Performance Evaluation of Closed-loop Industrial Applications Over Imperfect Networks

Sándor Rácz, Géza Szabó and József Pető

***Abstract*—5G networks provide technology enablers targeting industrial applications. One key enabler is the Ultra Reliable Low Latency Communication (URLLC). This paper studies the performance impact of network delay on closed-loop control for industrial applications. We investigate the performance of the closed-loop control of an UR5 industrial robot arm assuming fix delay. The goal is to stress the system at the upper limit of the possible network delay. We prove that to achieve the maximum accuracy of the robot at maximum speed, URLLC is a must have.**

***Index Terms*—Industrial Application, Robot Arm, URLLC, Network Delay, Trajectory Accuracy, Measurements, Performance Evaluation**

I. Introduction

Wireless networks are continuously replacing wired networks in several areas. Mobile Broadband is one of the most successful areas. As a next step, 5G networks also provide technology enablers targeting industrial automation and control applications. One key enabler is the Ultra Reliable Low Latency Communication (URLLC). URLLC should be capable of successfully transmitting messages over radio interface within 1 ms with a 99.999% success probability and should be capable to achieve a latency of 0.5 ms on average for multiple transmissions [1].

Industrial automation and control applications require significantly different latency and reliability [2]. Least demanding applications like diagnostics and maintenance do not require latency lower than 15 ms and reliability around 99.99%. Closed-loop applications require latency between 1 ms and 15 ms and ultra high reliability. Special applications, e.g., printing machine, typically require even lower latency (<1 ms).

A well-designed Industrial-Ethernet based solution provides reliable and low latency connection [6]. A potential problem is the cable cut like events. To overcome this, aliveness of the connection is continuously monitored. In case of a connection problem, the application executes emergency action, typically, the robotic cell is stopped. For example, in ProfiNet RT [16], data frames are sent periodically (update time specifies the period) and when a predefined number of consecutive frames are not arrived in time (retry parameter specifies the threshold, typical value is 3) then the application is notified.

From controller point of view, industrial applications over wired links are designed based on the assumption of perfect communication environment, e.g., non-delayed sensing and actuation. In contrast to wired networks, providing high quality services over wireless networks is resource demanding. Wireless networks have to deal with non-negligible transmission disturbances due to e.g. interference, fading and shadowing over the radio link. Several radio mechanisms, e.g., retransmission mechanisms, active queue managements, multiconnectivity, power control, link adaptation, try to compensate disturbances, and finally the network provides high quality services.

Wired link can be replaced by wireless link without touching the control algorithm of the application when

- wireless link can guarantee the same transmission requirements as the wired link provides, or
- characteristics of wireless link fulfill the design requirements of the control algorithm.

The first case is more conservative and much more challenging to realize by a wireless network, because in several applications the underlying industrial protocol provides strict guarantees that are much higher than the application requires. In the second case, the characteristics of connection provided by wireless link are adapted to the application requirements.

The joint optimization of application control loop and wireless network can improve efficiency. If the network is informed about the current latency requirement of the control loop, then the network can more efficiently assign resources. In this way, the wireless network can serve more applications simultaneously. We address the case when neither the link nor the application are optimized jointly. This paper evaluates the performance of a robot arm control application. The application includes closed-loop control of an UR5 industrial robot arm [10] and it is connected to the robot arm through a fixed delay connection. The main focus is on the effect of the link delay on the performance of robot arm movement quality measured by specific key performance indicators (KPIs).

Our target system on which the evaluation is done is a UR5 robot arm. The UR5 is an industrial grade robot arm and has an externally accessible velocity control interface. The robot arm accepts velocity commands for each joint (servo) and publishes joint state information with 8 ms update time. Investigated KPIs are response time and precision of trajectory execution, i.e., spatial and temporal deviations from the planed trajectory.

The paper is organized as follows. Section II discusses the state-of-the-art. Section III describes the measurement setup and provides measurement results. Section IV describes the measurement scenarios. Section V shows the performed evaluations. Section VI discusses the observations. Section VII concludes the paper.

Sándor Rácz and Géza Szabó are with Ericsson Research, Budapest, Hungary (e-mail: {sandor.racz, geza.szabo}@ericsson.com

József Pető is with Budapest University of Technology and Economics, Hungary (e-mail: pjoejoejoe@gmail.com)

DOI: 10.36244/ICJ.2019.2.4

II. RELATED WORK

The rest of the section gives an overview of Networked Control Systems (NCSs) focusing on the introduction of wireless link.

A. *Networked control systems*

In networked control systems, feedback control loops are closed via communication networks [11], [12]. A NCS consists of numerous coupled subsystems, which are geographically distributed, and individual subsystems exchange information over wired or wireless networks. In [11], an overview of recent developments on NCSs is presented. Three general configurations of networked control systems, i.e., centralized, decentralized and distributed configurations are discussed. Then, challenging issues from the study and application of NCSs are outlined from three aspects: communication, computation and control. In [12], several aspects of NCS was discussed, including sampling, network-induced delays, packets dropouts, quantization errors, sampled-data control, networked control, event-triggered control, network-based filtering in continuous or discrete-time domain.

The introduction of wireless links in a NCS requires us to revisit the design aspect of the system. There are two main means to integrate a wireless link into a NCS:

- adapting the control algorithms to the properties offered by wireless link [11], [13] and
- improving the wireless network to meet the current design assumptions [9], [14] .

In [11], authors proposed a fuzzy predictive control method to mitigate the network-induced delays from sensor-to-controller and controller-to-actuator links. At each time instance, the method evaluates the network delays and the proper control law is designed based on a predictive scheme. In [13], delay compensation scheme using classical and adaptive Smith predictor was applied to wireless NCS. The Markov model was proposed to compute the estimated network delay used in the classical predictor. In the adaptive predictor, the channel delay statistics using shift registers was proposed to update the estimated delay.

In [9], they provided a low complexity RTT skew MIMO control algorithm for 5G using multiconnectivity. In [14], authors considered all control loops as network applications (i.e. keeping controllers and devices unchanged) and developed a control-aware network uplink scheduler to handle the control performance degradation caused by communication delays.

B. *Wired communication*

Industrial automation and control applications use diverse technology to interconnect controller and devices. Industrial-Ethernet based protocols (e.g. ProfiNet, EtherCAT) and Field-Buses (e.g. ProfiBus) are the most widespread solutions with estimated market shares of about 46% and 48%, respectively [3]. In [4], authors compare the network protocols used nowadays in industrial applications. All investigated systems show similar basic principles, which are solely implemented in different ways. Shared memory is applied and most systems require a master or a comparable management system which controls the communication. Shared memory is implemented via data distribution mechanisms that are based on a high frequency packet sending pattern. These packets have to be transmitted with strict delivery time with minimum jitter.

ProfiNet [5] distinguishes between two real-time classes with different services and target applications.

- Real-Time (RT) class: This class is suitable for applications with cycle times of 1-10 ms. Standard Ethernet components can be used to connect devices. Application, transmission and devices have their own not synchronous cycles, in this way jitter is not optimized.
- Isochronous Real-Time (IRT) class: This class is suitable for applications with cycle times of less than 1 ms. This class provides clock synchronized communication and provides jitter less than 1 μs, but needs hardware support via switch-ASIC.

C. *Wireless communication*

A lot of effort is put into improving radio algorithms of URLLC. In [7], authors reviewed recent advances in URLLC. In [8], authors discussed wireless channel models that are relevant for URLLC. For challenging services like URLLC, tailor-made methods are developed to achieve the strict performance requirements, e.g. [9]. The support of URLLC services comes at the cost of reduced spectral efficiency compared to mobile broadband services without latency and reliability constraints [1]. The spectral efficiency significantly depends on the provided quality, for example, URLLC providing 1 ms latency can have about 3 times lower spectral efficiency compared to URLLC providing only 10 ms latency. In networks, where the share of the URLLC traffic can be significant in the load, optimized use of URLLC can improve the network capacity.

III. EXPERIMENTAL ENVIRONMENT

In this section, we investigate the response time and the trajectory execution performance of an UR5 industrial robot arm in networked control scenarios. During measurements we used a real UR5 robot arm.

A. *Hardware components*

UR5 industrial robot arm is a 6-DOF, lightweight, flexible, and collaborative robot that allows to automate repetitive and dangerous tasks with payloads of up to 5 kg. The robot arm is ideal for optimizing low-weight collaborative processes, such as picking, placing and testing.

The controller of UR5 provides access to a wide range of low level functionalities. This makes the robot suitable to be included in custom networked control system. At lowest level, individual joints with brushless servo motors and harmonic drive reducers are controlled with 2.4 KHz frequency. Unfortunately, this interface is not accessible externally. The next control possibility is to command (receive) the velocity (state) of the individual joints at sampling frequency of 125 Hz and this interface is accessible externally (low level control API). The robot supports ProfiNet RT, ModBus and TCP/IP communication protocols.

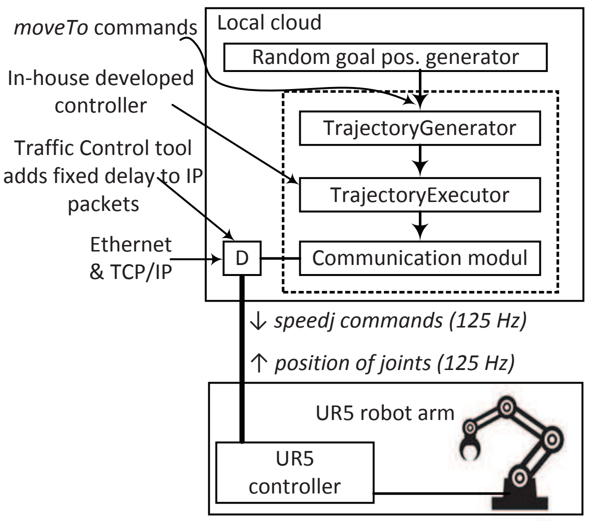


Fig. 1. Measurement setup

B. Software components

We have developed a robot arm controller from scratch based on the low level control API of the UR5. The main reason to do so is to facilitate the integration of our custom KPIs. Figure 1 shows our measurement setup. The controller runs on a Linux PC that is connected to UR5 over Ethernet. It uses TCP/IP protocol stack for communication and a 125 Hz controlling frequency. Furthermore, velocity control is applied which means that we send per joint velocity commands (*speedj*) to the robot arm every 8 ms, including rotation speed information of the 6 servo motors. Network delay that models latency aspect of URLLC link is inserted in the control loop, i.e., between the controller and UR5 robot arm, by the Traffic Control tool of Linux. We use fix delay to analyze the behavior of the system on the upper limit of the possible network delay. Note that jitter can be transformed to fix delay with a jitter-buffer.

Our controller implements trajectory generation, trajectory execution and communication modules, see Figure 1. Trajectory generator accepts *moveTo* commands. The parameters of a command specify the goal position and orientation in Cartesian space, target execution start time and maximum allowed joint velocity and acceleration. First, using inverse kinematics, the goal position and orientation are transformed into joint space. Then, a feasible path is determined from the current position of the robot to the goal position using tangent bug algorithm. Obstacles can be specified in joint space. Finally, the feasible path is sampled and cubic-spline interpolation is applied considering the specified maximum joint velocity and acceleration. The trajectory generator also supports smooth on-the-fly trajectory modification.

Trajectory executor receives trajectories. A trajectory is described by 6 splines, each spline describes individual joint position evolution in time. For each joint, a feed-forward velocity control is running with predefined update time for which default value is 8 ms. The position error is calculated from the target position coming from the spline and the current position extracted from the robot feedback. The baseline velocity is obtained from the spline by derivation and modified through a PID controller based on position error. We have tuned the parameters of the PID controller for zero network delay and we kept this setting unchanged during the investigations. The update timers of the robot and the trajectory executor are unsynchronized. This unsynchronized operation further increases the average response time with 4 ms and the standard deviation (i.e. jitter) with ~ 2.3 ms. Consequently, the average dead delay of the control loop of trajectory executor is about 18 ms and the jitter is about 3 ms. Trajectory executor also records the realized trajectory. After execution, it compares the planned and the realized trajectories and calculates KPIs.

The communication module sends the joint velocity commands to the robot. Commands are sent in clear text format, and each command message contains joint velocity values for all of the 6 joints. A velocity command message is valid until a new message received or an optionally specified timeout expired. The status feedback is encoded in binary format and has a size of about 1 Kbyte.

Network delay	Avg. response time	Stand. Dev.
0 ms	14.66 ms	1.84 ms
1 ms	14.99 ms	2.00 ms
2 ms	15.60 ms	2.50 ms
4 ms	20.61 ms	1.91 ms
8 ms	22.46 ms	1.66 ms
16 ms	30.51 ms	1.70 ms
32 ms	46.62 ms	1.81 ms
64 ms	78.50 ms	1.83 ms

TABLE I
Average and standard deviation of response time for different network delays

IV. Measurement scenarios

A. Response time

We started with the investigation of the response time (i.e, dead delay) of the robot. We sent a (non-zero) velocity command to standstill robot and inspected the received status messages sent by the robot. The response time is the time elapsed from the command transmission to the first received status message reporting joint movements. Table I shows the mean value and the standard deviation of response times for different network delays. Without network delay, the average response time is 14.66 ms and the standard deviation is 1.84 ms. The robot checks the incoming commands periodically with 8 ms period and also sends status messages with 8 ms period. Note that the standard deviation of a continuous random variable uniformly distributed over $[0, 8)$ is 2.3. The measurements show that the internal robot operation contributes to the jitter of control-loop about 2 ms. Table also shows that the network delay additionally increases the average response time and does not significantly modify the standard deviation. This means that the quick reaction on external events needs low network delay. For example, assume that the robot moves with 1 m/sec, then e.g., 10 ms additional network delay can end in up to 1 cm additional difference.

B. Precision of trajectory execution

We evaluated three main KPIs for each trajectory to measure execution quality. Two of them measure execution precision and the third one measures the execution time. Let $\underline{p}(t)$ and $\underline{r}(t)$ denote the position functions of the planned and the realized trajectories, respectively. Positions can be defined in Cartesian space or in joint space. In Cartesian space, the 3D coordinates (i.e. x, y and z) and the orientation of the tool center point are considered. In joint space, for example, $\underline{r}(0) = \{r_1(0), r_2(0), \ldots, r_6(0)\}$ denotes the start position, where $r_i(t)$ denotes the position of i-th joint at t. Denote T_p and T_r the durations of the planned and the realized trajectories, respectively. During $T_p < t \leq T_r$, the goal position refinement is being executed by the controller. The execution is finished when the predefined goal position accuracy has been achieved or predefined refinement time limit reached. In measurements, 10 sec maximum refinement time was configured. We introduce the following KPIs:

- Spatial deviation from the planned trajectory.

$$\Gamma(t) = \min_{\tau \in [-1,1]} \| \underline{r}(t) - \underline{p}(t+\tau) \|_2, \quad t \in [0, T_r].$$

  The $\Gamma(t)$ is the minimal distance between the robot position at time t and the corresponding segment of the planned trajectory around t. For orientation,

$$\Gamma_O(t) = \min_{\tau \in [-1,1]} \arccos\left[O_r^{-1}(t) \cdot O_p(t+\tau)\right], \quad t \in [0, T_r],$$

  where $O_r(t)$ and $O_p(t)$ are unit quaternions [15] representing the realized and the planed orientations of the tool center point, respectively.
- Temporal deviation from the planned trajectory.

$$\Delta(t) = \arg\min_{\tau \in [-1,1]} \| \underline{r}(t) - \underline{p}(t+\tau) \|_2, \quad t \in [0, T_p].$$

  The $\Delta(t)$ is the time difference between minimal distance point pair at time t.
- Refinement time. $\Upsilon = T_r - T_p$ is the extra time needed to approach the goal position in the predefined spatial accuracy.

The spatial and temporal deviations describe the distance between the realized and the planed trajectory. The spatial deviation measures the distance in Cartesian space or in joint space. By temporal accuracy, we refer to the timing accuracy of trajectory execution. The temporal deviation measures how accurately the planed trajectory is followed in time. For example, assume that the robot arm exactly moves along the planned path. In this case the spatial deviation is zero. Now assume that the robot arm moves on this path with 100 ms delay, i.e. $r(t) = p(t-0.1)$. In this case the temporal deviation is -100 ms.

V. EVALUATION OF THE MEASUREMENTS

During the measurements we have executed trajectories to randomly generated goal positions and orientations. The same trajectories were executed with varying parameter settings. We used different maximum allowed join speeds (from 22.5 to 112.5 deg/sec), goal accuracy in joint space (from 0.1 to 0.001 deg), controller update times (8, 16 and 24 ms) and a wide range of network delays (RTT: 0, 1, 2, 4, 8, 16, 32 and 64 ms). The high delay values, i.e. 32 and 64 ms, are included to see extreme cases as well. Figure 2 and Figure 3 highlight measurement results.

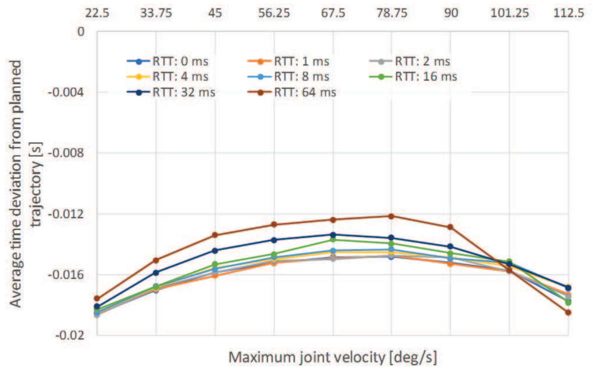
A. Affecting the temporal deviation

Figure 2(a) and Figure 2(b) show the average and the range of temporal deviation from planned trajectories for different network delays as a function of maximum allowed joint speed and using 8 ms update time and 0.1 deg accuracy. The average of temporal deviation hardly depends on the network delay, its absolute value is about 12-18 ms and the negative sign means that the robot is a little behind time in average. Note that this is approximately the dead delay of the control loop for non-delayed (i.e. RTT: 0 ms) case.

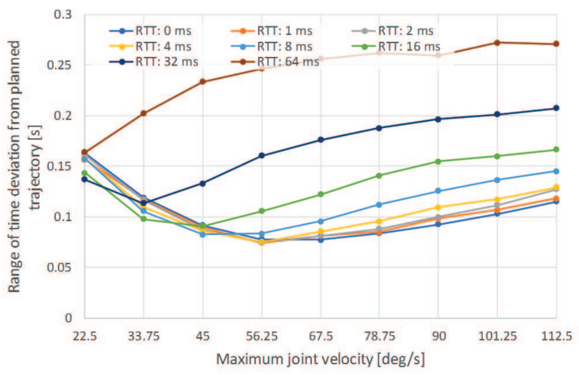
The range of temporal deviation is more sensitive to network delay. For each network delay we can observe a speed limit, e.g. for 16 ms delay it is about 45 deg/sec. If the speed is below this limit, the curve is close to the non-delayed curve. However above the limit, the range of temporal deviation curve goes above the non-delayed curve. Increased range value means that the robot is sometimes ahead of time and sometime behind time to the planed trajectory. It is also interesting that in low speed cases (e.g. 22.5 deg/sec) the range of temporal deviation is high ($\sim$ 150 ms) and hardly depends on the network delay. This can mean that in case of slow motion the high temporal deviation is probably caused by internal operation of the robot and the controller and not by the network delay. Summarizing, low network delay is required for use-cases where high temporal accuracy is crucial at high robot movement speed. For example, to avoid collision of more robot arms working close to each other.

B. Affecting the spatial deviation

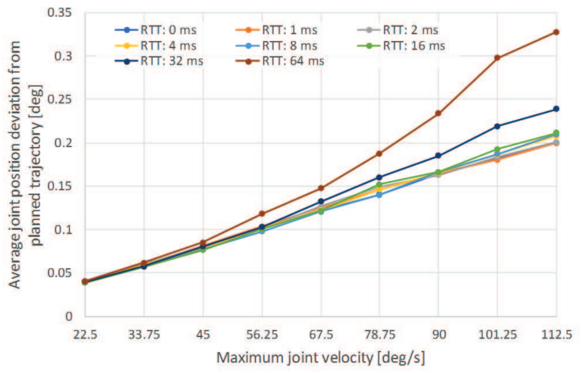
Figure 2(c) and Figure 2(d) show the average of spatial deviation from planned trajectory for different network delays as a function of maximum allowed joint speed and using 8 ms update time and 0.1 deg accuracy. In Figure 2(c), the measures are evaluated in joint space, in Figure 2(d) the measures are evaluated in Cartesian space. For higher speed, the same network delay causes higher degradation, as we expected. For network delays of 32 and 64 ms, the difference is significant. For lower network delays, the difference is relatively small. This can mean that from a certain network delay limit (in this measurement setup 32 ms) the network delay causes more intense degradation of accuracy. The two figures have similar shape. Note that, using forward kinematics formulae, joint space can be one-to-one mapped into Cartesian space. For inverse kinematics (Cartesian space to joint space transformation), a point in Cartesian space can have more than one image in the joint space. We conclude that applying lower joint speeds the spatial accuracy increases and also the controller is more tolerable to network delay. In this way, if low latency connection is available then the robot can be



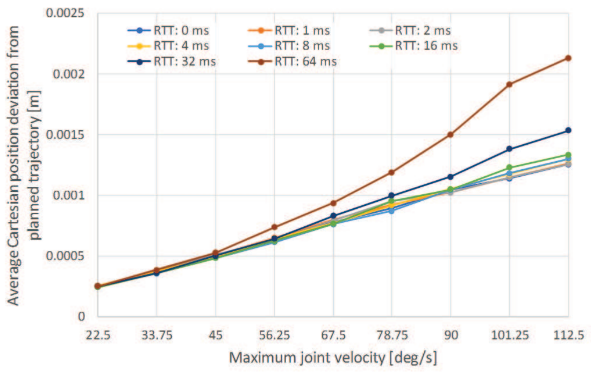
(a) Average of temporal deviation



(b) Range of temporal deviation



(c) Average of spatial deviation in joint space



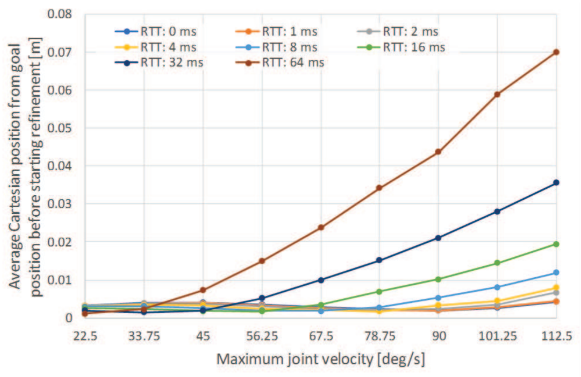
(d) Average spatial deviation in Cartesian space

Fig. 2. Statistics of temporal and spatial deviations of realized trajectory from planned trajectory for different network delays

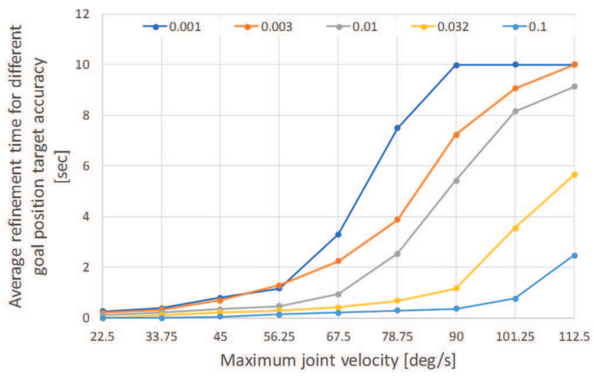
used at full speed. This also means that if only higher latency connection is available then using lower robot speed allows achieving the same spatial accuracy.

C. Affecting the refinement time

Figure 3 shows results for refinement time. Figure 3(a) shows the spatial distance at time T_p, i.e., when the planned trajectory ends and goal refinement starts. As we expected, this KPI has similar figure as average spatial deviation values. Figure 3(b) shows how refinement times depend on the required spatial accuracy (8 ms update time and 16 ms network delay). Refinement time is higher for stricter accuracy requirement and for higher robot speed. There are also cases when the required accuracy cannot be achieved within predefined time limit, e.g., 0.001 deg accuracy and > 90 deg/sec speed. This means that a deadline on execution time leads to requirement on maximum tolerable network delay. In contrary, using lower robot speed with the same spatial accuracy is achievable over a connection with higher latency. The cost is the increased execution time. Consequently, choosing proper required accuracy can improve execution time. In a robotic cell, the cyclic time is an important KPI. The cyclic time can be improved by reducing refinement time by specifying lower accuracy for cases where spatial accuracy is not crucial.



(a) Average spatial distance at the start of goal position refinement



(b) Average refinement time for different required goal position accuracy in joint space; accuracy requirements: from 0.001 deg to 0.1 deg

Fig. 3. Statistics of goal refinement phase

Max joint speed	Delay	8 ms tick	16 ms tick	24 ms tick
22.5 deg/sec	16 ms	0.116	0.116	0.115
	64 ms	0.143	0.186	0.141
45 deg/sec	16 ms	0.229	0.232	0.255
	64 ms	0.622	0.710	0.675
67.5 deg/sec	16 ms	0.340	0.374	0.492
	64 ms	1.464	1.571	1.568
90 deg/s	16 ms	0.477	0.608	0.808
	64 ms	2.531	2.660	2.793
112.5 deg/sec	16 ms	0.801	1.029	1.287
	64 ms	3.694	4.144	4.100

TABLE II
MAXIMUM SPATIAL DEVIATION IN JOINT SPACE [DEG] FROM PLANNED TRAJECTORY FOR DIFFERENT CONTROLLER UPDATE TIME (TICK), NETWORK DELAY AND MAXIMUM JOINT SPEED.

D. Experimenting with the update time of the controller

In the final measurement, we investigated the effect of update time of controller on the accuracy. In Table II, maximum spatial deviation in joint space is shown. For each trajectory, the maximum spatial deviation is calculated (i.e. $max_{t\in[0,T_B]}\Gamma(t)$) and averaged over executed trajectories. We observed that to utilize the advantage of lower update time requires low network delay as well. For example, in 64 ms network delay cases, using the lowest (i.e. 8 ms) update time has no significant effect on the performance. However, for lower network delay cases (e.g. 16 ms), lower update time leads to significant gain. This means that systems using low update time require strict latency requirements from the wireless link. Providing low latency connection for a system with high update time has no performance advantage.

VI. DISCUSSION ON THE OBSERVATIONS

This section summarizes and discusses observations and also suggests a method to handle loss and jitter.

A. Requirements on the network

In general, measurement results have shown that the network delay lower than 4 ms has no significant performance impact. This is because (a) the internal operation of the robot ends in about 2 ms standard deviation in response time, most probably, due to the internal sampling used in the robot and (b) the ticks of the robot and the controller are unsynchronized. The impact of network delay lower than 4 ms is masked by the background "noise" of measurement setup. The detailed analysis of 0-4 ms network delay range requires more sophisticated measurement apparatus, e.g., the robot and the controller should be synchronized, otherwise the randomness introduced by unsynchronized update times dominates the behavior or a robot arm with lower update time (e.g., < 1 ms) should be used.

The task of the robot arm can put requirements on the network delay:

- For tasks where robot arm should react on external events, low network delay is desired, because the network delay between robot and controller directly increases the reaction time.
- For tasks where time consuming goal refinement is not tolerable, low network delay should be provisioned. The deadline on trajectory execution time leads to a requirement on the maximum tolerable network delay. In general, higher network delay makes the refinement time longer and in this way increases the total trajectory execution time.
- Some tasks require accurate movement along the path, e.g. welding, and not only at the goal position. Another example is the collaboration of more robot arms where the precise and synchronized movements are crucial. For these tasks also low network delay is desired.

The internal mechanisms of robot arm can also put requirements on the network delay. In general, a low update time system requires lower network delay. The control of a robot arm with e.g. 20 ms update time, probably tolerates higher network delay than a more precise and faster robot arm with e.g. 1 ms update time. In addition to this, providing low latency connection for a system with relatively high update time has limited performance advantage.

Performance requirements of trajectory execution can also put requirements on the network delay. Faster robot movements require lower network delay for accurate movement. In other side, if only higher latency connection is available then using lower robot speed can compensate increased network delay for some extent.

Performance optimization can also give guidelines for required network delay. Choosing proper required accuracy can improve execution time. For example, if less accurate movement is enough, then relaxed accuracy can shorten refinement time.

B. Handling jitter, delay and packet loss

In the measurements, jitter and packet loss were not considered. We assumed negligible jitter and no packet loss was introduced by the network.

When the *jitter* is relatively small compared to the latency of the connection, then jitter buffer like methods can be used to transform jitter into extra delay. During the end-to-end delay budget calculation this extra delay should be taken into account. This method requires packet buffering capability.

Delayed or lost packets that were not arrived in time can end in performance degradation. The best solution is to minimize the occurrence of these events and to avoid bursty occurrence of them. One of the main goals of URLLC is to provide reliable connection and fulfill these requirements. In some extent, delayed or lost packets can also be handled at higher layers in the controller and at the device side. All correction methods reduce the accuracy of movements and can efficiently be used only for a limited time period.

In case of delayed or lost *status messages*, action should be taken at the controller side of the control loop. In case of trajectory execution, the controller uses joint positions from the status messages. The missing joint position values can efficiently be extrapolated, because trajectory generators intentionally generate smooth trajectories to reduce the load of the servos. Practically, the missing joint position value is extrapolated from the historical values of joint positions and from the remaining part of the trajectory. The position error caused by extrapolated values will be corrected by the PID control when the controller receives again correct status messages.

In case of delayed or lost *command messages*, action should be taken at the both sides of the control loop. At the *controller side*, the controller needs to be informed about the unsuccessful command transmission to keep itself up-to-date. A potential solution is that the wireless network informs controller about transmission status of down-link packets. When radio interface failed or predicted to fail to transmit a packet in time (e.g. radio related problems or congestion), then wireless network notifies the controller about this event. Relying on this information the controller updates its internal state and tries to avoid overreaction.

VII. Conclusion

We investigated the performance of the closed-loop control of an UR5 industrial robot arm at varying network characteristics. We run trajectories and measured the accuracy of realized trajectories as the function of network delay and robot movement speed. We introduced KPIs to evaluate the temporal and spatial accuracy of the realized trajectories. We observed that to achieve the maximum accuracy of the robot at maximum speed, there is a need for low latency communication. However, at lower speed or at relaxed accuracy, higher network latency is still tolerable. We also observed that, providing much lower latency than the update time of the robot has only moderate performance gain. Finally, we suggested a method to handle loss and jitter of robot control packets.

References

[1] J. Sachs, G. Wikstrom, T. Dudda, R. Baldemair and K. Kittichokechai, "5G Radio Network Design for Ultra-Reliable Low-Latency Communication," in IEEE Network, vol. 32, no. 2, pp. 24-31, March-April 2018.

[2] S. A. Ashraf, I. Aktas, E. Eriksson, K. W. Helmersson and J. Ansari, "Ultra-reliable and low-latency communication for wireless factory automation: From LTE to 5G," 2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA), Berlin, 2016, pp. 1-8.

[3] Automation Inside (March) 2017 [Online]. Available: http://www.automationinside.com/2017/03/industrial-network-market-shares-2017.html

[4] P. Danielis, J. Skodzik, V. Altmann, E. B. Schweissguth, F. Golatowski, D. Timmermann and J. Schacht, "Survey on real-time communication via ethernet in industrial automation environments," Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA), 2014, pages 1-8.

[5] PROFIBUS and PROFINET International. (April 2019) [Online]. Available: https://www.profibus.com/

[6] S. Nsaibi, L. Leursand H. D. Schotten, "Formal and simulation-based timing analysis of Industrial-Ethernet sercos III over TSN," 2017 IEEE/ACM 21st International Symposium on Distributed Simulation and Real Time Applications (DS-RT), Rome, 2017, pp. 1-8.

[7] Mehdi Bennis, Merouane Debbah and H. Vincent Poor, "Ultra-Reliable and Low-Latency Wireless Communication: Tail, Risk and Scale," CoRR abs/1801.01270 (2018)

[8] Patrick C. F. Eggers, Marko Angjelichinoski and Petar Popovski, "Wireless Channel Modeling Perspectives for Ultra-Reliable Communications," CoRR abs/1705.01725 (2017)

[9] R. A. Delgado, K. Lau, R. H. Middleton and T. Wigren, "Networked Delay Control for 5G Wireless Machine-Type Communications Using Multiconnectivity," in IEEE Transactions on Control Systems Technology, 2018, Early Access.

[10] https://www.universal-robots.com/products/ur5-robot/

[11] N. Vafamand, M. H. Khooban, T. Dragicevic and F. Blaabjerg, "Networked Fuzzy Predictive Control of Power Buffers for Dynamic Stabilization of DC Microgrids," in IEEE Transactions on Industrial Electronics, doi: 10.1109/TIE.2018.2826485

[12] X. M. Zhang, Q. L. Han and X. Yu, "Survey on Recent Advances in Networked Control Systems," in IEEE Transactions on Industrial Informatics, vol. 12, no. 5, pp. 1740-1752, Oct. 2016.

[13] Mahmoud Gamal, Nayera Sadek, Mohamed R.M. Rizk and Ahmed K. Abou-elSaoud, "Delay compensation using Smith predictor for wireless network control system," Alexandria Engineering Journal, Volume 55, Issue 2, 2016, Pages 1421-1428.

[14] Q. Liu, S. Zoppi, G. Tan, W. Kellerer and E. Steinbach, "Quality-of-control-driven uplink scheduling for networked control systems running over 5G communication networks," 2017 IEEE International Symposium on Haptic, Audio and Visual Environments and Games (HAVE), Abu Dhabi, 2017, pp. 1-6.

[15] Huynh, D.Q., "Metrics for 3D Rotations: Comparison and Analysis," J Math Imaging Vis (2009) 35: 155. doi: 10.1007/s10851-009-0161-2

[16] PROFINET Real-Time Protocol (PN-RT) https://wiki.wireshark.org/PROFINET/RT

Sándor Rácz received his MSc and PhD in electrical engineering from the Budapest University of Technology and Economics (BME), at the Department of Telecommunications and Media Informatics (TMIT) in 1997 and 2004 respectively. Since 2000, he has been a research fellow at the Ericsson Traffic Analysis and Network Performance Laboratory (Traffic Lab) in Budapest. His research interests include performance modelling and analysis of telecommunication systems. He published several patents, as well as conference and journal papers, for which he received more than 850 independent citations.

Géza Szabó is working as a Senior Researcher in the Artificial Intelligence research area in Ericsson Research and taking part in designing and implementing demonstrations for external events within our robotics team e.g., Mobile World Congress, Hannover Messe.

József Pető received his MSc. degree in Computer Engineering from the Budapest University of Technology and Economics in 2018. He is currently working toward his Ph.D. degree. His current areas of interest and research include cloud robotics, digital twin, robot simulation, machine learning applications in robotics.