Evaluating Distributed Computation Offloading Scalability for Multiple Robots

Fatima Ayoub
Department of Electronic Engineering
Maynooth University
Maynooth, Ireland
fatima.ayoub.2020@mumail.ie

Rudi Villing
Department of Electronic Engineering
Maynooth University
Maynooth, Ireland
rudi.villing@mu.ie

***Abstract*—Perception in robotics using modern deep learning approaches is often computationally expensive. Cloud robotics and edge robotics provide possible solutions to this. In particular, computation offloading permits compute-constrained robots to offload compute tasks from the robot itself to more capable servers (the remote brain). However, the scalability of computation offloading for robots sharing a WiFi network has typically not been considered. In this work, we investigated the scalability of offloading to the cloud and edge for deployments of multiple robots in real-world settings and network conditions. We interpret typical network performance metrics such as latency, throughput, and packet loss in terms of their effect on robot computations to help to understand this question. To characterize the problem further, we introduce three different static offloading profiles based on the sensing capabilities of currently available robot platforms and examined these in a number of experiments on both simulated and physical wireless networks. Our results show that WiFi network capacity is much less than advertised when subjected to offloading data traffic and indicates limitations on the number of robots that can concurrently use computation offloading in a given space and the amount of data that they can transfer. This has significant implications for the dense deployment of robots that depend on computation offloading to meet their required service level.**

Keywords—Robotics, Cloud robotics, Computational offloading, Wireless network performance.

I. INTRODUCTION

Today, autonomous robots can use external computational resources to enhance their performance and capabilities. Robots mostly utilize cloud resources to perform computationally expensive tasks. Autonomous mobile robots must balance computing capability with power consumption (which affects battery autonomy time) and cost. For this reason, there is often less onboard storage space and less capable computational resources on a robot than on a desktop or laptop computing system. Many commercially available robots do not have an onboard graphics processing unit (GPU), or the GPU has limited resources. To overcome the restrictions that this more constrained computing environment creates, a robot may offload some of its computational tasks to a remote system (the remote brain) at the edge or in the cloud, where resources can be considered essentially limitless [1]. This cloud enabled robotics approach [2] is attractive for a number of reasons, among them that it can enable lower cost and less computationally powerful robots to perform complex tasks such as care assistance and socially aware guiding or delivery.

Depending on the application, computation offloading may require substantial amounts of data to be transferred to the remote system. The largest data is usually vision data used for navigation, perception, manipulation control, and recognition among other tasks. Unlike video for human consumption, robot vision data is usually not suitable for high compression due to latency constraints, tolerance of frame loss, and potential sensitivity to artefacts that humans overlook.

In static offloading, the decision regarding what computations to offload is taken at design time, whereas in dynamic offloading, the decision is taken at run time depending on factors such as power consumption, network usage and status, among others [3]–[5]. Since the dynamic decision could be that offloading should not be done at a given time, dynamic offloading makes sense only if the computation to be offloaded is optional or if the robot can execute the computation itself (but prefers not to). If the computations are required for the robot's purpose and it cannot execute them itself, then static offloading is required. Since larger deployments of robots in settings such as healthcare will likely include lower cost and less computationally capable robots among others, static offloading is the strategy that we analyse in this work.

Many environments today that could benefit from autonomous mobile robots, including healthcare environments, typically have very few such robots. This is a consequence of both the cost and challenging nature of robotics in dynamic settings among other factors. Computation offloading provides the ability to both reduce cost and increase the robot's capabilities through sophisticated (remote) processing and this allows robots to be deployed more widely and applied to a greater range of tasks. In

This publication has emanated from research conducted with the financial support of Science Foundation Ireland (SFI) under Grant Number 18/CRT/6222.

healthcare and care home settings for example robots can help by participating in social interactions, delivering medicine, recording patient parameters, responding to nurse calls, and cleaning among other tasks, reducing the workload of the nurses and staff [6].

As autonomous mobile robots become more ubiquitous then, it is appropriate to investigate how well the currently installed communications infrastructure might support increasing numbers of robots and specifically those using computation offloading. Safe and reliable operation is an important consideration for all robot deployments, and when using computation offloading, the ability of the network to guarantee the required communication quality of service is an essential element of that. Moreover, when robots are both mobile and autonomous, it is entirely possible for multiple robots to enter a single space (for example a hospital ward or a care home common area) either because their task requires them to visit that space or they are simply passing through enroute to another destination. This means that the peak density of robots in a given area may be larger than the average.

To our knowledge the scalability of dense deployments of robots offloading to remote servers has not been subjected to study. Since the traffic pattern of offloading robots (dominated by upload) is different to typical traffic patterns considered in network design and since WiFi networks can degrade substantially when congested, we investigate (a) how many offloading robots a given wireless access point (AP) can sustain and (b) if static offloading is a viable strategy for distributed computation going forward. Furthermore, this study focuses on the ability of existing installed networks to handle the offloading traffic, since redesigning and reinstalling the local network just to handle additionl robots is often not justifiable or viable in the settings we consider.

As a target environment, we consider care settings such as hospitals and care homes. In these settings mobile autonomous service robots and human interactive robots are expected to play a greater role and current examples of such robots include the PAL Robotics TIAGo [7], Ari [8], and Kompai robotics Kompai [9]. Such robots rely heavily on processing data from both vision and audio sensors for tasks including person detection and recognition, socially aware guiding and navigation, and effortless, responsive, human interactions including taking routine measurements (such as temperature). All these tasks could potentially benefit from computation offloading.

To examine this question in detail, this paper makes three main contributions. We propose three different static offloading profiles grounded in the capabilities of available robots and consideration of typical tasks they might undertake. We design and execute experiments to evaluate how these offloading profiles are handled for configurations of 1 to 13 robots, in simulation and in two different physical networks. Finally, we interpret the results and their implications for larger deployments of robots using static offloading.

The remainder of this paper is organized as follows. Section II, briefly surveys related work on cloud robotics and offloading. We detail our offloading traffic profiles and the metrics we use in Section III. Section IV describes the experiments in detail, Section V the results and their interpretation for robot offloading, and thereafter, our conclusions.

II. Related Work

Distributed computation offloading is a key element of cloud enabled robotics. Cloud enabled robotics differs from many other cloud-based applications due to the large amount of data associated with robot sensing and perception and the real-time nature of the system (usually requiring all data communication and processing to be completed in a few tens of milliseconds). A general review of cloud robotics can be found in [10] including information about applications and challenges. There are two main techniques for offloading computation from the robot to the cloud, static and dynamic offloading. Static offloading refers to the pre-determination of which tasks will be executed on the robot and which tasks will be executed on the cloud before the robot starts performing the tasks. In contrast, dynamic offloading involves making the offloading decision at runtime based on the current robot and cloud resource utilization.

The use of static offloading in robotics systems enables better task planning and scheduling by making offloading decisions in advance. This approach can lead to more predictable task execution, thereby achieving low latency. Consequently, static offloading is frequently employed to achieve optimal performance in robotics systems. Authors in [5] considered a distributed computation offloading architecture for cloud robotics to improve the computational capacity and flexibility of robotic system. Overall, the performance of cloud robotics is improved with this technique: using offloading the robot can execute up to 30 frames per second while without offloading robot can only process up to 19 frames per second for an object tracking algorithm. To ensure smooth and efficient offloading, the architecture requires low latency communication between the edge, fog, and cloud layers which depends on the ability of the network to maintain this low latency.

Another study presents a promising approach for cooperative computation offloading for robot swarms in cloud robotics [4]. The method has potential applications in search and rescue, environmental monitoring, and industrial automation. The proposed QoS-aware framework can aid in the efficient and dependable offloading of computations in such systems. The main requirement of the proposed method is low-latency and dependable communication between robots and the cloud, which can be challenging in dynamic and unpredictable network conditions, particularly for mobile robots that move around the environment. The use of cloud computing to empower robots comes with a fundamental tradeoff between safety and performance. Many robotics applications require real-time computation, and any lag can result in failures with potential safety consequences.

Computation offloading has been extensively investigated in the context of Internet of Things (IoT) devices and mobile devices (e.g. [11], [12]). As the IoT domain tends to work with smaller data sizes, research related to smartphones and mobile

devices is more relevant here, particularly when applications related to face detection or other vision tasks are considered.

From an architectural perspective, computations can be offloaded to the cloud [13], [14], to a mix of local and internet cloud resource [15], to a cloudlet at the network edge (e.g. using a socially aware offloading algorithm [16]), or to peer devices [17]. There is also the question of whether to offload complete computation tasks or to subdivide the task so that parts may be offloaded [13].

Most research has focused on dynamic offloading methods, where the decision about which computations to offload is made at run time. Factors to consider when making such decisions include energy consumption (e.g. [13], [14], [18]), dynamic connectivity properties (e.g. [17], [18]), and maintaining or improving the quality of service (QoS) of the application offloading computation [18]–[20]. Typically, dynamic offloading has been formulated as an optimization problem [17], [19] but it can also be approached using evolutionary algorithms (e.g. [19]), or economic and game theoretic approaches [16], [22]. The latter are particularly relevant when robots or mobile devices must compete for limited offloading resources (whether the limit is due to computation resources or network connectivity limitations). Recently deep reinforcement learning approaches have been applied to learn the offloading decision-making process [23], [24].

Dynamic computation offloading relies on the key assumption that there are some criteria under which offloading should not be done. If not offloaded, the relevant computations must be optional, or they must be handled by the robot itself and the tasks requiring them may need to tolerate long delays due to slower computation. Dynamic offloading also adds complexity to the software architecture since computational capabilities must be replicated on the robot and in the cloud.

The static offloading strategy is presented much less frequently (e.g. [5]), but it has the benefit of simpler designs than dynamic offloading. Moreover, it is suitable for lower cost robots which do not have the computational capability to perform state of the art perception tasks. It does place higher demands on the network, but in exchange it provides a consistent and predictable QoS (important in care settings), unlike systems using dynamic offloading which must either be able to perform local computations as fast as offloaded computations, or must suffer degraded QoS when computations are performed locally. By definition, the QoS of robots using dynamic offloading must be lower than those using static offloading given the same onboard computing capability since sometimes they will decide not to offload. On the other hand, while offloading is taking place, the two strategies have identical network requirements and for all these reasons we examine the ability of WiFi networks to handle the static offloading strategy in this paper.

III. Evaluation Profiles and Metrics

A. Real-time Robot Characteristics

Robots generally operate within the framework of a high-level sense-think-act loop. This loop has periodic deadlines by which it must process sensor input, make decisions, and actuate its motors for the robot to behave safely, responsively, and correctly. It is well known that delays in control systems lead to degraded performance which may include slower response times, lack of stability, or oscillations in the response and this can affect the perception and trust in such robot systems. A frame rate of between 15 and 30 frames per second (fps) is the generally accepted requirement for safe and responsive performance for locomotion at humanlike speeds (e.g. [25]), for certain manipulation tasks, tasks requiring fast reaction times, or human robot interaction that depends on detecting brief eye movements or micro-expressions.

Latency must also be considered. In a robot system, there is usually a delay of up to one frame time in both sensing and actuation. Therefore it is important to minimize any additional latency introduced by offloading. For this work, we consider that the additional offloading latency should be less than 66.6 ms, that is a 1-frame delay at 15 fps.

In general, the TCP protocol is not suitable for real-time data transmission over wireless channels, since any lost packets must be retransmitted, and this can introduce substantial latency. For this reason, some form of UDP is typically used as the underlying transport between robots and the cloud [26]. As UDP is inherently unreliable, packet loss may render entire frames unusable. Application layer forward error or erasure correction (FEC) may mitigate this in some situations, but it introduces complexity and does not appear to have been used in the cloud robotics literature. For this reason, our experiments do not use FEC.

B. Offloading traffic profiles

This study assumes static offloading of the most computationally intensive tasks which are almost always associated with vision (and to a lesser extent audio) data. Other typical sensors in an autonomous robot, such as a 2D laser scanner, sonar, inertial management unit, and wheel rotation and joint sensors, do not produce significant amounts of data in comparison [26]. Therefore, we focus on the data produced by vision-based sensors (RGB and RGB-D cameras) and audio sensing (microphones). As an offloading robot generates highly asymmetric data flows, where the upload data (from robot to remote server) is about 150 times the download data [26], we ignore the download data in this work.

To reduce the vision data size, we assume that each camera frame is compressed using JPEG compression with an average compression ratio of 20:1. With JPEG, each frame is compressed independently, and it has relatively low computation requirements making it suitable for implementation on a compute constrained robot. For convenience, we assume identical compression for depth images. Under these conditions, vision data is generated with an approximately constant bit rate, R_{vision}, that can be calculated as

$$R_{vision} = w \times h \times n_{bpp} \times r_{frame} \times c \qquad (1)$$

where w and h are the image width and height in pixels, n_{bpp} is the number of bits per pixel, r_{frame}, is the frame rate (in frames per second), and c is the compression factor (the inverse of the compression ratio).

We assume that audio data is primarily focused on human interactive tasks and sampled at 16000 samples/second. As audio compression introduces additional latency, we do not use compressed audio data and we do not use voice/silence detection. Hence, the constant audio data bit rate is

$$R_{audio} = f_s \times n_{bps} \times n_{channels} \quad (2)$$

where f_s is the audio sampling frequency, n_{bps} is the number of bits per sample (assumed to be 16), and $n_{channels}$ is the number of audio channels.

For the purpose of evaluation, we proposed three different robot profiles, small, medium, and large, differentiated by the amount of vision and audio data to be communicated and based on the sensing capabilities of existing robots. The high profile is based on the vision and audio capabilities of a higher end robot such as the PAL Ari which includes multiple cameras and a microphone array for better audio detection and localisation in noisy environments. The medium and low profiles are representative of robots with more constrained hardware.TABLE I. details the specifications of all three profiles. The bit rates for vision and audio data are calculated according to (1) and (2).

TABLE I. OFFLOADING PROFILE SPECIFICATIONS

	Small Profile	**Medium Profile**	**High Profile**
Vision[a]	RGB camera: 320×240 @ 15 fps	RGB-D camera: 640×480 @ 15 fps	RGB camera: 1280×720 @ 30 fps RGB-D camera: 640×480 @ 30 fps
Audio[b]	1 channel	2 channel	4 channel
Vision bit rate (Mbps)	1.382	7.373	47.923
Audio bit rate (Mbps)	0.256	0.512	1.024
Total bit rate (Mbps)	1.638	7.885	48.947

a. Fps is frames/second, RGB is 24 bits/pixel, RGB-D is 24+8 bits/pixel

b. Each audio channel is 16000 samples/second

Throughput measures rate at which the data required for computation can be offloaded from the robot to the server. If the average throughput per robot is less than the total bit rate

C. Metrics

In this study we measure three metrics, for the upload flow only: throughput, packet loss, and latency. All metrics are measured at the remote server and therefore incorporate the contributions of the robot, the server, and the network between them. Throughput measures rate at which the data required for computation can be offloaded from the robot to the server. If the average throughput per robot is less than the total bit rate listed TABLE I. , then the robot will not be able to maintain the frame rate specified.

Packet loss presents an insidious problem for offloading data that includes vision data in particular because data will span multiple packets (e.g. 45 packets for a single frame of medium profile data given 1472 bytes of data per packet). In the absence of FEC, the loss of just one packet may render a camera image unusable. Therefore, packet loss rates as low as 10%, 2.2%, or 0.7% could corrupt all frames in the low, medium, and high profile data streams if packet loss were to occur at regular intervals. Packet loss is often somewhat bursty, however, so this represents worst case behaviour.

Communication latency is an important component of overall latency when offloading computation. Excessive latency will cause any real-time tasks that depend on offloading to degrade or fail. It should be as small as possible and not larger than 66.66 ms (as explained in subsection A above). In our experiments, which do not use external networks, we expect the network latency to be dominated by latency associated with the WiFi network to which the robots connect.

IV. EXPERIMENTS

To evaluate the scalability of computation offloading for dense robot deployments, we executed two experiments: one in a simulated environment and one in a physical environment. In both experiments all three offloading profiles from. TABLE I. were tested, and the number of offloading robots was varied from 1 to 13 to monitor the effects of increased WiFi traffic. Each offloading profile was evaluated as a separate condition and in each such condition, all robots used the same offloading profile. Each experimental condition was tested for 30 seconds and raw data were logged throughout for later analysis.

In both the simulation and physical environments all robots communicated with a single wireless AP using 802.11ac (the WiFi version used by our enterprise WiFi network and our robot stations). We used a maximum transmission unit size of 1500 bytes resulting in a payload data size of 1472 bytes for both experiments.

A. Simulation environment

The simulation experiment was conducted using ns-3 [27]. Multiple robot stations were connected to an 802.11ac WiFi network (using Minstrel-HT for rate control) via a single wireless AP. The AP was bridged to a CSMA network (approximating an ethernet LAN) and the remote server was also connected to this. All robot stations generated data at the constant bit rate according to the offloading profile. Data was communicated using UDP to the server which consumed data as fast as it arrived.

Fig. 1 shows the placement of robot stations in the physical environment (section B). The placement in the simulation environment was similar except that there were no simulated walls or rooms and station positions were quantised to the nearest meter. The simulation experiment was conducted before the physical experiment and was expected to represent ideal conditions as there was no competing traffic in the WiFi network and no interference from adjacent WiFi channels and no in-band interference.

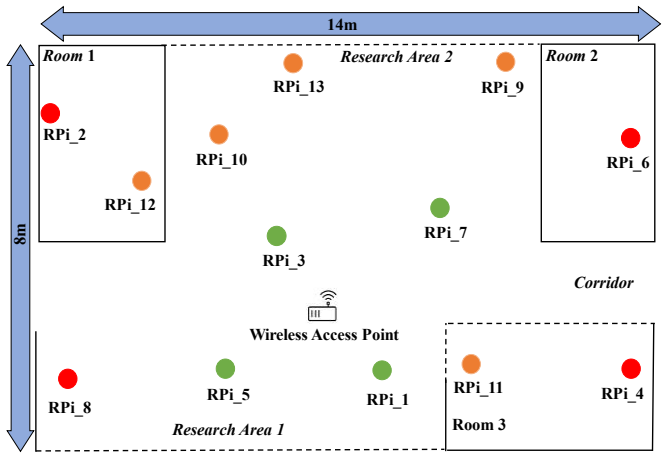

Fig 1. The physical environment used for experiments. Colour coding indicates distance from the wireless AP. RPi_1 and RPi_13 are raspberry pi 4 devices while the rest of the devices are raspberry pi 3B+. Dotted lines represent the glass walls and doors.

B. *Physical environment*

The physical environment corresponds to a working research area in a university incorporating open plan research desks, laboratory space, social space, and a corridor. This area features concrete walls, glass walls, glass doors and wooden doors, all of which might be found in care settings that are the target environment we consider. The environment was set up as shown in Fig. 1. The maximum distance from the wireless AP to any station is approximately 8 meters and the minimum distance is approximately 1 meter. Offloading robots were emulated using Raspberry Pi 3B+ and Raspberry Pi 4 devices [28] since both of these natively support 802.11ac and can produce and send data at data rates corresponding to each of the offloading profiles. All Raspberry Pi devices were configured to use 5 GHz WiFi communications and used their on-board antennae.

We also evaluated two different WiFi networks (private and enterprise) and two different times of day (afternoon and evening). The private WiFi network used a standalone Asus RT-AC66U Gigabit Router supporting up to 1300 Mbps and only carried traffic from the robot stations. The enterprise WiFi network was configured to provide a private Service Set Identifier (SSID) dedicated to the robot stations and was accessed via an Aruba AP-315 supporting up to 1733 Mbps and 4 spatial streams. Unlike the private network, the enterprise network also carried normal university traffic and for this reason all conditions of the enterprise network measurements were repeated on three different days.

The afternoon condition in the enterprise network arguably reflects the most relevant real-world scenario. Although the robots used a dedicated SSID, robots and other network users shared the same physical network infrastructure such as access points. During the afternoon sessions, multiple users accessed the network causing varying load and traffic conditions. To counter this, we conducted three readings for each experiment and reported the average measures as already described.

The server device was a Dell G5 15 5590 laptop (i7-9750, 16 GB memory, Gigabit ethernet). For the enterprise network condition, this was connected to the university's ethernet network while for the private network condition it was connected directly to the ethernet port of the private wireless AP. Prior to the start of each experimental run, the robot station and offloading server clocks were synchronized using the network time protocol (NTP) [29] with the offloading server acting as the NTP server.

Offloading itself was implemented by python code running on robot stations (Raspberry Pi devices) and the server device (Dell G5). The offloading client produced data in discrete offloading frames. Each offloading frame comprised data equivalent to one frame of vision data (from each of the cameras considered in the profile) and the corresponding time window of audio data. Offloading frames were produced 15 or 30 times per second according to the offloading profile and data for a frame was then sent as quickly as possible, fragmented as needed across multiple UDP packets. Each packet contained a sequence number and NTP synchronised timestamp, and these were used by the offloading server to detect lost packets and estimate communications latency. Parallel-SSH [30] was used to start the robot offloading client scripts on each robot station (Raspberry Pi) in parallel so that devices would send offloading traffic concurrently.

The purpose of synchronizing with NTP and sending data at the vision data frame rate is to closely model the implementation of real-time processing in robots. In a real robot implementation the challenge would be to offload frames as soon as possible after the sensing data was available and to receive the offloading server response as soon as possible thereafter, particularly where the response feeds into closed loop control algorithms for navigation, motion, and actuation. A further complication would be the asynchronous nature of responses with a high likelihood that two or more offloading frames may have been sent from a robot before the first response is received. In this study we avoided this complication by focusing on the offloading frames only and not dealing with responses (which have a minimal data size in comparison).

V. Results

The received signal strength at the robot stations varied between -65 dBm and -30 dBm in the simulation environment and between -61 dBm to -40 dBm in the physical environment. Fig. 2 shows the average throughput per robot station in both the simulation and physical environment experiments.

For the small profile, the throughput is never less than 94% of the nominal throughput suggesting that robots could largely operate at the required frame rate. The medium profile, throughput was more variable but remained above 81% of nominal with 9 robots or fewer. High profile, throughput dropped to 66% of nominal once 3 robots were offloading and continued to drop as the number of robots increased. In this case, frame rates would likely not be acceptable for robot applications requiring fast response or continuous control. Although we expected that the physical networks would consistently perform better in the evening (when there was less traffic from people at work than during the afternoon),

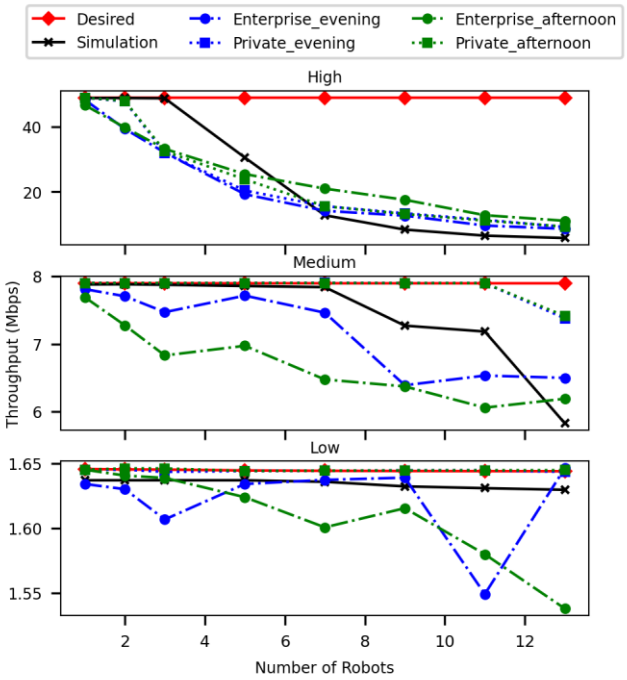

Fig 2. Average throughput per robot station on an 802.11ac WiFi network in different conditions. The desired line shows the expected throughput if no packets are lost.

there are no consistently better outcomes between times of the day or between the private and enterprise WiFi networks. It is worth commenting that the peak aggregate throughput achieved for multiple robots was much less than the advertised maximum: 125 Mbps (9.6% of maximum) on the private network and 158 Mbps (9.1% of maximum) on the enterprise network. Though perhaps surprising, this is likely a side effect of having robot stations at different distances from the AP (rather than at the ideal distance) and it is exacerbated by congestion effects as the aggregate offloading traffic increases.

In addition to throughput, packet loss must be considered. Moreover, in the absence of FEC, loss of a packet causes loss of a frame. To gain further insight, Fig. 3 shows packet loss suffered by robots that were at close, far, and intermediate distances from the AP (focusing on the enterprise network afternoon condition only as the most relevant case). Although there are differences none of the distances is consistently associated with the lowest or highest packet loss. Nevertheless, with 5 robots or fewer, far robots experience the highest packet loss.

For robot applications it is more useful to look at frames lost (as a result of packet loss), as shown in Fig. 4. Frame loss increases as the attempted aggregate offloading data rate increases, though not linearly. It is also clear that the worst frame loss experienced by an individual robot can be significantly worse than the average frame loss experienced by the robots. In all profiles, some individual robots experienced 100% frame loss in at least some experimental trials. The data show that very significant frame loss of more

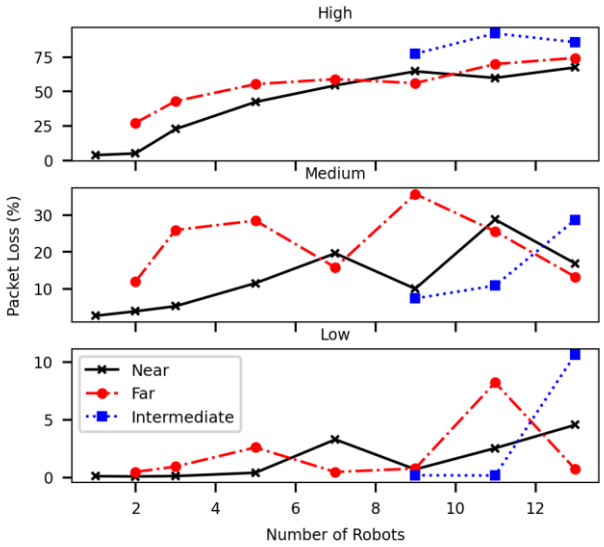

Fig 3. Average packet loss (percentage) per station for multiple robot stations sending via a single 802.11ac access point under the experimental conditions.

than 9 out of 15 fps can occur with average packet loss as low as 2.5%.

The final metric we considered was latency. In simulation, the latency is less than 4 ms for the low profile, but exceeds our threshold of 66.6 ms with 7 robots in the medium profile and 5 robots in the high profile. In physical tests with real network equipment the results were different. With the private network, the average latency per robot was quite usable at less than 12 ms for all configurations of the low profile, up to 11 robots on the medium profile, and just one robot in the high profile. All other configurations, however, had average latencies ranging from 158 to 245 ms which are likely too long to be usable.

The enterprise network exhibited more variable latencies. In the low profile average latencies were 16 ms up to 11 robot stations, but standard deviations of 80 to 277 ms for 7 robots or more, indicated that some robots were already suffering unusable latencies at this point. Results for the medium profile were similar. Average latencies for the high profile were a little longer (up to 44ms) and standard deviations exceeded

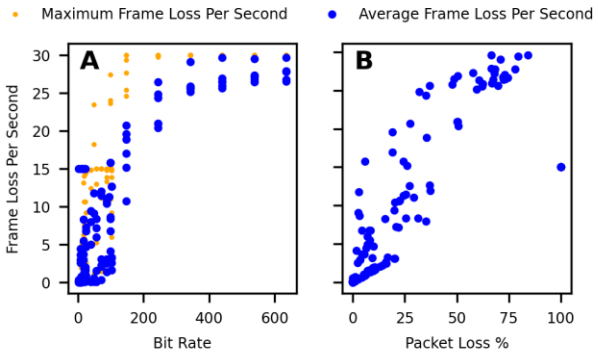

Fig 4. Frame loss rate per robot in relation to the attempted aggregate bit rate from multiple robots (A) and the packet loss percentage per robot (B).

100 ms in configurations with 5 robots or more. In summary, these results show that WiFi latency can become a problem for at least some robots once there are between 5 and 11 robots using a single AP, depending on the amount of offloading data transmission being attempted.

Our experimental findings pertain to static clients connected to a single access point and was designed to be a representative snapshot in time of the positioning of multiple mobile robots. Using mobile robots would have introduced more variability into the results. Sometimes the results may potentially have been better, but in many situations they may have been worse, for example in scenarios where all robots are positioned far from the access point.

In scenarios involving two or more access points, performance improvement is possible by distributing clients across access points. These scenarios can potentially enhance the overall system performance in comparison to the single access point setup reported here. However, we contend that using mobile robots the single access point scenario is still the most important bottleneck to understand. Unless specific techniques are employed to limit the number of robots moving into the physical area covered by any given access point, there will always (and perhaps frequently) be situations where a number of robots all use a single access point and this is reason we chose to explore this scenario in our work.

VI. Conclusion

In this work we examined the scalability of computation offloading (using the static offloading strategy strategy and three different offloading traffic profiles) for deployments of 1 to 13 robots within range of a single wireless AP in a realistic 802.11ac WiFi environment. The purpose of this work is to help designers of robot systems understand if computation offloading is viable in realistic network environments and to be aware of the difference between predications based on simulation or controlled laboratory settings relative less controlled real-world settings.

Our results show that WiFi network capacity is far less than might be expected (based on advertised AP data rates) and that issues of increased packet loss (and frame loss) and packet latency occur quickly as the attempted aggregate bit rate of offloading data increases. While the problem of frame rate degradation due to frame loss might be mitigated with FEC, this would further increase the attempted aggregate data rate from multiple robots and worsen congestion related issues.

In general, our results show that static computation offloading is a viable strategy only in limited circumstances. It is not viable with high profile robots, while its viability for low and medium profile robots is limited by the robot application's tolerance for dropped frames and latency. Although dynamic computation offloading appears to offer an alternative to static offloading, it is not directly comparable.

In particular, if QoS is not to degrade when dynamic computation offloading cannot take place, then the robot must be able to perform the computations that it would have offloaded onboard and with the same completion deadlines. We conclude that WiFi capacity has significant implications for the deployment of robots that depend on computation offloading to achieve their required quality of service levels and that further studies with computation offloading schemes in real-world networks are warranted to understand what service levels can be reliably achieved.

References

[1] A. Rahman, J. Jin, A. Cricenti, A. Rahman, and D. Yuan, "A cloud robotics framework of optimal task offloading for smart city applications," in *2016 IEEE Global Communications Conference (GLOBECOM)*, IEEE, 2016, pp. 1–7.

[2] J. Kuffner, "Cloud-enabled humanoid robots," in *Humanoid Robots (Humanoids), 2010 10th IEEE-RAS International Conference on, Nashville TN, United States, Dec.*, 2010.

[3] A. Rahman, J. Jin, A. L. Cricenti, A. Rahman, and A. Kulkarni, "Communication-aware cloud robotic task offloading with on-demand mobility for smart factory maintenance," *IEEE Trans Industr Inform*, vol. 15, no. 5, pp. 2500–2511, 2018.

[4] Z. Hong, H. Huang, S. Guo, W. Chen, and Z. Zheng, "QoS-aware cooperative computation offloading for robot swarms in cloud robotics," *IEEE Trans Veh Technol*, vol. 68, no. 4, pp. 4027–4041, 2019.

[5] R. Chaari, O. Cheikhrouhou, A. Koubâa, H. Youssef, and H. Hmam, "Towards a distributed computation offloading architecture for cloud robotics," in *2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC)*, IEEE, 2019, pp. 434–441.

[6] K. Obayashi and S. Masuyama, "Pilot and feasibility study on elderly support services using communicative robots and monitoring sensors integrated with cloud robotics," *Clin Ther*, vol. 42, no. 2, pp. 364–371, 2020.

[7] J. Pages, L. Marchionni, and F. Ferro, "Tiago: the modular robot that adapts to different research needs," in *International workshop on robot modularity, IROS*, 2016.

[8] S. Cooper, A. Di Fava, C. Vivas, L. Marchionni, and F. Ferro, "ARI: The social assistive robot and companion," in *2020 29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, IEEE, 2020, pp. 745–751.

[9] P. Caleb-Solly, S. Dogramadzi, C. A. G. J. Huijnen, and H. van den Heuvel, "Exploiting ability for human adaptation to facilitate improved human-robot interaction and acceptance," *The Information Society*, vol. 34, no. 3, pp. 153–165, 2018.

[10] B. Kehoe, S. Patil, P. Abbeel, and K. Goldberg, "A survey of research on cloud robotics and automation," *IEEE Transactions on automation science and engineering*, vol. 12, no. 2, pp. 398–409, 2015.

[11] K. Akherfi, M. Gerndt, and H. Harroud, "Mobile cloud computing for computation offloading: Issues and challenges," *Applied computing and informatics*, vol. 14, no. 1, pp. 1–16, 2018.

[12] P. Pandey, D. Pompili, and J. Yi, "Dynamic collaboration between networked robots and clouds in resource-constrained environments," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 2, pp. 471–480, 2015.

[13] X. Ma, Y. Zhao, L. Zhang, H. Wang, and L. Peng, "When mobile terminals meet the cloud: computation offloading as the bridge," *IEEE Netw*, vol. 27, no. 5, pp. 28–33, 2013.

[14] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang, "Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading," in *2012 Proceedings IEEE Infocom*, IEEE, 2012, pp. 945–953.

[15] T. Zhao, S. Zhou, X. Guo, Y. Zhao, and Z. Niu, "A cooperative scheduling scheme of local cloud and internet cloud for delay-aware mobile cloud computing," in *2015 IEEE Globecom Workshops (GC Wkshps)*, IEEE, 2015, pp. 1–6.

[16] L. Tang and X. Chen, "An efficient social-aware computation offloading algorithm in cloudlet system," in *2016 IEEE Global Communications Conference (GLOBECOM)*, IEEE, 2016, pp. 1–6.

[17] C. You and K. Huang, “Mobile cooperative computing: Energy-efficient peer-to-peer computation offloading,” *arXiv preprint arXiv*, vol. 1704, 2017.
[18] S. Barbarossa, S. Sardellitti, and P. Di Lorenzo, “Computation offloading for mobile cloud computing based on wide cross-layer optimization,” in *2013 Future Network & Mobile Summit*, IEEE, 2013, pp. 1–10.
[19] A. Rahman, J. Jin, A. Cricenti, A. Rahman, and M. Panda, “Motion and connectivity aware offloading in cloud robotics via genetic algorithm,” in *GLOBECOM 2017-2017 IEEE Global Communications Conference*, IEEE, 2017, pp. 1–6.
[20] W. Chen, Y. Yaguchi, K. Naruse, Y. Watanobe, and K. Nakamura, “QoS-aware robotic streaming workflow allocation in cloud robotics systems,” *IEEE Trans Serv Comput*, vol. 14, no. 2, pp. 544–558, 2018.
[21] Y. Zhai, B. Ding, P. Zhang, and J. Luo, “Cloudroid swarm: A qos-aware framework for multirobot cooperation offloading,” *Wirel Commun Mob Comput*, vol. 2021, 2021.
[22] K. Xie *et al.*, “Distributed multi-dimensional pricing for efficient application offloading in mobile cloud computing,” *IEEE Trans Serv Comput*, vol. 12, no. 6, pp. 925–940, 2016.
[23] S. Chinchali *et al.*, “Network offloading policies for cloud robotics: a learning-based approach,” *Auton Robots*, vol. 45, no. 7, pp. 997–1012, 2021.
[24] M. Penmetcha and B.-C. Min, “A deep reinforcement learning-based dynamic computational offloading method for cloud robotics,” *IEEE Access*, vol. 9, pp. 60265–60279, 2021.
[25] Pal Robots, “ARI: The new generation of AI powered robots.” http://pal-robotics.com/wp-content/uploads/2020/01/ARI-the-new-generation-of-AI-powered-robots-1.pdf
[26] R. C. Mello, W. M. Scheidegger, M. C. Múnera, C. A. Cifuentes, M. R. N. Ribeiro, and A. Frizera-Neto, “The PoundCloud framework for ROS-based cloud robotics: Case studies on autonomous navigation and human–robot interaction,” *Rob Auton Syst*, vol. 150, p. 103981, 2022.
[27] NS-3 Consortium, “ns-3 documentation.” https:www.nsnam.org
[28] “Raspberry Pi 4 .” https://www.raspberrypi.com/products/raspberry-pi-4-model-b/ (accessed Aug. 11, 2022).
[29] D. L. Mills, “Network time protocol (NTP),” 1985.
[30] “Parallel-SSH.” https://parallel-ssh.org/ (accessed Oct. 24, 2022).