An Experience Report on the Design and Implementation of an Ad-hoc Blockchain Platform for Tactical Edge Applications

Nguyen Khoi Tran[1], M. Ali Babar[1],
Julian Thorpe[2], Seth Leslie[2], and Andrew Walters[2]

[1] The University of Adelaide, Adelaide SA 5005, Australia
{nguyen.tran, ali.babar}@adelaide.edu.au
[2] Defence Science and Technology Group, Adelaide, Australia

Abstract. The success of task groups operating in emergency response or military missions largely depends on their ability to access, process, and exchange information, which is enabled by tactical edge software applications. Designing such applications poses challenges for architects due to the need for resilience, trust, and decentralisation of information processing and exchange. Nevertheless, these challenges also present opportunities for exploring new architectural alternatives like blockchain-based decentralised systems. This paper reports the experience from an R&D partnership to architect and implement a decentralised ad-hoc platform for deploying and operating tactical edge software applications, which facilitate information processing and exchange among tactical edge task groups. The platform utilises a peer-to-peer architecture, aided by ad-hoc blockchain networks, to enable secure and decentralised information storage, update, and distribution. By leveraging a kernel-based, service-oriented architecture, the platform can be adapted to different mission contexts and integrated with various task groups' systems and data sources. Additionally, we present a process and heuristics for architecting tactical edge software applications to operate on the platform. We further share our experience in deploying and operating the platform in a government-led field experiment and conclude the report with lessons learned from this case study.

Keywords: blockchain · smart contract · platform · information sharing · common operating picture · emergency response.

1 Introduction

The tactical edge environment of a military or emergency response mission contains various task groups from different organisations. The mission's success depends on the task groups' ability to access, process, and exchange tactical information in a timely and secure manner, despite network bandwidth, processor power, data storage, and electrical power constraints [5, 17, 18]. Secure and resilient information exchange within and across task groups can enable collaborative information processes, such as maintaining a shared library of dynamic

reference information (DRI) about environments and objects of interest within a mission to improve situation understanding and facilitate superior decision processes [4]. Within an increasingly digitalised tactical edge environment, collaborative information processes are increasingly aided by software applications, which we denote in this paper as tactical edge software applications.

Tactical edge software applications require an underlying software platform that can support information processing and exchange within and across task groups while ensuring the information's integrity, confidentiality, traceability, and provenance. This paper presents the findings and experiences of a long-standing academia-government partnership focused on architecting and developing a decentralised software platform for information processing and exchange at the tactical edge. The platform functions as an ad-hoc information platform established across computing nodes of task groups prior to or during a mission. It allows for the secure distribution, update, and storage of shared information artefacts without relying on a remote intermediary. One of the platform's key features is its ability to allow task groups to deploy an extensible application suite for facilitating collaborative information processes such as DRI.

The platform is built on ad-hoc blockchain networks [16] and peer-to-peer (P2P) content distribution networks like the Interplanetary File System (IPFS) [1]. We utilise a kernel-based, service-oriented architecture to encapsulate the platform's complexity and enable task groups to adapt the platform to different mission contexts and resource availability by changing the application suite. We have also developed a design process and heuristics for architecting software applications that operate on the platform.

This paper is structured as follows. Section 2 presents the current centralized architecture of a tactical edge environment and proposes an alternative decentralized information platform. Section 3 describes the architecture of the platform that implements the decentralized information platform. Section 4 details a process for creating tactical edge software applications that function on the platform. Sections 5 and 6 describe the experiences and lessons gained from implementing and operating the platform during a government-led field experiment. Section 7 provides an overview of related work. Finally, Section 8 summarizes and concludes the paper.

2 Context and Architectural Vision

2.1 Current Architecture: Centralised Information Platform

Figure 1 depicts the current architecture of a tactical edge environment consisting of two task groups working together. The computing devices used by these groups are divided into three tiers, named and classified based on MITRE's Tactical Edge Characterisation Framework [5]. The higher a device is in the hierarchy, the more stable its network connectivity, power supply, and computing resources. Devices in the *Dismounted systems* category include wearable devices, smartphones, mobile workstations, and remote-controlled or autonomous vehicles used by first responders during missions. *Mobile centres* are workstations or

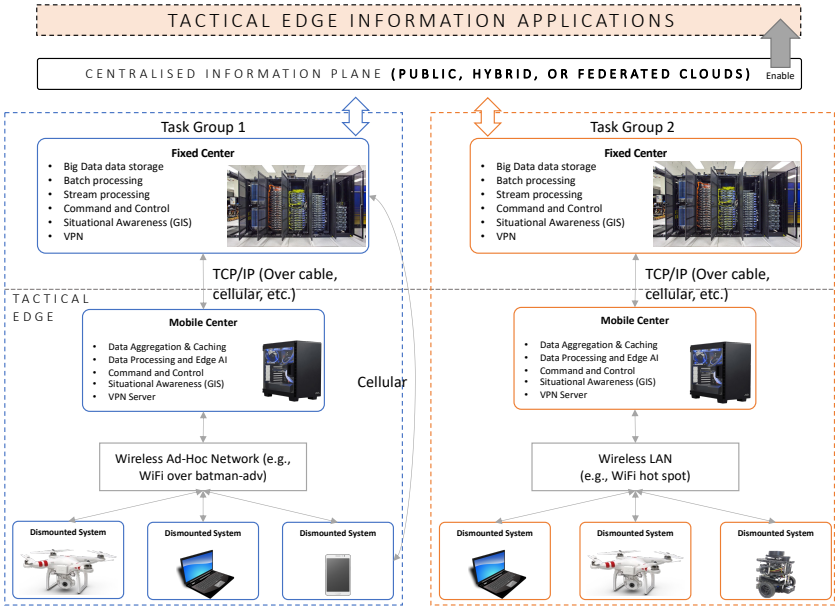

Fig. 1. Architecture of an exemplary tactical edge environment and infrastructure

servers used by field-deployed command and control (C2) to provide situational awareness tools (like Geographic Information System) and C2 facilities to receive updates from first responders, monitor missions, and issue instructions. These centres can also serve as cloudlets [8] for offloading data and AI/ML operations from dismounted systems. *Fixed centres* operate from a remote location away from the tactical edge. They provide commanders with situational awareness and C2 capability over an entire task group. Fixed centres can also serve as a private cloud, providing big data storage and processing capability.

Enabling information exchange and processing among task groups requires connecting their infrastructures. The current cloud-centric solutions leverage intermediary public clouds, hybrid clouds [9], or a federated design to establish a *centralised information platform* to store, distribute, and secure shared information. The exchange of and processing upon shared information can be encapsulated by *tactical edge information applications* operating within the information platform.

Example: Dynamic Reference Information Management DRI is an evolving information library that describes objects of interest, their attributes, and their connection to the observable world, such as the tactical edge environment where a mission occurs [4]. DRI can be maintained by diverse teams to improve their situational understanding and decision-making abilities. Military and emergency service platforms commonly utilize DRI to configure their systems and

sensors for a mission. Due to its critical role, ensuring the authenticity, integrity, provenance, and availability of this shared information artefact is essential [18]. The storage and update of DRI serve as an excellent example of a tactical edge information application.

2.2 Vision: Decentralised Secure Information Platform

Having presented the current architecture, the remainder of this paper focuses on our effort to architect and engineer an alternative alternative to address the following issues of the current architecture that arose in practice:

Dependence on network uplink. C2 and first responders might not have access to up-to-date information on a remote cloud due to intermittent connectivity. Moreover, the exchanged information must cross multiple trust boundaries, thus increasing its exposure to cyber threats.

Single point of failure. The centralised information platform represents an intermediary within the information flow within and across task groups. Even though a defence-in-depth strategy can protect the information platform, there is still a risk that programming errors, insider threats, or cyber attacks can disrupt the information platform and, thus, deny C2 and responders from accessing and sharing vital information.

Ownership, control, and accountability. A major issue we faced when architecting tactical edge information exchange and processing services was the ownership and control over the services and the intermediary cloud infrastructure. The task groups might not be willing or able to host the shared data and processes. They might also not trust others or an intermediary enough to hand over the data and processes.

We developed the idea of a *Decentralized Secure Information Platform* after researching the utility of blockchain technology in Internet of Things (IoT) systems [14] and successfully establishing and leveraging an *ad-hoc blockchain network* to collect sensor data during a field experiment [13]. The architecture of this envisioned information platform is shown in Figure 2. The platform can be deployed in the field and operated directly by tactical edge task groups. Upon the platform, they can deploy *decentralised* tactical edge software applications to facilitate collaborative information processes such as DRI management. The backbone of the platform is a private ad-hoc blockchain network established amongst task groups to provide a decentralised root of trust and synchronisation. Due to the use of an ad-hoc blockchain, we refer to the envisioned decentralised information platform as the *"Ad-hoc Blockchain Platform"* or simply *"platform."*

3 Design and Implementation of the Platform

This section presents the platform's architecture and implementation details and elaborates on how the platform achieves the vision described in Section 2.2.

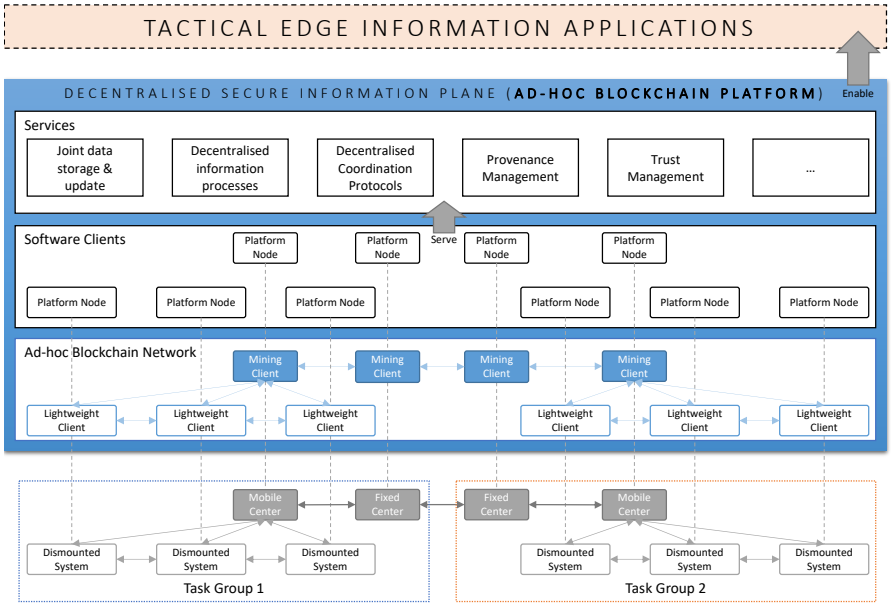

Fig. 2. Conceptual view of the Adhoc Blockchain Platform for Tactical Edge

3.1 Requirements and Constraints

The platform was architected towards the following requirements, which we derived from the vision described in Section 2.2.

Decentralisation. The platform should be operable directly by the tactical edge task groups to avoid the issues of ownership, control, accountability, and single point of failure.

Proximity to the tactical edge. The platform should operate at or close to the tactical edge to improve availability and reduce latency.

Information trustworthiness. We consider information trustworthy if a receiver can ascertain the identity of the sender (authenticity), verify that it has not been modified (integrity), and be sure that the sender cannot falsely deny sending the message (non-repudiation) [10].

Provenance of processing results. Use cases such as DRI management require updating shared artefacts based on the incoming information. The provenance showing how and by whom the updates were made is vital for both in-situ trust assessment and postmortem analysis.

Potentially large data volume. The platform must accommodate data payloads captured and exchanged at the tactical edge, such as video and audio recordings.

Interoperability with applications and data sources. The platform must support diverse tactical edge information applications. Doing so requires interoperability with various software systems and data sources of task groups.

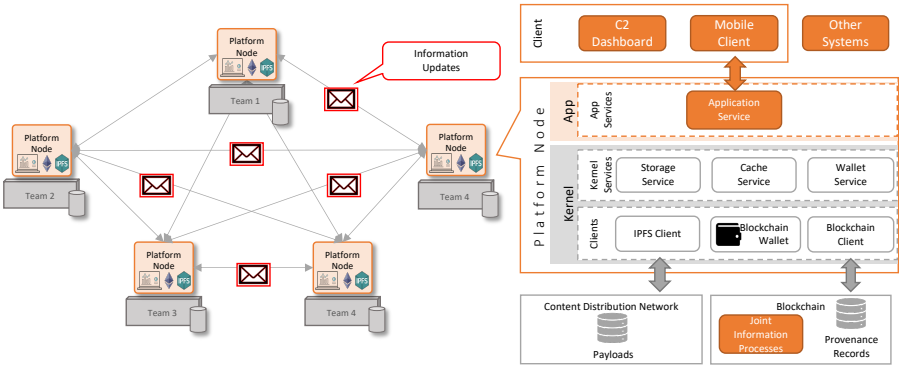

Fig. 3. Platform's architecture comprising multiple peer nodes

3.2 Platform Architecture

We designed the platform's architecture by leveraging well known architectural styles including peer-to-peer (P2P), service-oriented (SOA), and Microkernel. As a P2P system, the platform is made up of multiple identical software clients called **Platform Nodes** (Figure 3), which are deployed across the computing devices of task group participants. Each node represents a participant within the platform and retrieves or updates information on their behalf. Participants can interact with the information through decentralised tactical edge applications running locally on their devices. Nodes communicate via pre-established data links like tactical data links, ad-hoc wireless networks, and cellular networks. To ensure the security of the information exchanged, we have adopted a blockchain protocol like Ethereum [19]. Additional capabilities such as data storage and distribution can be enabled by integrating other protocols like IPFS [1]. By adopting a P2P architecture and deploying nodes within the tactical edge, we addressed the first two requirements. We addressed the described security requirements regarding information exchanges by leveraging the built-in security mechanisms of a blockchain protocol.

To facilitate the adaptation of platform nodes to mission contexts and participant devices, we leveraged the microkernel architecture to design platform nodes. The software stack of a node is separated into *kernel space* and *application space*. Both layers comprise software services that abstract the complexity behind predefined service interfaces to simplify the development and integration of applications and data sources. The application space contains *application-specific services*, which are key components of tactical edge information applications running on the platform. This layer's content can be modified based on the application suite of a mission and the information needs and authorisation levels of a participant. We will elaborate on developing and deploying a tactical edge application on the platform later in Section 4.

The kernel space contains *clients* and *kernel services* that are vital for platform's operation and, thus, unchanged across missions and participants. The *clients* layer contains software clients necessary for the P2P protocols utilised by the platform (e.g., blockchain client and wallet for blockchain protocol, IPFS client for IPFS). The *kernel services* layer provides a uniform interface for application services to interact with the clients. The *wallet* service encapsulates a blockchain client and a wallet, providing the ability to query a blockchain and submit transactions. The *storage* service encapsulates data payload stores (e.g., IPFS client or local storage) and implements the secure offloading process for large data payloads (Section 3.2). The `Cache` service provides the ability to cache on-blockchain records and data payloads. We introduced the cache service to the platform because we found that the read latency from a blockchain, specifically from an Ethereum-based blockchain via the Go-Ethereum, was high and impeded the user experience from the graphical user interface at the application layer.

Secure Data Offloading To save valuable blockchain storage space for critical records, we applied the Off-chain Data Storage pattern [21] to architect a secure mechanism to store data payloads outside the blockchain. In particular, the platform can store incoming audio and video recordings on a node's local storage or a decentralised content network established among platform nodes (e.g., using IPFS). The platform then includes the cryptographic hashes of these data payloads in blockchain transactions to prove their existence. These hashes can be used to identify, locate, and verify the integrity of the data payloads. The storage service can transparently handle this offloading process.

3.3 Platform Implementation

Our platform's concept demonstrator uses the Go-Ethereum (Geth) client implementation to support the Ethereum protocol. To handle the creation of Ethereum transactions and interaction with smart contracts, we used the `Ethers.js` for our Wallet service. We implemented the services as NodeJS modules and packaged them as Docker containers. Containerisation helps improve portability across diverse systems that task groups utilize. Additionally, we utilized Docker Compose to provide a programmatic and automated way to deploy the entire stack.

3.4 Platform Bootstrapping Process

The platform can be field-deployed by *deployers*, IT operators from participating task groups. Deployers bootstrap the platform on behalf of their task groups and exert no control over the platform after deployment. The process is as follows:

1. *Configure the architecture of the ad-hoc blockchain network.* Platform deployers can tune the network's performance with design decisions such as protocol choice and client deployment. Our preliminary collections of common blockchain network designs provide some templates to aid the design process [16].

2. *Deploy and configure blockchain clients on all computing nodes that participate as platform nodes.* Deployers can leverage an blockchain deployment automation framework such as NVAL [15] to automatically deploy and configure blockchain clients according to the architecture from the previous step.
3. *Deploy and configure the kernel service layer on all platform nodes.*
4. *Deploy tactical edge applications.* This step consists of deploying the smart contract backend of applications and starting application services on platform nodes. Section 4.3 presents more details about the application deployment process.

4 Develop and Deploy Applications with the Platform

Tactical edge information applications like DRI management can be developed separately from the platform and deployed upon the platform during the bootstrapping process. This section presents the architecture, development, and deployment processes of tactical edge information applications.

4.1 Application architecture

An application is made up of two parts: on-blockchain and on-platform components. The on-blockchain components are *smart contracts* that store and process information like a DRI library. The on-platform components include *software clients* that users can interact with and an *application service* that connects software clients with smart contracts. These components are installed on the platform node of authorized users.

4.2 Application development process

After analyzing the project logs, we developed the following process to assist developers in designing and developing applications for the platform.

1. *Determine the suitability of the application.* To determine if an application is suitable for the platform, it is important to identify its information needs. Only shared information artefacts and their updates will benefit from being hosted on the platform.
2. *Identify the data payload and provenance records.* The provenance records benefit from being managed by a blockchain, whilst the payloads represent candidates for offloading to off-chain data stores.
3. *Model the collaborative information processes.* Because blockchains operate like deterministic state machines, it is helpful to model a collaborative information process as a set of actions targeting the shared information artefacts, transitioning them between various states in their life cycle. The artefacts can be represented in a blockchain by smart contracts. Artefacts' stages can be stored within the contracts' variables, and actions can be implemented as smart contract functions. Whenever a user sends a blockchain transaction to a smart contract function to act on the shared information artefact, the transaction becomes part of the provenance record.

4. *Develop an application service.* This service provides an Application Programming Interface (API) that connects external software clients to smart contracts. To make it easier for integration with external software clients, we recommend designing the API at a higher level of abstraction to allow users to retrieve information instead of blockchain transactions.
5. *Develop user-facing software clients (optional).*

4.3 Application deployment process

Before the platform bootstrap (Section 3.4), deployers must acquire and integrate software components of all utilised applications into the deployment package of platform nodes. Each node can possess a different application suite according to its users' information needs and authorisation. Once the ad-hoc blockchain network has been established during platform bootstrapping, deployers can deploy the smart contracts of all applications on the blockchain. They can then use the smart contract identifiers to finish configuring application services. Finally, deployers start the application services and corresponding software clients on platform nodes, completing both application deployment and platform bootstrapping.

5 Case Study

We developed and deployed a proof-of-concept demonstration of our platform, along with an application called Decentralised Dynamic Reference Information (DDRI), as part of a government-led field experiment. Due to the space constraint of this paper, our report in this section will focus on the design and technological decisions of the proof-of-concept platform and the application utilised in the experiment. Detailed quantitative analysis of the platform would be presented in a forthcoming paper.

5.1 Context

The case study was part of the Real-time Information Superiority Experimentation (RISE) initiative [17]. RISE was created to test and explore information management, exploitation, and exchange technologies like the platform and DDRI in a field experiment setting. The experiment took place at Port Elliot with Surf Life Saving South Australia and involved three emergency service scenarios. These scenarios included a search and rescue mission for a drifting inflatable, a search and rescue mission for an emergency beacon attached to a boat in distress, and a repeated version of the first scenario with limited communication capabilities. The platform and DDRI application were utilised to maintain a DRI library based on information from three viewpoints, simulating three task groups. Figure 4 presents the map of the experiment area and locations of platform nodes (PESLSC, Commodore's Point, and Lady Bay).

Fig. 4. Node locations at Port Elliot Surf Life Saving South Australia (PESLSC), Commodore's Point, and Lady Bay

5.2 Implementation

Background We designed the software components of DDRI based on the following DRI formalisation by Consoli and Walters [4]. An reference information library is a set I_{AP}, consisting of four subsets: known objects O, objects' attributes or features X_A, known sensors S, and environment's parameters X_{env}, such as geographical location.

$$I_{AP} = \{O, X_A, S, X_{env}\}$$

The library is updated due to an observation process Θ. This process generates an observation about a stimulus event S_e at time t. The observation is made by a sensor S and describes the attributes $X_A(S)$ of an object of interest $objInt_i$. The observation happens within an environment described by parameters S_{env}.

$$\Theta(S_e, t) = \{objInt_i, X_A(S), S, S_{env}\}$$

.

An observation becomes a reference information update (RI update) if it happens within the mission environment ($S_{env} \in X_{env}$) and either the object of interest is unknown ($objInt_i \notin O$) or attributes of a known object ($X_A(S)$) has changed, the information generated by the observation represents an update to the library. Therefore, a dynamic reference information library can be considered an I_{AP} that can be modified by RI updates during a mission.

Functional Scope DDRI's design began by identifying the elements of DRI management that the platform should host. We decomposed DRI management into four steps: observation, processing, storage, and propagation. While observation and processing can be done manually or with AI-assisted tools, storage and propagation require information exchange among members. Thus, DDRI focused on the storage and propagation steps.

Smart Contract Design Since all observations would come from task groups in the same mission, we assumed they happen in the same observable world. In other words, $S_{env} \in X_{env}$ for all observations, so we omitted S_{env} and X_{env} from our data model without losing information. Therefore, a DRI library comprises the set of available sensors S, objects of interest $objInt_i$, and attributes of those objects $X_A(S)$. The set of available sensors, objects, and their attributes describe the *state* of DRI at a time. The DRI transition between states is based on incoming RI updates, reporting observations about changes in attributes of an object of interest. The assessment and appending of RI updates into a DRI form the actions making up the collaborative information process among task groups. The attributes within RI updates represent data payloads. The details regarding the sensor, the observed objects, and attributes' representations (e.g., cryptographic hashes) within RI updates triggering the update of DRI represent the provenance of the shared information artefact.

We designed the smart contract backend of the DDRI application based on the above analysis. The smart contract classes and objects (`struct`) that make up the on-blockchain components of DDRI are presented in the class diagram shown in Figure 5. We assumed each mission has one DRI and task groups can simultaneously participate in multiple missions. DDRI uses one instance of the `Mission` contract to represent a mission and manage its DRI. The `DDRI` contract maintains a list of registered missions and provides the `createMission` function for deployers to register a new mission. For each mission, the DDRI contract creates a new instance of the `Mission` contract. This contract includes a list of `RIUpdate` objects to capture a DRI's development. Since the latest state of a DRI could be derived from the RI updates, we left the presentation of DRI to off-blockchain software clients and knowledge representation tools. The `addRIUpdate` function handles the processing and appending of RI updates. Future implementations of DDRI could extend the `addRIUpdate` function with collaborative verification and validation protocols.

Services and Clients The next step was designing the application service that exposes DDRI smart contracts to external software clients. This service internally calls the storage service to offload the RI update payload and then calls the wallet service to construct and submit the relevant transaction. Additionally, it calls the cache service to check and update the cache before finally returning the sequence result to the upper layer for responding to users. We also developed two types of software clients tailored to different users. The Mobile client is a web-based application accessible via mobile phones for forward-deployed agents

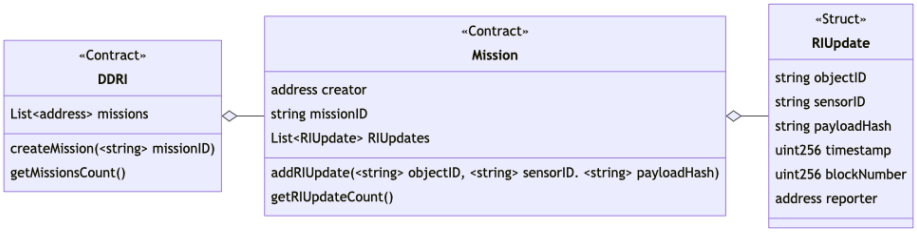

Fig. 5. Smart contracts for DRI management

to report and record RI updates from the field rapidly. The C2 Dashboard is a web-based dashboard used by team leaders to access and view the status and history of every object of interest in the mission (Figure 6). The dashboard presents both the RI update list and the latest state of DRI derived from the RI updates.

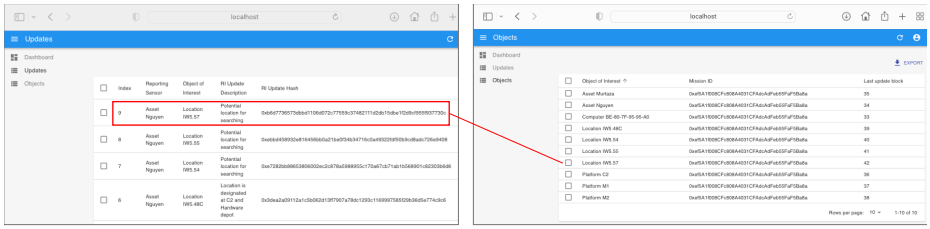

Fig. 6. C2 Dashboard showing an RI update list and a corresponding list of objects of interest

5.3 Deployment and Operation

The prototype platform utilised in the experiment comprised three on-site nodes, placed at vantage points in Figure 4, and a remote located 85 kilometres from the experiment site to simulate a remote fixed centre. Initially, we had planned to deploy the nodes directly on designated computers and connect them via cellular network links using a virtual private network. However, we had to switch to a fall-back configuration due to weather conditions and technical challenges. As a result, all four nodes were deployed on separate cloud-based virtual machines connected via a VPN. The designated computers could control their corresponding platform nodes through a 4G uplink.

During the experiment, the authors took on the responsibility of operating platform nodes to record RI updates. The updates were provided by mission participants who acted as observers and information processors. These updates

included verbal instructions and visual observations of the environment and rescue. For instance, an event such as 'Jet Ski 1 pulling the lost inflatable back to the beach to treat the patient' would involve describing the inflatable, patient details, rescue craft and crew status, and the time associated with each action in the situation. Before each experiment, participants were assigned roles as observers and processors, which were also documented in the RI updates.

The entire experiment lasted for 5 hours and 45 minutes. In summary, 361 RI updates were recorded and exchanged through the platform throughout the experiment, with an average time interval of 44 seconds between updates. The ad-hoc blockchain that supported the platform retained a steady rate of 15 seconds between blocks, which indicated that the blockchain could keep up with the information. At the end of the mission, the distributed ledger maintained by the platform only grew by 10MB, which suggested that the platform had adequate scalability.

6 Lessons Learned

Blockchain network architecture design as a means to achieve software requirements. The quality attributes of blockchain-based software applications depend on the characteristics of their underlying blockchains. The characteristics cannot be modified for public blockchains, so they must be treated as constraints. For private blockchains, architects may be able to swap blockchain implementations or develop new consensus protocols. However, caution should be taken due to these protocols' complexity and adversarial context [2].

We discovered that tweaking the *architecture of a blockchain network* can offer a promising alternative to the approaches mentioned above. For instance, one can reduce the amount of resources consumed by a blockchain network on dismounted systems by disabling the mining feature of their blockchain nodes and tweaking the protocol configurations by switching from Proof-of-Work (PoW) to Proof-of-Authority (PoA). Architects can also embed an organisation's structure into a blockchain network's architecture by controlling the assignment of blockchain nodes and roles (full nodes, archive nodes, miners) to participants. By taking control of such design decisions, architects can optimise the blockchain network's characteristics to ease the constraints at the application design level without developing new blockchain protocols or replacing blockchain implementations altogether. Furthermore, blockchain network architectures provide new research opportunities, such as identifying a design space [13] and patterns (e.g., [16]) that can help practitioners design blockchain networks tailored to their applications.

Autonomy versus Control. Decentralisation implies the autonomy of participants. Such autonomy is realised by bringing the abilities to access, process, and exchange information to the tactical edge and replicating them across participants. While this approach can help strengthen the resilience of information access and processing, it can also create challenges regarding coordinating and

controlling a mission. Additionally, participants must take on the responsibility of bootstrapping and managing their software infrastructure to be a part of the platform.

A solution we adopted in the project is introducing a bootstrapping phase to coordinate the platform launch before the mission. During this phase, platform deployers (who are technicians from task groups) can use automation tools like NVAL [15] to deploy and configure platform nodes on task groups' computers according to the mission and their information needs. After deployment, the platform remains decentralised and autonomous. We are also exploring other decentralised monitoring and governance mechanisms so that privileged participants (such as task group commanders) can monitor and take action on the platform, such as disabling platform nodes when compromise is detected.

7 Related Work

Blockchain technology has been increasingly leveraged in military and emergency response contexts to prevent cyber attacks, manage complex supply chains, and strengthen the resilience of communication links both within and outside tactical edge environments [3,12]. For instance, Siemon et al. [11] proposed a framework to leverage blockchains as a resilient communication infrastructure among C2 in an emergency response mission. Fend et al. [6] utilised blockchains as a secure mechanism for updating the retransmission policies between military networks (Tactical Data Links or TDL). Rather than focusing on individual use cases, our proposed ad-hoc blockchain platform provides a foundation for developing and operating multiple use-case-specific applications. For instance, task groups can use the platform to secure TDL retransmission policies by deploying a tactical edge information application that manages the policy updates and exposes service endpoints for TDL gateways to retrieve trusted and up-to-date policies.

The platform was designed and developed based on the existing software architecture knowledge about blockchain-based software. Inspired by Xu et al. [20], we leveraged a blockchain as a decentralised software connector for bridging distributed platform nodes. The off-chain data storage pattern [21] and caching tactic [22] were utilised to implement the platform's secure data offloading and caching capabilities. We also leveraged the smart contract registry and factory patterns [21] to design the exemplary application, DDRI. The model-driven approach by Jurgelaitis et al. [7] inspired our proposed heuristics for identifying and modelling the shared information processes for developing smart contracts of tactical edge information applications.

8 Conclusions

In this experience report, we discussed a software platform that enables secure information exchange and processing among task groups operating in a tactical edge environment. We provided insights into the platform's analysis, architecture, and implementation. Additionally, we discussed the design process and

supporting heuristics for creating tactical edge information applications that can operate on the platform. The platform's applicability was also explored, highlighting its usefulness for further academic-government partnership on research and development of innovative concepts like DRI.

Our platform was designed with decentralisation, adaptability, and integration in mind. Using a peer-to-peer architecture and an ad-hoc blockchain as a software connector, we created a platform that enables task group members to exchange and process information without relying on remote intermediaries. To achieve adaptability, we separated information processing logic from information exchange, storage, and processing services. The Microkernel architecture we chose reinforced this separation. We also ensured ease of integration by adopting a service-oriented architecture for interactions within and outside the platform. Our platform has been evaluated through a field experiment. During platform's development and deployment, we discovered that blockchain network architecture can be leveraged to achieve non-functional requirements at the application level. We also faced the challenge of balancing autonomy and control when operating a decentralised platform. In the future, we plan to investigate decentralised monitoring and governance mechanisms to address this trade-off. We will also explore systematic approaches to decommissioning and archiving the ad-hoc blockchain after a mission.

References

1. Benet, J.: Ipfs-content addressed, versioned, p2p file system. arXiv preprint arXiv:1407.3561 (2014)
2. Cachin, C., Vukolić, M.: Blockchain consensus protocols in the wild. arXiv preprint arXiv:1707.01873 (2017)
3. Clementz, G.W.A.Q.R.V.G.: Blockchain in defence: A breakthrough? Tech. rep. (2020), https://finabel.org/blockchain-in-defence-a-breakthrough/
4. Consoli, A., Walters, A.: Dynamic reference information: Formalising contextual actionable information for contested environments. In: 2020 14th International Conference on Innovations in Information Technology (IIT). pp. 154–159. IEEE (2020)
5. Dandashi, F., Higginson, J., Hughes, J., Narvaez, W., Sabbouh, M., Semy, S., Yost, B.: Tactical edge characterization framework, volume 1: Common vocabulary for tactical environments. Tech. rep. (2007)
6. Feng, W., Li, Y., Yang, X., Yan, Z., Chen, L.: Blockchain-based data transmission control for tactical data link. Digital Communications and Networks **7**(3), 285–294 (aug 2021). https://doi.org/10.1016/j.dcan.2020.05.007
7. Jurgelaitis, M., Butkienė, R., Vaičiukynas, E., Drungilas, V., Čeponienė, L.: Modelling principles for blockchain-based implementation of business or scientific processes. In: CEUR workshop proceedings: IVUS 2019 international conference on information technologies: proceedings of the international conference on information technologies, Kaunas, Lithuania, April 25, 2019. vol. 2470, pp. 43–47. CEUR-WS (2019)
8. Lewis, G., Echeverria, S., Simanta, S., Bradshaw, B., Root, J.: Tactical cloudlets: Moving cloud computing to the edge. In: 2014 IEEE Military Communications Conference. IEEE (oct 2014). https://doi.org/10.1109/milcom.2014.238

9. Mansouri, Y., Prokhorenko, V., Babar, M.A.: An automated implementation of hybrid cloud for performance evaluation of distributed databases. Journal of Network and Computer Applications **167**, 102740 (oct 2020). https://doi.org/10.1016/j.jnca.2020.102740
10. Schneier, B.: Applied Cryptography Protocols, Algorithms and Source Code in C. Wiley and Sons, Incorporated, John (2015)
11. Siemon, C., Rueckel, D., Krumay, B.: Blockchain technology for emergency response. In: Proceedings of the 53rd Hawaii International Conference on System Sciences (2020)
12. Tosh, D.K., Shetty, S., Foytik, P., Njilla, L., Kamhoua, C.A.: Blockchain-empowered secure internet-of-battlefield things (iobt) architecture. In: MILCOM 2018-2018 IEEE Military Communications Conference (MILCOM). pp. 593–598. IEEE (2018)
13. Tran, N.K., Babar, M.A.: Anatomy, concept, and design space of blockchain networks. In: 2020 IEEE International Conference on Software Architecture (ICSA). IEEE (mar 2020). https://doi.org/10.1109/icsa47634.2020.00020
14. Tran, N.K., Babar, M.A., Boan, J.: Integrating blockchain and internet of things systems: A systematic review on objectives and designs. Journal of Network and Computer Applications **173**, 102844 (jan 2021). https://doi.org/10.1016/j.jnca.2020.102844
15. Tran, N.K., Babar, M.A., Walters, A.: A framework for automating deployment and evaluation of blockchain networks. Journal of Network and Computer Applications **206**, 103460 (oct 2022). https://doi.org/10.1016/j.jnca.2022.103460
16. Tran, N.K., Babar, M.A.: Taxonomy of edge blockchain network designs. In: European Conference on Software Architecture. pp. 172–180. Springer (2021)
17. Walters, A., Anderson, R., Boan, J., Consoli, A., Coombs, M., Iannos, A., Knight, D., Pink, M., Priest, J., Priest, T., Smith, A., Tobin, B., Williamson, S., Laurenson, B.: Conducting information superiority research in an unclassified surrogate defence environment. Tech. rep. (Sep 2018)
18. Walters, A., Bou, S., Consoli, A., Priest, T., Davidson, R., Priest, J., Williamson, S.: Building trusted reference information at the tactical edge. Tech. rep. (2019)
19. Wood, G., et al.: Ethereum: A secure decentralised generalised transaction ledger. Ethereum project yellow paper **151**(2014), 1–32 (2014)
20. Xu, X., Pautasso, C., Zhu, L., Gramoli, V., Ponomarev, A., Tran, A.B., Chen, S.: The blockchain as a software connector. In: 2016 13th Working IEEE/IFIP Conference on Software Architecture (WICSA). pp. 182–191. IEEE (2016)
21. Xu, X., Pautasso, C., Zhu, L., Lu, Q., Weber, I.: A pattern collection for blockchain-based applications. In: Proceedings of the 23rd European Conference on Pattern Languages of Programs. pp. 1–20. ACM (jul 2018). https://doi.org/10.1145/3282308.3282312
22. Yánez, W., Bahsoon, R., Zhang, Y., Kazman, R.: Architecting internet of things systems with blockchain: A catalog of tactics. ACM Transactions on Software Engineering and Methodology (TOSEM) **30**(3), 1–46 (2021)