Optimal Command Ordering for Serial Link Manipulators

Christian Smith and Yiannis Karayiannidis
Centre for Autonomous Systems, Royal Institute of Technology
Stockholm, Sweden. e-mail: {ccs|yiankar}@kth.se

***Abstract*—Reducing the number of cables needed for the actuators and sensors of humanoid and other robots with high numbers of degrees of freedom (DoF) is a relevant problem, often solved by using a common bus for all communication, which may result in bandwidth limitation problems. This paper proposes an optimized method to re-order the commands sent to the joint-local controllers of a serial manipulator. The proposed method evaluates which local controller would benefit the most from an updated command given a cost function, and sends a command to this controller. As is demonstrated in both simulation and on a real robot, the resulting scheme can significantly improve system performance, equivalent to increasing the communication frequency by up to 3 times.**

I. Introduction

Humanoid robots present some of the most difficult challenges for robot design. Not only do they need to be self-contained with no or limited physical connections to external power or processing units, but they also need to fit a human form factor of limited dimensions, often with hard constraints on weight. Furthermore, the extremities are expected to have similar motion and manipulation capabilities to a real human, with high requirements for dexterity, strength, and precision.

Historically, the solutions to these problems have had different trade-offs between performance measures. Placing the actuators in the torso and moving the joints via cables enables low mass for the arms [1], but the torso may have to be excessively bulky to accomodate for the motors and control units [2], while the heat generated from centralized motors may be difficult to dissipate. Some robots have central controller units and run individual power cables to actuators located near each joint, and individual signal cables from each sensor or joint angle encoder back to the control unit [3]. Even in the simpler case, this results in four cables per actuated/encoded degree of freedom. For a high DoF arm with an advanced hand this may pose a major problem, both in terms of weight and space inside the arms, and mechanical problems of passing cables through each joint.

To reduce the amount of cables running along a manipulator, it has become common practice to use local controllers for each actuator and/or sensor [4]–[12], reducing the cabling need to a total of four (or a similar small number of) cables: two for power and two for signals between the local controllers and the centralized control computer. However, this has introduced a new problem. Since all communications between local controllers and the central computer share the same bus, the communication speed becomes inversely proportional to the number of connected local controllers. The latter implies that advanced controllers requiring high frequency control loops are increasingly difficult to implement on systems with a high number of DoFs. As a result, robotic systems cannot combine the advantages of using local controllers with shared cables with high frequency control on low-cost standardized communication buses.

In this paper we observe that for many applications, the communication needs are unevenly distributed between different local controllers, and propose a locally optimized message priority scheme that allows different local controllers to communicate at different frequencies. The performance of the scheme is evaluated in simulation, and an implementation on a real robot demonstrates the performance of the proposed scheme for a real velocity controlled 7 DoF robot arm for a hybrid force/velocity control type task.

II. Related Work

It is common practice to place local controllers at (or near) each joint, and send messages to these on a common bus. This section presents works using these schemes and how problems of limited bandwith have been dealt with.

Several robot systems have been implemented using the Controller Area Network (CAN) bus, motivated by its robustness and reliability. However, CAN is limited to a theoretical maximum of approximately 7000 messages per second, with many implementations performing significantly slower. The KHR-2 humanoid uses distributed control and a 1 Mbaud CAN bus. Having a total of 41 DoFs, the overall control loop is closed at 100 Hz [13]. The HRP-3 humanoid robot uses centralized motordrivers and controllers for the arms for heat dissipation reasons, but the lower body and legs use joint-local controllers that are connected on a CAN bus [3]. For the HRP-4 humanoid, joint-local controllers are used for the arms as well, but a total of 10 seperate CAN buses are used for the 34 actuators to achieve sufficient communication throughput [12]. In [14], 4 parallel CAN buses are used close the main control loop at 600 Hz for a 6 DoF manipulator. By using several parallel buses, these implementations still have to cope with problems of running large numbers of cables through the kinematic structure. Other implementations use different kinds of serial buses. Paredis et al. use a 5 MBaud RS485 bus to achieve 400 Hz control of an arm using distributed controllers [4]. Jia et al use a RS422 bus and distributed controllers to velocity control a 6 DoF arm at

20 Hz [15]. Jiang et al use RS485 to send commands to the joint-local controllers of a humanoid robot at 20 Hz. [10].

Implementations requiring higher frequencies of the main control loop use more advanced, faster buses. Lamarche and Zhu state that closing the main control loop at 1000 Hz for a modular robot with a distributed control system requires a 100 Mbps bus [5]. Similarily, the main control loops of the Justin humanoid are closed at 1000 Hz, using joint-local controllers and a 100Mbit/s SERCOS-Bus [9].

III. Proposed Method

In this section we propose a novel scheme to reorder the messages on the control bus, to make more efficient use of the limited bandwith of communication links. State of the art schemes (see Section IV-A for a listing) address all local controllers in every iteration of the main loop, while the scheme proposed here is based on the assumption that the required update frequency of commands to the joint-local controllers may differ between joints, and depend on the current task and controller.

We assume an n-DoF serial link with a local controller at each joint, and that we can send $\mathcal{F}$ commands per second, and that each command can set the target angular velocity for the local controller of one joint. We also assume that a controller keeps the setpoint velocity until a new command is received. Thus, with a classical approach where a command is sent to each joint in each iteration of the control loop, the control loop can be run at a frequency denoted by f:

$$f = \frac{\mathcal{F}}{n} \tag{1}$$

We denote the generalized position of the robot with $\mathbf{p}$, and we assume that we formulate our control objectives in term of the generalized velocity, $\mathbf{v}$. The objective for our system is the minimization of the norm of the error $\mathbf{e}$, i.e.

$$\min \|\mathbf{e}\| \tag{2}$$

where $\mathbf{e}$ is defined as follows:

$$\mathbf{e} = \mathbf{W}(\mathbf{v} - \mathbf{v}_d) \tag{3}$$

where $\mathbf{W}$ is a user-defined weight matrix, $\mathbf{v}$ is the current generalized velocity, and $\mathbf{v}_d$ is the desired velocity. The generalized task space velocity $\mathbf{v}$ is related to the current joint space velocity $\dot{\boldsymbol{\theta}}$ through the Jacobian $\mathbf{J}(\boldsymbol{\theta})$ as follows:

$$\mathbf{v} = \mathbf{J}(\boldsymbol{\theta})\dot{\boldsymbol{\theta}} \tag{4}$$

Under the assumption that $\mathbf{J}(\boldsymbol{\theta})$ is invertible, the desired joint velocity $\dot{\boldsymbol{\theta}}_d$ can be calculated as follows:

$$\dot{\boldsymbol{\theta}}_d = \mathbf{J}^{-1}(\boldsymbol{\theta})\mathbf{v}_d \tag{5}$$

Clearly, when all the joints can be simultaneously given new velocities, then the choice $\mathbf{u} = \dot{\boldsymbol{\theta}}_d$, with $\mathbf{u} \in \Re^n$ denoting the input at the joint level, drives $\mathbf{e}$ to zero.

We assume that the velocity in joint space has been altered by the most recent velocity set-point command. This means that the velocity is a sum of the velocity of the previous time-step, $\dot{\boldsymbol{\theta}}_{\text{old}}$, and the change $\delta\dot{\boldsymbol{\theta}}$, which in turn implies that the generalized velocity in the robot workspace can be expressed as follows:

Frame	α	a	θ	d
Base	$0°$	0.274	$90°$	0
Base	$90°$	0	$0°$	0
1	$0°$	0	θ_1	0
2	$-90°$	0	$\theta_2 + 180°$	0
3	$-90°$	0	$\theta_3 + 180°$	0.313
4	$-90°$	0	$\theta_4 + 180°$	0
5	$-90°$	0	$\theta_5 + 180°$	0.2665
6	$-90°$	0	$\theta_6 + 180°$	0
7	$-90°$	0	$\theta_7 + 180°$	0
Tool	$0°$	0	$0°$	0.42

TABLE I: DH parameters of the 7-DOF arm (using Craig's convention).

$$\mathbf{v} = \mathbf{J}(\boldsymbol{\theta})(\dot{\boldsymbol{\theta}}_{\text{old}} + \delta\dot{\boldsymbol{\theta}}) \tag{6}$$

Substituting (6), (5) in to the weighted error (3) we get:

$$\mathbf{e} = \mathbf{W}\mathbf{J}(\boldsymbol{\theta})(\dot{\boldsymbol{\theta}}_{\text{old}} + \delta\dot{\boldsymbol{\theta}} - \dot{\boldsymbol{\theta}}_d) \tag{7}$$

From (7), it is clear that the error vanishes when $\delta\dot{\boldsymbol{\theta}}$ is equal to the difference between actual and desired velocity, i.e. $\delta\dot{\boldsymbol{\theta}} = \dot{\boldsymbol{\theta}}_d - \dot{\boldsymbol{\theta}}_{\text{old}}$ which is equivalent to $\mathbf{u} = \dot{\boldsymbol{\theta}}_d$. However, the system communication constraint implies that in each loop, only the velocity one of the joints is allowed to be modified. In other words, all elements but one in $\delta\dot{\boldsymbol{\theta}}$ have to be zero. In order to choose which joint's velocity to modify, we define a diagonal matrix $\boldsymbol{\mathcal{E}}_\theta$ as follows:

$$\boldsymbol{\mathcal{E}}_\theta = \text{diag}[\dot{\boldsymbol{\theta}}_d - \dot{\boldsymbol{\theta}}_{\text{old}}] \in \Re^{n\times n} \tag{8}$$

and the matrix corresponding to task space velocity errors:

$$\boldsymbol{\mathcal{E}} := [\mathbf{e}_1 \ \cdots \ \mathbf{e}_n] = \mathbf{W}\mathbf{J}(\boldsymbol{\theta})\boldsymbol{\mathcal{E}}_\theta \in \Re^{m\times n} \tag{9}$$

The joint to have its velocity updated corresponds to index k of the maximum entry of vector $\text{vec}_{i=1,\cdots n}[\|\mathbf{e}_i\|]$, i.e.

$$u_k = \dot{\theta}_{dk} \tag{10}$$

with

$$(\|\mathbf{e}_k\|, k) = \max_{i=1,\cdots n}[\|\mathbf{e}_i\|] \tag{11}$$

where the $\max$ function over a vector returns two outputs: a) the maximum element and b) the corresponding index. The joint with index k is thus the one that contributes the most to the error $\|\mathbf{e}\|$ if left unchanged, and choosing joint k to recieve the next velocity update command will therefore minimize $\|\mathbf{e}\|$, as desired.

IV. Simulation

As a first evaluation, the proposed scheme was implemented in simulation. We simulate a 7 DoF manipulator, modeled after one arm on our physical platform (see Figure 1, DH parameters are shown in Table I).

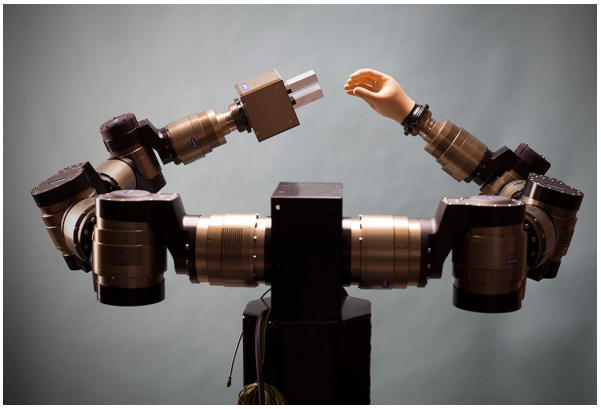

Fig. 1: The Dual 7 DoF manipulator used in the experiments.

In Cartesian space, we define the 7 DOFs as the 3 Cartesian coordinates of the position of the end effector (x, y, z), the 3 angles parameterizing the orientation of the end effector (α, β, γ), and the angular position ϕ of the elbow along a circle around the vector from shoulder to wrist. We assume that each joint has a local velocity controller that will accelerate with constant acceleration until it reaches the most recently commanded target velocity. When this velocity is reached, it will be kept until a new velocity command is received. We assume all links of the manipulator to be rigid.

A. Command ordering schemes

In our simulation, we compare the following 5 methods of sending commands to the joint-local velocity controllers

A Sequential with same control This approach is common in the literature, and consists of calculating the control signal once in each control loop, and sending each joint controller its individual velocity command in sequence [5], [11], [15].

Algorithm A Sequential commands with same control

```
while not done do
    Calculate u
    for i = 1 → n do
        send u_i to joint i
        joint i executes u_i
    end for
end while
```

B Sequential with updated control This approach is similar to **A**, but assuming that we have high-frequency state updates from sensors, and that calculating $\mathbf{u}$ is computationally inexpensive, we can recalculate the control signal before sending each command

C Synchronized non-predictive If the communication bus allows commands to be broadcast to all units, the velocity commands can be synchronized by letting the distributed controllers wait, and not start executing the latest command until a *synchronize* command is recieved. This means that the overall frequency of the control loop will be lower, with $f = \frac{\mathcal{F}}{n+1}$, but that motion is synchronized, so that the resulting velocity in cartesian space is always in the same direction as was commanded [16].

Algorithm B Sequential commands with updated control

```
while not done do
    for i = 1 → n do
        Calculate u
        send u_i to joint i
        joint i executes u_i
    end for
end while
```

Algorithm C Synchronized non-predictive

```
while not done do
    for i = 1 → n do
        Calculate u
        send u_i to joint i
    end for
    broadcast synchronize command
    joint i executes u_i, ∀i
end while
```

D Synchronized predictive Since the synchronization command will effectively delay the execution of the control command by one complete period of the control loop, performance may suffer. To counteract this, we predict the desired command for the next iteration of the control loop denoted by $\mathbf{u}_{\text{predicted}}$ and use it instead. This method may be very difficult to implement outside of simulation for tasks with significant feedback, but it is included here for completeness.

Algorithm D Synchronized predictive

```
while not done do
    for i = 1 → n do
        Calculate u_predicted
        send u_i to joint i
    end for
    broadcast synchronize command
    joint i executes u_i, ∀i
end while
```

E Proposed optimized ordering With this proposed method, we use the procedure described in Section III to determine which joint should be updated with the next command.

B. Free motion

We start with the simplest case, with free motion following a predefined generalized position trajectory $\mathbf{p}_d(t)$ and the corresponding derivative with respect to time $\dot{\mathbf{p}}_d(t)$. We track this with the following kinematic control law consisting of

Algorithm E Optimized ordering

while not done **do**
  calculate $\mathbf{u}$
  calculate optimal choice of $i = k$
  send u_k to joint k
  joint k executes u_k
end while

a proportional term of the position error plus a feedforward term of the desired velocity:

$$\mathbf{v}_d(t) = a\left[\mathbf{p}_d(t) - x(t)\right] + \dot{\mathbf{p}}_d(t) \tag{12}$$

where $\mathbf{v}_d(t)$ is the commanded generalized velocity, and a is a constant positive gain. The closed loop system at the kinematic level is a first order linear differential equation of the position error. We let $\mathbf{W}$ be a 7×7 identity matrix.

In the first test, we use the following target trajectory along a straight line parallel to the y axis:

$$\begin{array}{lllllll} x_d & = & -0.2 & & y_d & = & 0.15 + 0.3\frac{t}{5}, \\ z_d & = & 0.05 & & & & t = 0 \ldots 5\ s \\ \alpha_d & = & \pi/2 & & \beta_d & = & 3\pi/4 \\ \gamma_d & = & 0.05 & & \phi_d & = & -\pi/4 \end{array} \tag{13}$$

This is a simple trajectory to follow, and as we can see in Figure 2, most approaches perform similarily. While the delayed methods **A** and **C** perform slightly worse, we see acceptable behavior, with less than 1 mm root mean square deviation (RMSD) from approximately 100 msgs per second, or approximately 15 Hz in the main loop for all conventional methods. The proposed method is equivalent to conventional methods for this case.

We then proceed to make the target trajectory more challenging, by adding a small amplitude but high frequency sinusoidal wave to the orientation angle γ:

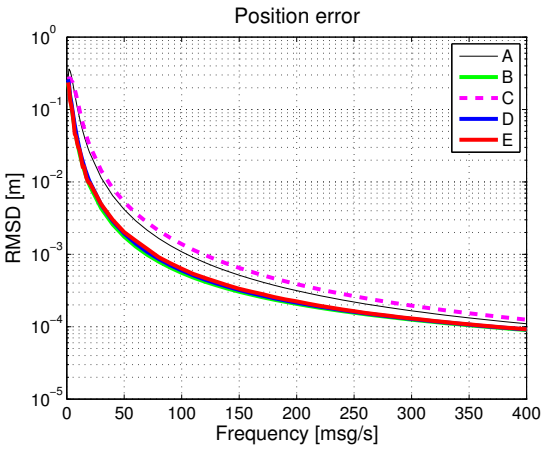

Fig. 2: Results from straight line trajectory following. The plot shows the root mean square deviation from the target trajectory. We note that methods A and C, which have inherent delays perform worse than the rest, and that the performances of methods B, D, and E are equivalent.

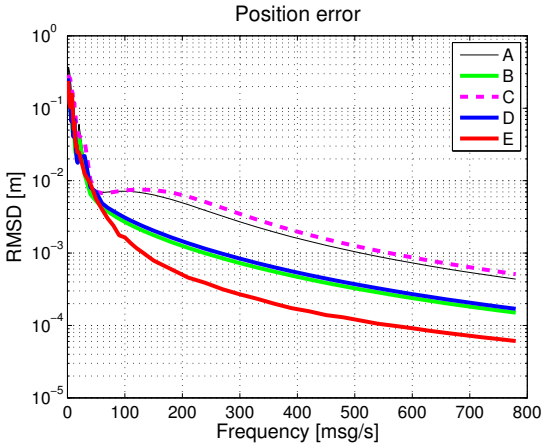

Fig. 3: Results from following straight line trajectory with oscillatory orientation. The plot shows the root mean square deviation from the target trajectory. Methods A and C, which have inherent delays, perform poorly, with resonance problems around 100 msg/s, while the performances of methods B and D are equivalent. The proposed approach E performs significantly better for this task.

$$\gamma_d \quad = \quad 0.05 + 0.005 \cdot \sin(20t) \tag{14}$$

The remaining parameters of the trajectory are unchanged from (13). As is shown in Figure 3, the position error for a given communication frequency is now significantly smaller for the proposed method **E**, while the relative performance of the other methods is similar to the previous task. The angle γ is here proportional to the 7th joint of the robot, and the required control frequency for the 7th joint is thus higher than for the other joints, fitting the proposed method.

C. Force Control

In the second simulation scenario we examine the performance for a hybrid force/position controlled task. Here, we use the same controller as above for controlling position in x, y, and the rotation angles, but use the following PI controller in the z direction to control contact force against the surface:

$$v_{dz}(t) = k_P F_e(t) + k_I \int_0^t F_e(\tau) d\tau \tag{15}$$

where the force error $F_e(t)$ is defined as the difference between desired and actual force, $F_d(t) - F(t)$, and k_P and k_I are positive constants. We set the weight matrix $\mathbf{W}$ to be a 7×7 identity matrix. We assume a surface with the normal in the z direction, located at $z = 0.05$, which gives a similar trajectory as in the previous section. The interaction stiffness is defined as a linear spring with stiffness 10 kN/m. We make the problem more challenging by adding step-shaped disturbances of an amplitude of 1 mm, defining the surface with the following function:

$$z \quad = \quad 0.05 + \text{round}[0.001 sin(5t)] \tag{16}$$

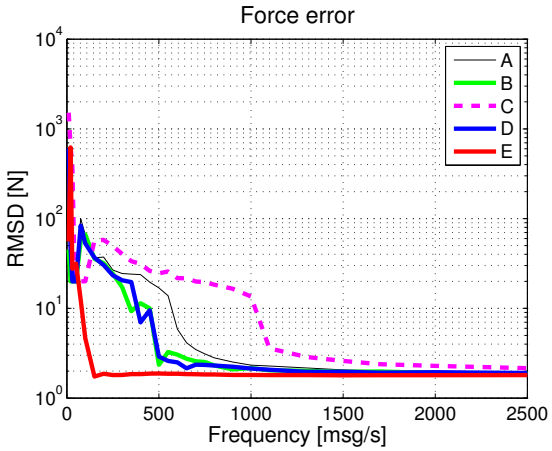

Fig. 4: Results from tracking a target force along a straight line trajectory with step-function disturbances. The plot shows the root mean square deviation from the target force. Methods A and C, which have inherent delays, perform poorly, while the performances of methods B and D are roughly equivalent. The proposed approach E performs significantly better for this task.

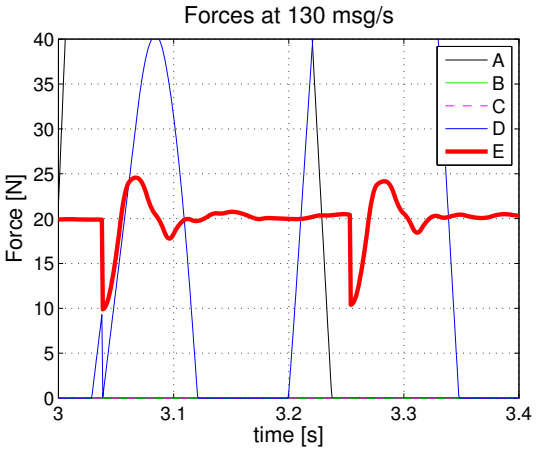

Fig. 5: An illustration of the force tracking performance at 130 msg/s. The proposed method is stable and handles the disturbances well, while the other methods perform very poorly and are mostly outside the plotted region. The target force is 20 N, and stepfunction disturbances are given at t=3.04 s and t=3.25 s.

This is a more challenging control problem than the previous section, and the system requires higher messenging frequencies to perform well. Figure 4 shows the force error as a function of messenging frequency.

We see that the proposed method **E** has significantly lower messenging frequency requirements for acceptable performance, and gives stable performance at 130 Hz, see Figure 5. At approximately 3 times higher messenging frequencies, 450 Hz, methods **B** and **D** begin to stabilize, see Figure 6. Note that the predictive method **D** may not be realistic to implement for this scenario, unless a very detailed model of the force disturbance is available. The non-predictive synchronized approach **C** needs more than 1000 messages per second for stable performance in this scenario, which is approximately 8 times more than the proposed method.

A look into the relative frequencies of the different joints illustrates how the proposed method works. Figure 7 shows how the relative frequency is temporarily raised for joints 2 and 4, that have a high influence on the z position and thus for the force compensation. When the force error has been corrected, other joints are needed to correct the pose, and have their relative frequency temporarily raised. When all deviations have been handled, the relative joint frequencies return to the distribution before the disturbance.

D. Discussion

In this section we have shown the relative performance for different methods to address the joint-local velocity controllers of a simulated robot arm. We have shown that for simple, straight-line trajectories, any approach performs well with relatively low update frequencies. When the trajectory becomes more complex, by adding a high frequency component, the proposed method begins to perform better than the traditional approaches. When we attempt force control with step-shaped disturbances, we see a significant performance increase with the proposed method.

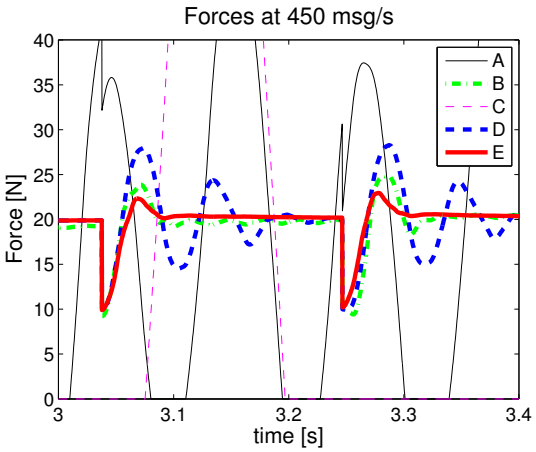

Fig. 6: An illustration of the force tracking performance at 450 msg/s. The proposed method is stable and handles the disturbances well, while the other methods perform worse. Methods B and D start to perform acceptably at this frequency. The target force is 20 N, and stepfunction disturbances are given at t=3.04 s and t=3.25 s.

Note that the control law functions for the simulation were chosen arbitrarily, and that the performance as a function of communication frequency depends on the choice of function, tracked trajectory, and interaction stiffnesses. However, the relative performance between the different approaches is similar for different choices for these parameters. E.g, for stiffer interaction with higher gains, the required frequencies will be higher, but the relative order of when the methods give stable performance will remain unchanged.

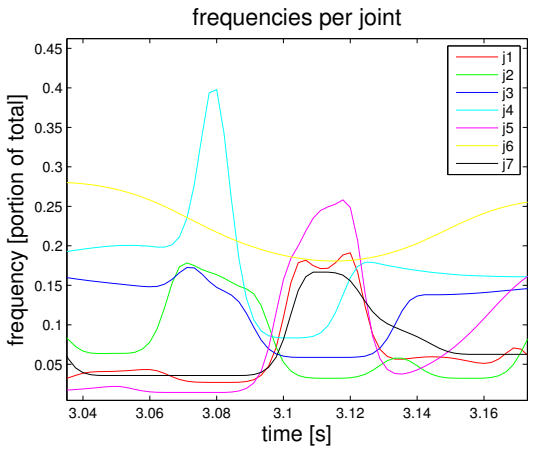

Fig. 7: An illustration of the relative messenging frequencies of the different joints 450 msg/s. A stepfunction disturbance is given at t=3.045 s and the realitive frequencies of joints 2 and 4 are raised to counter the force offset, and when this has been handled, the frequencies of joints 1, 5, and 7 are temporarily raised to correct the pose error caused by the force compensation. The plots have been smoothed with an adaptive moving average.

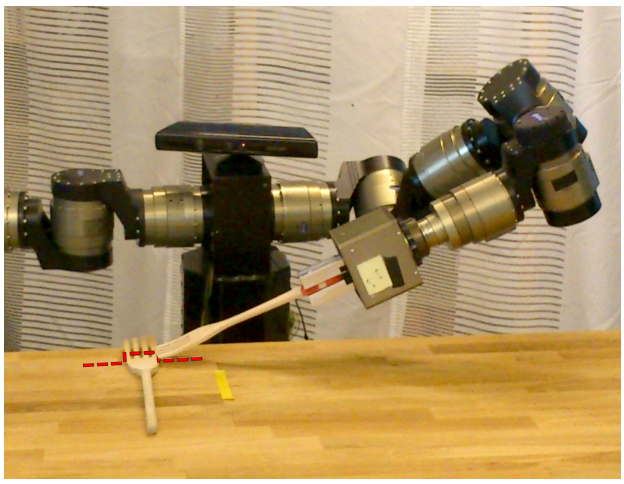

Fig. 8: In the experiment, the robot traced a straight line (dotted red in image) along a counter with a wooden fork, trying to apply a constant force of 5 N.

V. Experiments

As a validation of the proposed optimized serial addressing scheme, we implemented it on our dual-arm mobile robot.

A. Experimental Setup

The performance specifications of the real robot are the same as the simulated robot, but the real robot has upper bounds on motor torques, and the physical system contains more sources of error and noise. The 7 DoF arms of the robot have joint-local velocity controllers, and are connected via a 500 kbaud CAN bus and support broadcasted synchronization commands. Including the time it takes for the joint-local control modules to respond to a command, it is possible to send up to 1200 commands per second on the bus. In this setup, forces were measured with a ATI mini 45 force/torque sensor located at the wrist of the robot and connected to a separate CAN bus that did not interfere with the bus sending the velocity commands.

The experiment was carried out by letting the robot hold a wooden fork and trace it along a countertop with a constant normal force. A second fork had been placed on the countertop to provide a disturbance similar to the periodic step function in Section IV-C, see Figure 8. The trajectory to follow was given as:

$$\begin{aligned} x_d &= 0.3 - 0.03t, & t &= 0 \ldots 7\ s \\ y_d &= 0.4 & z_d &= -0.11 \\ \alpha_d &= \pi & \beta_d &= \pi/12 \\ \gamma_d &= 0 & \phi_d &= -\pi/4 \end{aligned} \tag{17}$$

The table surface is located at $z = -0.11$, and the second fork is approximately 9 mm thick. The controller gains were set to $\alpha = 1$, $k_P = 0.03$, and $k_I = 0.01$. The interaction stiffness of the setup was measured to be 1.73 kN/m, and the target force was set to 5 N. This is deliberately chosen to be less than in the simulation, so that the performance of all schemes would be acceptable in the measurable range. We compare the performance of schemes **A**, **B**, **C**, and **E** from Section IV. Scheme **D**, with prediction, was left out of this treatment, as it was not possible to implement an accurate enough predictor for the force measurements.

The experiment was run several times for each scheme, and artificial delays of different were inserted into the communication loop to evaluate the performance of different messenging frequencies. The different schemes also insert different delays into the system, for calculating the control signal. Schemes **A** and **C**, could thus be run at slightly higher messenging frequencies, as they calculate the control signal less often. The messenging frequencies presented here were the actual achieved frequencies, as measured while running the experiments. Figure 9 shows the results, in terms of root mean square deviation (RMSD) from desired force. The error does not reach zero for higher frequencies, since the physical robot system takes a finite amount of time to correct the step function disturbances, and due to noise and errors in the physical setup.

B. Results and Observations

The proposed scheme performs significantly better than the others. For this specific task and controller, the system is stable down to less than 200 messages per second, while the second best performing scheme (**B**) needs more than 50% higher communication speeds for stable performance. The two remaining schemes require several times higher communication speeds for the same performance. Keeping in mind that there are 7 DoFs on the robot, for methods A and C the effective control frequency is 1/7th of the communication frequency. Comparing the proposed method E with

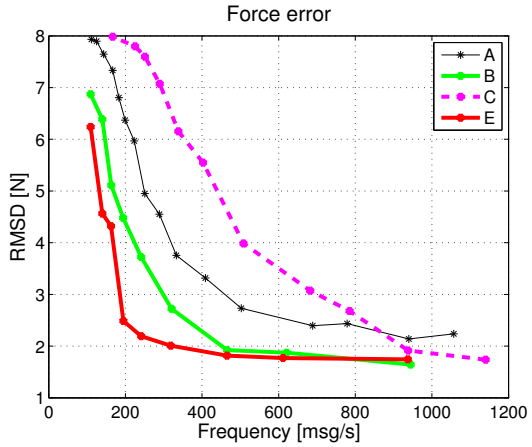

Fig. 9: Results from the force tracking experiment. The plot shows the root mean square deviation from the target force. Methods A and C, which have inherent delays, perform poorly, while the performance of method B is better. The proposed approach E performs significantly better for this task. Method D is not implemented.

the synchronized method C, the effective control frequency for the proposed method is approximately 1/2 of the the communication frequency for this task. Thus, for a given communication frequency, the proposed method will allow more than 3 times as many units to be connected to the same bus without loss of performance.

VI. Conclusion

This paper proposes a method for determinining the optimal joint to target with updated velocity commands for a serial link manipulator with joint-local velocity controllers. The aim of the method is to improve performance for systems at low messenging frequencies. Simulations show that for the simplest case of undisturbed free motion along a straight line, the proposed scheme performs as good as any previously published method. For more complex motions, including high-frequency free motion, and hybrid velocity/force controlled motion, the proposed method performs significantly better than previous methods. Finally, the proposed method was implemented on a robot, and results for the non-trivial task of hybrid velocity/force control are compared for different methods at different communication frequencies, again showing that the proposed method performs significantly better for lower messenging frequencies. Implementing the proposed method on a humanoid robot with a high number of DOFs enables one to take advantage of the low cabling demands for connecting all joint-local controllers to a common bus, while providing performance equivalent of using several parallell buses.

Acknowledgment

The work presented in this paper has been funded by The Swedish Research Council and the European Union FP7 project RoboHow.Cog (FP7-ICT-288533). The authors gratefully acknowledge the support.

References

[1] A. Albers, S. Brudniok, J. Ottnad, C. Sauter, and K. Sedchaicham, "Upper body of a new humanoid robot - the design of ARMAR III," in *IEEE-RAS International Conference on Humanoid Robots*, Dec 2006, pp. 308–313.

[2] A. Hernandez-Herdocia, A. Shademan, and M. Jägersand, "Building a mobile manipulator from off-the-shelf components," in *IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, july 2010, pp. 1116–1121.

[3] K. Kaneko, K. Harada, F. Kanehiro, G. Miyamori, and K. Akachi, "Humanoid robot HRP-3," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, sept. 2008, pp. 2471–2478.

[4] C. J. Paredis, H. B. Brown, and P. K. Khosla, "A rapidly deployable manipulator system," in *International Conference on Robotics and Automation*. Minneapolis, Minnesota: IEEE, 1996, pp. 1434–39.

[5] T. Lamarche and W.-H. Zhu, "A virtual decomposition control based communication network for modular robots applications," in *Proceedings of 16th International Conference on Computer Communications and Networks*, aug. 2007, pp. 1321–1326.

[6] H. Brown, M. Schwerin, E. Shammas, and H. Choset, "Design and control of a second-generation hyper-redundant mechanism," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2007, pp. 2603–2608.

[7] A. Flores-Abad and A. Arpidez, "Embedded control system for a 5-DOF manipulator by means of SPI bus," in *Electronics, Robotics and Automotive Mechanics Conference*. IEEE, 2009, pp. 123–128.

[8] J. de Gea, J. Lemburg, T. Roehr, M. Wirkus, I. Gurov, and F. Kirchner, "Design and control of an intelligent dual-arm manipulator for fault-recovery in a production scenario," in *IEEE Conference on Emerging Technologies Factory Automation*, sept. 2009, pp. 1–5.

[9] M. Fuchs, C. Borst, P. Giordano, A. Baumann, E. Kraemer, J. Langwald, R. Gruber, N. Seitz, G. Plank, K. Kunze, R. Burger, F. Schmidt, T. Wimboeck, and G. Hirzinger, "Rollin' justin - design considerations and realization of a mobile platform for a humanoid upper body," in *IEEE International Conference on Robotics and Automation*, may 2009, pp. 4131 –4137.

[10] C. Jiang, W. Chen, Q. Shi, and B. Xu, "Research and design on distributed controllers for mechanical arms of humanoid robot," in *2nd International Asia Conference on Informatics in Control, Automation, and Robotics*. IEEE, 2010, pp. 88–91.

[11] Y. Zhang, X. Zeng, and X. Wang, "Control system design based on CANopen framework for multi-legged robot with hand-fused foot," *Applied Mechanics and Materials*, vol. 42, pp. 307–312, 2011.

[12] K. Kaneko, F. Kanehiro, M. Morisawa, K. Akachi, G. Miyamori, A. Hayashi, and N. Kanehira, "Humanoid robot HRP-4 - humanoid robotics platform with lightweight and slim body," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, sept. 2011, pp. 4400–4407.

[13] J.-Y. Kim, I.-W. Park, J. Lee, M.-S. Kim, B. kyu Cho, and J.-H. Oh, "System design and dynamic walking of humanoid robot KHR-2," in *IEEE International Conference on Robotics and Automation*, april 2005, pp. 1431–1436.

[14] C. Smith and H. I. Christensen, "Constructing a high performance robot from commercially available parts," *Robotics and Automation Magazine*, vol. 16, pp. 75–83, Dec 2009.

[15] H. Jia, W. Zhuang, Y. Bai, P. Fan, and Q. Huang, "The distributed control system of a light space manipulator," in *International Conference on Mechatronics and Automation*, aug. 2007, pp. 3525–3530.

[16] K. Xiang, Z. Sun, H. Dai, Q. Chen, and J. Liu, "Can-bus based distributed control system for hydraulic turbine blade repairing robot," in *Proceedings of the Third international conference on Intelligent robotics and applications*, 2010, pp. 695–704.