International Journal of Cloud Applications and Computing
Volume 14 • Issue 1

Custom Network Protocol Stack for Communication Between Nodes in a Cloudlet System

Manoj Subhash Kakade, BITS Pilani, India*
https://orcid.org/0009-0007-8009-1553

K.R. Anupama, BITS Pilani, India
https://orcid.org/0000-0002-6334-4228

Sushil Nayak, BITS Pilani, India

Swarnab Garang, BITS Pilani, India

ABSTRACT

With the advent of internet of things (IoT), new network paradigms have emerged. One such technology is cloudlets. Cloudlets are being increasingly used in various IoT-based applications such as smart homes, smart cities, healthcare, and industrial automations. Cloudlets have an advantage of proximity to the end-device while offering services similar to the cloud. Existing cloudlets use IEEE 802.11 for communication between nodes. In this paper, the authors present a protocol customized for usage in cloudlets, which also considers various limitations of the node that constitute the cloudlet. The nodes on the cloudlet are generally constrained in terms of power and memory when compared to nodes on a cloud. The custom protocol also incorporates fault-tolerance, time synchronization, and factors such as task affinities for communication. The protocol proposed in this paper gave an excellent packet delivery ratio, the lowest being 91% even with increased bandwidth usage when compared to IEEE 802.11.

KEYWORDS

Cloudlets, Cross-Layer Network Stack, Dynamic TDMA, Time Synchronization, Internet of Things (IoT), Industrial IoT

With advances in internet of things (IoT) research, new computing paradigms have emerged (Pham et al., 2022). Billions of edge devices are part of multiple application domains in IoT. One of the primary areas of applications of IoT (Malik et al., 2021) is in industrial systems. Industrial IoT currently uses either edge or server-based computing due to the necessity of securing data. While large industrial complexes may employ a private cloud (Prajapati et al., 2018), medium or small-scale industries prefer edge computing (Khan et al., 2019). Edge computing requires powerful coordinators connected to

DOI: 10.4018/IJCAC.339891
*Corresponding Author

the end device to process the data (Cao et al., 2020). Industrial networks are inherently hierarchical (Galloway & Hancke, 2012); one possible solution is using cloudlet systems. The cloudlet layer is an extra layer introduced between the edge of the IoT system and the cloud. The cloudlet layer could handle short-term data processing and control decisions, while long-term data storage can be done on the cloud.

We have built a cloudlet system using multiple Qualcomm DragonBoard 410c development boards (Kakade, 2023). Our earlier work (Kakade et al., 2023) details the distributed task-sharing algorithm used on the cloudlet framework. For tasks and data to be distributed efficiently among the nodes in the cloudlet, an ideal network protocol stack is required for communicating between the nodes in the cloudlet. Currently, most cloudlet systems use IEEE 802.11 for communication between nodes in the cloudlet.

The custom algorithm described in this paper is a novel algorithm which considers multiple parameters such as:

- processing load on the node
- storage space availability of the node
- number of active network connections
- type of node used in building a communication protocol between various nodes in the cloudlet

The distributed task-sharing algorithm requires regular communication between various nodes in the cloudlet to obtain information regarding the state of the node's processing power and available storage, as task migration decisions are made based on these parameters. The exchange of information is done at regular intervals so that the decisions are not based on stale information. Whether tasks are migrated or not, the nodes will communicate consistently between them based on their current CPU and storage status. This requires a standardised network protocol to achieve constant communication between the nodes.

The network protocol stack is a set of protocols that are used to enable communication between nodes in a cloudlet. These protocols define the rules and standards for transmitting data over the network, ensuring that information is sent and received correctly (Xiang & Shaobin, 2020). By using a network protocol stack, nodes in a cloudlet can communicate with each other efficiently, enabling seamless data transmission and ensuring the integrity of the information being exchanged. Additionally, the network protocol stack allows for effective resource management within the cloudlet, enabling optimal utilisation of available network resources and ensuring smooth operation of various services within the cloudlet.

The significant contribution of this paper is novel custom-built network protocols explicitly designed for communication between nodes in a cloudlet environment. These custom-built protocols are optimised for cloudlet networks' unique characteristics and requirements, ensuring efficient data transmission and low latency. The development of this custom protocol is a significant contribution to the field, as it addresses the specific challenges and optimisations needed for effective communication within a cloudlet. Furthermore, the paper provides an in-depth analysis of the performance and effectiveness of this custom protocol compared to traditional network protocol stacks, demonstrating the superiority in terms of speed, reliability, and resource utilisation. Overall, this paper highlights the importance of a network protocol stack in enabling effective communication between nodes in a cloudlet and demonstrates the value of a custom-built protocol.

To the best of our knowledge, this paper is the first to propose and implement a custom network protocol stack specifically designed for communication within a cloudlet. The results of our experiments show significant improvements in data transmission efficiency and network resource utilisation compared to traditional network protocol stacks.

The rest of the paper is organised as follows: the next section gives a brief introduction to cloudlets and the proposed cloudlet architecture, followed by a detailed description of the operation of the

custom network protocol stack; the implementation details are discussed in the following section along with the results and analysis; and we conclude with the significant contribution of this paper and the direction of our future work.

CLOUDLETS

A cloudlet is a small-scale data centre or a cluster of computing devices designed to provide cloud services to primarily constrained devices close to it (Babar et al., 2021). The industrial IoT system is made up of many heterogeneous nodes that are connected using different wired and wireless networks (Lou et al., 2021). An industrial system has at least three levels of network hierarchy. The lowest level comprises sensors and actuators, while the second level comprises programmable logic controllers and computerized numerical control systems; the third level generally includes complex computing devices that can form edge devices in IoT systems. Industrial IoT systems are an ideal application for a localised cloudlet architecture as they guarantee data privacy, security, and predictable latency jitters (Fernández-Caramés et al., 2018). The proposed cloudlet system is a low-cost solution that is easily adaptable to small-scale and medium-scale industries. This solution is also scalable for large-scale industries since the architecture and the network protocol provide a consistent performance irrespective of the number of nodes that are part of the cloudlet system. The complete cloudlet architecture is shown in Figure 1.

The solution eliminates the need to pay for higher-cost, less secure cloud services that may be inaccessible due to connectivity issues. Most industrial complexes are situated at the periphery of cities, so the network connectivity may need to be more consistent. Also, cloudlet systems solve the significant and unpredictable jitter in task execution by allowing tasks with hard real-time deadlines to be scheduled on a priority basis (Ramasubbareddy & Sasikala, 2021). Tasks with long-running and soft real-time deadlines may be transferred to the cloud. Generally, industrial networks have three levels of hierarchy, as shown in Figure 2.

Figure 1. The Complete Architecture of Cloudlet

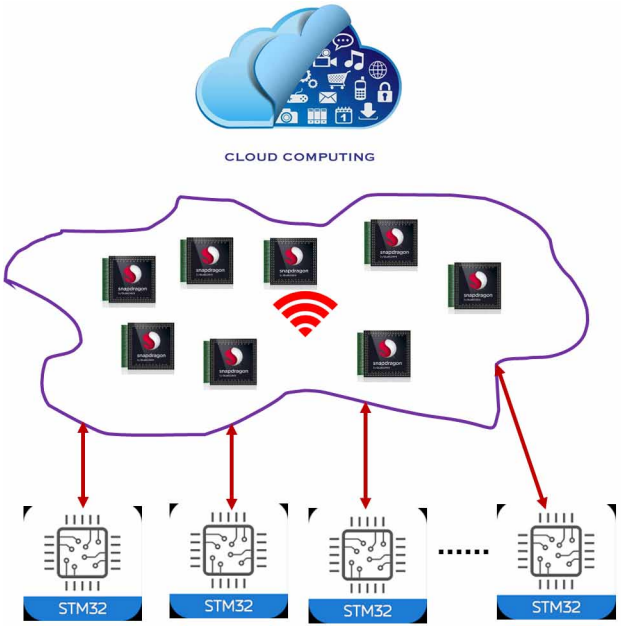
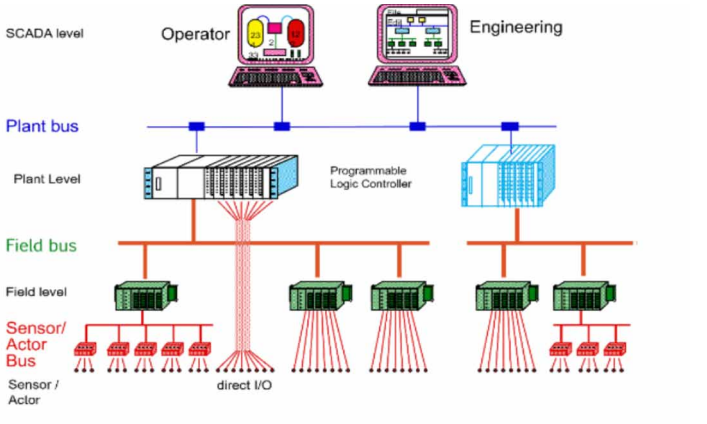


Figure 2. Hierarchical Model of Industrial Networks



1. The device level, i.e., the network of sensors and actuators. The primary function at the device level is data acquisition and control.
2. The field level, which is made up of controllers and programmable logic controllers. This level controls the manufacturing process or industrial equipment. The processing capabilities required are high because complex control functions are implemented at this level.
3. The plant level comprises two or more complex computing devices that form the edge of the IoT system. We propose replacing level 3 with a cloudlet system comprising distributed system on chips (SoCs) connected in an ad-hoc network.

The cloudlet system meets the requirements of industrial IoT: responsiveness, scalability, usability, flexibility, and security (Ren et al., 2019).

The cloudlet system collects industrial data in real time from the end devices in Levels 1 and 2.

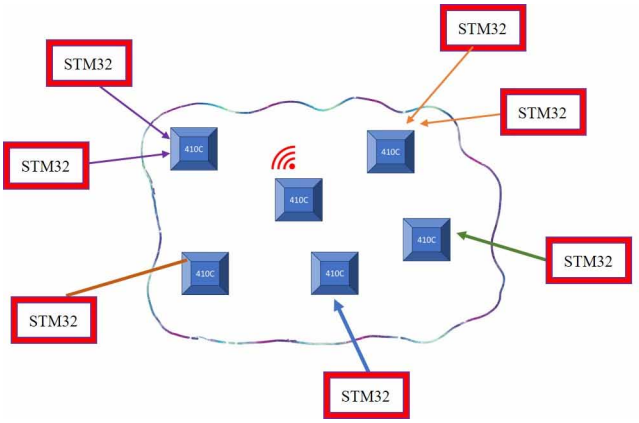
Distributed Cloudlet Architecture

This section gives a brief overview of the distributed cloudlet architecture. Distributed cloudlet architecture is a decentralised system where computing resources are distributed across multiple nodes in a cloudlet. By distributing computing resources this way, the architecture aims to reduce latency and improve the overall performance of applications and services (Shaukat et al., 2016). As seen in Figure 3, the end devices are connected to the individual nodes in the cloudlet architecture rather than relying on a central node for data processing. In addition, distributed cloudlet architecture also offers improved reliability and fault tolerance. By distributing computing resources across multiple cloudlets, the architecture reduces the impact of potential failures or disruptions of a single node on the overall system. This architecture also enables scalability, as additional nodes can be easily added to the network to accommodate increasing demands.

In addition to the hardware components, the individual nodes in the cloudlet architecture are equipped with software components essential for efficient data processing and management. These include operating systems tailored for the specific hardware architecture, virtualisation technologies for creating isolated environments, and distributed processing frameworks for coordinating and executing computational tasks across interconnected nodes.

The architecture of the individual node in the cloudlet is designed to provide a robust and efficient platform for handling diverse data processing tasks. The combination of powerful hardware and optimised software components enables each node to contribute effectively to the overall functioning

Figure 3. Distributed Cloudlet Architecture



of the cloudlet system, ensuring seamless collaboration and efficient data processing across the interconnected nodes.

By utilising the Qualcomm Snapdragon 410c SoC and integrating it into the cloudlet architecture, we can leverage the processing power, connectivity options, and storage capabilities of the individual nodes to achieve scalable and decentralised data processing, thereby enhancing the overall performance and efficiency of the cloudlet system.

Generally, in the case of industrial systems, there is a requirement of nearly zero jitter, which a cloud-based system cannot provide with varying networking latencies and connectivity. Cloudlets provide low-power computing capability with limited processing and memory, but as a distributed connected system, they can handle complex control algorithms while meeting hard real-time deadlines. The distributed architecture for task scheduling and data storage was described in our earlier paper (Kakade et al., 2023). In this paper, we only concentrate on the performance of the network protocol stack.

NETWORK PROTOCOL STACK FOR DISTRIBUTED CLOUDLET SYSTEMS

The network of cloudlets will be medium-sized and sparsely distributed. The nodes in the network are primarily SoCs. They may be based on Qualcomm Snapdragon 410c (Basic Kit for DragonBoard-410c, n.d.), Qualcomm Snapdragon 820c (DragonBoard 820c Hardware, Documentation, n.d.) or Qualcomm Snapdragon 888 (Snapdragon 888 Mobile Hardware Development Kit, n.d.). Each node in the cloudlet is connected to multiple end devices using one of the standard IoT communication protocols. On the other end, the cloudlet nodes are connected to the cloud using standard wired or wireless communication based on the availability of the infrastructure (Tawalbeh et al., 2015; Jia et al., 2015). Mobile ad-hoc network (MANET) protocols (Khan et al., 2018) are available, but the MANET network's primary focus is large and densely populated mobile nodes.

Similarly, multiple protocols are available for communication and networking in wireless sensor network (WSN) (Ketshabetswe et al., 2019). However, they work better with energy-constrained end devices and have limited processing power and memory compared to the SoCs. WSNs are meant for networks that cover a large geographic area (Ali et al., 2017). The nodes in the cloudlets will be placed nearby; hence, we have developed our own communication protocol for various nodes in the cloudlet.

The network protocol stack described here is the local network between various nodes in the cloudlet. While developing the network protocol stack, the primary aim was to keep it as lightweight as possible and ensure that multiple node characteristics such as task affinity, bandwidth availability,

storage space, and CPU availability were factored in. It is not possible to use the entire IEEE 802.11 stack to connect the various nodes in the cloudlet for multiple reasons:

1. The nodes have limited memory, which might already be in use for processing or storage.
2. The nodes in the cloudlet need to communicate in real time and require guaranteed latencies and delivery rates, which cannot be provided by collision-based protocols such as carrier sense multiple access (CSMA) collision detection or CSMA collision avoidance.

The cloudlet architecture is also dynamic, with new nodes added as required, while some nodes may shut down or be removed entirely from the network. This was taken into account while designing the network protocol stack. We went in for a cross-layer protocol stack rather than a layered network stack to ensure the protocol is lightweight while providing real-time guarantees. Every node in the cloudlet was assigned an ID based on its role in the cloudlet system. The node ID is a vital part of the network protocol design. SoCs such as Qualcomm Snapdragon 410c, which are capable of limited processing, are usually used for data gathering. Qualcomm Snapdragon 820c, being slightly more powerful, can run pruned machine learning and deep learning algorithms. Qualcomm Snapdragon 888, with its excellent graphic capability, can be used for computer vision.

The size of the network may vary from 20 nodes to 50 nodes. This is the largest cloudlet system attempted to the best of our knowledge. The network protocol stack should be able to scale irrespective of the network size.

Network Protocol Stack

In this sub-section, we present the complete network protocol stack connecting the various nodes in the cloudlet.

Neighbour Discovery

As the cloudlet architecture is entirely decentralised, every node has to discover its role in the network. This process starts with neighbour discovery. Every node in the network is in communication range; hence, they are all "neighbours" in network terms.

The issue with this is that as the network's size increases, the scalability of any protocol stack will be affected. If it is a contention-based protocol, the number of contentions and packet loss due to contention will be high. The energy consumption due to the control overhead and packet loss will also be high. If the communication protocol is contention-less, like time division multiple access (TDMA), then the number of time slots required will be very large, and communication latencies will be high. Hence, we have divided the network into subregions to improve the scalability of the network, as shown in Figure 4.

Henceforth, the term "neighbour" describes nodes in a subregion.

The node in the first subregion, which carries the minimum ID, starts initialising the network. The node broadcasts information about: its task affinities, its ID, available network bandwidth, available data storage (lightly loaded, moderately loaded, heavily loaded), and CPU utilisation (lightly loaded, moderately loaded, heavily loaded).

The nodes that receive the broadcast make an entry in a neighbour cache and wait for a random amount of time before broadcasting their information to their neighbours.

For example, if node A in Figure 5 initiates the broadcast, nodes B, C, and D will receive the broadcast and wait for a random duration of time before broadcasting their information to their neighbours E, F, and G in the following subregion. The random duration is calculated based on the available bandwidth, the data load status, and the CPU utilisation status. To this, a random number is added.

The delay relation can be represented as follows:

Figure 4. Division of Network Into Subregions

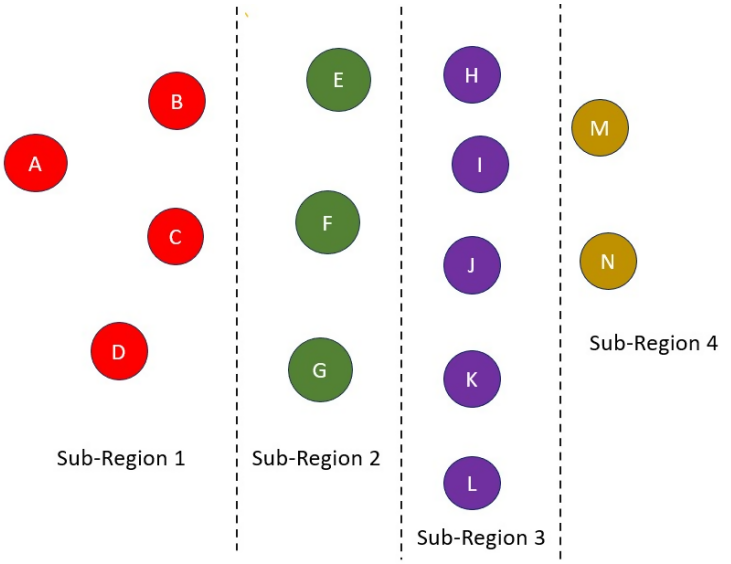
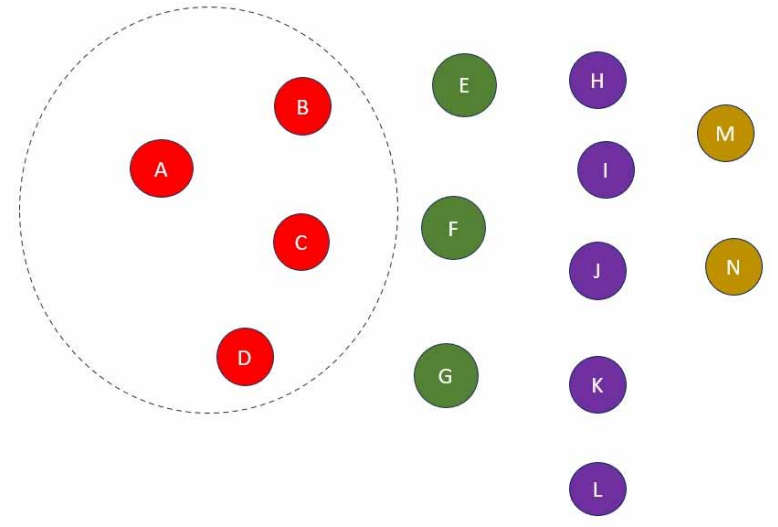


Figure 5. Initial Broadcast of Node A to Its Neighbours



Delay $\propto$ (1/CPU Utilisation)
Delay $\propto$ Available data storage
Delay $\propto$ Available Bandwidth (Provided there is task affinity)

After the delay, the nodes will send the same information as node A. The advantage of such randomisation is that lightly-loaded nodes end up at the top of the neighbour table compared to moderately-loaded or heavily-loaded nodes. Also, nodes with higher task affinities will end up at the top of the table.

The random value is added to introduce randomness in the initial stage of the cloudlet when all resources will be free.

For example, in Figure 6, if C is lightly loaded and can execute the same set of tasks as A, C will broadcast its information first. Hence, A will know C's availability and place it at the top of the network cache. If A now needs to offload data or tasks because it is heavily loaded, node C is most likely to receive that additional work.

We are not considering energy in this scheme, as all nodes on the cloudlet will be powered using standard power lines. Power conservation is not a significant factor in this protocol stack. Hence, we cannot adapt existing ad-hoc network or WSN protocol stacks as they are more oriented towards reducing energy consumption, which is a non-issue in our case.

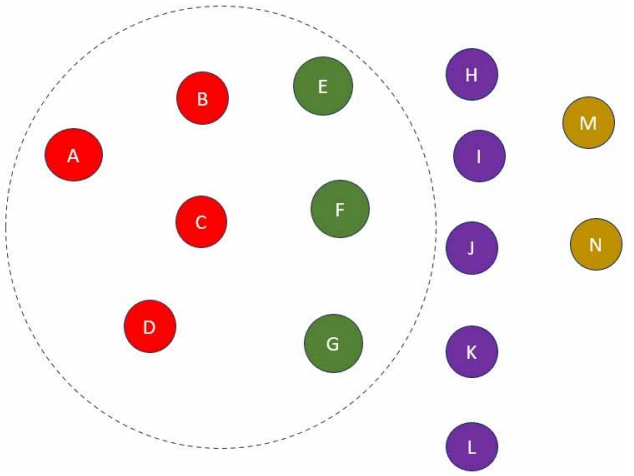
The broadcasts continue until all nodes in all subregions have information about their neighbours in their cache. In case there is a node that has yet to hear from its neighbours for a long duration of time, it can broadcast a discovery message to its neighbours. When a node receives a neighbour-discovery message, it will wait for a random delay (based on the data load, CPU utilisation, and task affinity). Once neighbour discovery has been completed, the nodes can now communicate with each other.

Suppose node A wants to offload some of its data or tasks to another node in the cloudlet. In that case, it will examine the neighbour cache and select the node with maximum task affinity, minimum data load, and CPU utilisation. The priority here will be given to CPU utilisation and task affinity if it is the task that is being migrated. If it is the data that is being migrated, then more weigh will be assigned to the memory space available on the neighbours. Therefore, while node C may have better task affinity and less CPU utilisation, the amount of data space available may not be sufficient. Hence, A might choose any of its other neighbours, B or D, depending upon the availability of space. An essential factor that each node considers before forwarding the packet is the available network bandwidth. Communication occurs if the network bandwidth is low or medium; otherwise, a different neighbour is selected.

Media Access Control

Initially, while the network is being set up, the media access control (MAC) protocol will be pure CSMA. Once neighbour discovery has been completed, a new MAC protocol, which is TDMA-based, will take over. Communication is divided into cycles. Each communication cycle is further divided into time slots.

Figure 6. Node C Broadcasts Its State's Information to Its Neighbours



There are two types of time slots: static and dynamic, as shown in Figure 7. In this manner, our MAC protocol resembles Flexray (Wang et al., 2015), which is used for intravehicular time-based communication.

Static Time Slot Allotment

The node with the minimum node ID assigns itself the first time slot and then broadcasts the information to the neighbours, as shown in Figure 8. In the example shown in Figure 8, Node A has the minimum ID; hence, it assigns itself the first time slot. A then broadcasts its information to its neighbour.

They, in turn, will select a time slot close to A; how close to A depends upon the time delay calculated during the neighbour-discovery phase. Node C, which would have transmitted with minimum time delay, will pick the second slot. This process will continue, and nodes B and D will forward the selected time slots and the free time slots to their neighbours, who will then choose their residual time slots.

The time slots being selected here are the static time slots. After the second neighbour, the time slots can overlap because the possibility of contention when a node is three subregions away is negligible. If tasks or data are to be offloaded to any nodes, the nodes with the subregion or the subregions that are closer to each other are considered.

This means there is only a need for a limited number of static time slots, making the protocol scalable.

In Figure 8, A and L take over the same time slot so that collision may occur. To avoid this, nodes transmit first so that they can reach nodes in their subregion or adjacent subregion. If node A wants to send data in the subregion, it will first try within Subregions 1 and 2 by limiting the communication energy. Node L, which is in Region 3, will try 3 and 4. If it cannot find a node, then there will be a collision. This is a limitation, but such a situation rarely arises.

Figure 7. Static and Dynamic Time Slots

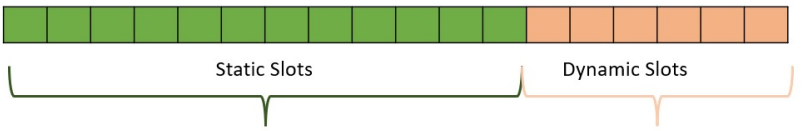
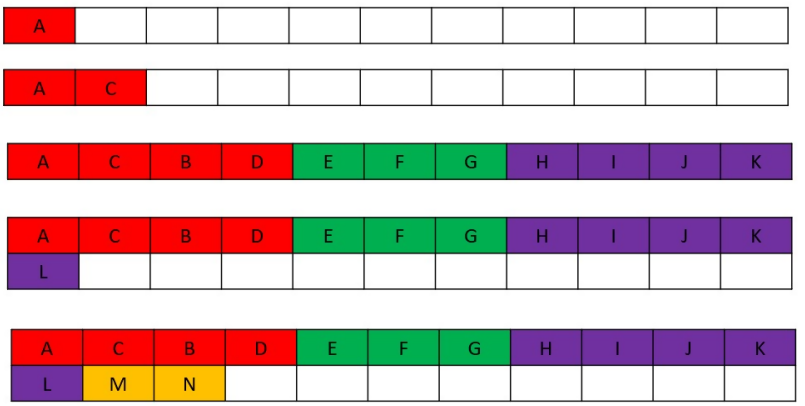


Figure 8. Static Allotment of Slots



This process is shown in Figure 8.

Dynamic Time Slots

The dynamic time slots are not pre-reserved. If any node wants to use the dynamic time slot, it should do so by making a reservation for the dynamic time slot at the end of its communication in the static time slot. So if A wants to use Dynamic Slot 3, then at the end of its communication at Static Slot 1, it will advertise that it will use Dynamic Slot 3.

Dynamic time slots are usually used for non-real-time communication. Even though dynamic time slot reservation is made on the fly, there will not be any contention as the nodes will choose the dynamic time slot they want to communicate in at the end of the static time slot. The number of dynamic time slots is less than the static time slots. We have experimented with various size ratios and found the ideal static-to-dynamic time slot ratio to be equal to half the number of nodes in the network. If the dynamic time slots become full in the current communication cycle, the node can wait until the next communication cycle to use the dynamic time slot. It must be noted here that dynamic time slots are not used for real-time communication.

Fault Detection

To have secure and reliable communication, we have added two bus guardians with a high communication range who will monitor all communications on the cloudlet system. If any cloudlet node sends corrupt data or data with incorrect timing, the bus guardians will take over and send an error message. The bus guardians are also responsible for maintaining time synchronisation between the various nodes. Two time slots at the beginning of every communication cycle are reserved for the two bus guardians in which they transmit timing and control information. We have used precision time protocol (PTP) (Watt et al., 2015) for time synchronisation, which is common in industrial networks. The communication between the nodes in the cloudlets will primarily be wireless and use the industrial, scientific, and medical band. So, each SoC will be equipped with a transceiver capable of communicating in the 2.4 GHz band.

Algorithm

```
Node
1: for each node, do
2:   if node_id = minimum_node_id then
3:     call broadcast(id, task_affinity, available_data_storage,
      available_cpu, available_network_bandwidth)
4:   else
5:       call wait_for_neighbour_broadcast()
6:   end
7: end

Neighbour Broadcast
1: if the neighbour's broadcast is  received,  then
2:   index ← call calculate_neighbour_table_index_value()
3:   call insert_neighbour_parameters_into_neighbour_table()
4:   delay ← call calculate_random_delay(neighbour.cpu_utilization,
     neighbour.data_storage,1 / (neighbour.network_bandwidth))
5:   call broadcast(id, task_affinity, available_data_storage,
     available_cpu,available_network_bandwidth)
6: end
```

```
Assignment of Static Time Slot
1: if node_id = minimum_node_id, then
2:          time_slot ← call get_time_slot(1)
3:          call broadcast_time_slots()
4: else
5:          call wait_for_static_time_slot_broadcast()
6: end
7: if received static time slot broadcast, then
8:   time_slots←callcheck_for_free_time_slots()
9:          n← call get_first_free_time_slot()
10:   call choose_time_slot(n + delay)
11:   call broadcast_time_slots(time_slots)
12: end
```

NS 2 Implementation

The simulation of the cloudlet was performed using Network Simulator 2 (NS 2.35) (About NS2 v2.35, n.d.). We used NS 2.35 as it is a universally accepted simulator (Alkenani & Nassar, 2022). We did not use Version 3 because it needs to implement various factors, such as node grouping according to task affinity. It also does not allow variation in cross-layer protocol design where individual node characteristics are used for determining the source and destination of communication. Also, NS 2.35 has a precise modular implementation, allowing us to implement our custom network protocol stack quickly.

To test the scalability of the network implemented, we varied the number of nodes from 10 to 50. We envisage that the number of nodes in a cloudlet will not exceed 50, as cloudlets are supposed to be a miniature version of a cloud-based system.

All nodes in the cloudlet are within communication range of each other. Hence, the primary implementation has been in the MAC layer. Additionally, a time synchronisation protocol was implemented to coordinate the timing between various nodes in the network. The time synchronisation protocol implemented was the standard industrial PTP. Since the network is expected to be fault tolerant, two additional nodes with snooping capability were included to monitor all the traffic in the network. These nodes are termed bus guardians. The bus guardians were added to the general operation directory (GOD.cc) since GOD.cc is aware of all the transactions in the network. The traffic generation model was also modified to ensure that nodes with similar task affinities formed the source-destination pair. Even when the traffic was generated randomly, if the criteria of task affinity were not satisfied, the source-destination pair was replaced with nodes with similar task affinities.

Variations were made to the TDMA protocols to introduce the concept of mini-time slots. To implement the mini-time slots in which time-critical data was sent, the major slots were taken as integer multiples of the mini-time slots. The nodes were dynamically allocated the major time slots and were required to reserve the mini-time slots in advance. All changes were implemented in mac-tdma.cc.

As all nodes were in communication range of each other, we used a DumbAgent as the routing protocol. In literature (Khanh et al., 2020), IEEE 802.11 is used mainly for communication between nodes in a cloudlet. Hence, we did a comparative study of our custom protocol with IEEE 802.11, non-persistent CSMA (Ma et al., 2018), Aloha (Munari et al., 2015), and TDMA (Le et al., 2017).

We have compared the packet delivery ratio, the control overhead, and the latency incurred by the various protocols. The detailed results and their implications are discussed in the next section.

RESULTS AND DISCUSSION

This section presents the result of the simulation of the network protocol stack for varying data loads. The analysis presented here is restricted to the packet delivery ratio, control overhead, and latency.

This analysis aims to evaluate the performance of the network protocol stack under different data loads, specifically focusing on packet delivery ratio, control overhead, and latency at the MAC layer, which was extracted using practical extraction and reporting language scripts. The energy was not considered, as the assumption was made that the nodes are powered using a standard power outlet. This cloudlet architecture was primarily used in an industrial environment where each node is powered using a standard power outlet.

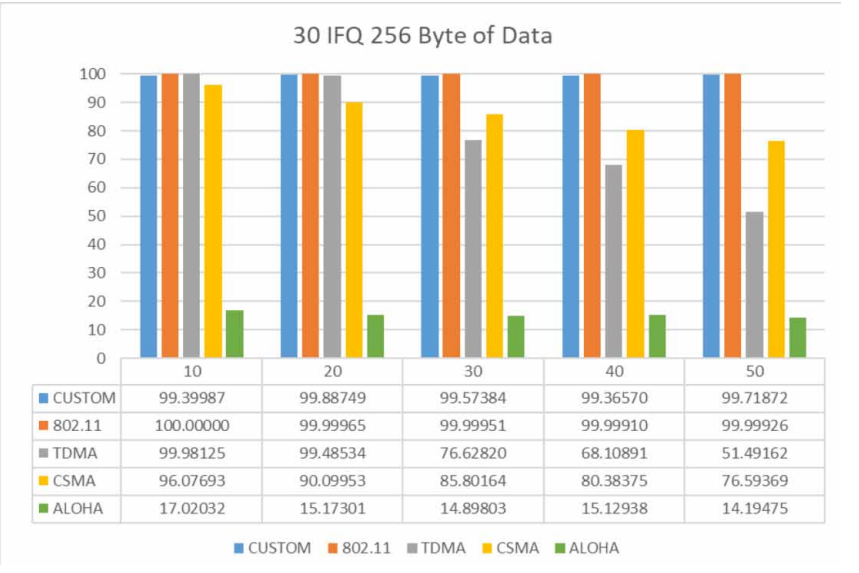
The simulation results showed a direct correlation between data load and packet delivery ratio, control overhead, and latency.

The detailed graphs of the packet delivery ratio for varying packet sizes of 256, 512, 1024, 2048, and 4096 bytes were compared. The number of packets generated was 16 packets/ second. Hence, the data was generated at a rate of 32786 B/s, 65536 B/s, 131072 B/s, 2062144 B/s, and 524288 B/s, respectively. The reason for using large amounts of data is to demonstrate the migration of tasks from one node to another in the cloudlet; generally, the task size would be large. Also, the data required for processing the task would have to be migrated along with the task. We have maintained the interface queue length (IFQ) for the graphs shown to be 30 packets. We have compared varying IFQ sizes, but the difference was negligible. We changed the IFQ length from 10 to 50 in steps of 10, with minor variations in results; hence, we retained data for an IFQ length for a mid-value of 30. We have used connectivity of about 70% of the total number of nodes. This demonstrates the protocol's scalability despite the heavy bandwidth use since the cloudlet system is supposed to perform well whether the bandwidth usage is low, medium, or high. The data rate was set to the standard value of 1 Mbps.

Packet Delivery Ratio

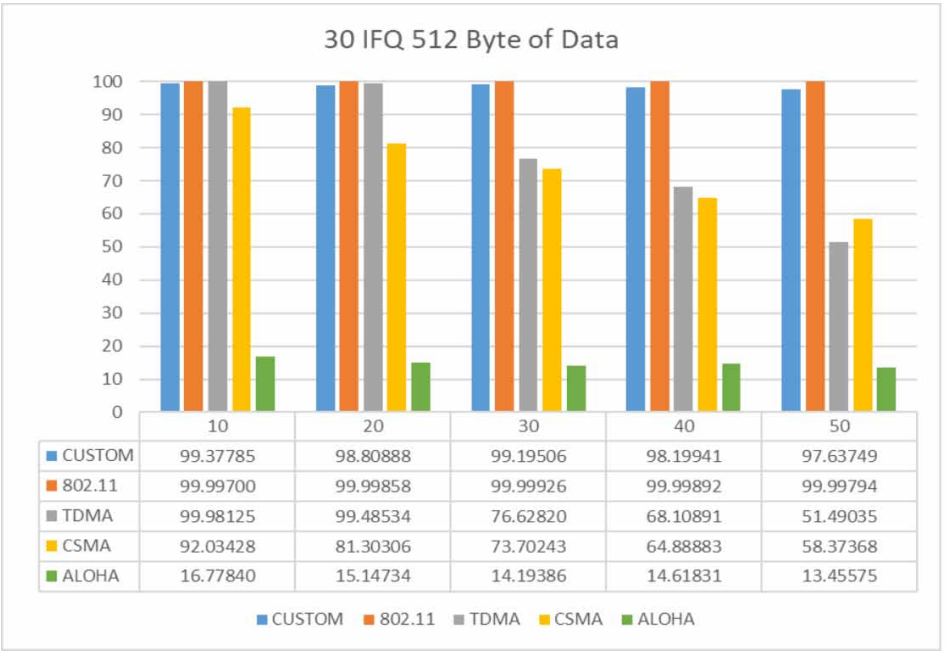
As can be seen from Figures 9 through 13, for packet delivery ratio (PDR), there is minimal variation between the custom protocol and IEEE 802.11 for smaller packet sizes. However, as the packet size increases and the number of nodes in the network increases, the custom algorithm achieves better scalability, which gives a minimum PDR of 91.1% with a data size of 4096 bytes. Both TDMA and Aloha, as expected, do not perform well. The same is true of non-Persistent CSMA. The protocol was compared against IEEE 802.11 as most cloudlets use IEEE 802.11 for inter-nodal communication. The comparison against TDMA was performed as the protocol developed is a dynamic variation of

Figure 9. PDR for Packet Size = 256 Bytes



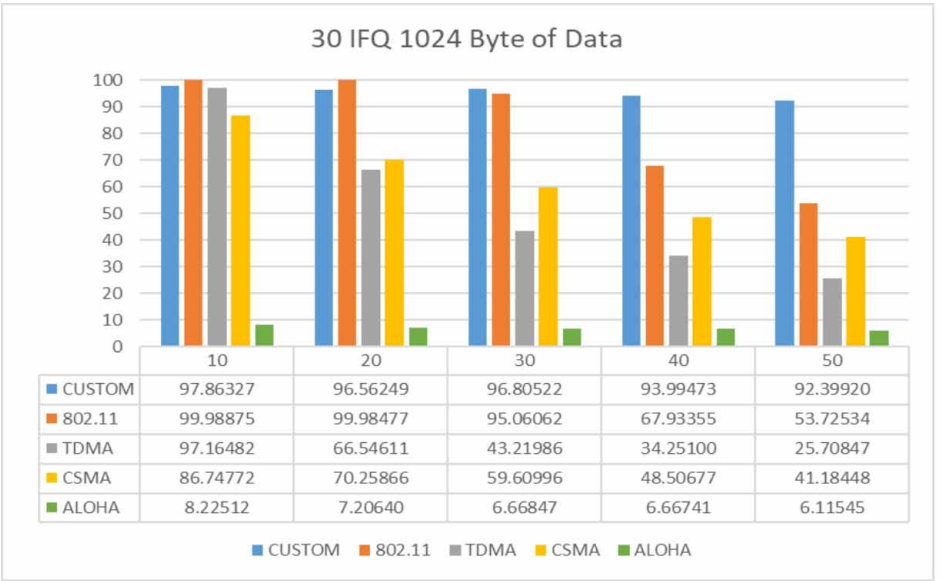
	10	20	30	40	50
CUSTOM	99.39987	99.88749	99.57384	99.36570	99.71872
802.11	100.00000	99.99965	99.99951	99.99910	99.99926
TDMA	99.98125	99.48534	76.62820	68.10891	51.49162
CSMA	96.07693	90.09953	85.80164	80.38375	76.59369
ALOHA	17.02032	15.17301	14.89803	15.12938	14.19475

Figure 10. PDR for Packet Size = 512 Bytes



	10	20	30	40	50
CUSTOM	99.37785	98.80888	99.19506	98.19941	97.63749
802.11	99.99700	99.99858	99.99926	99.99892	99.99794
TDMA	99.98125	99.48534	76.62820	68.10891	51.49035
CSMA	92.03428	81.30306	73.70243	64.88883	58.37368
ALOHA	16.77840	15.14734	14.19386	14.61831	13.45575

Figure 11. PDR for Packet Size = 1024 Bytes



	10	20	30	40	50
CUSTOM	97.86327	96.56249	96.80522	93.99473	92.39920
802.11	99.98875	99.98477	95.06062	67.93355	53.72534
TDMA	97.16482	66.54611	43.21986	34.25100	25.70847
CSMA	86.74772	70.25866	59.60996	48.50677	41.18448
ALOHA	8.22512	7.20640	6.66847	6.66741	6.11545

Figure 12. PDR for Packet Size = 2048 Bytes

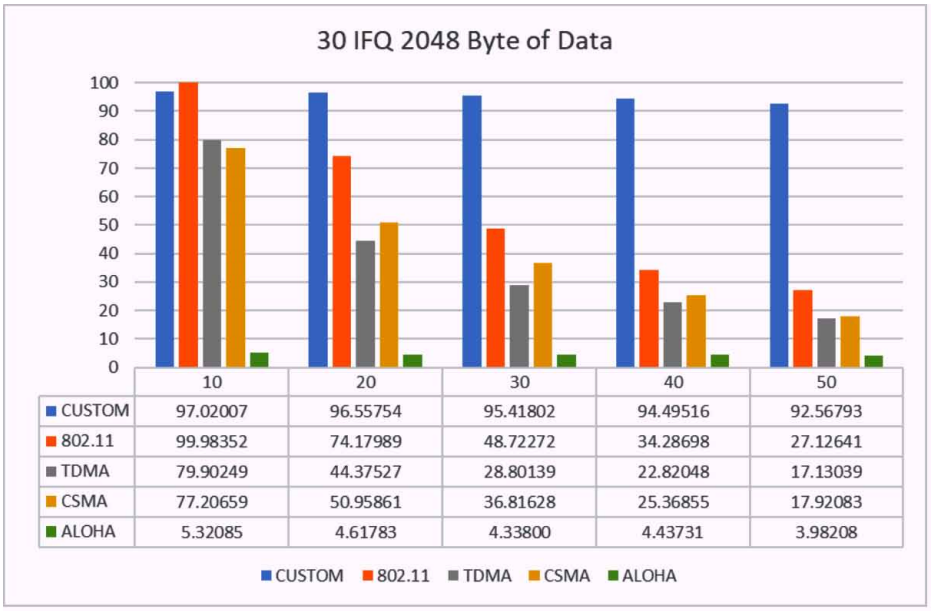
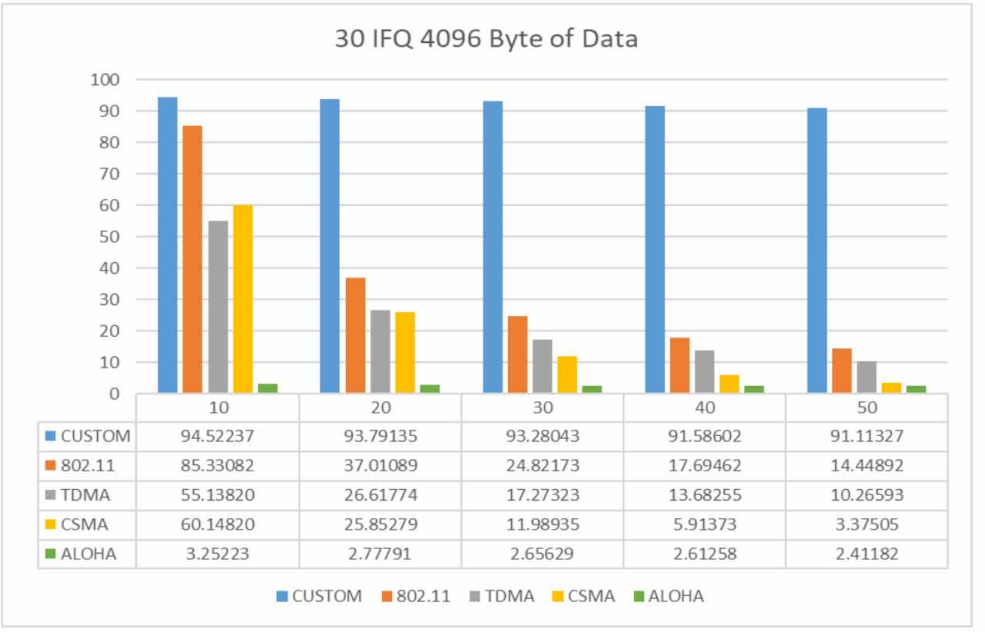


Figure 13. PDR for packet size = 4096 bytes

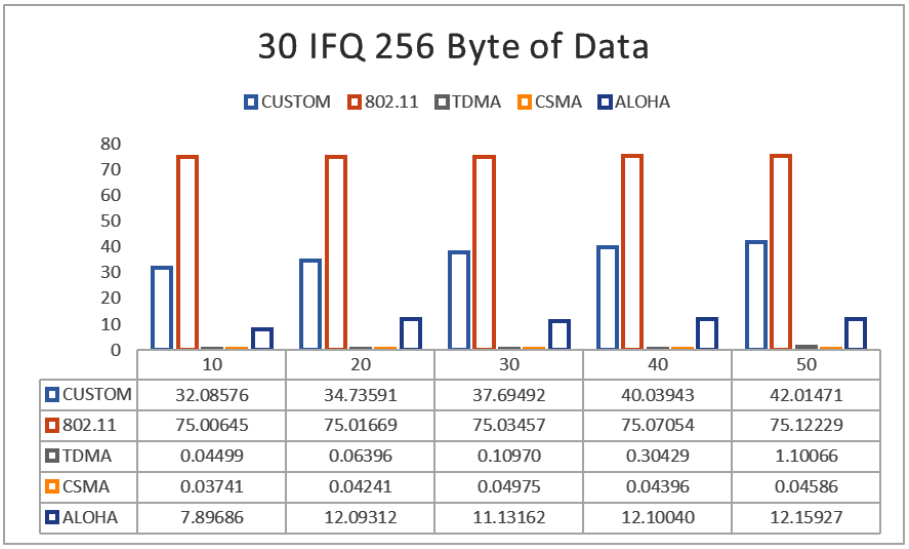


TDMA. CSMA and Aloha were used mainly for historical reasons and because they have very low control overheads compared to the custom protocol and IEEE 802.11.

Control Overhead Ratio

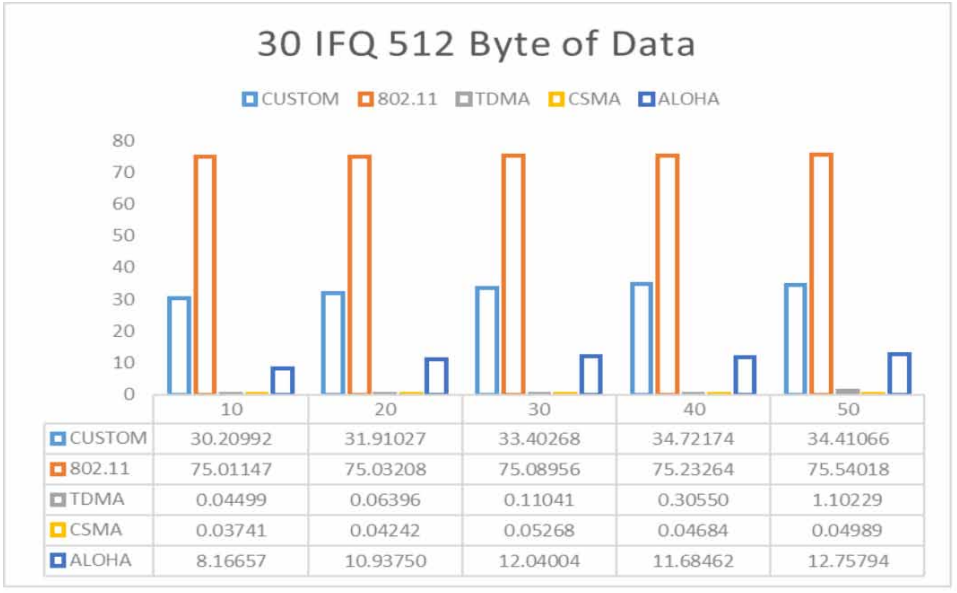
We analysed the control overhead ratio in the network, as shown in Figures 14 through 18. The routing overheads are not considered as all nodes are within communication range, and our routing protocol

Figure 14. COR for Packet Size = 256 Bytes



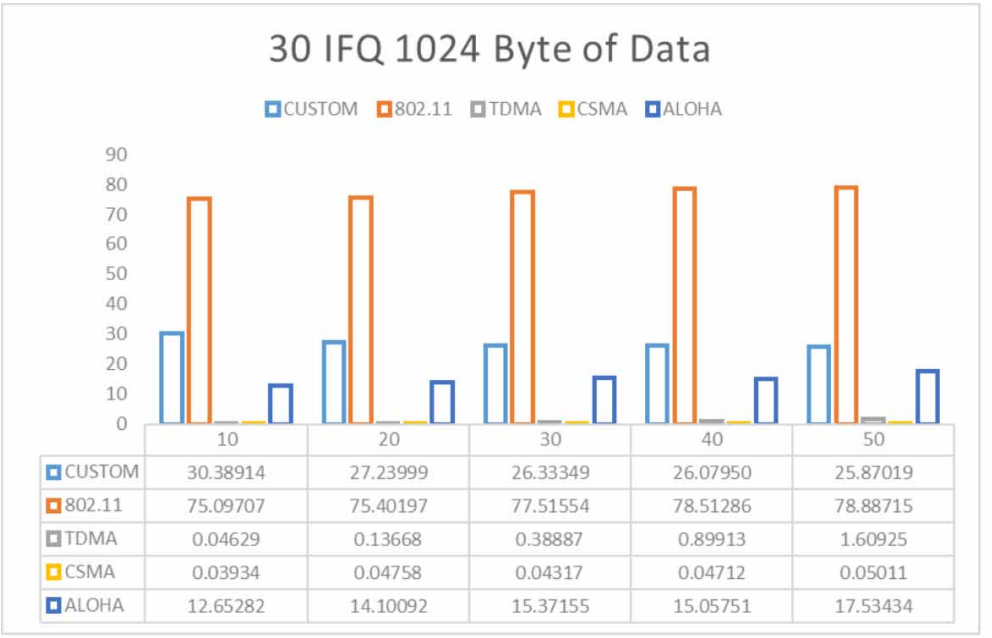
	10	20	30	40	50
CUSTOM	32.08576	34.73591	37.69492	40.03943	42.01471
802.11	75.00645	75.01669	75.03457	75.07054	75.12229
TDMA	0.04499	0.06396	0.10970	0.30429	1.10066
CSMA	0.03741	0.04241	0.04975	0.04396	0.04586
ALOHA	7.89686	12.09312	11.13162	12.10040	12.15927

Figure 15. COR for Packet Size = 512 Bytes



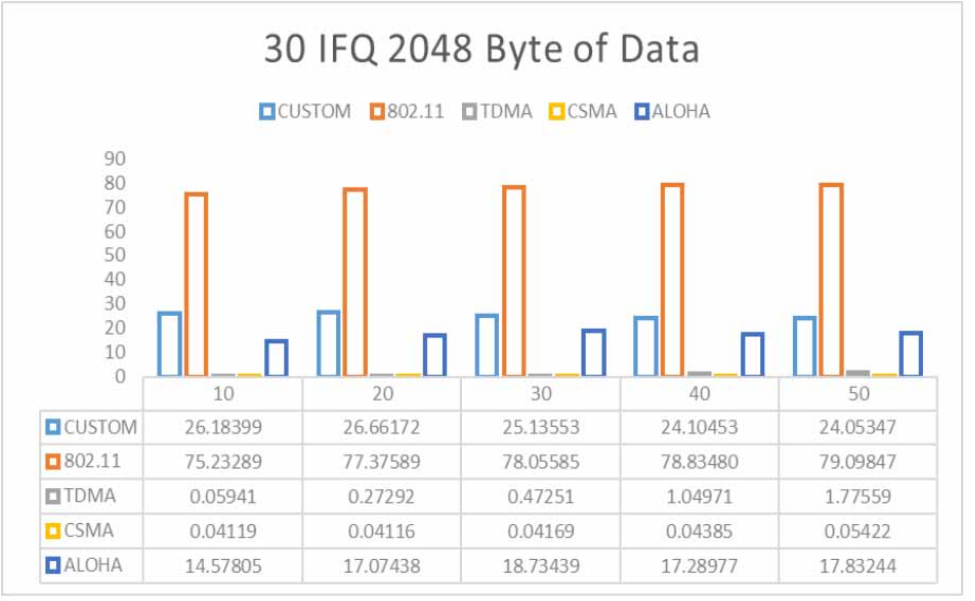
	10	20	30	40	50
CUSTOM	30.20992	31.91027	33.40268	34.72174	34.41066
802.11	75.01147	75.03208	75.08956	75.23264	75.54018
TDMA	0.04499	0.06396	0.11041	0.30550	1.10229
CSMA	0.03741	0.04242	0.05268	0.04684	0.04989
ALOHA	8.16657	10.93750	12.04004	11.68462	12.75794

Figure 16. COR for Packet Size = 1024 Bytes



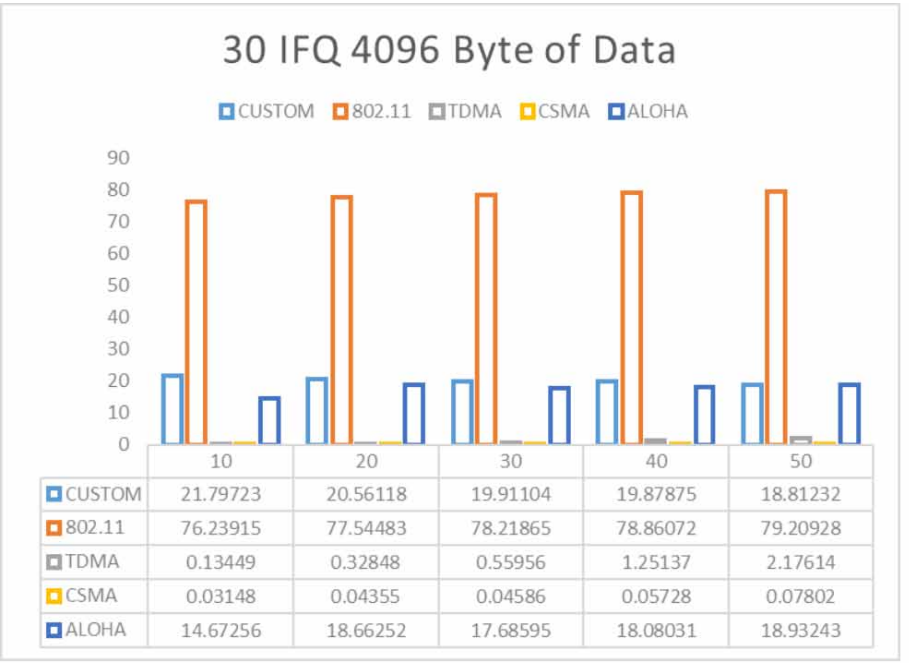
	10	20	30	40	50
CUSTOM	30.38914	27.23999	26.33349	26.07950	25.87019
802.11	75.09707	75.40197	77.51554	78.51286	78.88715
TDMA	0.04629	0.13668	0.38887	0.89913	1.60925
CSMA	0.03934	0.04758	0.04317	0.04712	0.05011
ALOHA	12.65282	14.10092	15.37155	15.05751	17.53434

Figure 17. COR for Packet size = 2048 Bytes



	10	20	30	40	50
CUSTOM	26.18399	26.66172	25.13553	24.10453	24.05347
802.11	75.23289	77.37589	78.05585	78.83480	79.09847
TDMA	0.05941	0.27292	0.47251	1.04971	1.77559
CSMA	0.04119	0.04116	0.04169	0.04385	0.05422
ALOHA	14.57805	17.07438	18.73439	17.28977	17.83244

Figure 18. COR for Packet Size = 4096 Bytes



	10	20	30	40	50
CUSTOM	21.79723	20.56118	19.91104	19.87875	18.81232
802.11	76.23915	77.54483	78.21865	78.86072	79.20928
TDMA	0.13449	0.32848	0.55956	1.25137	2.17614
CSMA	0.03148	0.04355	0.04586	0.05728	0.07802
ALOHA	14.67256	18.66252	17.68595	18.08031	18.93243

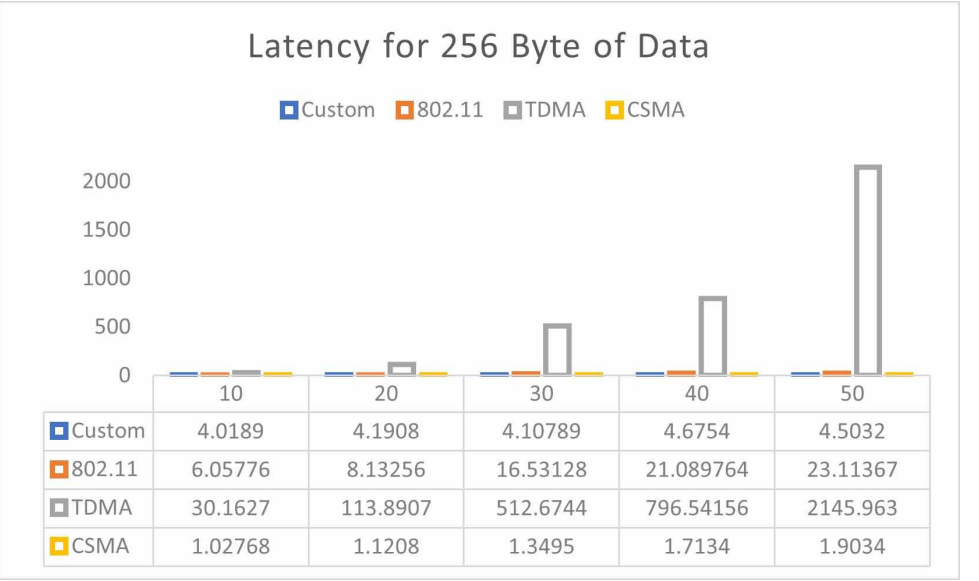
is a DumbAgent. The control overhead of the custom protocol for smaller-size packets varies in the range of 30–40%. In the case of IEEE 802.11, the control overhead remains consistent at around 75%. TDMA and CSMA have negligible control overhead, which is less than 1%. Due to the exchange of "hello" packets, Aloha has a control overhead of less than 15%. The number of hello packets in Aloha increases with the data packets. Hence, there is a slight increase in control overhead with packet size. The packets are fragmented when the packet size is large, as in the case of 1024, 2048, and 4096. IEEE 802.11 has a significant control overhead primarily because of the handshaking before and after the data transmission. The request-to-send, clear-to-send, and acknowledgement frames add to the overhead, which increases with the number of nodes and connections.

In the case of the custom protocol, the control overhead decreases with the data size and the number of nodes. This is because the number of control packets that will be exchanged remains constant. As the number of nodes increases, the data size increases, the number of connections increases, and the number of data packets increases, but the number of control packets remains relatively constant. Hence, it appears that there is a drop in control overhead.

Latency

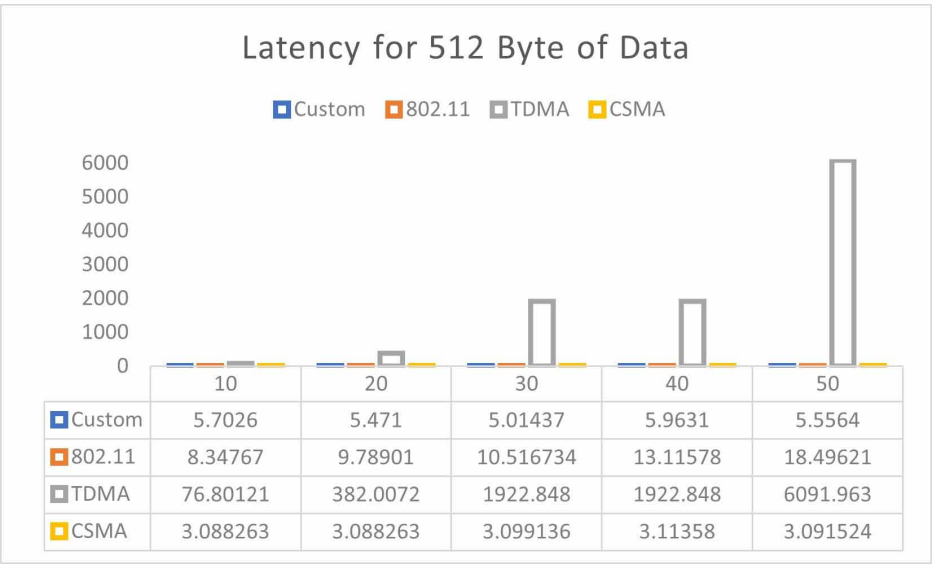
Latency is a crucial factor to consider when evaluating network protocols. It refers to the delay or time a message or data packet takes to travel from its source to its destination within a network. We analysed the latency in the network, as shown in Figures 19 through 23. The latency of the custom protocol for varying packet size and varying the number of nodes in the network is better than IEEE 802.11 and TDMA. The latency of custom protocol varies in the range of 5–25% based on the data packet size and the number of nodes in the network. In the case of IEEE 802.11 and TDMA, the variations are more with the increment in data packet size and the number of nodes in the network.

Figure 19. Latency for 256 Bytes of Data



	10	20	30	40	50
Custom	4.0189	4.1908	4.10789	4.6754	4.5032
802.11	6.05776	8.13256	16.53128	21.089764	23.11367
TDMA	30.1627	113.8907	512.6744	796.54156	2145.963
CSMA	1.02768	1.1208	1.3495	1.7134	1.9034

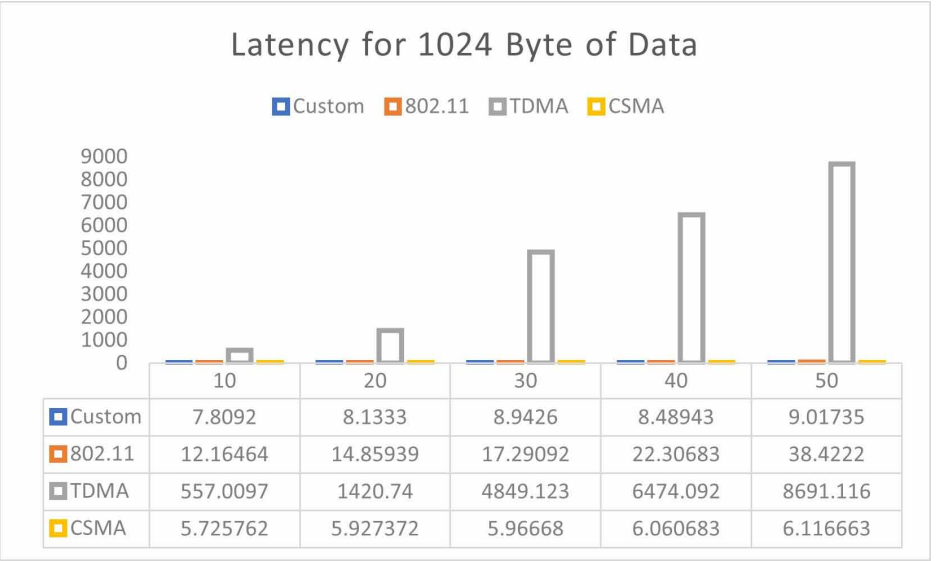
Figure 20. Latency for 512 Bytes of Data



	10	20	30	40	50
Custom	5.7026	5.471	5.01437	5.9631	5.5564
802.11	8.34767	9.78901	10.516734	13.11578	18.49621
TDMA	76.80121	382.0072	1922.848	1922.848	6091.963
CSMA	3.088263	3.088263	3.099136	3.11358	3.091524

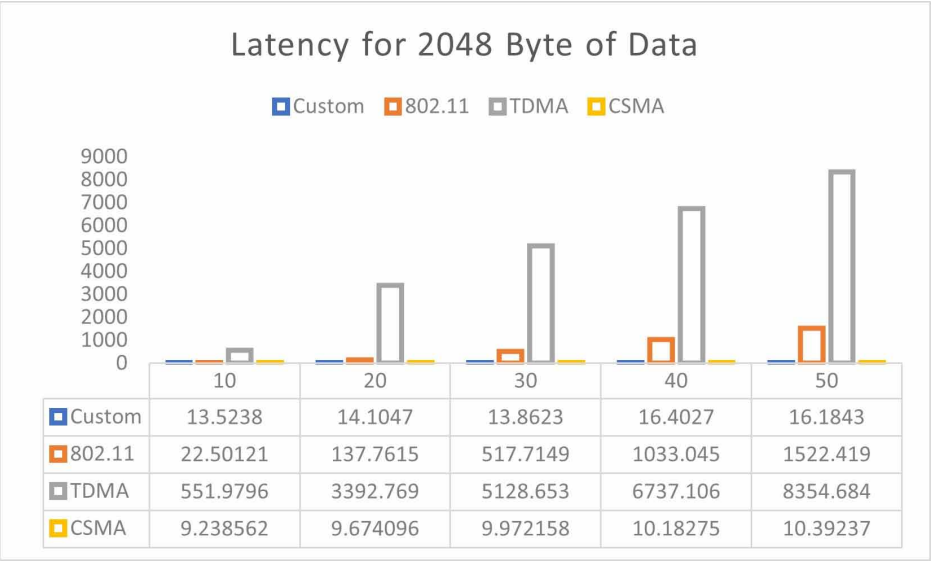
As the CSMA has negligible overhead, it has better latency for smaller data packet sizes; however, in the case of 4096 byte data packet, CSMA and the custom protocol stack give comparable results in terms of latency. With an increase in the number of nodes in the network custom protocol stack performs better in terms of latency because the control overhead seems to decrease with the data size and the number of nodes.

Figure 21. Latency for 1024 Bytes of Data



	10	20	30	40	50
Custom	7.8092	8.1333	8.9426	8.48943	9.01735
802.11	12.16464	14.85939	17.29092	22.30683	38.4222
TDMA	557.0097	1420.74	4849.123	6474.092	8691.116
CSMA	5.725762	5.927372	5.96668	6.060683	6.116663

Figure 22. Latency for 2048 Bytes of Data

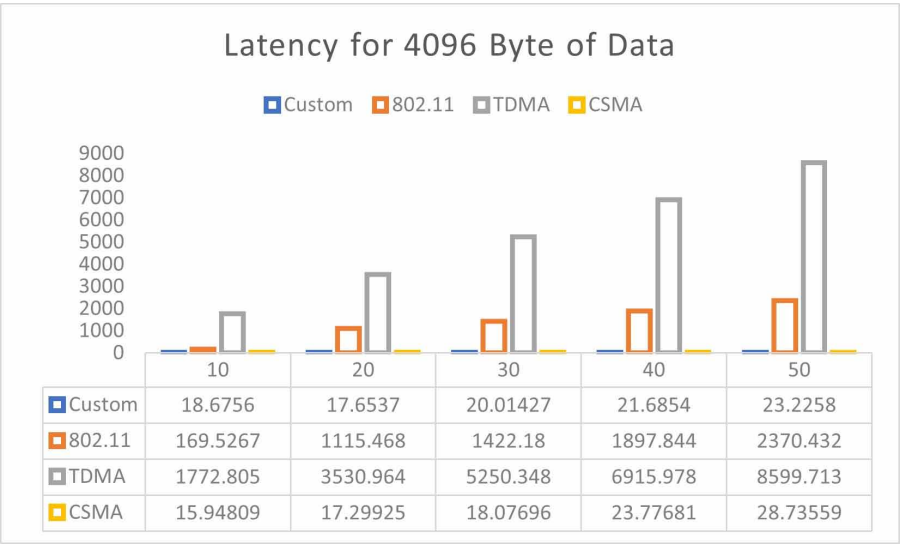


	10	20	30	40	50
Custom	13.5238	14.1047	13.8623	16.4027	16.1843
802.11	22.50121	137.7615	517.7149	1033.045	1522.419
TDMA	551.9796	3392.769	5128.653	6737.106	8354.684
CSMA	9.238562	9.674096	9.972158	10.18275	10.39237

CONCLUSION AND FUTURE WORK

This paper presents a novel custom protocol for communication within nodes in a cloudlet. We have shown that the protocol is scalable even with many nodes in the cloudlet. The protocol requires a significantly smaller memory footprint as no complex arithmetic is involved in the computation of roots or connections. The protocol was tested on NS 2.35 so we could verify the scalability. The protocol was designed for a cloudlet system built using 410c, which has limitations in terms of memory space. It has good computational power as it is a quad-core Kryo processor. The major limitation

Figure 23. Latency for 4096 Bytes of Data



	10	20	30	40	50
Custom	18.6756	17.6537	20.01427	21.6854	23.2258
802.11	169.5267	1115.468	1422.18	1897.844	2370.432
TDMA	1772.805	3530.964	5250.348	6915.978	8599.713
CSMA	15.94809	17.29925	18.07696	23.77681	28.73559

of this protocol is the higher control overhead than CSMA or TDMA, which does not increase with an increase in packet size or number of nodes. Hence, the protocol is scalable in terms of control overhead as well. The protocol has dedicated bus guardians that snoop on all communication and ensure minimal errors. The communication of the bus guardians has been included in the control overhead calculations. We have also implemented time synchronisation using the PTP. We have not compared the time synchronisation protocol against TDMA, as TDMA already runs with perfect simulation time. PTP can be integrated with any MAC protocol with minimum control overhead increase.

To the best of our knowledge, this is the only cross-layer protocol that has been specifically designed for peer-to-peer communication between nodes in a cloudlet. While the performance of the protocol has also been verified on Qualcomm Snapdragon 410c, the number of nodes was limited to eight; this was because of the application of sensor digital twinning, which we had selected as proof of concept of implementation of digital twining-based application on cloudlets did not require more than seven nodes.

Future Work

In the future, we plan to increase the number of nodes, make the network more heterogeneous for various industrial IoT applications, and test the network in real time.

The major challenge was integrating the protocol stack into the existing distributed architecture framework and task scheduling algorithms. This requires modifications of the operating system; we have added the scheduling algorithm plus network protocol stack as a modular layer over the Debian system. The version is still unstable and requires more testing with multiple scenarios.

Security is a major issue in any cloud-based system. Using cloudlets instead of clouds makes the system more secure and less susceptible to attacks. Cloudlets play a crucial role in strengthening the security of cloud-based systems. Organisations can significantly reduce their susceptibility to potential cyber-attacks and data breaches by leveraging cloudlets instead of traditional clouds.

One of the key reasons cloudlets are effective in enhancing security is their ability to bring computation and data storage closer to the network's edge. This reduces the distance data needs to travel, minimising the exposure to potential security vulnerabilities and threats. Hence, no security protocol was integrated into the network protocol stack of the cloudlets. Communication

between the end devices and nodes in the cloudlets uses standard authentication methods to ensure secure communication between the cloudlet and the external world. At this point, we have yet to implement security protocols in the cross-layer network protocol stack. However, it is essential to note that implementing security protocols in the network protocol stack of cloudlets is still an ongoing challenge.

REFERENCES

About NS2 v2.35. (n.d.). *Information sciences institute*. Retrieved September 19, 2023, from https://www.isi.edu/nsnam/ns/edu/index.html

Ali, A., Ming, Y., Chakraborty, S., & Iram, S. (2017). A comprehensive survey on real-time applications of WSN. *Future Internet*, *9*(4), 77. doi:10.3390/fi9040077

Alkenani, J., & Nassar, K. A. (2022). Enhanced system for ns2 trace file analysis with network performance evaluation. *Iraqi Journal of Intelligent Computing and Informatics*, *1*(2), 119–130. doi:10.52940/ijici.v1i2.22

Babar, M., Khan, M. S., Ali, F., Imran, M., & Shoaib, M. (2021). Cloudlet computing: Recent advances, taxonomy, and challenges. *IEEE Access : Practical Innovations, Open Solutions*, *9*, 29609–29622. doi:10.1109/ACCESS.2021.3059072

Basic Kit for DragonBoard-410c. (n.d.). 96 Boards. Retrieved October 19, 2023, from https://www.96boards.org/documentation/consumer/dragonboard/dragonboard410c/getting-started/basic-kit/

Cao, K., Liu, Y., Meng, G., & Sun, Q. (2020). An overview on edge computing research. *IEEE Access : Practical Innovations, Open Solutions*, *8*, 85714–85728. doi:10.1109/ACCESS.2020.2991734

DragonBoard 820c Hardware Documentation. (n.d.). 96 Boards. Retrieved October 19, 2023, from https://www.96boards.org/documentation/consumer/dragonboard/dragonboard820c/hardware-docs/

Fernández-Caramés, T. M., Fraga-Lamas, P., Suárez-Albela, M., & Vilar-Montesinos, M. (2018). A fog computing and cloudlet based augmented reality system for the industry 4.0 shipyard. *Sensors (Basel)*, *18*(6), 1798. doi:10.3390/s18061798 PMID:29865266

Galloway, B., & Hancke, G. P. (2012). Introduction to industrial control networks. *IEEE Communications Surveys and Tutorials*, *15*(2), 860–880. doi:10.1109/SURV.2012.071812.00124

Jia, M., Cao, J., & Liang, W. (2015). Optimal cloudlet placement and user to cloudlet allocation in wireless metropolitan area networks. *IEEE Transactions on Cloud Computing*, *5*(4), 725–737. doi:10.1109/TCC.2015.2449834

Kakade, M. S., Karuppiah, A., Mathur, M., Bibekar, M., Basu, G., Raghavan, A., Raghav, A., & Lekshminarayanan, P. (2023). A completely distributed cloudlet framework using 410c SoC. In *2023 14th International Conference on Computing Communication and Networking Technologies* (pp. 1–7). IEEE. doi:10.1109/ICCCNT56998.2023.10307503

Kakade, M. S., Karuppiah, A., Mathur, M., Bibekar, M., Basu, G., Raghavan, A., Raghav, A., Lekshminarayanan, P., & Garang, S. (2023). Multitask scheduling on distributed cloudlet system built using SoCs. *Journal of Systemics, Cybernetics and Informatics*, *21*(1), 61–72. doi:10.54808/JSCI.21.01.61

Ketshabetswe, L. K., Zungeru, A. M., Mangwala, M., Chuma, J. M., & Sigweni, B. (2019). Communication protocols for wireless sensor networks: A survey and comparison. *Heliyon*, *5*(5), e01591. Advance online publication. doi:10.1016/j.heliyon.2019.e01591 PMID:31193432

Khan, B. U. I., Olanrewaju, R. F., Anwar, F., Najeeb, A. R., & Yaacob, M. (2018). A survey on MANETs: Architecture, evolution, applications, security issues and solutions. *Indonesian Journal of Electrical Engineering and Computer Science*, *12*(2), 832–842. doi:10.11591/ijeecs.v12.i2.pp832-842

Khan, W. Z., Ahmed, E., Hakak, S., Yaqoob, I., & Ahmed, A. (2019). Edge computing: A survey. *Future Generation Computer Systems*, *97*, 219–235. doi:10.1016/j.future.2019.02.050

Khanh, T. T., Nguyen, V., Pham, X. Q., & Huh, E. N. (2020). Wi-fi indoor positioning and navigation: A cloudlet-based cloud computing approach. *Human-Centric Computing and Information Sciences*, *10*(1), 1–26. doi:10.1186/s13673-020-00236-8

Le, H. Q., Al-Shatri, H., & Klein, A. (2017). *Efficient resource allocation in mobile-edge computation offloading: Completion time minimization. In 2017 IEEE international symposium on information theory*. IEEE. doi:10.1109/ISIT.2017.8006982

Lou, D., Holler, J., Patel, D., Graf, U., & Gillmore, M. (2021). *The industrial internet of things networking framework*. Industry IoT Consortium Networking Task Group. https://www.iiconsortium.org/wp-content/uploads/sites/2/2022/08/IINF_Update_2022_08_03.pdf

Ma, X., Lin, C., Zhang, H., & Liu, J. (2018). Energy-aware computation offloading of IoT sensors in cloudlet-based mobile edge computing. *Sensors (Basel)*, *18*(6), 1945. doi:10.3390/s18061945 PMID:29914104

Malik, P. K., Sharma, R., Singh, R., Gehlot, A., Satapathy, S. C., Alnumay, W. S., Pelusi, D., Ghosh, U., & Nayak, J. (2021). Industrial internet of things and its applications in industry 4.0: State of the art. *Computer Communications*, *166*, 125–139. doi:10.1016/j.comcom.2020.11.016

Munari, A., Clazzer, F., & Liva, G. (2015). *Multi-receiver aloha systems-a survey and new results. In 2015 IEEE international conference on communication workshop*. IEEE. doi:10.1109/ICCW.2015.7247493

Pham, Q. V., Ruby, R., Fang, F., Nguyen, D. C., Yang, Z., Le, M., Ding, Z., & Hwang, W.-J. (2022). Aerial computing: A new computing paradigm, applications, and challenges. *IEEE Internet of Things Journal*, *9*(11), 8339–8363. doi:10.1109/JIOT.2022.3160691

Prajapati, A. G., Sharma, S. J., & Badgujar, V. S. (2018). *All about cloud: A systematic survey. In 2018 international conference on smart city and emerging technology*. IEEE. doi:10.1109/ICSCET.2018.8537277

Ramasubbareddy, S., & Sasikala, R. (2021). RTTSMCE: A response time aware task scheduling in multi-cloudlet environment. *International Journal of Computers and Applications*, *43*(7), 691–696. doi:10.1080/1206212X.2019.1629098

Ren, J., Zhang, D., He, S., Zhang, Y., & Li, T. (2019). A survey on end-edge-cloud orchestrated network computing paradigms: Transparent computing, mobile edge computing, fog computing, and cloudlet. *ACM Computing Surveys*, *52*(6), 1–36. doi:10.1145/3362031

Shaukat, U., Ahmed, E., Anwar, Z., & Xia, F. (2016). Cloudlet deployment in local wireless networks: Motivation, architectures, applications, and open challenges. *Journal of Network and Computer Applications*, *62*, 18–40. doi:10.1016/j.jnca.2015.11.009

Snapdragon 888 Mobile Hardware Development Kit. (n.d.). Qualcomm Developer Network. Retrieved October 19, 2023, from https://developer.qualcomm.com/hardware/snapdragon-888-hdk

Tawalbeh, L. A., Jararweh, Y., Ababneh, F., & Dosari, F. (2015). Large scale cloudlets deployment for efficient mobile cloud computing. *Journal of Networks*, *10*(1), 70–76. doi:10.4304/jnw.10.01.70-76

Wang, W., Nešić, D., & Postoyan, R. (2015). Emulation-based stabilization of networked control systems implemented on FlexRay. *Automatica*, *59*, 73–83. doi:10.1016/j.automatica.2015.06.010

Watt, S. T., Achanta, S., Abubakari, H., Sagen, E., Korkmaz, Z., & Ahmed, H. (2015). *Understanding and applying precision time protocol. In 2015 Saudi Arabia smart grid*. IEEE. doi:10.1109/SASG.2015.7449285

Xiang, L., & Shaobin, L. (2020). Hybrid cloud networking design based on Openstack architecture. In Journal of Physics: Conference Series (Vol. 1693, No. 1, p. 012011). IOP Publishing. doi:10.1088/1742-6596/1693/1/012011

Manoj Subhash Kakade is an Assistant Professor in the Department of EEE at Work Integrated Learning Programmes Division of BITS Pilani. He holds Masters of Engineering in VLSI & Embedded systems. His area of expertise is Embedded system design, IoT and has been engaged with research and teaching in the field of Embedded systems for the last 15 years. He has spent more than over four years in industry in Embedded hardware domain where in, he was involved in design and development of various embedded systems based product for process, consumer electronics, and healthcare industries. He has published several research articles in international journals and conferences of repute. His research interests include Embedded systems, IoT, Edge Computing.

K. R. Anupama is a Sinor Professor in the Department of EEE at BITS-Pilani, K K Birla Goa Campus, Goa, India. She received her Ph.D. from BITS-Pilani, Pilani Campus, Rajasthan, in 2005, on the topic of GPS based Routing Protocols for Mobile Ad Hoc Networks. She had a research fellowship from Nokia Research Centre, Boston for a period of 4 years (2001-2005). She has developed a series of protocols using predictive source routing for unicast, multicast, geocast and inter-zone routing. She was Co-Investigator for 'Intelligent Water Resource Management Project at BITS, Pilani -Pilani Campus', funded by Microsoft Research (2005-2008) and Principal Investigator for 'Wireless Sensor Networks for Inaccessible Terrains', funded by UGC Major Research Project (2010-2012). She has completed two major projects with GAIL India, based on 'Design and Development of Condition Based Monitoring of Pipelines using Wireless Sensor Networks'. She has more than 100 research publications including, journal papers, and conference papers. Ten of her students have completed their PhDs, two students are in the process of completing in the area of Embedded System, machine learning algorithms and optimization.

Sushil Nayak is a student at BITS Pilani K. K. Birla Goa Campus, Goa, India currently pursuing a bachelor's in engineering degree in Electronics and Communication. He has a keen interest in Network Computing and is pursuing various projects in the same domain.

Swarnab Garang is a student at BITS Pilani K. K. Birla Goa Campus, Goa, India currently pursuing a bachelor's in engineering degree in Electronics and Communication. He has a keen interest in Edge computing, digital twinning and is pursuing various projects in the same domain.