www.irbas.academyirmbr.com
International Review of Basic and Applied Sciences
October 2022
Vol. 10 Issue.4

Processing IoT Application in the Edge Computing and Overall Effect on Infrastructure Cost

MUHAMMAD MUDASSAR
(Corresponding Author)
Department of Computer Science, COMSATS University Islamabad Vehari Campus, Pakistan.
Email: muhammad.mudassar@cuivehari.edu.pk

MUHAMMAD ALTAF
Department of Computer Science, COMSATS University Islamabad Vehari Campus, Pakistan.

SAMI ULLAH
Department of Management Science, COMSATS University Islamabad Vehari Campus, Pakistan.

Abstract

Internet of Things (IoT) along with smart processing devices used to provide edge computing environment supports several daily life applications including smart healthcare, city traffic management, automated cars, and smart power grids for efficient supply of electricity. These applications usually process the data using cloud computing resources that increases application latency. Application execution on the Edge computing environment allows to perform application processing inside the edge network hence reducing the latency and increasing the utilization of local and available computing machines. This also helps to avoid heavy infrastructure cost or renting out the cloud resources. The problem of processing big size data and computational oriented applications can be resolved by combining the resources of devices inside the edge network executing resource intensive task. The traditional clustering methods are centralized have overhead of creating and maintaining the cluster. For this paper a decentralized grouping of edge nodes is provided that groups devices on arrival using an organizer node who have some high end job, and handle the execution in parallel on group devices by distributing the tasks to them. The proposed method reduces traffic amount communicated to cloud and allow application processing in the edge network resulting in reduction of latency as well as the cost in terms of the infrastructure. The decentralized group formation proposed here along with task partitioning works in the edge computing as the results are provided for this. Detailed results ensure reducing overall latency, limiting bandwidth and use of local resources hence this results in saving the economic cost.

Keywords: *Computing, IOT, Processing, Effect, Application.*

Introduction

Internet of Things (IoT) also termed as cyber systems (Miorandi et al., 2012) becoming common in the form of applications like smart health tracking and healthcare, city traffic management using IoT, automated cars driving. These IoT application produce vast amount of data needed to be quickly processed to fulfill the application requirements. The traditional big data processing technologies tend to use resources available at cloud but for smart applications this is not suitable for as it may not provide quick communication hence effecting the real-time decision making. Edge computing, Shi and Dustdar (2016) addresses these shortcomings well by allowing the computations to be performed at edge of the network on

the smart and available devices, as the devices are closer to the user hence reducing the network traffic as well. The figure 1 shows the .typical model of IoT based processing with importance of edge computing.

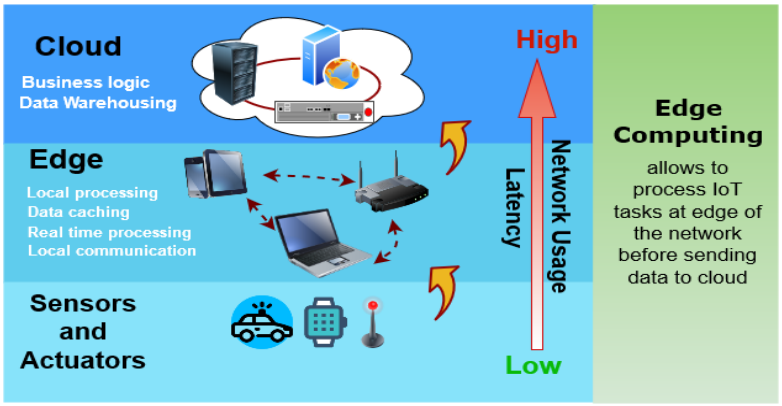

Fig. 1: IoT and edge computing

The cloud offers resources and usually unlimited and resides at fix location in comparison to the devices used in the edge computing termed edge nodes have low resources as well as different in terms of resource types and maker, additionally these are mobile or have mobility property. Running IoT applications directly on nodes in the edge network can low the quality of service (QoS). Additionally the smart health monitoring, smart city in terms of traffic management, latest agriculture using IoT devices and smart grids for power management are inherently distributed in nature (Shi & Dustdar 2016). To address these challenges, an efficient strategy is required to group or cluster edge nodes, taking into account their resource availability.

Current clustering methods often rely on centralized approaches, which are unsuitable for heterogeneous edge nodes. These methods impose constant overhead costs, as cluster members must frequently communicate with the cluster-head. Furthermore, over-reliance on one cluster head can lead to resource depletion, causing the cluster head to fail and rendering the entire cluster inoperative. To overcome these limitations, a decentralized method for grouping edge nodes is essential. This decentralized approach should support IoT task execution within the edge network while minimizing latency and optimizing network resource usage.

Smart devices in edge computing, also referred to as edge nodes, play versatile roles. They can store and cache data, process it locally, offload computational tasks when required, and redirect requests to the cloud. This enables real-time processing and allows time-sensitive applications to make decisions locally within the edge network, eliminating delays. Additionally, edge nodes can filter incoming data to extract initial insights before sending it to the cloud, thereby reducing backhaul link stress and enhancing bandwidth efficiency.

A relatable analogy can be drawn from the way Muslim communities gather for prayers. Five times a day, people assemble at local mosques when the prayer time begins, following a public announcement called the Azan. Upon hearing the Azan, individuals move toward their nearest mosque, where they participate in a group prayer led by an Imam who leads the Salat. After prayer completion attendees disperse to resume their daily activities (Ali et al., 2018). The concept is presented in the figure 2. This natural congregation inspired the development of a decentralized strategy for grouping heterogeneous edge nodes to collaboratively execute large IoT tasks.

In this research, we propose an effective approach involving an organizer node to initiate the grouping process. Each edge node independently decides whether to join a group based on its available resources,

without requiring knowledge of the complete network topology. Once grouped, nodes collaborate to execute a task and return to their routine operations once completion of the task. The organizer node manages task allocation and negotiates with group members to ensure all task requirements are met. Simulation results of our proposed algorithm for dynamic edge node grouping demonstrate its effectiveness in meeting latency requirements and minimizing network resource usage, achieving the desired objectives for IoT applications in decentralized edge environments.

Figure 2: People gathering to offer Salat (Paryer). Azan and gathering is provided in b while once people are gathered they offer salat as in fig a, after salat people disperse provided in fig c.

Related Work

With the rapid adoption of 5G technology and Internet of Things (IoT) a 75.44 billion of devices will be connected till 2025 (Latre et al., 2016). The high growth can generate vast amounts of data to be processed. Currently, most IoT solutions rely on remote cloud infrastructure for data processing (Wang & Ranjan, 2015; Truong & Dustdar, 2015). While this approach supports large-scale computation, it also incurs high costs in terms of bandwidth and increases latency. To mitigate these issues, edge computing has emerged as a promising solution, receiving considerable attention in the literature (Gu et al., 2015;Wang et al., 2016;Chen et al., 2015). Advances in processing capabilities in smart devices like smartphones, smart TVs, Raspberry Pi, and smart gateways have enabled the provision of computational services closer to the network edge (Su et al., 2018; Satyanarayanan et al., 2015;Zhang et al., 2018).

Edge nodes exhibit significant heterogeneity with respect to available resource and geographic distribution, ranging from servers and routers to smart phones and Raspberry Pi (Yi et al., 2015). Additionally, IoT traffic is heavily influenced by mobility and spatial factors such as population density and user activity. These factors impact the load distribution and capabilities of edge devices, posing scalability challenges for Smart Cyber-Physical Systems (SCPS) applications (Qazi et al., 2016). To address these issues, edge devices and resources must operate in a distributed manner, independent of external cloud support (Guo et al., 2018). This distributed model can leverage the computational potential of edge nodes, enabling efficient handling of IoT challenges.

To execute resource-intensive applications that surpass the capacity of individual edge nodes, it is advantageous to form groups of edge devices. Prior research on edge computing concepts, such as cloudlets (Satyanarayanan et al., 2009), Femtoclouds (Habak et al., 2015), and Fog Computing (Yi et al., 2015), advocates resource pooling among mobile devices. For instance, Cloudlets allow devices to offload processing tasks to a central edge resource, but they do not distribute tasks across multiple edge devices (Flores et al., 2017). Similarly, Femtoclouds use centralized controllers for cluster management (Kumar &

Zaveri 2018), and Fog Computing deploys dedicated servers at the edge, incurring higher costs and requiring specialized management to adapt to fluctuating user demands (Gedawy et al., 2018). While these systems have notable advantages, their centralized nature limits scalability and flexibility.

In the context of Mobile Edge Computing (MEC), research has explored clustering techniques to optimize communication and processing. For example, a graph-based clustering approach in (Mudassar et al., 2019) partitions geographical areas to localize communication within clusters, reducing cloud dependency. Guo et al. (2018) proposed a transparent computing architecture to enhance scalability and manageability for IoT applications. Similarly, clustering methods in Bouet & Conan, (2018) explore bottom-up and top-down approaches tailored to specific IoT application needs. However, these studies rely on centralized resource management models for aspects like operating systems, applications, and data.

Although efforts to migrate and distribute computing closer to end devices have shown some promise, most rely on static and centralized organization schemes, such as master-slave or manager-worker models. As the number of heterogeneous devices grows, it becomes essential to adopt decentralized techniques for grouping resources and allocating tasks based on the individual capabilities of edge devices. While numerous research projects have addressed edge node clustering and resource utilization, a universally applicable model for grouping edge nodes across diverse IoT applications remains lacking. Furthermore, effective resource coordination within heterogeneous edge node groups has not been comprehensively addressed.

Methodology

This section describes the heterogeneous node grouping so that the resource intensive tasks can be executed on the nodes in the edge network. At start the task handling is discussed.

a) Task Handling

To execute an application having task ω on the edge nodes in the edge network a task have $\omega = \{i_k, \mu_k, d_k, t_k\}$ where i_k is the edge node processing the request and this is the node where application is arrived, μ_k denotes workload size of the task, d_k is size of data of the task and t_k denotes the time required to the completion of ω. The task ω is partitioned to m independent subtasks $\omega = \{\delta_1, \delta_2, \ldots \delta_m\}$, there is not dependency between tasks and can be executed in parallel in the edge computing on the edge nodes.

The task ω executed fairly Mudassar et al. (2019) means that resources needed by the task are provided to meet the time requirement. An task ω at any time t have resources $\Re this_\omega (t_k) \in [0,t_k]$ of η running on accumulated edge nodes shown in 1.

$$\Re this_\omega(t_k) = \int_0^{t_k} this_\omega(t)dt \qquad 1$$

Where $this_\omega(t)$ shows resources currently allocated to the task ω at t time. To get the job done required resources $\Re req_\omega (t_k) \in [0,t_k]$ can be accumulated as

$$\Re req_\omega(t_k) = \int_0^{t_k} req_\omega(t)dt \qquad 2$$

Where $req_\omega(t)$ shows at time t these resources are required to accomplish ω . To execute application fairly fd_ω of task ω at time t_k is given by.

$$fd_{\omega}(t_k) = \frac{\Re this_{\omega}(t_k)}{\Re req_{\omega}(t_k)} = \frac{\int_0^{t_k} this_{\omega}(t)dt}{\int_0^{t_k} req_{\omega}(t)dt} \qquad 3$$

The degree of fairness is as $d_{\omega}(t_k) \in [0,1]$, and 1 is when all required resources are allocated to complete the job in time ensuring the latency is reduced.

b) Decentralized grouping in edge

To efficiently handle a large task on an edge network, we propose distributing it across a federation of edge nodes, denoted as $G_i^E(\omega)$, where i represents i-th group, E denotes the edge nodes within the group, and ω is the task to be processed. The grouping of edge nodes is inspired by the way Muslims gather at a mosque for congregational prayers. An edge node, designated as the organizer E^{Org}, either has a task to perform or receives a task request from a user, as illustrated in figure 3. This organizer node collaborates with other edge nodes to complete the task and sends the results back to the user.

Our proposed algorithm operates in a decentralized manner, where E^{Org} merely initiates the formation of the federation of edge nodes. Each edge node independently decides whether to execute a subtask based on its local resource information. Once a node opts to execute a subtask, it becomes part of the federation, collaborating with other nodes to complete the distributed task. The organizer node (E^{Org}) may also execute part of the task but does not directly control the allocation of subtasks to other nodes.

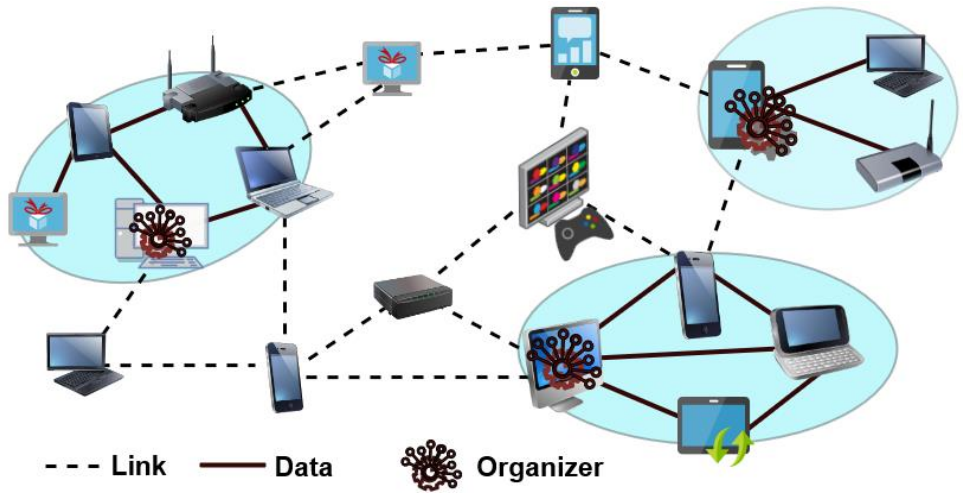

Figure 3: Edge node grouping

For a task ω comprising m independent subtask. Let c_i represent the number of computations required to complete a single subtask. The total workload for ω is provided as follows.

$$\mu_{\omega} = \alpha \sum_{i=1}^{m} c_i \qquad 4$$

Where α is a constant factor. The time (t) required to execute ω on node in the edge network i depends on its computational speed s_i, and is expressed as below.

$$t_{\omega} = \frac{\mu_{\omega}}{s_i} \qquad 5$$

The decision for executing ω distributed on different edge nodes depends on the condition as provided above in the equation 3 and ensuring the following condition.

$$t_{\omega} > t_{k} \qquad 6$$

If condition in equation 6 holds E^{Org} broadcast for node resources to form a group $G_i^E(\omega)$ inside the edge U and ensuring requirements to execute ω. The nodes in the execute $\omega = \{\delta_1, \delta_2, \ldots \delta_m\}$, and return the results to the E^{Org}. Each edge node determines its resource (R_{ni}) which it can allocate to the task. The assumption is that one edge node will receive one task and completes it independently. The algorithm is given for the group formation and task distribution.

Algorithm 1: Edge node group formation and task execution

Data: Organizer node E^{Org}; G(ω) = {}; U as the edge nodes; θ as empty queue
Output: Group $G_e^i(\omega)$ of edge nodes to execute task ω

For $E^{Org} \in$ U **do**
  θ ← E^{Org}
  E^{Org} broadcast to all edge nodes (u ∈ U)
  nodes (u′ ∈ U) in locality response back to E^{Org}
  for all nodes u′ **do**
    θ ← u′
  end for
  while $R_{\omega} \neq$ null && θ≠null **do**
    for $n_i \in$ θ **do**
      if $R_{\omega} \cap R_{ni}$ = resources **then**
        n_i pull $\delta_i \in \omega$
        $R_{\omega} = R_{\omega} - R_{ni}$
        G(ω) ← n_i
      end if
    end for
    for $n_i \in$ G(ω) **do**
      if (neighbor $n_k \in n_i$) ∩ G(ω) == null **then**
        n_i will send R_{Req}= (E^{Org}, R_{ω}) to n_k
        θ ← n_k
      end if
    end for
  end while
  $G_e^i(\omega)$ ← G(ω)
end for

Edge node E^{Org} requires group formation and sends a message the respective edge nodes in the edge network responds the request and inserted to queue θ. Based on the resources tasks are assigned to the edge nodes by the organizer node. Neighbors can also be added in the group G(ω). this will continuous until job is completed.

Results and Discussion

For the evaluation of our methodology we have simulated the video surveillance system for object tracking. This simulation is based on the number of cameras generating video and corresponding processing

modules, as described in (Gupta et al., 2017). This application involves a distributed system of cameras monitoring an area, applicable to various domains to provide security, ensure public safety, and manufacturing process monitoring. Main requirements for such a system are low latency, reduced network traffic, timely processing, and guaranteed reliability and availability. The application require lot of resources and requiring low-latency our methodology ifedge node grouping is well-suited for running this application.

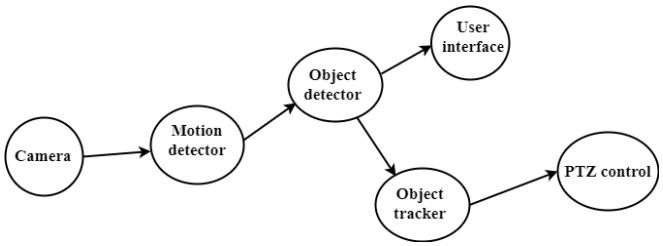

Fig. 4 : Simulation for surveillance tasks

The surveillance application comprises six modules, illustrated in the workflow shown in Figure 4. These modules (tasks) include *camera, motion detector, object detector, object tracker, user interface, and pan, tilt, and zoom (PTZ) control*. We conducted our simulation using iFogSim (Gupta et al., 2017), a simulation tool that runs on top of CloudSim (Calheiros et al., 2011). iFogSim supports executing application scenarios on edge devices with different configurations and allows for measuring various application-related statistics, as well as device and network metrics.

To test the simulation, the scenario was repeated with different numbers of workflows in the system, thereby increasing the number of tasks to be executed. Specifically, we tested the field view task, which processes video captured by cameras, detects objects, and identifies any instances of violence. For each workflow, an organizer node was designated to group edge nodes in the area. The organizer also ensured fault tolerance for nodes executing the workflow, as per the proposed methodology. A set of edge devices was deployed in the network to facilitate group formation. Since iFogSim does not support direct communication between devices at the same level, we introduced a gateway responsible for inter-device communication.

Each edge node had a processing capacity of 2400 MIPS (Millions of Instructions Per Second) and was connected to the gateway through a 10,000 Kbps bandwidth link with a 2 ms latency. The gateway was used solely for communication and had a processing capacity of 1000 MIPS. The link between edge devices and the gateway also had a bandwidth of 10,000 Kbps with a 4 ms latency, while the link between the gateway and the cloud featured 10,000 Kbps bandwidth and 100 ms latency. Sensors connected to the edge nodes operated over a 10,000 Kbps link with a 2 ms latency.

For task allocation, our methodology considered the CPU requirements of tasks. If the CPU requirements matched the capacity of an edge node, the task was processed at that node, which was then added to the group. We modified iFogSim's task allocation policy to incorporate resource-aware task placement, which was executed before the application started. To the best of our knowledge, no existing mechanism addresses the grouping of edge nodes for distributed processing of large tasks using iFogSim. Existing solutions primarily focus on task placement in edge networks based on edge node processing capabilities, such as the Edge-ward placement approach (Gupta et al., 2017). This approach attempts to place smaller jobs within the edge network, while larger jobs are routed to the cloud.

We compared our approach to the traditional cloud placement method and the Edge-ward placement policy. Metrics such as latency, data transfer volume over the network, and the deadline-miss ratio (percentage of tasks that miss their deadlines) were analyzed. Figure 5 presents a comparison of application latency. The

results demonstrate that latency is significantly higher when applications are executed on the cloud, as all data must be transferred to the cloud for processing. With an increase in the number of workflows, the latency for Edge-ward placement also rises because large tasks are offloaded to the cloud when edge nodes have insufficient processing capacity. In contrast, our approach processes large tasks in a distributed manner using edge node groups near the end user, resulting in much lower latencies. This advantage becomes even more pronounced as the number of workflows increases.

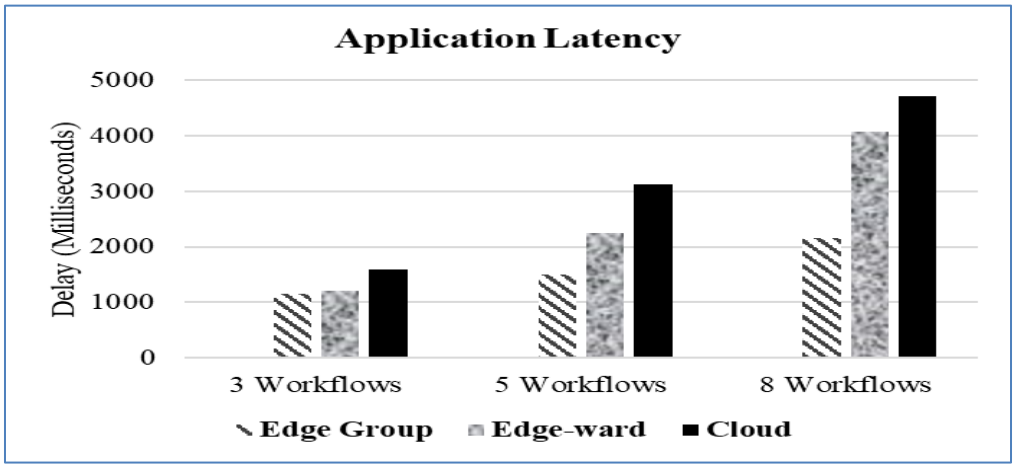

Fig. 5 : Application Latency comparison for proposed approach vs only edge node vs cloud

The reduced delays facilitated timely completion of application tasks using our methodology, as demonstrated in the figure 6, which compares the deadline-missing ratio across different approaches. The results clearly show that our approach significantly outperforms the others, maintaining a consistently low deadline-missing ratio in all cases. In contrast, both the Edge-ward placement and cloud-based approaches exhibit poor performance, particularly as job sizes increase. For Edge-ward placement, the deadline-missing ratio rises with larger jobs because tasks are offloaded to the cloud when edge network nodes lack sufficient capacity. In our approach, the formation of edge node groups ensures that tasks are processed within the edge network, allowing even large jobs to meet their deadlines. Conversely, the cloud-based approach suffers due to high latency between the local network and the cloud, causing many tasks to miss their deadlines. This issue becomes more severe as the number of workflows increases, leading to congestion on the link between the gateway and the cloud as sensor devices generate higher traffic.

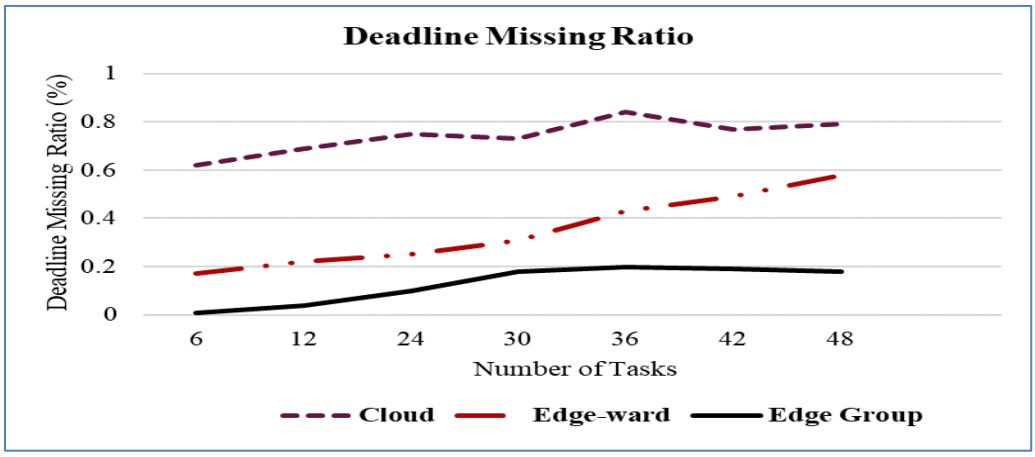

Fig. 6: Comparison of deadline missing ratio

Our proposed methodology also impacts overall network usage, as shown in the figure 7. As the tasks process in a distributed manner across edge node groups, the bandwidth use decreases compared to single edge and cloud approach. By processing tasks within the edge network, our approach conserves bandwidth. The results indicate that forming groups of edge nodes enables timely task execution while reducing both deadline-missing instances and total data transfer over the network.

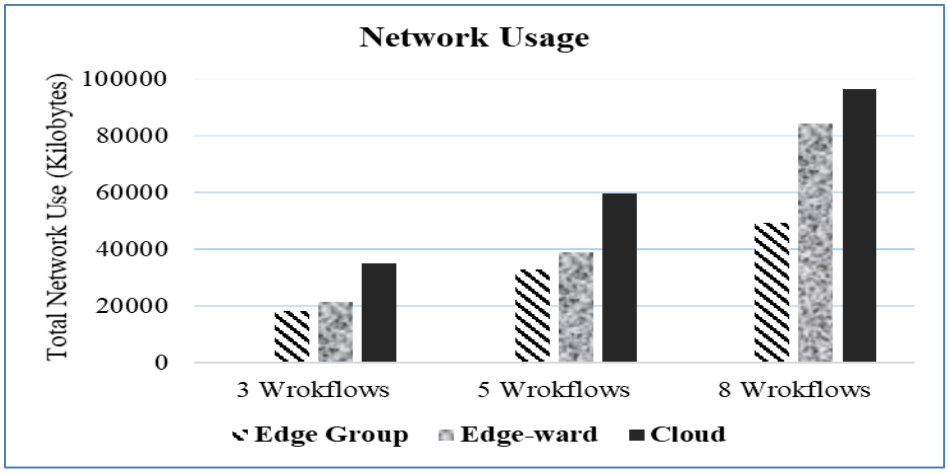

Fig. 7: Bandwidth comparison

The decentralized group formation mechanism activates upon task arrival, and task is executed on nodes in the edge network according to their resources. While resource leveraging upon job arrival may initially seem to add latency, the parallel execution of tasks across edge node groups mitigates this effect. This demonstrates that latency-oriented IoT applications can be executed efficiently in a distributed edge network without requiring a pre-established clustering mechanism for edge nodes.

Conclusion

A decentralized edge node grouping is proposed that groups the smart devices inside the edge network to combine their resources. These resources can be used to process the IoT applications in the edge network. This will reduce the latency and increase the IoT application efficiency along with reduction in the bandwidth utilization. Compared with other available approaches to cluster the edge nodes our proposed approach works without any global information hence the cluster head selection time will be eliminated, the edge node grouping immediately starts when task is arrived at organizer node who collaborate with nodes in the edge network to execute sub-tasks. Using parallel execution of subtasks the task deadlines are achieved. Additionally, by using benefit of edge computing to execute the IoT applications will save the bandwidth. Results shows that proposed methodology is effective and the performance is also effective one. Compared to cloud based execution of big tasks a significant improvement is resulted for the deadline missing ration, network traffic reduction and reduced latency measures. The edge node utilization will also lower the overall economic cost.

References

Ali, M., Shafi, M., & Farooq, U. (2018, April). Salat activity recognition using smartphone triaxial accelerometer. In *2018 5th International Multi-Topic ICT Conference (IMTIC)* (pp. 1-7). IEEE.

Bouet, M., & Conan, V. (2018). Mobile edge computing resources optimization: A geo-clustering approach. *IEEE Transactions on Network and Service Management*, *15*(2), 787-796.

Calheiros, R. N., Ranjan, R., Beloglazov, A., De Rose, C. A., & Buyya, R. (2011). CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and experience*, *41*(1), 23-50.

Chen, X., Jiao, L., Li, W., & Fu, X. (2015). Efficient multi-user computation offloading for mobile-edge cloud computing. *IEEE/ACM transactions on networking*, *24*(5), 2795-2808.

Flores, H., Hui, P., Nurmi, P., Lagerspetz, E., Tarkoma, S., Manner, J., ... & Su, X. (2017). Evidence-aware mobile computational offloading. *IEEE Transactions on Mobile Computing*, *17*(8), 1834-1850.

Gedawy, H., Habak, K., Harras, K., & Hamdi, M. (2018, July). An energy-aware iot femtocloud system. In *2018 IEEE International Conference on Edge Computing (EDGE)* (pp. 58-65). IEEE.

Gu, L., Zeng, D., Guo, S., Barnawi, A., & Xiang, Y. (2015). Cost efficient resource management in fog computing supported medical cyber-physical system. *IEEE Transactions on Emerging Topics in computing*, *5*(1), 108-119..

Guo, H., Ren, J., Zhang, D., Zhang, Y., & Hu, J. (2018). A scalable and manageable IoT architecture based on transparent computing. *Journal of Parallel and Distributed Computing*, *118*, 5-13.

Guo, H., Ren, J., Zhang, D., Zhang, Y., & Hu, J. (2018). A scalable and manageable IoT architecture based on transparent computing. *Journal of Parallel and Distributed Computing*, *118*, 5-13.

Gupta, H., Vahid Dastjerdi, A., Ghosh, S. K., & Buyya, R. (2017). iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments. *Software: Practice and Experience*, *47*(9), 1275-1296.

Habak, K., Ammar, M., Harras, K. A., & Zegura, E. (2015, June). Femto clouds: Leveraging mobile devices to provide cloud service at the edge. In *2015 IEEE 8th international conference on cloud computing* (pp. 9-16). IEEE.

Habak, K., Zegura, E. W., Ammar, M., & Harras, K. A. (2017, October). Workload management for dynamic mobile device clusters in edge femtoclouds. In *Proceedings of the second ACM/IEEE symposium on edge computing* (pp. 1-14).

Kumar, J. S., & Zaveri, M. A. (2018). Clustering approaches for pragmatic two-layer IoT architecture. *Wireless Communications and Mobile Computing*, *2018*(1), 8739203.

Latre, S., Leroux, P., Coenen, T., Braem, B., Ballon, P., & Demeester, P. (2016, September). City of things: An integrated and multi-technology testbed for IoT smart city experiments. In *2016 IEEE international smart cities conference (ISC2)* (pp. 1-8). IEEE.

Miorandi, D., Sicari, S., De Pellegrini, F., & Chlamtac, I. (2012). Internet of things: Vision, applications and research challenges. *Ad hoc networks*, *10*(7), 1497-1516.

Mudassar, M., Zhai, Y., & Liao, L. (2019). Efficient state management for scaling out stateful operators in stream processing systems. *Big data*, *7*(3), 192-206.

Qazi, Z. A., Penumarthi, P. K., Sekar, V., Gopalakrishnan, V., Joshi, K., & Das, S. R. (2016, March). KLEIN: A minimally disruptive design for an elastic cellular core. In *Proceedings of the Symposium on SDN Research* (pp. 1-12).

Satyanarayanan, M., Bahl, P., Caceres, R., & Davies, N. (2009). The case for vm-based cloudlets in mobile computing. *IEEE pervasive Computing*, *8*(4), 14-23.

Satyanarayanan, M., Simoens, P., Xiao, Y., Pillai, P., Chen, Z., Ha, K., ... & Amos, B. (2015). Edge analytics in the internet of things. *IEEE Pervasive Computing*, *14*(2), 24-31.

Shi, W., & Dustdar, S. (2016). The promise of edge computing. *Computer*, *49*(5), 78-81.

Su, X., Li, P., Riekki, J., Liu, X., Kiljander, J., Soininen, J. P., ... & Li, Y. (2018, March). Distribution of semantic reasoning on the edge of internet of things. In *2018 IEEE International Conference on Pervasive Computing and Communications (PerCom)* (pp. 1-9). IEEE.

Truong, H. L., & Dustdar, S. (2015). Principles for engineering IoT cloud systems. *IEEE Cloud Computing*, *2*(2), 68-76.

Wang, L., & Ranjan, R. (2015). Processing distributed internet of things data in clouds. *IEEE Cloud Computing*, *2*(1), 76-80.

Wang, Y., Sheng, M., Wang, X., Wang, L., & Li, J. (2016). Mobile-edge computing: Partial computation offloading using dynamic voltage scaling. *IEEE Transactions on Communications*, *64*(10), 4268-4282.

Yi, S., Hao, Z., Qin, Z., & Li, Q. (2015, November). Fog computing: Platform and applications. In *2015 Third IEEE workshop on hot topics in web systems and technologies (HotWeb)* (pp. 73-78). IEEE.

Zhang, W., Flores, H., & Pan, H. U. I. (2018, March). Towards collaborative multi-device computing. In *2018 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)* (pp. 22-27). IEEE.