Dynamically Allocating Network Resources for Teleoperation Applications using Intent-based Inference

Dung N. H. Pham, Mohammed S. Elbamby, and Viet Duc Le

***Abstract*— Due to the limitation of network resource capacity, maintaining high-quality communication for large-scale teleoperation applications, in terms of ultra-low latency and ultra-reliability requirements, is challenging. Most network infrastructures are not designed for such extreme demands compared to other applications. However, if the future behavior of the teleoperation application is captured, we can allocate network resources more efficiently. Using the intent-based inference mechanism to predict future behaviors, the goal of this research is to deliver a low latency environment while guaranteeing high-reliability requirements. Considering a group of remote-controlled robots working in an industrial environment, the intent is interpreted as the target destination to which the robot is heading. This research proposes a Recurrent Neural Network (RNN) solution for network systems, which predicts the target destination and allocates network resources based on the predicted result. The objective is to minimize the unproductive time of robots, which is the duration robots stay idle. To quantify the performance, the proposed solution is tested under a comprehensive simulator that allows multiple actors and data streams to work simultaneously. Results show that the RNN integration for robotic applications can significantly improve network performance under different scenarios. However, care should be taken when applying since not all network systems benefit positively and equally from proposed solutions.**

***Index Terms*— Robot Destination Prediction, Recurrent Neural Network, Scheduling and Matching, Network Simulation**

I. Introduction

THE use of remote-controlled robots, also known as teleoperation [1], has been applied widely across multiple fields, both in the research and industrial environment. From massive space discovery expeditions [2] to complex microsurgery operations [3], teleoperation has been a significant assistance in many hazardous or inaccessible circumstances. Moreover, with the recent advancements in sensor technologies, computer technologies, artificial intelligence, and network architecture systems, the scale of applicable teleoperation has reached a level that could not be achieved in the past.

Dung N. H. Pham is with the University of Twente, Enschede, The Netherlands and Nokia Bell Labs, Espoo, Finland (e-mail:phamnguyenghoangdung@student.utwente.nl).

Mohammed S. Elbamby is with Nokia Bell Labs, Espoo, Finland (e-mail: mohammed.elbamby@nokia-bell-labs.com).

Viet Duc Le is with the University of Twente, Enschede, The Netherlands (e-mail: v.d.le@utwente.nl).

Although applicable in multiple areas, large-scale teleoperation still faces many challenges. The most demanding one is maintaining the high-quality communication connection between the control station and multiple working machines [4]. Without guaranteeing a high-quality connection, the remote operation is likely to be inaccurate and ineffective. Here, the term high-quality communication can be interpreted as low latency and high reliability. Latency is the required waiting time to transfer an information packet from a source endpoint device to a destination endpoint device [5]. With teleoperation, the challenge is even more complicated since the requirements for high-quality communication are demanding. The waiting time of receiving incoming request signals, processing and sending responses is sometimes unacceptable for certain applications. With reliability, it is defined as the percentage of information packets that are successfully delivered from a source device to a target device given a time frame [5]. For most teleoperation applications, the latency requirement should be lower than 5 to 10 milliseconds [5]. In some extreme cases, for example, the robotic motion control application, the latency must not exceed 1 millisecond [5]. With reliability, the usual requirement falls from 99.9999% to 99.999999% [5]. Fulfilling these ultra-low latency and/or ultra-reliability requirements for all devices in a network system where multiple devices sending and receiving information packets in multiple datatypes is extremely challenging. Therefore, a viable solution can be sending the response to destination devices before it is even requested by predicting the future behavior of the device. However, developing this solution is application-dependent. Since the behavior of the local application is unknown to the network, the network can hardly infer the needed response based on the behavior of the application by itself. Moreover, the difference in technical design of network systems and local applications, for example, the architecture or coding languages, makes local application predictions undesirable to be handled by the network. Therefore, for each local application, an intent-based inference solution that can infer future behavior should be developed to minimize the burden of processing information for the network.

Despite being unsuitable, the term intent-based has been mentioned at the network layer before. An intent is defined as the future operational objectives of an application. The intent-based network system is a technology that aims to automatically formulate and orchestrate the network based on the intent of local applications or services [6]. To achieve that,

the local application requires certain resources that the network is expected to provide, without indicating how to achieve them [7]. Integrating intent-interpretation solutions at the network layer allows for a higher level of abstract objectives, thus handling a wider range of local applications. However, the interpretation of intent, both for humans and applications, depends not only on behaviors but also on the environment and shared knowledge in the surrounding community. As such, no network system can analyze the meaning of intent without the ability to comprehend the context, knowing the possible options to interact with the scenario, and being aware of the potential consequence for each option [8]. To fully understand intent meanings, the network should leave the interpretation process to local applications since applications contain the needed context information. By doing so, intents can be realized more efficiently by inferring in advance, for example, predicting the future behaviors of an application. For teleoperation, there are thousands of different applications, each with separated operational behaviors and purposes. Interpreting the application behavior meaning depends heavily on the context of the situation. Therefore, instead of understanding teleoperation intents in general, this research topic will focus on interpreting a specific teleoperation intent and introducing a comprehensive simulator to measure the effectiveness of applying intent-based inference mechanisms in network resources allocation.

For the specific teleoperation intents, this research topic chooses moving robots as the study case. By predicting robot's next destination using robot trajectories in different scenarios, this research aims to fulfill the reliability and ensure the latency requirement in data transmission for the mentioned application. Using Recurrent Neural Network (RNN) as the backbone of the machine learning model, an accuracy of 99.8% for destination prediction problem is achieved. Moreover, this research emphasizes the practicality of intent-based prediction model by designing and simulating an intent-based wireless teleoperation system to optimize content delivery rate and minimize latency. The designed system can ideally self-configure the wireless connection between base stations and moving robots to maintain reliability and latency requirements. In conclusion, the main research question that we focus on is whether the network benefits from the predictive machine learning model by knowing the meaning of future behavior from local applications.

Concisely, the main contribution of this research can be summarized as follow:

- Utilizing a machine learning architecture that handles spatial and temporal data of a moving robot to predict its future destination to minimize the operational waiting time.
- Implementing a real-time matching and scheduling algorithm between different devices and control stations to maintain latency and reliability requirements.
- A comprehensive simulation for multiple devices and multiple data streams is designed to analyze the performance of the proposed solution.

This research is made under the supervision of University of Twente in Netherlands; Nokia Bell Labs and Aalto University in Finland.

II. Related Works

Recently, the topic of network resource allocation for teleoperation applications has been the focus area of many research. In this work, we explore generally the network resource allocation topic for robotic devices in Section II-A. Then, following few interesting research, we want to further improve the network performance by introducing intent-based inference mechanism in Section II-B. Section II-C is dedicated to describe a use case of intent-based inference, the future destination prediction which this research focuses on, with several related works.

A. Resource allocation for robotic devices

In recent years, robot control has become an exciting term in smart industrial environments. To control a robot, signals have to be sent, wired or wireless, via a network system. In a network system where multiple robots are controlled, optimizing resource allocation for all robots becomes extremely challenging [9]. This includes limitations in robots and communication resources [9], the demanding data stream processing requirements, the trade-off between computation and communications, security and safety qualification, etc. [10]. Addressing these problems require multiple processes, which the work [10] gathers into three categories: resource pooling, task offloading and task scheduling. Resource pooling indicates the reservation and distribution of resources for multiple applications in the network [10], which is highlighted as on-demand arrangement [11]–[13] and reserved arrangement [14], [15]. Task offloading refers to the action of moving computation tasks from local computers to cloud processors [16]. Noticeable works that can be mentioned are [17]–[19]. Task scheduling refers to the order task requests should be handled, as presented in [14], [15], [17], [20].

Over the wide area of resource allocation topics, this research focuses on task scheduling by further studying some above-mentioned works. The first one [17] focuses on optimizing latency and reliability constraints by designing a caching system that can reduce computational time. The work indicates that, by proactively choosing popular tasks and arranging connected devices based on task preference, the computational latency can be reduced significantly. The other work [20] demonstrates a network system design that is applicable specifically for the virtual reality application. In that work, they design a three-layer architecture, with the virtual reality application acting as the edge device, the mmWave small base station as the middle layer transmitting data to the edge devices and remote cloud, with the cloud server acting as the computation and optimization process handle. Although these works illustrate a great foundation for the mentioned problem, several limitations are needed to be addressed. First, compared to static devices, remote control robots continuously move from one place to another, thus requiring other intent interpretations to be created. Second, addressing optimization problems with multiple requirements

from numerous application types can raise issues with the methods mentioned in these papers since they are designed to function with only one type of application. In this research, we aim to activate an intent-based inference mechanism as a dynamic component which is integrated in an update version of the task scheduling methods from [20] and [17].

B. Intent Utilization

The term intent is not a new definition. In real life, intent is referred to as an objective that is being described by a person, capturing the final goal or state that needs to be achieved without identifying how to achieve it [8]. Although being understood effortlessly by humans, intent was hardly adopted in technology fields due to its abstraction in interpreting the intended meaning. The human brain can process information abstractly and ambiguously. A machine, however, understands a context or an event by logical input, thus limiting its ability to generalize abstract concepts, like an idea or a term. However, with the advancement of artificial intelligence, we can now express abstract terms by logical input. As such, an open door is made to interpret intents of machines.

For teleoperation applications, multiple works have been done to capture future behaviors and their meaning, each with different context information. The works from [21], [22], and [23] develop different types of task-parameterized generative models that support controlling a robot device remotely. The work introduces a hidden semi-Markov model to capture the intent of the user, thus assisting the user by independently executing tasks by a shared control mechanism. Another work from Yoojin Oh et al. predicts the intent of a robot arm user by identifying the targeted object that the user wants to graph [24]. Using a perception module, the model specifies the objective from sequential tasks, then the robot autonomously conducts a retrieving motion using trajectory optimization.

Although applicable in many scenarios, not many works focus on quantifying the impact of intent analysis in a large-scale network system. Obviously, intent interpretation can benefit the operation individually. However, further study needs to be conducted to evaluate the improved performance at the multi-agent level. This research focuses on the above study by employing destination prediction as an intent-based inference use case and evaluate the overall performance at the network system level.

C. Future Destination Prediction

The topic of future destination prediction has received significant attention recently, due to its impactful contribution to many location-based services, such as traffic-flow prediction, weather forecast and network resources optimization. With the current popularity of mobile devices and wireless networks, an enormous amount of human movement data is produced every day. Using this big location-based data, researchers are putting much effort into predicting the human next location prediction. As a result, many works have been accomplished to demonstrate the robustness of next location prediction systems.

There have been several attempts to apply machine learning-based methods to predict the future destination of human behaviors [25]. A noticeable approach from Yujie W. et al. combines multiple machine learning techniques – support vector machines (SVM), decision tree (DT) and logical regression (LR) to detect unlicensed taxi services based on large-scale vehicle mobility data [26]. The pattern embedding model proposed by M. Chen et al. integrates traffic trajectory data with other types of information (object, location, and time), outputting a low-dimensional latent space, which can be applied with other future location predictions [27]. In the field of deep learning-based methods, much more research has been produced using different techniques. Various types of deep learning models have been applied, for example, Convolutional Neural Network (CNN) [28], [29], Graph-based Neural Network (GNN) [30], [31], etc. However, most of the works focus on employing RNN to extract the time-dependent characteristics of positional data [32]–[38]. The main idea of these studies is to obtain meaningful information from human dynamic spatial and temporal moving data in different scenarios, combining with different context features, such as social media interaction, to predict the next destination of a decision-making agent. Even though these studies achieve appreciable results in destination prediction, they all focus on investigating the dynamic of human behaviors. However, one major difference between human and robot behavior is the dynamicity of the movement. Human behavior is less predictable since the ability to change the current action and intent of a person is more versatile. On the other hand, mobile teleoperation devices are usually set up in specific industrial environments, where the intent is pre-determined and the moving pattern is more noticeable. Applying the above-mentioned models may overfit the scenario, thus reducing the overall performance of the system.

To the best of my knowledge, no prior work studied the topic of robot future intent prediction based on trajectories data. Although there are several works studying the robot movement prediction problem [39], [40], their main goal is to predict the future path of a robot agent in a certain time frame. Therefore, the applicable abilities of these works for future destination prediction are needed to be revised, especially in a high obstacle-density environment. For these reasons, this research will focus on developing a predictive machine learning model that can leverage the future destination prediction ability using positional robot data.

III. Problem

In this session, we design a use case where remote-controlled robots and machines request data transmission simultaneously. A moving robot is demanded to carry an item from one machine to another. While traveling, the robot continuously receives controlling data packets from the edge server. On the other hand, machines require data packet only after the robot delivers the item at the correct position. When machine data packet transmission is completed, the machine starts executing tasks with the delivered item. After the machine finishes its job, the moving robot carries the item to the next machine.

The above scenario establishes multiple network requirements to maintain a stable workflow. Under these require-

ments, the goal of this use case is to minimize the total robot idle duration, denoted as robot waiting time, from the moment the robot arrives targeted machine to the point when machine data packet transmission is completed. To address this problem, this research proposes a centralized predictive solution, which aims to predict targeted machines in advance, thus sending machine packet requests before robots arrive at the machine. With this approach, we divide the problem into two parts, predicting robot destinations and minimizing robot waiting time under certain network constraints. While the predictive problem is handled locally by each robot, the optimization problem is managed by the cloud server globally. Section III-A describes the working scenario as a whole. Section III-B and III-C respectively explain the robot destination prediction and optimizing robot waiting time problem mathematically.

A. Scenario explanation

To deliver the research objective, a wireless system that focuses on interpreting the intent of a specific teleoperation application needs to be designed comprehensibly. Since intent inference requires context information, which can only be inferred ambiguously, it is critical that the designed scenario contains visible and understandable contexts. This section will describe the working scenario that requires the wireless system as a viable solution.

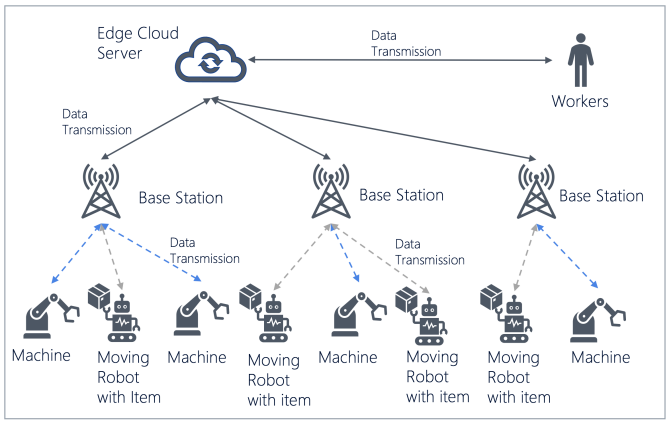


Fig. 1: Network Layers and Packet Transmissions

In an smart factory domain, a component based machinary process with multiple machines are deployed at fixed positions [41]. Each machine is designed to complete a specific work piece of an item, which is represent as a task. A item is completed when all required tasks is finished. A task can be finished using a pre-determined machine but the item has to be carried from one machine to another. To carry the item, the worker must control a remote-controlled robot and drive it to the machine. Each item needs a list of tasks that is unknown to the network system. After finishing a task, the worker immediately controls the robot heading to the next machine to execute the next task.

Consider a set of robots R and a set of machines M initially distributed over the working area. The network system consists of an edge cloud server multiple base stations $b \in B$ at fixed locations distributed in the same area. These base stations are connected to the edge cloud server that processes the controlling data between robots and workers. Additionally, base stations also serve as a communication device between fixed machines and the edge cloud server, thus transmitting the processing data to the fixed machine before the machine executes tasks (as described in Figure 1). The network system applies the time-slotted system approach with fixed time slots to handle data transmission.

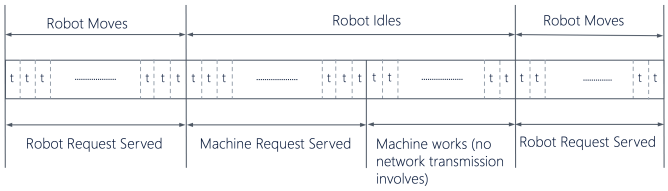


Fig. 2: Working Process of a Robot
This image illustrates the working process of a robots. However, in the simulation, multiple robots are operated at the same time, which the duration is different in each phase.

In this scenario, the network system needs to handle two distinctive applications, each with a different datatype. The first application is the remote-controlled robot, which transmits controlling data. To handle the robot remotely, the controlling data need to be received continuously. This datatype includes visual data, represented by a video stream, so that workers are aware of the surrounding environment. The second datatype is fixed machine processing data. Machines are designed to execute certain tasks, thus limited in storage and processing power. As such, we utilize the edge cloud processing power to handle processing data by transmitting required data packets from the server to dedicated machines. When a robot arrives at the targeted machine, the machine needs to load the information about the task that is going to be executed from the cloud server. After finishing loading processing data, the machine can start handling the assigned task. When the task is finished, the robot carries the item to the next machine. Figure 2 describes this process.

Handling multiple data streams in a wireless network requires complex algorithms. At a certain time instant, the network system needs to decide which request it needs to serve and how to handle it. Moreover, each data stream has its own requirements. Guaranteeing those requirements is a prerequisite for a stable operation. Under these requirements, we aim to minimize *total robot idle duration*, denoted as robot waiting time, from the moment the robot arrives targeted machine to the point when machine data packet transmission is completed.

To address this optimization problem, this research introduces a multi-stage scheme incorporated with a predictive model. Notice that the machine can only send requests when moving robots arrive. If the machine can send requests before that moment, then request can be registered sooner, thus give network system more time to arrange and serve its waiting request list. However, inaccurate predictions can

lead to unwanted packet registration, hence slowing down the system. To formulate this idea, a predictive problem is defined in Section III-B as finding which machine is the next destination of the current moving robot. The later stage of the scheme addresses robot waiting time optimization problem by handling registered request list. Section III-C defines the cost function and related constraints for all data streams in the environment.

Although the data to be transferred includes downstream and upstream datalink, in this research, we want to focus solely on measuring the optimization performance of downstream data transfer when applying intent-based prediction. Designing and measuring system performance for both downstream and upstream data is desirable but the complexity of the simulator will increase substantially. Therefore, the scope of this research only includes downstream datalink. However, the upstream datalink can be added in the future using the same research method.

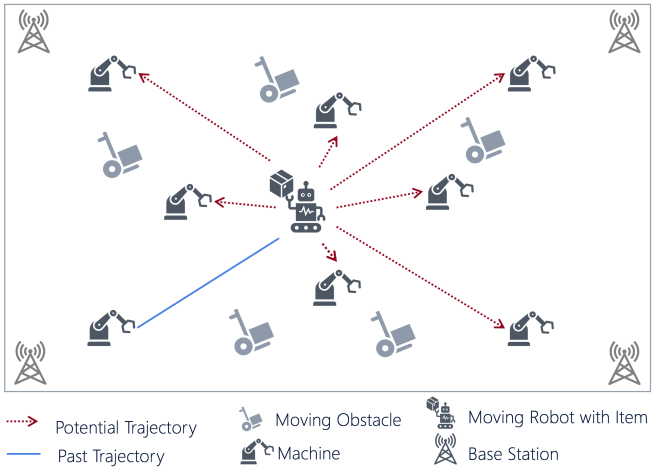


Fig. 3: Working Space of a Robot
This image illustrates the working space and a robots. However, in the simulation, multiple robots are operated in this working space.

B. Destination Prediction

Let the set of fixed machines be denoted as:

$$M = [m_1, m_2, ...m_n].$$

Each machine is positioned at a separated location, with $[x_{m_i}, y_{m_i}]$ denoted as latitude and longitude of machine m_i, in the indoor working environment. The past trajectory of a controllable robot at run i is described as:

$$r_i = \left[r_{i_1}, r_{i_2}, ...r_{i_{T-1}}, r_{i_T}\right]$$

where T is the moving time from the starting point to the destination of a run. The run indicates the moving direction of the robot, which contains the information of the destination machine. The positional input of the robot at each time frame is a 2-dimensional sequence, indicating the positional latitude and the longitude $r_{i_t} = [x_{i_t}, y_{i_t}]$.

This research goal is to emphasize the network efficiency optimization aspect. Therefore, to simplify the intent prediction problem, we assume each robot operates independently from the others. As such, by building one prediction model, we can apply to multiple robots with linear scaling in computational time complexity. However, care should be taken that, in a real industrial environment, multiple robots may cooperate in executing a task, or a machine can only handle a certain amount of robot tasks at a time, thus introducing potential dependencies between robot operations. For such cases, more sophisticate collations between robots have to be defined.

C. Waiting Time Mitigation Problem

Remote-controlled robots require control data packet continuously while moving. Thus, at a certain time instant, robots request to receive the controlling packet from the server. Under this request time-to-live, the nearby base station needs to send the corresponding controlling packet to the robot. If this transmission does not finish within the request time-to-live, the packet is marked as dropped.

For the processing machine, we design the restriction to be less stringent. When the moving robot finishes its run and approaches the targeted machine, the machine starts sending a request to the nearby base station. The base station then sends request to the edge server for processing. The edge server queues the request until the network has available resources to execute the request. This request type does not have time-to-live constraints. In other words, the robot must wait at the targeted machine until the data packet is fully transmitted. After that, the machine can execute its task, then robot can move to the next targeted machine.

Since there are two different data stream running simultaneously, the teleoperation data stream for remote robot control and the task processing data stream for the fixed machine, the optimization problem is divided into two smaller sections.

*1) **Mitigate Machine Data Transmission**:* At the time instant t, for each machine $m_i \in M$ connected to a base station $b_j \in B$, the transmission data rate when operating in time-division duplex is computed as:

$$D_{mb}(t) = W \log_2(1 + \frac{P_m h_{mb}}{I_m + N_0}) \quad (1)$$

where $D_{mb}(t)$ is the data rate from machine m to base station b (bps), W is the channel bandwidth, P_m is the transmit power of machine m, h_{mb} is the channel gain between machine m and base station b, I_r is the interfering power from other base stations at the machine m, and N_0 is the total noise power over the channel with bandwidth W.

The packet delivery time that a base station $b_j \in B$ can provide to a machine $m_i \in M$ is:

$$L_{mb}(t) = x_{mb}(t) \cdot \frac{S_M}{D_{mb}(t)}$$

with S_M is a machine packet data size and $x_{mb}(t)$ is the binary variable that takes the value 1 or 0 whether the base station b_j is serving the machine m_i at the current time instant.

Although all base stations can provide services to machine m_i at an instant, at most one connection between machine

m_i and all base stations is established. In other words, at a moment, only one base station serves the request from machine m_i, which is denoted mathematically as:

$$\sum_{b \in B} x_{mb}(t) = 1.$$

Consequently, the delivery time of transmission packet from a machine $m_i \in M$ connected all base stations in B is:

$$L_m(t) = \sum_{b \in B} L_{mb}(t). \quad (2)$$

Each machine needs to connect with at most one base station to send and receive processing data. The machine also prefers to be connected with the station which can provide the lowest delivery time. As such, the main goal is to minimize the robot waiting time as much as possible. To achieve this goal, we incorporate a predictive model to predict the next targeted machine before the robot reach the machine and send processing data request packet. A correct prediction can reduce the waiting time by sending the processing data ahead of time. However, a wrong prediction forces the system to send mismatch data, thus increasing the overall waiting time of the system. Therefore, for each task, the robot waiting time is computed by:

$$T_m(t) = L_m(t) - Z \cdot H \cdot A_m + Z \cdot H \cdot (1 - A_m), \forall m \in M \quad (3)$$

with $T_m(t)$ is the total waiting time of a run, Z is a binary variable indicates if a predictive model is used or not, H is the amount of time the system decides to run the prediction before the robot arrives at the destination, and A_m takes the value of 1 or 0 indicating the rightfulness in prediction. For each task, the minimum waiting time can only be reduced to 0, thus a lower constraint of $T_{mb} \geq 0$ needs to be applied.

Overall, this optimization problem is defined as minimizing the total waiting time of the system, which can be formulated as:

$$\min \sum_{m \in M} T_m(t). \quad (4)$$

*2) **Maintain Robot Packet Reliability**:* At the time instant t, for each robot $r_k \in R$ and a base station $b_j \in B$, the transmission data rate is similarly computed by:

$$D_{rb}(t) = W \log_2(1 + \frac{P_r h_{rb}}{I_b + N_0}). \quad (5)$$

The packet delivery time that a base station $b_j \in B$ can provide to a machine $r_k \in R$ is:

$$L_{rb}(t) = x_{rb}(t) \cdot \frac{S_R}{D_{rb}(t)}$$

with S_R is a robot packet data size and $x_{rb}(t)$ is the binary variable that takes the value 1 or 0 whether the base station b_j is serving the robot r_k at the current time instant.

Similarly, we can denote the packet delivery time for robot r_k at instant t as:

$$L_r(t) = \sum_{b \in B} L_{rb}(t) \quad (6)$$

with

$$\sum_{b \in B} x_{rb}(t) = 1.$$

Operating a robot remotely requires the robot to maintain connection with at most one base station. Intuitively, the robot prefers to be connected with the station which can provide the lowest delivery time. Moreover, the connection delivery time has to be below certain threshold for the robot control application to be executed correctly [5]. Thus, the reliability of the system is defined as the percentage of time the connection delivery time goes above the requirement threshold [5]. Based on the delivery time and reliability requirement that needed to operate the remote robot control application successfully, the constraint for robot reliability is defined as:

$$\Pr(L_r(t) \geq \alpha) \leq \epsilon, \forall r \in R \quad (7)$$

where α is the maximum packet time-to-live requirement and $1 - \epsilon$ is the minimum reliability requirement.

*3) **Problem Combination**:* At a time instant, each base station can handle a request separately. Therefore, the number of connected devices (including remote-controlled robots and machines) are equal or smaller than the number of base station. As such:

$$\sum_{r \in R} \sum_{b \in B} x_{rb} + \sum_{m \in M} \sum_{b \in B} x_{mb} \leq |S|. \quad (8)$$

Overall, the optimization problem is (4) is:

$$\min \sum_{m \in M} T_m(t)$$

under the constraints of the inequality (2), (3), (6), (7), (8),

$$T_m(t) = L_m(t) - (Z \cdot H \cdot (2A_m - 1)) \geq 0, \forall m \in M$$
$$\Pr(L_r(t) \geq \alpha) \leq \epsilon, \forall r \in R$$
$$\sum_{r \in R} \sum_{b \in B} x_{rb} + \sum_{m \in M} \sum_{b \in B} x_{mb} \leq |S|$$
$$L_r(t) = \sum_{b \in B} x_{rb}(t) \cdot \frac{S_R}{D_{rb}(t)}$$
$$L_m(t) = \sum_{b \in B} x_{mb}(t) \cdot \frac{S_M}{D_{mb}(t)}$$

IV. Approach

This section presents a multi-stage machine learning integration scheme to address the above problem. Section IV-A describes characteristics of the dataset that we use throughout this research. Section IV-B presents the machine learning model architecture for the destination prediction problem. The waiting time mitigation problem is handled in Section IV-C using a matching and scheduling scheme.

A. Dataset

This research aims to design an applicable teleoperation network system that delivers machine learning as the solution to improve operational performance. To achieve this goal, acquiring relevant datasets with context information is critical. However, such dataset is hardly obtainable from

the industry for several reasons. First, the dataset has to contain interpretable context information to recognize patterns and make predictions using machine learning. Most available public teleoperation datasets are either non-pattern or contain incomprehensive context information. Second, highly relevant industrial datasets are confidential. Most private organizations do not public their dataset, especially one related to the manufacturing environment. For these reasons, this research develops the solution based on several simulated datasets containing self-explanatory context information designed to be as dynamic and realistic as possible.

The dataset for the machine learning solution is simulated using the model predictive control (MPC) solution. In the robotic area, MPC is an optimization-based control method, which depends on a process model to predict the future behavior of a controlled agent in a finite time horizon [42]. By solving an optimization problem that reflects the control goals and specifications under certain constraints, MPC is guaranteed to deliver close-to-optimal actions. An essential characteristic of MPC is that the solution is optimized repeatedly at each time step. This makes MPC a viable solution to handle the dynamic environment, thus mimicking human behaviors better than other global planners, which can only deal with a non-changing environment. To utilize these qualities of MPC in the path planning method, we gather the dataset from the implementation of the work [42], in which the teleoperation agents are the drones with Parrot Bebop 2 quadrotor, and Forces Pro [43] as the problem solver.

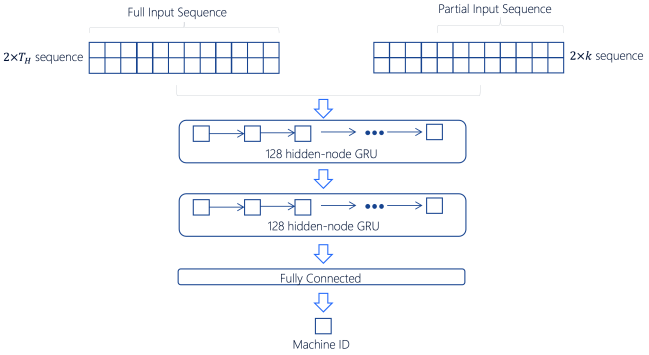


Fig. 4: Time sequence to Categorical Neural Network architecture. Input is a $2 \times n$ list represent trajectory data. Output is a scalar represents the Machine ID.

B. Prediction Model

Although the dataset contains full paths from the starting point to the destination for each robot run, we use a subset of the full path in the training phase, indicating the prediction should be made before the remote-controlled robot reaches a machine. The incomplete positional data:

$$r_i = \left[r_{i_1}, r_{i_2}, ...r_{i_{T_H}}\right]$$

is extracted from the complete run, with $T_H = T - H$ as the prediction moment. The H value is the prediction horizon counting backward from when robots arrive at machines. The larger H value is, the sooner the prediction is made. Newly acquired data is fed into the prediction model as the input data.

Given this problem as predicting a deterministic machine m from a set of machine M, it can be referred to as a multiclass machine learning problem. Since the input data is a time-dependent positional sequence, it is appropriate to consider RNN as the predictive model. RNN has proven its effectiveness in handling sequential data. In this work, we consider Gated Recurrent Unit (GRU), a variant of RNN, as the primary layer since GRU has shown to be suitable for small datasets [44].

This work utilizes the model architecture idea from [20], by combining 2 GRU layers. The input data: $\mathbf{X} \in \mathbb{R}^{N \times T_H \times d}$ (where N is the number of runs, T_H is the time length of each run, d is the input data dimensions) is fed into a layer GRU layer with 128 hidden nodes, followed by another same-sized GRU layer. The final output of the sequence is then fed to a fully connected linear layer (as described in Figure 4).

Model output: $\mathbf{y} \in \mathbb{R}^N$ indicates the predicted target machine m'. The model is trained under cross-entropy loss which compares the difference between the predicted machine m' with the target machine m.

We train and validate the predictive model with two different input datasets. The first set contains positional vectors from the starting point to the prediction point, which is referred as *full input sequence*. The second set only includes the later positional vectors of the run, which are the most critical information:

$$r_i = \left[r_{i_{T_H-k}}, r_{i_{T_H-k+1}}, ...r_{i_{T-H}}\right]$$

where k is the extracted number of time instants, denoted. We denote it as *partial input sequence*. By training only partial data, we want to quantify the impact of early positional input on the prediction, thus choosing the best process for the predictive model.

C. Scheduling and Matching Scheme

The delivery time mitigation introduced in the last session is a combinational problem. The cost function of the problem is non-convex and the constraints contain probabilistic inequations. The reason behind non-convexity is due to the data rate term in Equation (1) and (5) being the function of the interference from other connected devices. Since the number of constraints grows linearly with the number of devices (including robots and machines), finding the optimal solution is computationally complex [45].

We apply Markov's inequation to transform the probabilistic constraints (7) into tractable linear constraints [45]. Markov's inequation states that:

$$\Pr(\{.\} \geq \alpha) \leq \frac{\mathbb{E}\{.\}}{\alpha}.$$

If the upper bounds of Markov inequation satisfy the reliability threshold, then the probability constraints should follow the same pattern:

$$\frac{\mathbb{E}\{.\}}{\alpha} \leq \epsilon \Rightarrow \mathbb{E}\{.\} \leq \epsilon \cdot \alpha.$$

Hence, the constraints can be expressed as:

$$\begin{aligned} &\mathbb{E}\{L_r(t)\} \leq \epsilon \cdot \alpha, \forall r \in R \\ \Rightarrow &\sum_{b \in B} x_{rb}(t) \frac{S_R}{\mathbb{E}\{D_{rb}(t)\}} \leq \epsilon \cdot \alpha, \forall r \in R \end{aligned} \tag{9}$$

The newly acquired constraints indicate that to reach the desired reliability, a maximum value of delivery time $\mathbb{E}\{L_r(t)\}$ is allowed to be queued from the admitted request moments. Since being affected by interference from other connected devices, which is unknown to the network system before choosing the served devices at the current time instant, the estimated data rate D_{rb} has to be computed empirically by past obtained data rate. The estimated value at each base station b for each robot r and machine m is measured using a time-average estimation method [17]

$$\mathbb{E}\{D_{rb}(t)\} = v(t) \cdot D_{rb}(t-1) + (1 - v(t)) \cdot \overline{D}_{rb}(t-1) \tag{10}$$

where $v(t)$ is the learning rate at the time r, and $\overline{D}_{rb}(t-1)$ is the time-average estimation for data rate from the last time instant.

Following the above analysis, this research proposes a joint scheduling and matching scheme to solve the optimization problem. The first stage is registering newly arrived requests based on predictive model. Then, the system decides which request types should be prioritized at that moment. Finally, a matching algorithm is introduced between serverd devices (robots and machines) and base stations in the final stage.

*1) **Register Device Request**:* Our system handles robot requests and machine requests separately. For robot requests, when a new request arrives, network system registers the request in the robot request queue. For machine requests, we apply Machine Learning prediction model to handle new requests in advance. When a robot arrives at prediction horizon (denoted by H), we predict which machine is the targeted one. A new request with the robot and the predicted machine is registered in the network system. When the current robot arrives at the targeted machine, we can confirm whether our prediction is correct. If the prediction is accurate, no action is taken. However, if the prediction is wrong, the prediction request is de-registered and replaced by the correct one. The frequency of wrong predictions depends on the accuracy of the machine learning model. Algorithm 1 depicts the process in detail.

*2) **Scheduling Served Devices**:* The primary purpose of the scheduling process is to decide when the network system should prioritize the robot's packet and vice versa. To determine what packet type should be prioritized, the network system needs to compute the estimated delivery time of the following instant and adjust its decision based on the acquired information. Our primary assumption concerns whether the system can guarantee reliability if it reserves all the resources from the next instant until the end of the robot packet time-to-live. From the Equation (9), if the estimated delivered time exceeds a certain threshold, there is a high chance that the reliability is not guaranteed. From that point of view, we compute all robots' total estimated delivery time.

$$\sum_{r}^{R} L_r(t+1) = \sum_{r}^{R} L_r(t) - |B|, \forall r \in R. \tag{11}$$

If the robot reliability is not maintained in the next instant,

$$\sum_{r}^{R} L_r(t+1) \geq \epsilon \cdot \alpha, \tag{12}$$

the network should prioritize processing the robot packet in the current one. After deciding, selected request packets are put in the scheduled device list, represented by Algorithm 2.

Algorithm 1 Register Request Packets

Require: List of robots, List of machines
Ensure: Robot request queue R, Machine request queue M
for Each time instant **do**
  if New request from robot r **then**
    Add robot to queue R
  if Prediction for machine m is made for robot r **then**
    Add machine request for robot r to queue M
  if Robot r arrives at machine m **then**
    if Request in M is not for robot r **then**
      Remove wrong predicted machine request for robot r' from M
      Add new machine request for robot r to M
    Register robot arriving time

Algorithm 2 Scheduling served devices

Require: Robot request queue R, Machine request queue M
Ensure: Scheduled Device List D
for $r \in R$ **do**
  Compute estimated data rate $D_{rB}(t)$ from Eq. (10)
  Compute estimated delivery time $L_{rB}(t)$ from Eq. (9)
Compute total estimated delivery time for all robots $L_R(t+1)$ from Eq. (11)
if $L_R(t+1) \geq \epsilon \cdot \alpha$ from Eq. (12) **then**
  Put unfinished all robot requests to D
  if $|D| \leq |B|$ **then**
    Put unfinished machine requests to S one by one until $|D| = |B|$
else
  Put all unfinished machine request to D
  if $|D| \leq |B|$ **then**
    Put unfinished robot requests to D one by one until $|D| = |B|$

*3) **Matching Served Device with Base Station**:* The next step is to distribute devices to nearby base stations that solve the optimization constraints in problem (4). This problem can be referred to as a matching game between base stations $b \in B$ and served devices $d \in D$. At each instant, a device in the scheduled list is matched to a base station that aims to minimize the packet delivery time. The matching theory has been introduced in the wireless network system, where each

player in one set tries to match with players from the opposite set [46]. The newly formed matching pair is based on the preference from the base station set B and device set D. The preference list of each member in D and B rank members of the opposite set. For example, a device d prefers b over b if d ranks b higher than b in u preference list.

Given the two disjoint sets of base station B and scheduled devices D, we define a matching pair as a one-to-one mapping Γ from base station set B to devices set D. The result of matching pairs is the set of Γ, which is a member of all possible combinations from setting B and D such that for each $b \in B$ and each $d \in D$:

- For each $d \in D$, $\Gamma(d) \in B \cup \{d\}$, where $\Gamma(d) = \varnothing$ means device is waiting to be matched, and $\Gamma(d) = d$ means that all base stations have been connected and there is not a match reserve to the current device.
- For each $b \in B$, $\Gamma(b) \in D \cup \{b\}$, where $\Gamma(b) = \varnothing$ means base station is waiting to be matched, and $\Gamma(b) = b$ means there are no devices left that needs to be serve in the scheduled list.
- $|\Gamma(d)| = 1$, $|\Gamma(b)| = 1$
- $\Gamma(d) = b \Leftrightarrow \Gamma(b) = d$

From the point of a matching game, the inequality (8) satisfies the one-to-one mapping condition. Moreover, by applying the matching theory, we can propose the preference utility function that matches the requirement of our problem. Reflecting on two constraints, one is to guarantee the reliability threshold and one is to minimize the delivery rate, we propose two different utility functions.

1. `Data Rate Utility Function.` This approach follows the Greedy Method, which prioritizes the connection that provides the highest possible data rate. In other words, this utility function seeks the matching that minimizes the delivery time of remote-controlled robots and machines without concerning the deadline of request packets. The preference utility function follows Equation (10):

$$Pref_{bd} = Pref_{db} = \mathbb{E}\{D_{db}(t)\}, \forall d \in D, \forall b \in B. \quad (13)$$

2. `Delivery Time Utility Function.` This approach tries to meet the reliability requirements by prioritizing the packet that has the most urgent deadline. Unlike the first utility function, this one considers the deadline of request packets, thus emphasizing system reliability. However, since the data rate is not considered, this approach is noticeably slower than the first. The preference utility function follows Equation (9):

$$Pref_{bd} = Pref_{db} = \frac{1}{\alpha - \mathbb{E}\{L_{db}(t)\}}, \forall d \in D, \forall b \in B. \quad (14)$$

The work [46] shows that, with the one-to-one matching game, at least one two-sided stable matching exists that satisfies the matching game requirements. Considering a match $|\Gamma(d)| = b$. If existing d' and b' that b prefers d' than d, and d prefer b' than b, then $\Gamma(d) = b'$ and $\Gamma(d') = b$ are blocking pairs of match $\Gamma(d) = b$. A two-sided stable matching exists if and only if there is no blocking pair between (b, d).

To guarantee all matches are stable matches, the work [46] refers deferred acceptance (DA) algorithm with polynomial time complexity for the one-to-one matching problem. Algorithm 3 describes DA algorithm in detail.

Algorithm 3 Matching devices with base stations

Require: Scheduled Device List D, Base station list B
Ensure: A stable match Γ
  Initialize unmatch base stations and devices.
  $\forall b \in B, d \in D, \Gamma(b) = \varnothing, \Gamma(d) = \varnothing$
  Initialize preference list for devices from Eq. (13) or (14).
  Initialize preference list for base station from Eq. (13) or (14).
  while $\exists d, Pref_{db} > Pref_{db'}, \Gamma(d) = b' \vee \Gamma(d) = \varnothing$ **do**
    d proposes to its most preferred b
    if $\Gamma(b) = \varnothing$ **then**
      Device d is accepted. $\Gamma(d) = b, \Gamma(b) = d$
    else if $\Gamma(b) = d'$ **then**
      if $Pref_{bd'} > Pref_{bd}$ **then**
        Rejected device d.
        Remove base station b from device d preference list.
      else if $Pref_{bd} > Pref_{bd'}$ **then**
        Accept device. d $\Gamma(d) = b, \Gamma(b) = d$.
        Unmatch device d'. $\Gamma(d') = \varnothing$.
        Remove base station b from device d' preference list.
  for $d \in D, \Gamma(d) = \varnothing$ **do** $\Gamma(d) = d$

V. Performance Evaluation

This part illustrates results for our multi-stage scheme. We first introduce dataset parameters in Section V-A, then continue presenting and explaining collected results. Section V-B shows the machine learning model performance individually, while Section V-C describes the results from network simulation in multiple cases, including non-machine learning and machine learning approaches.

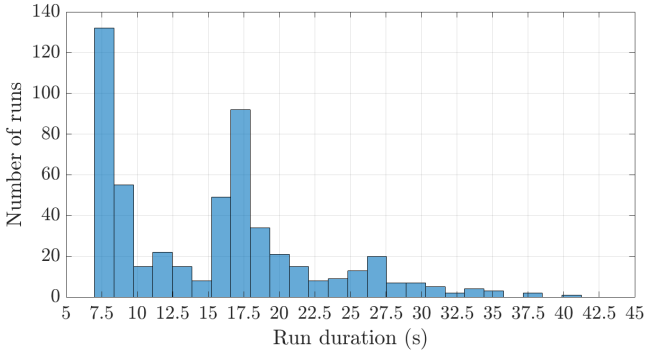


Fig. 5: Robot's Runs Duration

A. Data Analysis

The MPC model generates a dataset that contains 8 fixed machines at different positions in an environment size of $15 \times 15m^2$ (as depicted in Figure 3). 20 remote-controlled robots are working simultaneously in this environment. Robots

have size $0.5 \times 0.5m^2$. A robot run starting point is randomized from 1 of 8 possible machine positions. Each robot's target is 1 of the other 7 machines. A robot is considered to have reached its goal when the Euclidian distance between the robot and the targeted machine is equal to or lower than 1 m. When a robot arrives, the robot run is marked as finish. Since machines do not require network resources while performing tasks (the required data is pre-loaded), we exclude this phase when running network resource allocation simulation. The next run is initiated randomly the moment after the previous one. The robot's average speed is $1.5m/s$ with a sampling rate of $0.05s$. Several moving obstacles are introduced into the environment to make the path more dynamic. The MPC model handles the collision-avoiding algorithm, thus changing the deterministic path from one machine to another. 5 obstacles, with size $0.8 \times 0.8m^2$, move at a speed uniformly distributed between $0.4m/s$ and $0.8m/s$ heading to random directions throughout the dataset generation process.

With the above configuration, robot's run duration is described in Figure 5. We notice that the lower end of run duration is about 7.5 seconds, thus choosing the H value to be 7 seconds to 0 seconds to measure the performance of machine learning models.

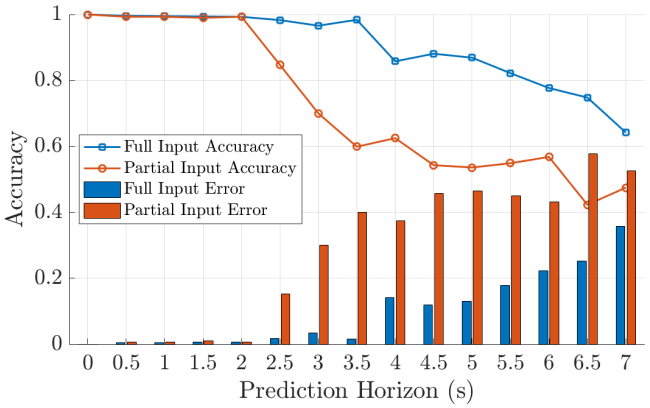


Fig. 6: Destination Prediction Result based on Prediction Horizon (H values)

B. Machine Learning Model

Figure 6 shows results from the destination prediction model with different H values and two pre-processing methods. We can notice that the prediction accuracy drops according to the H value. As described, H value represents the prediction horizon counting backward from when robots arrive at targeted machines. The larger H value is, the earlier the prediction is made. Earlier predictions mean the robot is further from the targeted machine, thus making predictions less accurate. The drop in accuracy is not linear due to the appearance of moving obstacles in the environment, making robots avoid them, thus enlarging robot paths dynamically and uniquely. Between the two data pre-process methods, we see a significant decline between H values 2 and 3, thus indicating the less critical data points still contribute to accurate predictions, especially when the prediction is made early.

C. Network Simulation

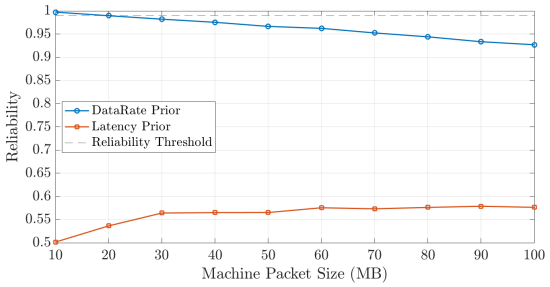
In this section, we compare different utility functions described in Algorithm 3 against each other as baseline in Section V-C.1. In Section V-C.2 and V-C.3 respectively, the most prominent baseline is improved further with the reliability control scheme introduced in Algorithm 2 and applied machine learning model in Algorithm 1. We gradually increase the workload the network system has to process by changing the machine packet size from 10MB to 100MB to demonstrate the proposed scheme performance in a different scenario. The default parameters are listed in the footnote unless stated otherwise. [1]

*1) **Utility Function Comparison**:* This experiment compares results between the data rate utility function and delivery time utility function following the matching procedure described in Algorithm 3. Simulation results in Figure 7 show that the delivery time utility function performs significantly worse than the data rate utility function (Figure 7b). Even though the delivery time utility function prioritizes more urgent requests, it still cannot meet the reliability requirements (Figure 7a). This can be explained, since matching based on the request urgency does not utilize the full power of the network system, thus making the network system perform much worse than its capability.

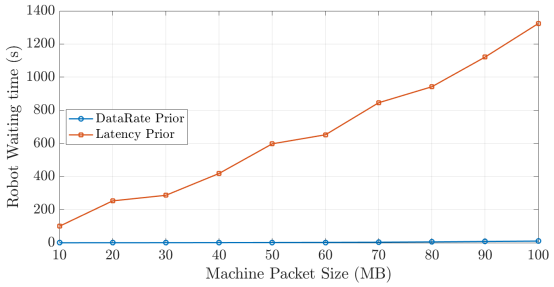
The data rate utility function allows the fastest packet transmission across the system without concerning packet deadlines. Therefore, robot waiting time is substantially lower (Figure 7b). However, Figure 7a shows that, with the increase in machine packet size, the system reliability decreases accordingly. Using the same parameter, we can safely assume the network serving capability is approximately the same in every case. Our assumption is, on average, for each time instant, the workload that the network system has to handle starts growing larger than its capability, thus resulting in a reliability decrease. This indicates that the system has to handle more workload than it can provide using the baseline method. Notice that this baseline method focuses solely on maximizing the data transmission rate, thus minimizing the robot waiting time. Therefore, this is the lowest robot waiting time we can achieve using the DA algorithm.

*2) **Reliability Control**:* Based on the result of the first experiment, we choose the data rate utility function to apply from here onwards. This experiment addresses the reliability issue that the Algorithm 3 cannot meet without significantly increasing the robot waiting time. The reliability control method is introduced in Algorithm 2 to achieve that. Due to the limited capacity, we demonstrate the result with the reliability threshold of 99% ($\epsilon = 0.99$) instead of 99.99%. Nevertheless, the result in Figure 8a shows a significant improvement in reliability compared to the data rate baseline. With the new reliability control method described in Algorithm 2, we identify a "secured boundary", preventing the reliability drops under the 99% threshold in all cases. Therefore, the network system can meet the reliability requirement. Regarding robot

[1] R = 20, M = 8, S = 4, Time instant length (t) = 0.0005s, Network bandwidth (W) = 10MHz, Robot packet size (S_R) = 200KB, Robot packet time-to-live (α) = 100ms, Learning Rate ($v(t)$) = 0.7

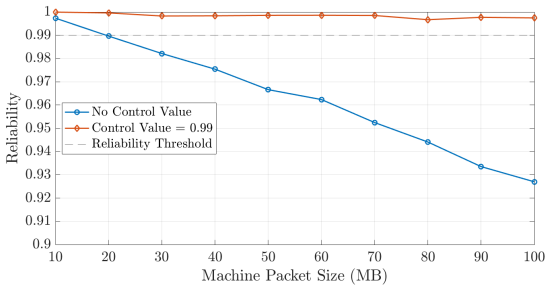


(a) Reliability Comparison

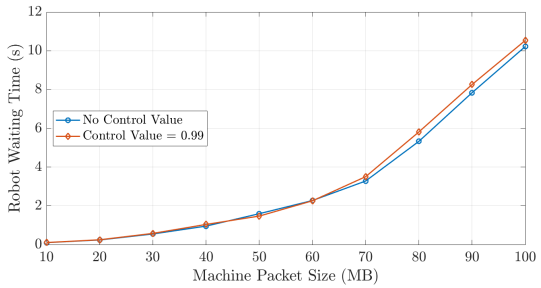


(b) Robot Waiting Time Comparison

Fig. 7: Comparison between Data Rate and Latency Utility Function



(a) Reliability Comparison



(b) Robot Waiting Time Comparison

Fig. 8: Comparison with and without Reliability Control using Data Rate Utility Function

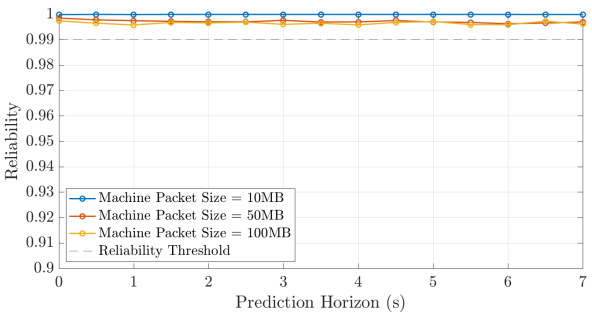
waiting time, Figure 8b shows that the highest additional time compared to the baseline is under 10% (when machine packet size is 80MB), indicating the excellent performance of the proposed scheme. [2]

*3) **Reliability Control and Machine Learning Prediction**:* This section shows the result of applying the machine learning model to the proposed scheme as described in Algorithm 1. In addition to the variation of machine packet sizes, we measure the network system performance with different H values to identify the suitable prediction horizon in each circumstance. In all scenarios, the reliability is controlled perfectly according to the requirement (Figure 9).

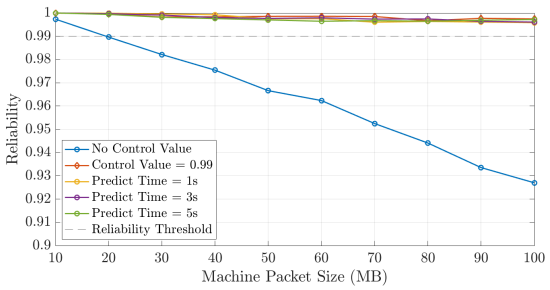
Figure 10 shows machine learning integration results with fixed machine packet size (Figure 10a - 10MB, Figure 10b - 50MB, Figure 10c - 100MB) and increased H values. We can notice significant improvements when the packet size is small with fixed sizes. Compared to the non-machine-learning approach (depicted as the dot line), with the 10MB machine packet size ($S_M = 10$MB) (Figure 10a), the robot waiting time is reduced by nearly 100% in the best performance. Even with the highest H following with the worst accuracy, the network system still benefits from the predictive model. However, with the increase in machine packet sizes, the predictive model's performance decreases significantly. Only 20% of improvement is recorded when $S_M = 50$MB and 2% is reported with $S_M = 100$MB. Witnessing this circumstance, we assume that when machine packet size is small, the network system can serve requests in a short time, thus improving the performance overall. However, with the increment in machine packet sizes, less available power remains for additional requests. Therefore, the machine learning approach only raises the improvement slightly.

Figure 11 presents simulation outcomes in another viewpoint, where H values is fixed ($H = 1, 3, 5$ seconds corresponding to Figure 11a, 11b, 11c respectively) and changing machine packet sizes. With different H values, we notice significant improvements when the packet is small. However, when the packet size grows, machine learning becomes less impactful, until a certain point, the performance becomes worse than the non-machine learning approach. This point is vastly different depending on the H value. This indicates that machine learning is viable for scenarios where the network system has unused resources. When applying machine learning, models result in correct or incorrect predictions. With correct prediction, data requests are registered in advance, providing the network system more time to handle them. However, with wrong predictions, redundant requests are registered, slowing the transmission process overall. A powerful network system can utilize the additional time that accurate predictions provide while still having available resources to handle redundant requests. However, for a less powerful system, not much transmit power remains to take advantage of extra time from correct prediction. Moreover, the system becomes more crowded as unnecessary requests appear, making the prediction model unproductive.

[2]The simulation parameter is designed specifically for the reliability threshold of 99% ($\epsilon = 0.99$) . The threshold above needs more fined parameters (i.e., smaller instant length) and longer simulation time to guarantee correctness.

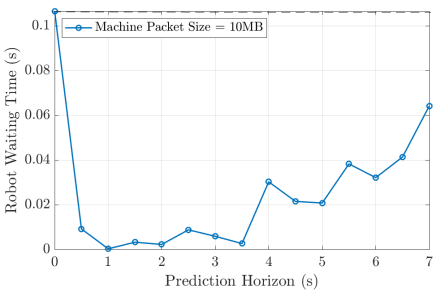


(a) Fixed Machine Learning Packet Sizes and different H values

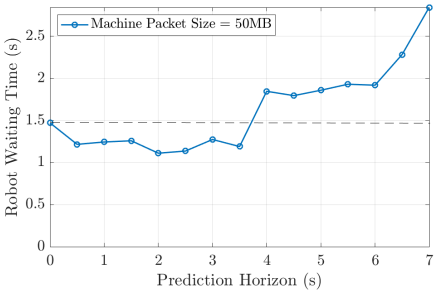


(b) Different Machine Packet Sizes and fixed H values

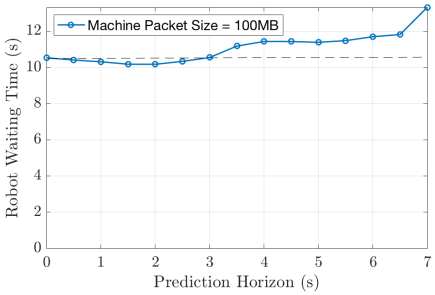
Fig. 9: Reliability Comparison with Machine Learning Packet Sizes and H values



(a) Machine Packet Size = 10MB

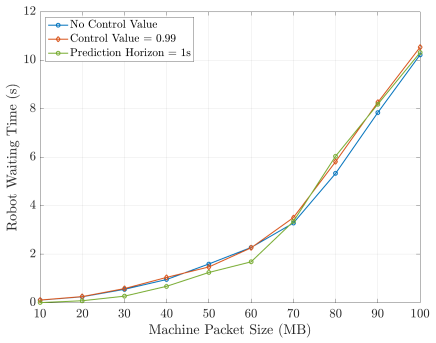


(b) Machine Packet Size = 50MB

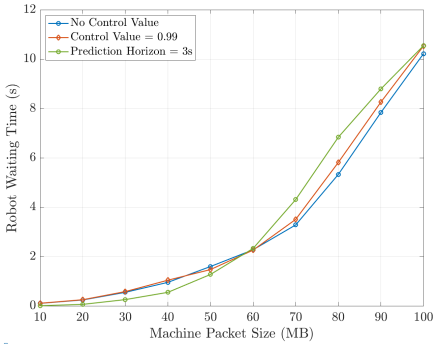


(c) Machine Packet Size = 100MB

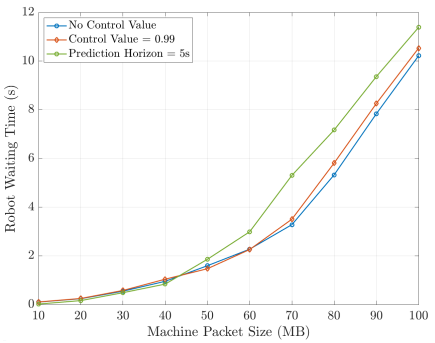
Fig. 10: Robot Waiting Time Comparison with fixed Machine Packet Size and different H values



(a) H value = 1s



(b) H value = 3s



(c) H value = 5s

Fig. 11: Robot Waiting Time Comparison with different Machine Packet Sizes and fixed H value

VI. Limitations

With simulations in different scenarios, we notice a system workload boundary, of which applying machine learning results in worse performance. Depending on the stand-alone performance of machine learning (i.e., accuracy, prediction horizon), this boundary changes accordingly. We assume that machine learning approaches produce additional redundant requests by raising wrong predictions. With an overloaded system, these requests can increase the burden that the system is taking, thus slowing down other request processes. However, more research needs to be conducted to address this assumption. Moreover, since this boundary change is dynamic, future experiments can be designed methodically to study the behaviors and impact of particular workloads on the network system.

Another issue we notice while designing this experiment is deciding the prediction moment. This research focuses on measuring the impact of different prediction horizons and accuracy on a robotic use case. In this case, performances with various prediction horizons are recorded after the moving robot finishes its run. With the pre-defined prediction horizons, we can correctly quantify the impact of the machine learning model for resource allocation purposes. However, in reality, we cannot decide the prediction horizon value deterministically since we do not know when the moving robot arrives at its targeted machine. Even by converting prediction horizons from time-wise to distance-wise measured from the targeted machine, the problem remains since the machine learning model cannot guarantee to raise correct targeted machine predictions. Addressing this problem requires additional resources and

depends on particular cases, thus making it a worthwhile issue for further studies.

VII. Conclusion

This research proposes an intent-based inference approach for managing teleoperation applications in a network system. The proposed scheme can proactively allocate network resources throughout multiple base stations to optimize the overall performance of local applications under reliability constraints of teleoperation applications. To achieve that, a reliable matching and scheduling scheme is introduced. Moreover, a machine learning model that utilizes Recurrent Neural Network structures is used to further enhance the proposed scheme's efficiency. Simulation results show that our proposed scheme can function well under multiple scenarios with different network capabilities and local applications workload. Our method significantly improves reliability and overall performance compared to other described methods. Although the intent-based inference, reflected by the machine learning model, depends on local applications, the improved matching and scheduling algorithm can generally be applied to other applications.

References

[1] T. B. Sheridan, "Human–robot interaction: Status and challenges," *Human Factors*, vol. 58, no. 4, pp. 525–532, 2016. PMID: 27098262.

[2] Z. Zhu and H. Hu, "Robot learning from demonstration in robotic assembly: A survey," *Robotics*, vol. 7, 04 2018.

[3] J. Legrand, M. Ourak, T. Vandebroek, and E. Vander Poorten, "A large displacement model for superelastic material side-notched tube instruments," *International Journal of Mechanical Sciences*, vol. 197, 02 2021.

[4] J. Torsner, K. Dovstam, G. Miklós, B. Skubic, G. Mildh, T. Mecklin, J. Sandberg, J. Nyqvist, J. Wang, B. Zhang, C. Martinez, and J. Neander, "Industrial remote operation: 5g rises to the challenge," vol. 93, pp. 54–69, 01 2016.

[5] L. Xia, X. Hou, G. Li, Q. Li, L. Sun, X. Liang, B. Liu, and A. K. C. Chan, "Verticals urllc use cases and requirements," 2020.

[6] A. Clemm, L. Ciavaglia, L. Z. Granville, and J. Tantsura, "Intent-Based Networking - Concepts and Definitions," Internet-Draft draft-irtf-nmrg-ibn-concepts-definitions-09, Internet Engineering Task Force, Mar. 2022. Work in Progress.

[7] K. Mehmood, H. V. K. Mendis, K. Kralevska, and P. E. Heegaard, "Intent-based network management and orchestration for smart distribution grids," *CoRR*, vol. abs/2105.05594, 2021.

[8] P. Szilágyi, "I2bn: Intelligent intent based networks," *J. ICT Stand.*, vol. 9, pp. 159–200, 2021.

[9] G. Hu, W. P. Tay, and Y. Wen, "Cloud robotics: architecture, challenges and applications," *IEEE Network*, vol. 26, no. 3, pp. 21–28, 2012.

[10] M. Afrin, J. Jin, A. Rahman, A. Rahman, J. Wan, and E. Hossain, "Resource allocation and service provisioning in multi-agent cloud robotics: A comprehensive survey," *IEEE Communications Surveys Tutorials*, vol. 23, no. 2, pp. 842–870, 2021.

[11] Y. SUN, X.-s. ZHOU, and G. YANG, "Cost aware offloading selection and resource allocation for cloud based multi-robot systems," *IEICE Transactions on Information and Systems*, vol. E100.D, pp. 3022–3026, 12 2017.

[12] L. Wang, M. Liu, and M. Q.-H. Meng, "A hierarchical auction-based mechanism for real-time resource allocation in cloud robotic systems," *IEEE Transactions on Cybernetics*, vol. 47, no. 2, pp. 473–484, 2017.

[13] M. El Hariri, I. H. Elhajj, C. Mansour, E. Shammas, and D. Asmar, "Environment-motivated real-time bandwidth allocation for collaborative robots teleoperation," in *2016 18th Mediterranean Electrotechnical Conference (MELECON)*, pp. 1–6, 2016.

[14] J. Lwowski, P. Benavidez, J. Prevost, and M. Jamshidi, *Task Allocation Using Parallelized Clustering and Auctioning Algorithms for Heterogeneous Robotic Swarms Operating on a Cloud Network*. 05 2017.

[15] S. Li, Z. Zheng, W. Chen, Z. Zheng, and J. Wang, "Latency-aware task assignment and scheduling in collaborative cloud robotic systems," in *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*, pp. 65–72, 2018.

[16] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "Maui: Making smartphones last longer with code offload," vol. 2010, pp. 49–62, 10 2010.

[17] M. S. Elbamby, M. Bennis, and W. Saad, "Proactive edge computing in latency-constrained fog networks," in *2017 European Conference on Networks and Communications (EuCNC)*, pp. 1–6, 2017.

[18] W. Li, C. Zhu, L. T. Yang, L. Shu, E. C.-H. Ngai, and Y. Ma, "Subtask scheduling for distributed robots in cloud manufacturing," *IEEE Systems Journal*, vol. 11, no. 2, pp. 941–950, 2017.

[19] X.-F. Liu, B. Lin, Z.-H. Zhan, S.-W. Jeon, and J. Zhang, "An efficient ant colony system for multi-robot task allocation with large-scale cooperative tasks and precedence constraints," pp. 1–8, 12 2021.

[20] M. Abdelrahman, M. Elbamby, and V. Räisänen, "Proactive scheduling and caching for wireless VR viewport streaming," *CoRR*, vol. abs/2110.02653, 2021.

[21] A. K. Tanwani and S. Calinon, "A generative model for intention recognition and manipulation assistance in teleoperation," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, p. 43–50, IEEE Press, 2017.

[22] K. H. Khokar, R. Alqasemi, S. Sarkar, and R. V. Dubey, "Human motion intention based scaled teleoperation for orientation assistance in preshaping for grasping," in *2013 IEEE 13th International Conference on Rehabilitation Robotics (ICORR)*, pp. 1–6, 2013.

[23] X. Bao, S. Wang, X. Ye, Y. Cai, T. Lu, and R. Wang, "A robotic shared control teleoperation method based on learning from demonstrations," *International Journal of Advanced Robotic Systems*, vol. 16, p. 172988141985742, 07 2019.

[24] Y. Oh, M. Toussaint, and J. Mainprice, "A system for traded control teleoperation of manipulation tasks using intent prediction from hand gestures," *CoRR*, vol. abs/2107.01829, 2021.

[25] A. Chekol and M. Fufa, "A survey on next location prediction techniques, applications, and challenges," *EURASIP Journal on Wireless Communications and Networking*, vol. 2022, 03 2022.

[26] Y. Wang, X. Fan, X. Liu, C. Zheng, L. Chen, C. Wang, and J. Li, "Unlicensed taxis detection service based on large-scale vehicles mobility data," in *2017 IEEE International Conference on Web Services (ICWS)*, pp. 857–861, 2017.

[27] M. Chen, X. Yu, and Y. Liu, "MPE: A mobility pattern embedding model for predicting next locations," *CoRR*, vol. abs/2003.07782, 2020.

[28] M. Chen, Y. Zuo, X. Jia, Y. Liu, X. Yu, and K. Zheng, "Cem: A convolutional embedding model for predicting next locations," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 6, pp. 3349–3358, 2021.

[29] J. Lv, Q. Li, Q. Sun, and X. Wang, "T-conv: A convolutional neural network for multi-scale taxi trajectory prediction," in *2018 IEEE International Conference on Big Data and Smart Computing (BigComp)*, pp. 82–89, 2018.

[30] H. Sun, C. Yang, L. Deng, F. Zhou, F. Huang, and K. Zheng, *PeriodicMove: Shift-Aware Human Mobility Recovery with Graph Neural Network*, p. 1734–1743. New York, NY, USA: Association for Computing Machinery, 2021.

[31] G. Wang, D. Liao, and J. Li, "Complete user mobility via user and trajectory embeddings," *IEEE Access*, vol. 6, pp. 72125–72136, 2018.

[32] C. Yang, M. Sun, W. X. Zhao, Z. Liu, and E. Y. Chang, "A neural network approach to jointly modeling social networks and mobile trajectories," *ACM Trans. Inf. Syst.*, vol. 35, aug 2017.

[33] A. Karatzoglou, A. Jablonski, and M. Beigl, "A seq2seq learning approach for modeling semantic trajectories and predicting the next location," in *Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, SIGSPATIAL '18, (New York, NY, USA), p. 528–531, Association for Computing Machinery, 2018.

[34] C. Wang, L. Ma, R. Li, T. S. Durrani, and H. Zhang, "Exploring trajectory prediction through machine learning methods," *IEEE Access*, vol. 7, pp. 101441–101452, 2019.

[35] X. Zhang, Z. Zhao, Y. Zheng, and J. Li, "Prediction of taxi destinations using a novel data embedding method and ensemble learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 1, pp. 68–78, 2020.

[36] N. Bahra and S. Pierre, "Rnn-based user trajectory prediction using a preprocessed dataset," pp. 1–6, 10 2020.

[37] D. Yao, C. Zhang, J. Huang, and J. Bi, "Serm: A recurrent model for next location prediction in semantic trajectories," in *Proceedings of the*

2017 ACM on Conference on Information and Knowledge Management, CIKM '17, (New York, NY, USA), p. 2411–2414, Association for Computing Machinery, 2017.

[38] A. Al-Molegi, M. Jabreel, and B. Ghaleb, "Stf-rnn: Space time features-based recurrent neural network for predicting people next location," in *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 1–7, 2016.

[39] H. Zhu, F. M. Claramunt, B. Brito, and J. Alonso-Mora, "Learning interaction-aware trajectory predictions for decentralized multi-robot motion planning in dynamic environments," *CoRR*, vol. abs/2102.05382, 2021.

[40] S. Eiffert and S. Sukkarieh, "Predicting responses to a robot's future motion using generative recurrent neural networks," *CoRR*, vol. abs/1909.13486, 2019.

[41] W. Wang, X. Zhu, L. Wang, Q. Qiu, and Q. Cao, "Ubiquitous robotic technology for smart manufacturing system," *Computational Intelligence and Neuroscience*, vol. 2016, pp. 1–14, 01 2016.

[42] M. Schwenzer, M. Ay, T. Bergs, and D. Abel, "Review on model predictive control: an engineering perspective," *The international journal of advanced manufacturing technology (2021). doi:10.1007/s00170-021-07682-3*, 2021.

[43] A. Domahidi and J. Jerez, "Forces professional," *embotech GmbH (http://embotech. com/FORCES-Pro)*, 2014.

[44] J. Chung, Ç. Gülçehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *CoRR*, vol. abs/1412.3555, 2014.

[45] A. Mukherjee, "Queue-aware dynamic on/off switching of small cells in dense heterogeneous networks," in *2013 IEEE Globecom Workshops (GC Wkshps)*, pp. 182–187, 2013.

[46] Y. Gu, W. Saad, M. Bennis, M. Debbah, and Z. Han, "Matching theory for future wireless networks: fundamentals and applications," *IEEE Communications Magazine*, vol. 53, no. 5, pp. 52–59, 2015.