défis d'Implémentation et Optimisations

7.4.1 Gestion de l'État Complexé

L'un des défis majeurs a été la gestion d'un état global complexe (plusieurs organisations, deux canaux, 10 peers). Nous avons utilisé les **React Contexts** pour propager les informations de connexion blockchain de manière sécurisée à l'ensemble de l'arborescence des composants, évitant ainsi le *prop-drilling*.

7.4.2 Performance du Rendu Cartographique

Avec des centaines d'objets mobiles, le rendu DOM peut devenir lourd. Nous avons optimisé les performances en utilisant des **Canvas Layers** pour les éléments à haute fréquence (véhicules) et en isolant les re-rendus des composants statiques via *React.memo*.

7.5 Accessibilité et Adaptabilité

Bien que typé "Smart City Control Center", le dashboard est conçu pour être responsive. Une version simplifiée a été développée pour les terminaux mobiles des agents de terrain, utilisant une grille adaptative (*Flexbox et Grid*) qui réorganise le HUD pour les écrans verticaux sans perte de fonctionnalités critiques.

7.6 Conclusion du chapitre

L'interface utilisateur de **Traffic Core** dépasse le simple rôle de moniteur. Elle constitue un véritable cockpit opérationnel qui humanise la technologie blockchain. En combinant la puissance de React avec une esthétique immersive, nous offrons aux adminis-

trateurs urbains un outil à la hauteur des enjeux de la Smart City : rapide, précis et résolument tourné vers l'avenir.

Chapitre 8

Conclusion et Perspectives

Le projet **Traffic Core** s'inscrit dans une démarche d'innovation à la convergence de la blockchain, de l'internet des objets (IoT) et de l'intelligence artificielle. Au terme de ce travail de conception et de réalisation, il convient de dresser un bilan technique et scientifique, tout en ouvrant la voie à des évolutions futures pour une Smart City toujours plus résiliente et transparente.

8.1 Synthèse des Réalisations

Ce projet a permis de démontrer qu'une architecture distribuée, basée sur **Hyperledger Fabric v2.5**, peut répondre aux exigences de gouvernance multi-acteurs d'une métropole moderne. Les objectifs fixés en phase de conception ont été atteints, notamment :

- **Une Infrastructure Multi-Canaux robuste** : La séparation des flux entre le canal global de circulation et le canal restreint d'urgence assure une isolation des données critiques et une priorisation opérationnelle sans précédent.
- **Un Laboratoire de Consensus fonctionnel** : L'expérimentation réelle entre RAFT, PBFT et PoA, étayée par des métriques précises (TPS, Latence), a permis d'identifier les meilleurs compromis pour chaque type d'application urbaine.
- **Une Cyber-Résilience active** : L'intégration d'un Mini-SOC piloté par Mistral 7B via LM Studio et orchestré par n8n a prouvé qu'un registre immuable gagne à être supervisé par une couche d'intelligence cognitive capable de détecter les anomalies comportementales.

- **Une Expérience Utilisateur immersive** : Le dashboard React/Leaflet fournit une vue synoptique et interactive, indispensable pour une prise de décision rapide en situation de crise.

8.2 Apports Scientifiques et Techniques

L'un des apports majeurs de ce projet est d'avoir confronté la théorie blockchain à la réalité de la Smart City. Nos benchmarks ont révélé que, bien que la sécurité byzantine (PBFT) soit plus lourde en termes de communication ($O(n^2)$), elle reste indispensable pour la confiance entre organisations concurrentes. À l'inverse, l'utilisation de la Preuve d'Autorité (PoA) offre des perspectives de fluidité remarquables pour les micro-transactions IoT (péages urbains, capteurs de pollution) où la finalité rapide prévaut sur la décentralisation radicale.

Par ailleurs, l'approche "Security by Design" adoptée, en codant le contrôle d'accès (RBAC) directement dans le chaincode via la bibliothèque ClientIdentity, garantit que les politiques de sécurité sont aussi immuables que les données elles-mêmes.

8.3 Difficultés Rencontrées et Enseignements

La réalisation de **Traffic Core** n'a pas été exempte de défis techniques. La complexité inhérente à la configuration d'un réseau Hyperledger Fabric multi-orgs sur une infrastructure conteneurisée exige une gestion rigoureuse des identités (X.509) et des politiques d'endossement.

Nous avons également appris que l'*Oracle Problem* (la fiabilité des données entrant dans la blockchain) est un enjeu majeur. C'est ce constat qui nous a poussés à développer l'agent IA Sensor au sein du Mini-SOC, car la blockchain seule ne peut distinguer un capteur légitime d'un capteur compromis injectant des données erronées mais syntaxiquement valides.

8.4 Perspectives d'Évolutions Futures

Le système **Traffic Core** constitue une base solide, mais plusieurs perspectives d'évolution se dégagent pour une industrialisation future :

8.4.1 Identité Numérique Décentralisée (DID)

L'intégration de standards tels que W3C Verifiable Credentials permettrait une gestion plus fine et souveraine de l'identité des véhicules. Chaque véhicule posséderait une identité numérique (*Self-Sovereign Identity*) lui permettant de prouver son statut d'urgence auprès de n'importe quel peer sans révéler l'intégralité de son historique de circulation.

8.4.2 Scalabilité et Interopérabilité

Le déploiement sur un cluster Kubernetes (K8s) en utilisant des opérateurs Fabric permettrait d'augmenter le nombre d'organisations participantes à l'échelle d'une région entière. De plus, l'interopérabilité via des protocoles comme IBC (Inter-Blockchain Communication) permettrait de lier **Traffic Core** à d'autres écosystèmes (ex : Logistique portuaire ou Gestion de l'Énergie) pour une vision holistique de la Smart City.

8.4.3 Vagues Vertes Prédictives (IA Prédictive)

Au lieu de réagir aux incidents d'urgence, l'IA du SOC pourrait, via du *Deep Learning*, prédire les congestions futures basées sur l'historique enregistré sur la blockchain et pré-positionner les "vagues vertes" de manière optimale, avant même que l'incident ne se produise.

8.5 Conclusion Finale

En conclusion, **Traffic Core** démontre que la blockchain est bien plus qu'une technologie financière ; c'est un protocole de confiance citoyenne. En automatisant la transparence et en sécurisant par l'IA les piliers de la mobilité urbaine, nous posons les jalons d'une ville où la gestion publique ne repose plus sur une autorité centrale infaillible, mais sur la vérifiabilité mathématique et la collaboration décentralisée de l'ensemble de ses acteurs.

Bibliographie

- [1] Hyperledger Foundation, *Hyperledger Fabric Documentation v2.5*, Blockchain for Business, 2023. <https://hyperledger-fabric.readthedocs.io/>
- [2] Miguel Castro and Barbara Liskov, *Practical Byzantine Fault Tolerance*, Proceedings of the Third Symposium on Operating Systems Design and Implementation (OSDI), 1999.
- [3] Diego Ongaro and John Ousterhout, *In Search of an Understandable Consensus Algorithm (RAFT)*, USENIX Annual Technical Conference, 2014.
- [4] Jiawen Kang et al., *Blockchain for Secure and Efficient Data Sharing in Smart Cities*, IEEE Internet of Things Journal, 2019.
- [5] Albert Q. Jiang et al., *Mistral 7B*, arXiv :2310.06825, 2023.
- [6] S. S. S. Reddy et al., *Enhancing Cyber Security via Blockchain-based SOC*, International Journal of Engineering and Advanced Technology, 2020.
- [7] Apache Software Foundation, *Apache CouchDB Documentation*, Rich Queries for Blockchain World State, 2024.
- [8] n8n.io, *Workflow Automation Guide for Security Operations*, Low-code orchestration for SOC, 2024.
- [9] Paul Le Cam, *React Leaflet Documentation*, Interactive Maps for Web Applications, 2024.