Modeling and Implementation for Embedded DC Motor Ethernet Control System

Lingbo Zhu, Guanzhong Dai, Li Shi, Xuefang Lin-Shi, Jean-Marie Rétif

Modeling and Implementation for Embedded DC Motor Ethernet Control System

Lingbo Zhu, Guanzhong Dai
School of Automation
Northwestern Polytechnical University
Xi'an, 710072, China
zhulingbo.insa@gmail.com

Li Shi
China Ship Development and Design Center
Wuhan, 430064, China

Xuefang Lin-shi, Jean-marie Rétif
Laboratory Ampère
INSA de Lyon
Villeurbanne, 69621, France

Abstract—**In this paper, a model of Embedded DC Motor Ethernet Control System is proposed with analysis on transmission time delay and data packet dropout. Based on this model, the ECS is described as a two-state asynchronous dynamical system with output feedback control and then implemented on a platform which uses a PC as a central controller and an ARM9 kit as a remote controller. As a key part of developments, embedded programs including client program and Linux Module are realized on the kit. The platform is proved to be flexible to run different control algorithms and extensibility to add nodes. The experiments and results demonstrate the validity of the system.**

Keywords-Ethernet Control System(ECS); Embedded Linux; Delay; DC Motor

I. INTRODUCTION

With the integration of communication networks and distributed control in the modern manufacturing and process industries, networked control systems (NCSs) are becoming increasingly important due to its simplicity, scalability, flexibility, maintainability, and cost effectiveness. However, there are still significant challenges that result in the application limitation of NCS technologies, such as transmission time delay and data packet dropout [1,2].

To reveal the performance of NCSs, researchers built various experimental platforms based on different software and hardware environment. Chow [3] used a PC running RTLinux working as a central controller and a Siemens C-515C microcontroller board serving as a remote controller with RS-232 connection. Nie [4] used ARM7TDMI with running uCLinux to work as a remote controller and a PC with Matlab Simulink(RTW) to work as a central controller. Hu [5] proposed a kind of NCS simulation platform based on switched Ethernet though using two computers to simulate actual controller and plant by Matlab. Liu [6] designed a simulation platform of NCS using Matlab to build models of real systems. However, in Chow's case, for the lack of RS-232 protocol, distance control based on this system is impossible. And there isn't OS (Operation System) on the microcontroller, the flexibility and expansibility are restricted. In Nie's case, the delays of the system are simulated. In Hu's case, the plant is a simulated one by Matlab and VB, as well as Liu's. Therefore, to achieve more authenticity and universality, a physical NCS platform with advanced protocol such as Ethernet with TCP/IP is needed.

Recently, using Ethernet to industrial field without exorbitant requirements is becoming a hot field [7]. Industrial Ethernet products begin to serve the communications requirements of industrial customers, replacing or supplementing legacy fieldbus such as Modbus, and Profibus which are exclusive mutually and are held by different companies with expensive prices. To solve the lacks described above and considering the popularity, a physical platform based on Ethernet, Samsung2410A [8] kit, DC motor is designed as well as implemented in this paper.

The layout of this paper is as follows. In section 2, considering transmission time delay and data packet dropout, the system is modeled as a two-state asynchronous dynamical system with output feedback control law. Section 3 shows implementation of the system. In section 4 the experiment results are illustrated. Conclusions are drawn in Section 5.

II. SYSTEM MODELING

As a type of NCSs, an ECS is shown in Fig.1. To analyze conveniently, a model of the system with transmission time delay and data packet dropout is proposed based on the following assumptions,

Assumption 1 The sensor is time-driven and has identical sampling period T.

Assumption 2 The controller and actuator are event-driven, which means calculating the control law or output signals as soon as which arrive.

Assumption 3 The total delay of closed-loop in the network is $\tau_k = \tau_{sc}^k + \tau_{ca}^k$, where τ_{sc}^k and τ_{ca}^k denote the sensor-to-controller and controller-to-actuator delay, respectively. If $0 \le \tau_k < T$, τ_k is disposed as short-delay. While if $\tau_k \ge T$, throw away this packet actively, and it is disposed as data packet dropout.

Assumption 4 Sampled data and control law of each period are transmitted by single packet. If the k[th] packet is lost, using $\bar{x}(k-1)$ to replace $\hat{x}(k)$.

Assumption 5 The probability of the network unblocked is $r(0 < r \le 1)$, which is a constant.

Supported by China Scholarship Council 2006[3074]

978-0-7695-3336-0/08 $25.00 © 2008 IEEE
DOI 10.1109/CSSE.2008.681

Assumption 6 The plant is assumed as a normal linear system, which is,

$$\begin{cases} \dot{x}(t) = Ax(t) + Bu(t) \\ y(t) = Cx(t) \end{cases} \tag{1}$$

Where, $x(t) \in R^n, u(t) \in R^m, y(t) \in R^p$ are the state, input, and measured output, respectively. A, B and C are system matrices with compatible dimensions.

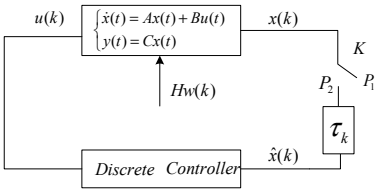

Fig. 1 A typical ECS

As seen in Fig.1, $x(k)$ is the state vector of the plant at time instant k. $\hat{x}(k)$ is the received state vector. $u(k)$ is the control law. $w(k)$ is the disturbance vector, and H is constant matrix with appropriate dimensions. Then the model consists of a discrete-time plant,

$$\begin{cases} x(k+1) = \Phi x(k) + \Gamma_0(\tau_k)u(k) + \Gamma_1(\tau_k)u(k-1) + \mathrm{H}w(k) \\ y(k) = C\hat{x}(k) \end{cases} \tag{2}$$

where, $\Phi = e^{AT}$, $\Gamma_0(\tau_k) = \int_0^{T-\tau_k} e^{As} B ds$, $\mathrm{H} = e^{DT}$,

$$\Gamma_1(\tau_k) = \int_{T-\tau_k}^{T} e^{As} B ds \ , \ x(k) = \begin{cases} \hat{x}(k), \text{unblocked} \\ \hat{x}(k-1), \text{blocked} \end{cases} .$$

The system has two states, S_1 –unblocked, S_2 –blocked. We assume that the controller is output feedback controller $u(k) = Ky(k) = KC\hat{x}(k)$, K is a constant matrix with appropriate dimensions, then (2) can be transformed as,

$$S_1 : \begin{cases} x(k+1) = (\Phi + KC\Gamma_0(\tau_k))\hat{x}(k) + KC\Gamma_1(\tau_k)\hat{x}(k-1) + \mathrm{H}w(k) \\ y(k) = C\hat{x}(k) \end{cases}$$

$$S_2 : \begin{cases} x(k+1) = \Phi\hat{x}(k) + KC(\Gamma_0(\tau_k) + \Gamma_1(\tau_k))\hat{x}(k-1) + \mathrm{H}w(k-1) \\ y(k) = C\hat{x}(k) \end{cases}$$

Let, $z(k) = [x^T(k), \hat{x}^T(k-1), w^T(k), w^T(k-1)]^T$

$$z(k+1) = \Omega_i \tilde{\Phi}_i z(k) \ , \ \Omega_1 = r, \Omega_2 = 1 - r \ \ i = 1, 2 \tag{3}$$

$$\tilde{\Phi}_1 = \begin{bmatrix} \Phi + KC\Gamma_0 & KC\Gamma_1 & H & 0 \\ I & 0 & 0 & 0 \\ 0 & I & 0 & 0 \\ 0 & 0 & I & 0 \end{bmatrix} \quad \tilde{\Phi}_2 = \begin{bmatrix} \Phi & KC(\Gamma_0 + \Gamma_1) & 0 & \mathrm{H} \\ I & 0 & 0 & 0 \\ 0 & I & 0 & 0 \\ 0 & 0 & 0 & I \end{bmatrix}$$

III. IMPLEMENTATION

A. Framework of the Implementation

For implementing a general system described by (3), we choose Ethernet with TCP/IP protocol as the network. The system is consisted by some hardware: (1)a PC running Fedora 6(Linux2.6) works as a central controller; (2) a kit made by Samsung 2410A(ARM9 core, running Linux2.4) serves as a remote controller; (3) The DC motor is ESCAP(28HSL18-219/204). Besides, an interface card realizes voltage conversion while sampling/outputting.

In the software side, there are three programs need be realized: a server program with control algorithm on central controller, a client program and a Linux module (named EMD) on ARM9.

As the most popular Embedded OS, Linux is an OS with multi-task. Thus, the processor is shared by a series of processes. To deduce the processing time on ARM9, we use interrupts to realize sampling and outputting, which can cut down the processing time from 1~1.5ms to 10~20us in each action.

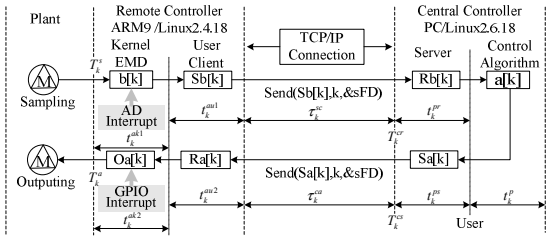

Fig. 2 Process of the system

Fig. 2 describes the process of one period. Where, arrays b[1..k], Sb[1..k], Rb[1..k] a[1..k], Sa[1..k], Ra[1..k], Oa[1..k] are all the buffers which are assumed to have enough space to storage the data. The process can be depicted as,

Step1: Build TCP/IP connection between Server and Client using Linux socket which is named as sFD.

Step2: Take the k^{th} cycle as an example. When kT time arrives, Client produces an interrupt signal to sample data by calling EMD. Then EMD samples the value and storages it to b[k].

Step3: Client copies b[k] to Sb[k] from kernel to user. And then send Sb[k] to server by calling send(Sb[k], k, &sFD).

Step4: Server receives Sb[k] using recv(Sb[k], k, &sFD). Then send Sb[k] to control algorithm for calculating the control law by calling Cal(a[0…k], $\vec{p}$, $\vec{g}$). Then put the result to a[k*], k*=k+1. Where $\vec{p}$ is the parameter vector of the system, and $\vec{g}$ is the external input.

Step5: Server gains Sa[k*] from a[k*]. Then send a[k*] to Client using send(Sa[k*], k*, &sFD).

Step6: Client receives Ra[k*] by recv(Sa[k*], k*, &sFD). If k*=k+1, then Client sends Ra[k*] to EMD by calling ioctl(), else throws away this packet.

Step7: EMD produces interrupt signal to output control law to the plant.

Step8: Client waits for $(k+1)T$ time arriving.

The programs on central controller are realized by C language as the same as normal Linux network coding. The following part, delay measurements and embedded programs are discussed.

B. Delay Measurement

As a NCS, delay measurement is a key issue of system evaluation. Taking kT time as an example, based on Fig.2, some remarks are listed below,

Remarks: the clock of ARM9 sampling- T_k^s ;the clock of EMD outputting- T_k^a ;the clock of PC receiving - T_k^{cr} ;the clock of PC sending- T_k^{cs} ; the time for processing sampling data in EMD and Client are t_k^{ak1} , t_k^{au1} , respectively; the time for processing control law in EMD and Client are t_k^{ak2} and t_k^{au2} . Then,

$$\tau_k = \tau_k^{sc} + \tau_k^{ca} = T_k^a - T_k^s - (t_k^{ak1} + t_k^{au1} + t_k^{ak2} + t_k^{au2}) - (T_k^{cs} - T_k^{cr}) \qquad (4)$$

Consider the speed of ARM9 is 200MHz, thus the time of $(t_k^{ak1} + t_k^{au1} + t_k^{ak2} + t_k^{au2})$ can be ignored. Then,

$$\tau_k = T_k^a - T_k^s - T_k^{cs} + T_k^{cr} \qquad (5)$$

Therefore, we just need mark the clock of these four points of each cycle, the delay can be calculated. Because the OS of the PC and ARM9 are both Linux system, the gettimeofday() function can be used to get the clock of the system. The precision of this function is 1 us, which is enough for our application.

Because TCP/IP protocol is used in the system, in the transmission there are no errors but long delays($\tau_k > T$) occur. To satisfy Assumption 3, we put a threshold of delay. Considering $T = 5ms$ and except the time of outputting, we choose the threshold as $\hat{\tau} = 4.9ms$. Thus, for the implementation,

If $\tau_k < \hat{\tau}$, the state of system is S_1 .

Else, the state of system is S_2 .

After this processing, the system can be described as (3). We define D_M as the max number of continuous data packet dropout. If data packet dropout happens, the used time(t_u) of k steps will be more than kT . We define $t_e = t_u - kT$.

C. Embedded Program

The embedded program in ARM9 kit should include two parts: one is user layer program and the other is EMD. User layer program includes: (1) TCP/IP communication with central controller; (2) data transmission with kernel layer program. EMD needs realize: (1) driving A/D to sample the voltage from motor; (2) driving GPIO to output control laws to motor; (3) realizing interrupts with parameters to sample and output data; (4) inter-communication with user layer program.

Client - A function named MyConnection() is established to implement communication with server by using Linux socket. In order to transmit data between Client and EMD, special functions read(), write() and ioctl() in EMD are implemented. The process of each period is realized by a function named promt_info() and which is listed as,

Step1: EMD samples voltage data. This step includes setting control segments, writing the segments to EMD, and producing interrupt signal by calling ioctl() function.

Step2: Client reads the A/D sampling value from EMD by using read() function

Step3: Client sends the data to server via socket.

Step4: Client receives control law from server.

Step5: Client sends control law to EMD.

EMD- EMD includes file_operations like a typical Linux module which defines some necessary functions, such as open, read, write, ioctl, etc. In the implementation, GPIO Driver, A/D Driver and interrupt functions should be realized.

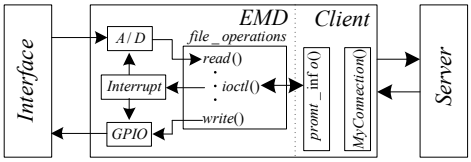

Fig.3 Relationship between EMD, Client, Server and Interface

Fig.3 shows the relationship between EMD, client, server and interface. The sampling data through interface are transmitted to A/D, then EMD can read the data using read(). And the data are transmitted to client by calling ioctl(). After that, the data reach server by socket connection. In the same way, the control law can be transmitted from server to interface.

D. The Plant and Controller

To decrease processing time, a simple control algorithm – discrete PI control is chosen. The motor's parameters can be found from its manual [9].

As the electric time constant $L/R = 8.62e-5$ is very small comparing to mechanical time $J/f = 0.1337$, the motor can be described by a first order model. Thus, the transfer function between the motor speed and the armature winding input voltage is,

$$\frac{\Omega(p)}{U(p)} = \frac{1/K_c}{1 + \frac{RJ}{K_c^2 \eta} s} \qquad (6)$$

It's determined by pole assignment by imposing a closed–loop dynamics which corresponds to a discrete–time first order with 15ms. The sampling period is set to 5ms. The

discrete time controller can be expressed as,

$$K(z)=\frac{u(z)}{\varepsilon(z)}=\frac{r_0+r_1z^{-1}}{1-z^{-1}} \tag{7}$$

where. $r_0=0.9775, r_1=-0.7183$ The corresponding margins are: Phase margin : $\Delta_\Phi=73.7^\circ$; Delay margin : $\Delta_T=0.029s\approx 5.9T_e$.

IV. EXPERIMENT

After developments, experiments are implemented based on a normal office Ethernet. The central controller and the remote controller are both connected to one port of the Switch(3COM Baseline Switch 2024). Other ports of the Switch are shared by the users of office for revealing the performance of the system with stochastic load. Initial voltage is 0 v. Set-point is 0.78125v. Sampling period is 5ms. 399 steps are done. The results of delay measurements(two independent groups) using method described in 3.2 are shown in Fig. 4. From it, we can get $r=97.24\%$ and $D_M=4$, while in group2 which are $r=97.99\%$ and $D_M=3$.

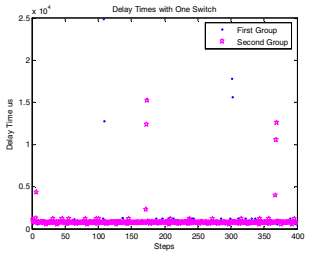

Fig.4 Delays of the System

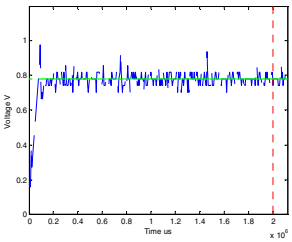

Fig.5 Results of the System

Fig. 5 shows the response of the system. From the figure the used time for 399 steps is beyond 1995000 us (the broken upright line of Fig.5), and we can get that $t_e=150010\text{us}$. The reason of this phenomenon is data packet dropout occurring.

V. CONCLUSION

In this paper, we have presented a model for ECS with transmission time delay and packet dropout. Then, in order to reveal the dynamical performance of the system, a platform is established based on embedded ECS applying to a DC motor. As a key part of developments, embedded programs including client program and Linux Module named EMD are realized on the ARM9 kit. The experiments and results show that this model is effective and valid for DC motor Ethernet control systems.

REFERENCES

[1] Yu-Chu Tian, David Levy, “Compensation for control packet dropout in networked control systems”, Information Science, 178(2008) pp:1263-1278.

[2] Matías G. R, Antonio B, “Analysis of networked control systems with drops and variable delays”, Automatica, 43 (2007), pp:2054-2059

[3] Mo-Yue Chow, Yodyium Tipsuwan, “Gain Adaptation of Networked DC motor Controller Based on QoS Variations”, IEEE Transactions on Industrial Electronics, Vol,50, October,2003

[4] Nie Xue-yuan,LIU Guo-ping, “Realization pf Embedded Networked Control Simulation Based on Simulink”, Journal of System Simulation,vol.17,No. 7,July 2005, 1613-1620

[5] Hu Xiaoya,Zhu Desen, and Wang Bingwen, “Simulation platform for networked control system based on switched Ethernet”, J.Huazhong Univ. of Sci. & Tech. (Nature Science Edition), Oct . 2005, Vol. 33 No. 10,pp:89-91.

[6] Liu Zhifei, Wang Shuqing, “Design of a Simulation Platform of Networked Control Systems”, Chinese Journal of Scientific Instrument, Jun. 2005, pp:597-600.

[7] Kweon S, Shin K G, Zheng Q. “Statistical Real-Time Communication over Ethernet for Manufacturing Automation Systems”, Proceedings of the Fifth IEEE Real-Time Technology and Application Symposium, 1999

[8] USER'S MANUAL S3C2410A – 200MHz & 266MHz 32-Bit RISC Microprocessor Revision 1.0, 2004 Samsung Electronics.

[9] Moteur C.C escap® 28L28, Réducteur escap® R32, http://www.portescap.com/