Model Predictive Path Tracking via Middleware for Networked Mobile Robot over IP Network

Yodyium Tipsuwan
Student Member, IEEE
ytipsuw@unity.ncsu.edu

Mo-Yuen Chow
Senior Member, IEEE
chow@eos.ncsu.edu

Department of Electrical and Computer Engineering, North Carolina State University, Raleigh, NC, USA, 27606

***Abstract*–The potential use of IP networks for real-time high performance robots and automation is enormous and appealing. A widely attractive objective for an IP-based mobile robot is to control a mobile robot over the IP network to track a predefined path. This paper proposes a model predictive path tracking control methodology over an IP network via middleware. In addition to the normal use of middleware, this paper utilizes middleware to schedule a control parameter for IP network delay compensation. The parameter is adjusted externally at the output of the path tracking algorithm with respect to the current network traffic conditions and a predictive performance measure computed by a neural network. Simulation results show that the mobile robot with neural network middleware provides significantly better IP networked control system performance.**

***Index Terms*–mobile robot, Internet, networks, adaptive control, control systems, DC motors, distributed control, real time system.**

I. Introduction

A RECENT and advancing trend in the networked control area is to replace specialized industrial networks with a general computer network such as Ethernet and wireless Ethernet to control high performance applications in the area of distributed control and teleoperation over the Internet or IP (Internet Protocol) networks [1]. A general protocol like Ethernet has advantages such as its affordability, widespread usage, and well-developed infrastructure for Internet connection. Nevertheless, once a networked control system is connected through the Internet, the network delays induced by IP can vary substantially, depending on network traffic conditions. Several methodologies to handle network delays have been applied on some robotic applications. For example, gain scheduling [2], wave variables [3, 4], and event-based approaches [5] have been applied on robotic manipulator control. Likewise, control of a mobile robot over an IP network is another robotic application that gains much attention. An widely attractive objective for an IP-based mobile robot is to control the mobile robot over an IP network to track a predefined path. Some techniques have been proposed to handle network delays in this particular problem such as gain scheduling [6] and predictive control [7].

This paper proposed a model predictive path tracking control methodology over an IP network via middleware. Middleware is an implementation to seamlessly link a network and a control application together [8]. In general, middleware is designed to monitor network traffic conditions, and handle network transmissions between a source and a destination. In addition to normal use of middleware, this paper utilizes middleware to schedule a control parameter externally for the network delay compensation. Thus, the path tracking algorithm used needs not to be modified. In this proposed approach, the middleware measures network traffic conditions to predict the current position of the robot. Then, the future performance of the robot is calculated by a neural network. If the performance of the robot cannot be satisfied as required, the middleware will virtually adapt the control parameter at the output of the path tracking controller.

II. System Description

A. System Configuration

In this paper, we consider a distributed networked mobile robot system configuration over an IP network as shown in Fig. 1.

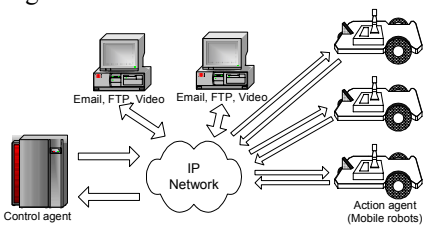

Fig. 1. An overall distributed networked mobile robot system over an IP network.

1) IP Network

The IP network under consideration links all action agents including mobile robots. In this paper, we assume that control and action agents use UDP (User Datagram Protocol) as the layer-4 protocol on the IP network to avoid additional delays from retransmissions.

2) Control Agent

Each control agent can be a high performance computing unit to manage operations of action agents. Periodically, the control agent converts the sensory signals in a packet sent across the IP network from each action agent to numerical feedback data for closed-loop control. The control or reference signal from the control agent is then sent back as a packet to each action agent via the IP network as well.

3) Action Agent

Each action agent such as a mobile robot contains an action agent controller and an action agent plant. The

action agent controller is a simple hardware unit to periodically convert the control or reference signal in a packet from the control agent to an actual signal to control the action agent plant. The sensory output of the action agent plant such as a motor speed is also monitored and sent back to the control agent.

B. Mobile Robot Model

The robot used to illustrate the proposed approach is a differential drive mobile robot with two driving wheels and one caster wheel [9] as shown in Fig. 2.

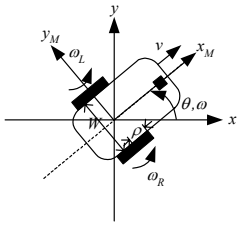

Fig. 2. Differential drive mobile robot.

The kinematics of the mobile robot in Fig. 2 is described as:

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} \frac{\rho}{2} & \frac{\rho}{2} \\ \frac{\rho}{W} & -\frac{\rho}{W} \end{bmatrix} \begin{bmatrix} \omega_R \\ \omega_L \end{bmatrix}, \quad (1)$$

$$\dot{x} = v\cos\theta, \quad (2)$$

$$\dot{y} = v\sin\theta, \quad (3)$$

$$\dot{\theta} = \omega, \quad (4)$$

where (x, y) is the position in the inertial coordinate, (x_M, y_M) is the position in the robot coordinate, θ is the azimuth angle of the robot, v is the linear velocity of the robot, W is the distance between two wheels, ρ is the radius of wheels, ω is the angular velocity of the robot, and ω_L and ω_R are the angular velocities of the left and right wheels, respectively.

C. Path Tracking Algorithm

A generalization of the quadratic curve approach proposed in [9] is used as the path tracking algorithm to illustrate the proposed approach. The main concept of this path tracking algorithm is to move the robot along a quadratic curve to a reference point on a desired path. A point on the path is described in the inertial coordinate as $(x_p(s), y_p(s))$, where s is the distance traveled on the path. By assuming that the orientation of the mobile robot comes close to the right value in the motion along the desired path, this algorithm controls only the position of the robot regardless to its orientation. This algorithm is suitable for real-time usage because of its simple computation with minimal amount of information compared to other approaches. The algorithm is outlined as:

1) Based on the current robot position $\mathbf{x}(i) = \begin{bmatrix} x(i) & y(i) & \theta(i) \end{bmatrix}^{\mathrm{T}}$, where $i \in \mathbb{N}^+$ is the iteration number, optimize:

$$\min_s \sqrt{\left(x_p(s) - x(i)\right)^2 + \left(y_p(s) - y(i)\right)^2}, \quad (5)$$

to find $s = s_0$ that gives the closest distance between the robot and the path. Depending on the forms of $x_p(s)$ and $y_p(s)$, this optimization could be performed in real-time by using a closed-form solution, or a lookup table and a numerical technique such as linear interpolation. The iteration number i can be thought of as the sampling time index of the path tracking controller if $t_{i+1} - t_i$ is constant

2) Compute the reference point for the robot to track by:

$$s(i) = s_0 + \frac{s_{\max}}{1 + \beta\left|\max \kappa(a)\right|}, \kappa(s) = \frac{d\theta_p(s)}{ds}, \quad (6)$$

$$\mathbf{x}_r(i) = \begin{bmatrix} x_r(i) & y_r(i) & \theta_r(i) \end{bmatrix}^{\mathrm{T}}, \quad (7)$$

where $s_{\max} \in \mathbb{R}^+$ and $\beta \in \mathbb{R}^+$ are positive constants, $a \in [s_0, s_0 + s_{\max}]$, $x_r(i) = x_p(s(i))$, $y_r(i) = y_p(s(i))$, $\theta_r(i) = \theta_p(s(i))$, and $\kappa(s)$ is the curvature of the path. A higher value of β will result in a shorter distance of $\mathbf{x}_r(i)$ from $(x_p(s_0), y_p(s_0))$ as well as a higher $\left|\max \kappa(a)\right|$.

3) Compute the error $\mathbf{x}_r(i) - \mathbf{x}(i)$, and transform this error to the error in the robot coordinate as:

$$\mathbf{e}(i) = \begin{bmatrix} e_x & e_y & e_\theta \end{bmatrix}^{\mathrm{T}} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \left(\mathbf{x}_r(i) - \mathbf{x}(i)\right). \quad (8)$$

4) Find a quadratic curve that links between $\mathbf{x}(i)$ and $\mathbf{x}_r(i)$ from:

$$y_M = A(i) x_M^2, \text{ where } A(i) = \mathrm{sgn}(e_x)\frac{e_y}{e_x^2}. \quad (9)$$

The robot will move forward if $\mathbf{x}_r(i)$ is ahead of the robot ($e_x > 0$). On the other hand, it moves backward when $\mathbf{x}_r(i)$ is at the back of the robot ($e_x < 0$).

5) Compute the reference linear and angular velocities of the robot along the quadratic curve. The original equations of the velocities are:

$$v_r(i) = \mathrm{sgn}(e_x)\sqrt{\dot{x}_M^2\left(1 + 4A^2(i) x_M^2\right)}, \quad (10)$$

$$\omega_r(i) = \frac{2A\dot{x}_M^3}{v_r^2(i)}. \quad (11)$$

Let x_M at $t_i \le t < t_{i+1}$ be given by:

$$x_M = K(i)(t - t_i), \quad (12)$$

where

$$K(i) = \mathrm{sgn}(e_x)\frac{\alpha}{1 + \left|A(i)\right|}, \quad (13)$$

and $\alpha \in \mathbb{R}^+$ is a positive constant used as a speed factor. The robot will move fast, if α is set to a high value, and vice versa. If $(t - t_i)$ is very small, $v_r(i)$ can be approximated during $t_i \le t < t_{i+1}$ by:

$$v_r^2(i) = K^2(i)\left(1 + 4A^2(i)K^2(i)(t - t_i)^2\right) \simeq K^2(i). \quad (14)$$

Thus, (10) and (11) can be approximated by:

$$\hat{v}_r(i) \simeq K(i), \quad (15)$$

$$\hat{\omega}_r(i) \simeq 2A(i)K(i). \quad (16)$$

The reference speeds of both wheels are calculated by:

$$\omega_{R,r}(i) = \frac{\hat{v}_r(i)}{\rho} + \frac{W\hat{\omega}_r(i)}{2\rho}, \quad (17)$$

$$\omega_{L,r}(i) = \frac{\hat{v}_r(i)}{\rho} - \frac{W\hat{\omega}_r(i)}{2\rho}. \quad (18)$$

6) Repeat all steps by going back to *1)* and set $i = i+1$.

III. Problem Formulation

In order to control a mobile robot to track a predefined path over a network, the control agent computes and sends the reference speed $\omega_{L,r}(i)$ and $\omega_{R,r}(i)$ in a packet across the network at every iteration *i* to the robot as shown in Fig. 3.

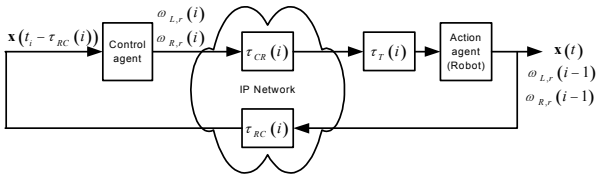

Fig. 3. Data flow of networked mobile robot.

The path tracking computation at iteration *i* starts when the control agent receives the feedback data in a packet from the mobile robot at time $t = t_i$. Compared with the network delays, the computation time at the control agent is usually relatively insignificant. Thus, the computation could be assumed to finish at $t = t_i$ as well. The basic arrival feedback data in this case is the robot position $\mathbf{x}(t_i - \tau_{RC}(i))$, where $\tau_{RC}(i)$ is the network delay from the robot to the control agent at *i*. The control agent then sends $\omega_{L,r}(i)$ and $\omega_{R,r}(i)$ to the robot once the computation is finished. Likewise, $\omega_{L,r}(i)$ and $\omega_{R,r}(i)$ are also delayed by the network. The network delay to send these reference speeds to the mobile robot is defined as $\tau_{CR}(i)$. The robot then periodically monitors and updates the reference speeds by the arrival data of $\omega_{L,r}(i)$ and $\omega_{R,r}(i)$ at every sampling time period *T*. The waiting time to update the reference speeds is defined as $\tau_T(i)$.

There are two concerns in the use of the original path tracking algorithm due to $\tau_{CR}(i)$ and $\tau_{RC}(i)$:

1) Due to $\tau_{RC}(i)$, the control agent does not have the current robot position $\mathbf{x}(t_i)$, but $\mathbf{x}(t_i - \tau_{RC}(i))$.

2) The reference speeds $\omega_{L,r}(i)$ and $\omega_{R,r}(i)$ are computed at $t = t_i$, but will be applied at $t = t_i + \tau_{CR}(i) + \tau_T(i)$.

If the control agent directly uses $\mathbf{x}(t_i - \tau_{RC}(i))$ as $\mathbf{x}(i)$ to compute $\omega_{L,r}(i)$ and $\omega_{R,r}(i)$, and if $\mathbf{x}(t_i - \tau_{RC}(i))$ and $\mathbf{x}(t_i)$ are very different, then the result may be far away from what it actually should be. In addition, even if the control agent uses $\mathbf{x}(t_i)$ to compute $\omega_{L,r}(i)$ and $\omega_{R,r}(i)$, the robot might have already moved to another position at $t = t_i + \tau_{CR}(i) + \tau_T(i)$ when $\omega_{L,r}(i)$ and $\omega_{R,r}(i)$ are applied. Thus, the robot response can be undesirable.

IV. Model Predictive Path Tracking

To handle these two delay concerns, position prediction and performance prediction are applied as follows.

A. Position Prediction

A model predictive approach is utilized to compute the future position of the mobile robot at $t = t_i + \tau_{CR}(i) + \tau_T(i)$ for using in the original path tracking algorithm. A future position at $t = t_i + \tau_{CR}(i) + \tau_T(i)$ defined as $\mathbf{x}(t_i + \tau_{CR}(i) + \tau_T(i))$ is predicted from $\mathbf{x}(t_i - \tau_{RC}(i))$. The predictive position defined as $\hat{\mathbf{x}}(t_i + \tau_{CR}(i) + \tau_T(i))$ is then used as $\mathbf{x}(i)$ in the path tracking algorithm so that $\omega_{L,r}(i)$ and $\omega_{R,r}(i)$ should be actually applied on time. The formulation is based on the difference between $\mathbf{x}(t_i + \tau_{CR}(i) + \tau_T(i))$ and $\mathbf{x}(t_i - \tau_{RC}(i))$ defined as:

$$\begin{aligned}\boldsymbol{\Delta}_\tau \mathbf{x}(i) &= \begin{bmatrix}\Delta_\tau x(i) & \Delta_\tau y(i) & \Delta_\tau \theta(i)\end{bmatrix}^{\mathrm{T}} \\ &= \mathbf{x}(t_i + \tau_{CR}(i) + \tau_T(i)) - \mathbf{x}(t_i - \tau_{RC}(i)) \quad (19)\end{aligned}$$

Before the mobile robot receives $\omega_{L,r}(i)$ and $\omega_{R,r}(i)$, both left and right wheels have been controlled by using $\omega_{L,r}(i-1)$ and $\omega_{R,r}(i-1)$ as the reference speeds. Thus, the robot can be assumed to move with a constant linear velocity *v* and a constant angular velocity ω if the controllers of both wheels work perfectly and the weight of the robot is light. From this assumption and (2)-(4), $\boldsymbol{\Delta}_\tau \mathbf{x}(i)$ can be approximated as follows.

$$\Delta_\tau \theta(i) \simeq 2A(i-1)K(i-1)(\tau_{CR}(i) + \tau_T(i) + \tau_{RC}(i)), \quad (20)$$

$$\begin{aligned}\Delta_\tau x(i) \simeq \frac{1}{2A(i-1)}\big[&\sin\theta(t_i + \tau_{CR}(i) + \tau_T(i)) - \\ &\sin\theta(t_i - \tau_{RC}(i))\big], \quad (21)\end{aligned}$$

$$\begin{aligned}\Delta_\tau y(i) \simeq \frac{1}{2A(i-1)}\big[&\cos\theta(t_i - \tau_{RC}(i)) - \\ &\cos\theta(t_i + \tau_{CR}(i) + \tau_T(i))\big], \quad (22)\end{aligned}$$

where $A(i-1)$ and $K(i-1)$ are the quadratic and speed factors in (9) and (13) computed at $i-1$, respectively. These factors can be easily found from (17) and (18) by using $\omega_{L,r}(i-1)$ and $\omega_{R,r}(i-1)$. Therefore, the mobile robot has to send $\omega_{L,r}(i-1)$ and $\omega_{R,r}(i-1)$ back to the control agent as the feedback data along with $\mathbf{x}(t_i - \tau_{RC}(i))$ for prediction. In addition, the delays $\tau_{CR}(i)$ and $\tau_{RC}(i)$ can be combined for simplicity.

B. Performance Measure

In addition to position prediction, the model predictive approach is also utilized to measure the robot performance. The performance of robot path tracking using model predictive position with respect to the network delays is directly dependent on $\boldsymbol{\Delta}_\tau \mathbf{x}(i)$, which can be measured by:

$$J(i) = \|\boldsymbol{\Delta}_\tau \mathbf{x}(i)\|, \quad (23)$$

where $\|\cdot\|$ is a vector norm. This measure indicates how much the robot moves away from $\mathbf{x}(t_i - \tau_{RC}(i))$ when the reference speeds have not been updated because of the

network delays. Large position and orientation difference could result in the high errors of $\omega_{L,r}(i)$ and $\omega_{R,r}(i)$, which lead the robot to a wrong position.

Actual $J(i)$ can be measured directly from the actual robot platform, or can be approximated from (20)-(22). For example, if 2-norm is used, (23) can be expressed as:

$$J(i) \simeq \sqrt{\Delta_\tau^2 x(i) + \Delta_\tau^2 y(i) + \Delta_\tau^2 \theta(i)}, \quad (24)$$

Using trigonometry rules, we can arrange:

$$\Delta_\tau^2 x(i) + \Delta_\tau^2 y(i) = \frac{1-\cos\Delta_\tau\theta(i)}{2A^2(i-1)}. \quad (25)$$

Therefore, the performance measure *J* becomes:

$$J(i) \simeq \sqrt{\frac{1-\cos\Delta_\tau\theta(i)}{2A^2(i-1)} + \Delta_\tau^2\theta(i)} \quad (26)$$

C. Performance Prediction

From (20) and (26), $J(i)$ can be viewed as a function of three variables: $\tau_{CR}(i)+\tau_T(i)+\tau_{RC}(i)$ defined as $\tau(i)$, $A(i-1)$, and $K(i-1)$. Since $J(i)$ depends on $A(i-1)$, $K(i-1)$, and $\tau(i)$, the control agent can also predict the future performance measure at $i+1$ defined as $\hat{J}(i+1)$ by using $A(i)$, $K(i)$, and the predictive delay at $i+1$ defined as $\hat{\tau}(i+1)$. If $\hat{J}(i+1) > \varepsilon$, where $\varepsilon \in \mathbb{R}^+$ is the maximal tolerance of the performance measure, the control agent will select the optimal *K* to replace $K(i)$ to ensure $\hat{J}(i+1) \le \varepsilon$. The control agent can then compute $\omega_{L,r}(i)$ and $\omega_{R,r}(i)$ from the optimal *K* and $A(i)$. The prediction can be easily done if $\hat{\tau}(i+1)$ is constant. However, for a random delay network such as an IP network, a more sophisticated algorithm is required to determine $\hat{J}(i+1)$ from $\hat{\tau}(i+1)$ based on delay characteristics, which will be described in a later section.

V. IP Network Delay Characterization

In order to determine $\hat{J}(i+1)$ from $\hat{\tau}(i+1)$ on an IP network, $\tau(i)$ is characterized and determined based on RTT (roundtrip time) delays as follows.

A. IP Network Delay Characteristics

To illustrate actual IP network delay characteristics, RTT delays are measured from an Ethernet network in ADAC (Advanced Diagnosis And Control) Lab at North Carolina State University (NCSU) to the destinations listed in Table 1 for 24 hours (00:00-24:00). Statistical measures of the RTT delays are also shown in Table 1. The corresponding histograms of the RTT delays to approximate probability densities are shown in Fig. 4.

B. Determine $\hat{\tau}(i+1)$

The shapes of the histograms in Fig. 4 indicate the higher probability to have RTT delays that are closer to the median than the mean. The median of RTT delays can be treated as $\hat{\tau}(i+1)$ for position prediction since the median

TABLE 1. STATISTICAL MEASURES (MINIMUM, MEDIAN, MEAN, AND MAXIMUM) OF RTT DELAYS MEASURED FROM ADAC LAB AT NCSU TO WWW.LIB.NCSU.EDU, WWW.VISITNC.COM, WWW.UTEXAS.EDU, AND WWW.KU.AC.TH.

Destination host	$\tau_{\min}$ (sec)	τ_{median} (sec)	τ_{mean} (sec)	$\tau_{\max}$ (sec)
www.lib.ncsu.edu	0.000435	0.000471	0.000580	0.0862
www.visitnc.com	0.0166	0.0232	0.0326	0.7562
www.utexas.edu	0.0622	0.0627	0.0629	0.1187
www.ku.ac.th	0.0045	0.3150	0.3730	227.7095

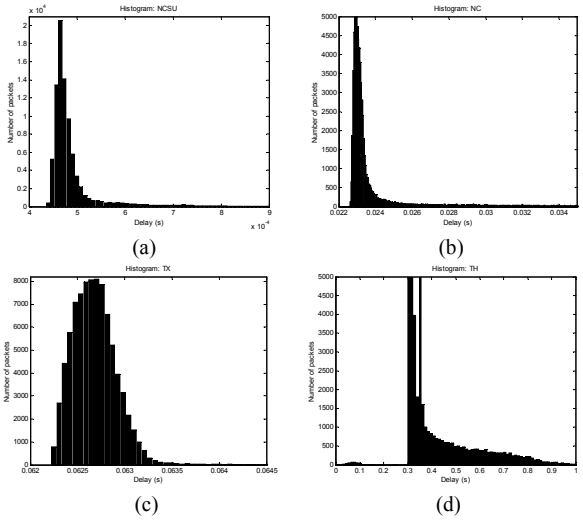

Fig. 4. Histograms of RTT delays measured from ADAC lab at NCSU to: (a) www.lib.ncsu.edu. (b) www.visitnc.com. (c) www.utexas.edu. (d) www.ku.ac.th.

is a good representation of the majority of RTT delays in this case. On the other hand, the mean of RTT delays is used for performance prediction for worse approximation of $\hat{J}(i+1)$.

C. Real-Time IP Network Traffic Condition Monitoring

The median and mean of RTT delays can be identified in real-time for position and performance prediction during a short time period by transmitting probing packets to the robot and by measuring their RTT delays. Probing packets are sent with a packet index *j*, $j = 0,1,2,\ldots$, and the current time defined as the sending time $t_s(j)$ to the mobile robot at every sampling time T_P. The robot will return a corresponding acknowledge packet including *j* and $t_s(j)$ after it receives a probing packet. The acknowledge packet arrives at the control agent at $t = t_A(i)$, where $t_A(i)$ is the arrival time of the acknowledge packet. Thus, the RTT delay of the roundtrip *j* can be computed by $\text{RTT}(j) = t_A(j) - t_S(j)$. All RTT delays are sorted and stored in a link-list data structure for convenience to compute the median and mean of RTT delays after *N* packets are acknowledged. The index *j* is reset to zero when $j > N$. In addition, both median and mean are also used to represent network traffic conditions.

VI. Neural Network Middleware

Position prediction and performance prediction can be

implemented in middleware separately from the tracking algorithm implementation. With middleware, the path tracking controller has more flexibility to be upgraded and used on different traffic conditions. The mobile robot system can be ported to other types of network protocols and environments by simply changing the controller parameters with respect to a suitable network traffic model. The overall process is depicted in Fig. 5.

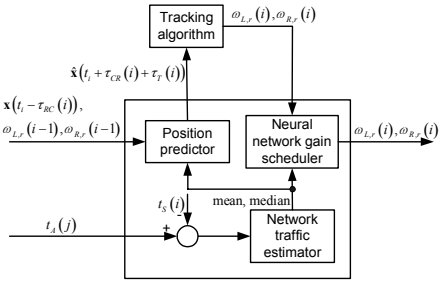

Fig. 5. Gain scheduler middleware for predictive path tracking.

The middleware is composed of three parts:

1) Traffic estimator

The traffic estimator is used to measure the median and mean of RTT delays of probing packets. The traffic delay median and mean are then used for position and performance prediction, respectively.

2) Position predictor

The position predictor is activated when the feedback data arrives at the control agent. The predictor computes $\hat{\mathbf{x}}\left(t_i+\tau_{CR}(i)+\tau_T(i)\right)$ using (20)-(22) and the median of RTT delays as $\hat{\tau}(i+1)$, and then forwards $\hat{\mathbf{x}}\left(t_i+\tau_{CR}(i)+\tau_T(i)\right)$ as $\mathbf{x}(i)$ to the path tracking algorithm in section I.

3) Gain scheduler

The gain scheduler is used to adjust the speed factor $K(i)$. If $K(i)$ computed from $\omega_{L,r}(i)$ and $\omega_{R,r}(i)$ of the original path tracking algorithm results in $\hat{J}(i+1)>\varepsilon$, the gain scheduler will replace $K(i)$ by the optimal *K*. Then, the gain scheduler computes $\omega_{L,r}(i)$ and $\omega_{R,r}(i)$ for sending out by using the optimal *K*, $A(i)$, and the mean of RTT delays as $\hat{\tau}(i+1)$. Otherwise, $\omega_{L,r}(i)$ and $\omega_{R,r}(i)$ from the original path tracking algorithm will be sent instead. The maximal tolerance ε is based on the user's requirement. If ε is small, the robot will track a path with the small difference between $\mathbf{x}\left(t_i+\tau_{CR}(i)+\tau_T(i)\right)$ and $\mathbf{x}\left(t_i-\tau_{RC}(i)\right)$, which implies the less effects of network delays. However, the robot may move slower, which may be undesirable in some cases.

The optimal *K* can be obtained by pre-computing $J(i+1)$ with respect to certain ranges of *K*, *A*, and τ from (26), and then searching for the set of optimal *K* that gives $J(i+1)\le\varepsilon$. However, using a lookup to store the optimal *K* may not be convenience since the table size may be very large, and the searching time for the optimal *K* can be long.

Computational intelligence techniques such as artificial neural networks (ANN) can be used to approximate the relationship between the optimal *K* with respect to ε, *A*, and τ. ANN is composed of simple mathematical elements called neurons, which operate in parallel. ANN can be trained so that they can provide target output values with a specific set of input values. Both input and output sets are together called as training sets. In this paper, the optimal *K* is the output of ANN, and the inputs are *A* and τ as shown in Fig. 6, where ε can be fixed as desired.

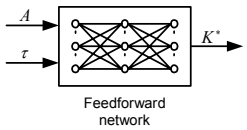

Fig. 6. Feedforward network configuration.

Different type of ANN can be used. This paper uses a feedforward network [10, 11] for illustration.

VII. Simulation Results

A robot simulation program is setup in a Mablab/Simulink environment to investigate the performance of the proposed scheme in the following environment

- The optimal *K* is prepared by computing $J(i+1)$ from $A\in[-300,300]$, $\tau\in[0.1,0.8]$, $K\in[0,5]$ with $\varepsilon=0.25$. A 3-layer feedforward network is trained by using Levenberg-Marquardt algorithm represent the optimal *K*. The number of hidden neurons used is 20. The network training reaches the error tolerance 1×10^{-6} at 135 epochs.
- RTT delays used for investigation are measured from ADAC Lab to Vienna University of Technology, Austria.
- The sampling time to send a probing packet $T_P=0.005$ s.
- The packet index to evaluate the median and mean of RTT delays $N=10$.
- The sampling time of the robot to process the reference speeds $T=0.01$ s.
- The elapsed time for simulation is 15 s.
- The initial position of the robot is $(-0.1,1)$.
- The simulation assumes that there is no packet loss.

In this simulation, the robot is assigned to track the reference path (Ref) as illustrated as the solid line in Fig. 7 from point A to point C. The robot will stop if the distance from the robot to point C is less than 0.05. Three cases are investigated:

- The robot tracks the path with neural network middleware under IP network delays (Md).
- The robot tracks the path without neural network middleware under IP network delays (No Md).
- The robot tracks the path without neural network middleware and IP network delays (No D).

Fig. 7 shows that the tracking algorithm performs very well without network delays. On the other hand, with network delays, the robot without neural network

middleware fails to track the path closely, and cannot reach the final position in 15 s. The robot can only move to point B at 15 s. On the other hand, the robot with neural network middleware can reach the destination in 15 s. Even though the robot shows small deviation from the path, it still tracks the path much closer than without middleware.

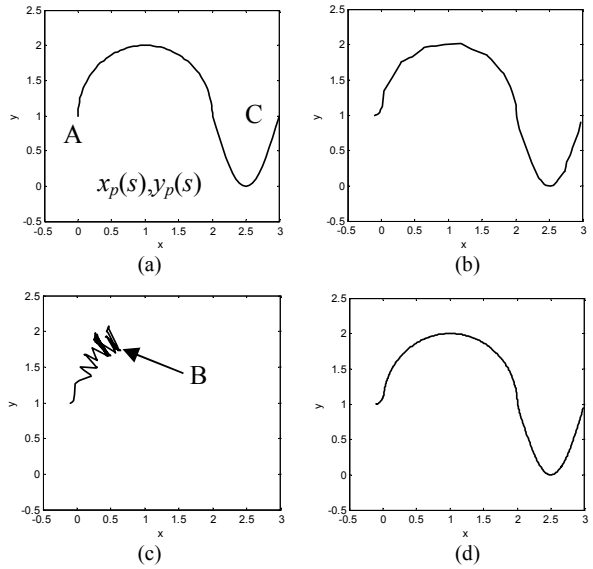

Fig. 7. Reference path and robot tracks:
(a) Reference path, (b) Md, (c) No Md, and (d) No D.

In addition, Fig. 8 shows the closest distance measure d from the robot to the point $x_p(s_0), y_p(s_0)$ on the path of all three cases when the robot is traveling along the path. The distance d provides a measure of how good the robot is tracking the path. The robot with neural network middleware tracks the path much closer than the one without neural network middleware. These results indicate that position and performance prediction by the neural network middleware can significantly improve the path tracking with network delay concerns. Because the effects of network delays still remain, the robot cannot track the path perfectly compared to the case without network delay despite of the middleware use. Nevertheless, closer tracking could be improved by reducing ε, but the robot would be slowed down. This is a trade-off design between accuracy and speed.

Furthermore, adjusting $K(i)$ to compensate the network delay effects by neural network middleware also help to improve the accuracy of the approximation in (14). With long network delays, $t - t_i$ may be large, and the approximation may be no longer valid. Reducing $K(i)$ can compensate the large $t - t_i$.

VIII. Conclusion

Model predictive path tracking scheme for networked mobile robot implemented in the neural network middleware can significantly improve the IP network delay compensation by adjusting the robot speeds. The

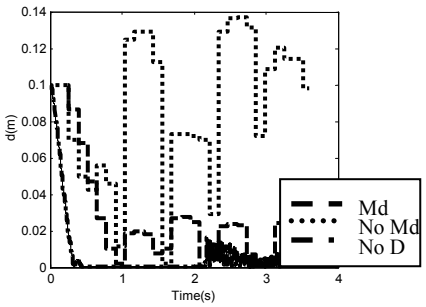

Fig. 8. Closest distance to the path d.

middleware structure allows convenient installation and improvement with an existing path tracking controller because it is implemented separately from the path tracking algorithm. An advanced delay prediction algorithm and a better model for prediction can be easily added or replaced in the middleware without modifying the path tracking algorithm. In addition, the proposed scheme could also take advantages from IP network QoS (Quality-of-Service) protocols since the mean and median of RTT delays, which are directly dependent on network QoS, are directly measured.

Acknowledgment

The authors would like to acknowledge the Royal Thai Government for partially supporting this study.

References

[1] G. Kaplan, "Ethernet's winning ways," *IEEE Spectrum*, vol. 38, pp. 113-115, 2001.

[2] A. Sano, H. Fujimoto, and M. Tanaka, "Gain-scheduled compensation for time delay of bilateral teleoperation systems," presented at IEEE International Conference on Robotics and Automation, Leuven, Belgium, 1998.

[3] S. Munir and W. J. Book, "Internet based teleoperation using wave variables with prediction," *IEEE/ASME Transactions on Mechatronics*, vol. 7, pp. 124-133, 2002.

[4] C. Benedetti, M. Franchini, and P. Fiorini, "Stable tracking in variable time-delay teleoperation," presented at IEEE/RSJ International Conference on Intelligent Robots and Systems, Maui, HI, 2001.

[5] K. Brady and T.-J. Tarn, "Internet-based teleoperation," presented at IEEE International Conference on Robotics & Automation, Seoul, South Korea, 2001.

[6] Y. Tipsuwan and M.-Y. Chow, "Gain Adaptation of Networked Mobile Robot to Compensate QoS Deterioration," presented at IEEE IECON2002, Sevilla, Spain, 2002.

[7] A. Ollero and G. Heredia, "Stability analysis of mobile robot path tracking," presented at IEEE/RSJ International Conference on Intelligent Robots and Systems, Pittsburgh, PA, 1995.

[8] B. Li and K. Nahrstedt, "A control-based middleware framework for quality-of-service adaptations," *IEEE Journal on Selected Areas in Communications*, vol. 17, pp. 1632-1650, 1999.

[9] K. Yoshizawa, H. Hashimoto, M. Wada, and S. M. Mori, "Path tracking control of mobile robots using a quadratic curve," presented at IEEE Intelligent Vehicles Symposium, Tokyo, Japan, 1996.

[10] D. E. Rumelhart and J. L. McClelland, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*. Cambridge, MA: The MIT Press, 1986.

[11] M.-Y. Chow, *Methodologies of Using Artificial Neural Network and Fuzzy Logic Technologies for Motor Incipient Fault Detection*. Singapore: World Scientific, 1998.