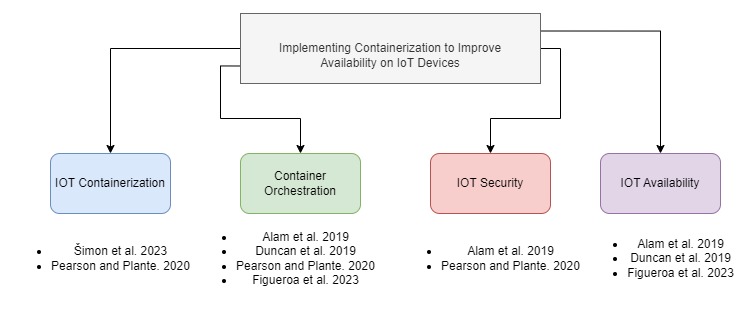
Implementing Containerization to Improve Availability on IoT Devices

Deploying containers on Internet of Things (IoT) devices introduces a highly effective approach towards improving the devices’ availability and security. IoT has revolutionised data collection, processing, and utilisation in several industries, including smart cities, healthcare, and agriculture. Deployment and management of such applications presents a considerable number of difficulties due to the unique nature of IoT ecosystems and specific limitations of IoT devices, such as computational resources and energy consumption.

**** Traditional strategies frequently fail to meet these challenges, leading to issues with service continuity and availability. Containerisation improves application mobility and can also be implemented in clusters for added redundancy and load-balancing, which makes it a highly viable solution for this scenario. This study builds upon the foundation of containerization’s effect on IoT devices by conducting a thorough review of the body of existing literature and technological frameworks, as cited below. It also emphasises four respective themes being containerized IoT, Container orchestration, IoT Availability, and IoT Security, which have been covered below respectively.

When making use of a containerized environment to deploy IoT devices such as Docker, system administrators are able to run simultaneous images on the same device and push them to a central repository for deployment. Updates pushed to these devices can be more efficiently by uploading to the specified repository where all images implied autonomously deploy changes accordingly. Data collection farmed from these devices can be accessed from a locally run database. As mentioned by Pearson and Plante [1] while conducting a research intended to verify whether containers are compliant with modern security principles , such containers are comprised of various layers, which are selectively updated so as to decrease overhead when deploying updates. Unlike traditional virtualization, containers are exclusively designed to operate on Linux-based hosts due to the lightweight nature of the platform. Whilst there are various options to choose from when deploying containerized clusters such as Kubernetes, Docker and Proxmox, it is clear that the Docker platform greatly exceeds its competition in a study conducted by M. Šimon et al [2], having the lowest number of failures whilst responding quicker to queries and downtime recovery. Due to the significance of these results, it is imperative to use Docker Swarm as the main high availability solution for the best possible results to be generated.

High Availability is achieved through the use of Container Orchestration Platforms, which seamlessly integrates these containers into a complete system that handles deployment, scalability, availability, and security in a highly efficient manner. Docker’s version of this orchestration suite called Swarm assigns worker or master roles to containers which either host the containers or manage configurations and deployment in the cluster respectively, as described in [3] and [4]. To maintain the system's seamless operation, many Kubernetes worker nodes are established. The reason for this is that in the event that one node fails, Kubernetes has the ability to move pods to other open nodes. A significant factor which aligns with the focus of this research is self-healing functionality as mentioned in [4], [5], which allows any node which is part of the cluster to be moved onto a different container and continue to as usual. Furthermore, this can be done with no downtime when a cloud infrastructure is implemented as discovered by Alam et al [3], which conducted a full service recovery migration from layer to layer in and during the failure occurrence.

Despite the benefits, integration of IoT devices with containerized environments creates complex security vulnerabilities related to Linux privilege escalation attacks or Man in the Middle attacks between devices. Various countermeasures can be taken to ensure robust security on the cluster as discovered in [1] and [3]. Communications between devices must be conducted used SSH and also TLS that makes use of X.509 certificates which will reject any non trusted communication and connections to the server that does not pass mutual authentication checks. The containers will also mitigate the affected area of any due the isolation capabilities of running multiple independent processes simultaneously. Linux platforms are also able to make use of encryption and hashing to packets in transit, covering all three objectives of security (CIA).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Study | Devices | Host Operating System | Cluster | Aim |
| [1] | 3 RPi | Linux | Docker | NA |
| [2] | NA | Linux | All | Benchmarking |
| [3] | NA | Linux |  | NA |
| [4] | 2 RPi | Linux | Kubernetes | Smart Home |
| [5] | 5 RPi | Linux | Kubernetes | Smart City |

A major research need exists in the integration of IoT devices with containerised applications in low-latency, unstable networks, and power-constrained contexts. The deployment of IoT and container technologies in environments with variable network stability and constrained energy resources presents issues that have not been fully explored, despite significant progress in these domains. Because containerised applications are dynamic and transient, they require consistent, dependable network connections for efficient management and orchestration of containers. This can be challenging to meet in settings with inconsistent network services. Furthermore, real-time data processing and decision-making applications require low-latency communication between dispersed IoT devices and containerised apps; however, current solutions fall short of these requirements in the setting of unstable networks.

Furthermore, energy efficiency is crucial since Internet of Things devices frequently operate in power-constrained environments. Long-term deployment and sustainability may be hampered by power consumption caused by the overhead brought on by container orchestration and the operational requirements of maintaining containerised applications. The ways in which energy-efficient containerised applications might be enhanced without sacrificing dependability or performance have not been extensively investigated in current research. This disparity suggests that novel strategies are desperately needed to solve the trifecta of low latency, unstable networks, and energy economy in containerized Internet of Things scenarios.

[1] B. Pearson and D. Plante, “Secure Deployment of Containerized IoT Systems,” in *2020 SoutheastCon*, Mar. 2020, pp. 1–8. doi: 10.1109/SoutheastCon44009.2020.9368276.

[2] M. Šimon, L. Huraj, and N. Búčik, “A Comparative Analysis of High Availability for Linux Container Infrastructures,” *Future Internet*, vol. 15, no. 8, Art. no. 8, Aug. 2023, doi: 10.3390/fi15080253.

[3] M. Alam, J. Rufino, J. Ferreira, S. H. Ahmed, N. Shah, and Y. Chen, “Orchestration of Microservices for IoT Using Docker and Edge Computing,” *IEEE Commun. Mag.*, vol. 56, no. 9, pp. 118–123, Sep. 2019, doi: 10.1109/MCOM.2018.1701233.

[4] C. Figueroa, T. Knowles, V. Kukreja, and C.-H. Lung, “IoT Management with Container Orchestration,” in *2023 IEEE 3rd International Conference on Electronic Communications, Internet of Things and Big Data (ICEIB)*, Apr. 2023, pp. 49–54. doi: 10.1109/ICEIB57887.2023.10170261.

[5] B. Duncan, Y. Lee, M. Westerlund, and A. Aßmuth, *CLOUD COMPUTING 2019 Proceedings of the Tenth International Conference on Cloud Computing, GRIDs, and Virtualization*. 2019.