Design and Application of Queue-Buffer Communication Model in Pneumatic Conveying

Liping Zhang
College of Electronic and Electrical Engineering
Shanghai University of Engineering Science
Songjiang District, Shanghai 201620, China

Haomin Hu
College of Electronic and Electrical Engineering
Shanghai University of Engineering Science
Songjiang District, Shanghai 201620, China

Abstract—**In order to communicate with a PLC (Programmable Logic Controller) flexibly and freely, a data communication model based on the PLC's free port is designed. In the structure of the model, a distributed data communication environment is constructed by using Ethernet and serial adapters. In the communication algorithm, a method of queue-buffer for multi-threaded is used to improve the real-time of the control. This communication model has good scalability and portability because the realization of it is not restricted by the number of PLC slave stations and the type of operating system of the host computer. A corresponding communication algorithm is applied to data collection and devices monitoring for a pipe pneumatic conveying system. The practice shows that not only the stability and reliability of the model can meet the needs of automatic control but also the communication performance and efficiency of the model is outstanding.**

Keywords—Queue-Buffer; Programmable Logic Controller; Freeport Communication; Critical resource; Mutex

I. Introduction

PLC as the industrial automatic control equipment encompassing the automation technology, computer technology, control technology and communications technology, has been widely used in various fields of industrial automation [1]. In the control system, how to achieve stable, efficient and flexible communication between the host computer and the PLC has become an important research topic. At the same time, PLC manufacturers also offer a variety of communication methods for users to choose. Taking Siemens S7-200 series as an example, a control system consisting of this series PLC can generally use configuration software monitoring, third-party monitoring-software monitoring, touch screen monitoring and other monitoring methods [2]. Meanwhile, in order to improve the flexibility of automated control systems, many kinds of PLC provide a communication patterns named "free port", such as Siemens s7-200 series PLC, Mitsubishi FX2 series PLC, OMRON CJM1 series PLC and so on.

Freeport communication is a method of using serial communication hardware and relevant instructions for customizing communication protocols which are provided by the PLC to realize data communication by PLC programming. The literatures [3] and [4] have described the implementations of the Siemens PLC communication method on free port. Currently the discussion about the free port communication method of the PLC mainly focused on the "point-to-point" technology, namely realizing the data acquisition and control of PLC through the serial communication of host computer. Literature [5] has described a method of using MSComm programming control for host computer to communicate with S7-200 series PLC. But in the field of industrial control, there will be many PLCs as slave stations, and not only the serial number of host computer is limited, but also the communication distance is restricted. This paper describes an approach which is based on Ethernet and serial adapters to achieve distributed data communication model through the PLC free port. This model has the advantage of being cross-platform and can be implemented in different operating systems. At the same time, in order to achieve the acquisition of data and the control of PLC efficiently and stably, the communication algorithm based on queue-buffer for multi-threaded is designed. Under the premise of ensuring real-time, this algorithm also improves the scalability of the number of communication nodes and the concurrency adaptability of archive database as far as possible.

In the following section, structure of the system and related communication model and algorithm are discussed.

II. System Structure

In the design of control system for a variety of powder dense phase pneumatic conveying engineering, a lot of devices need to be monitored, such as exhausting valve, feeding valve, pre-closing valve, outlet valve, upper intake valve, down intake valve, pneumatic hammer of pump and pulse dust collector, pneumatic hammer, fan of hopper. It's easy to use PLCs as control sources to debug and maintain solenoid valves and indicator lights by electric relays. The control system needs to detect and monitor the working status of pump pressure transmitters, material high level switch, pneumatic valve position switch, air pressure gauge and thermal resistance, etc. Many brands of PLC can control these devices, and they also provide the free port for communication, although different PLCs vary in communication protocol, but many PLC provides a free port of communication. The number of devices which PLC can control is limited and PLCs should be added as monitoring stations when the device exceeds a certain number. At the same time, the number of serial interfaces of host computer is limited too and communication will be restricted by the number of interfaces when PLCs' number beyond it. In order to monitor the devices with good scalability, we designed a communication system whose structure is shown in Figure 1.

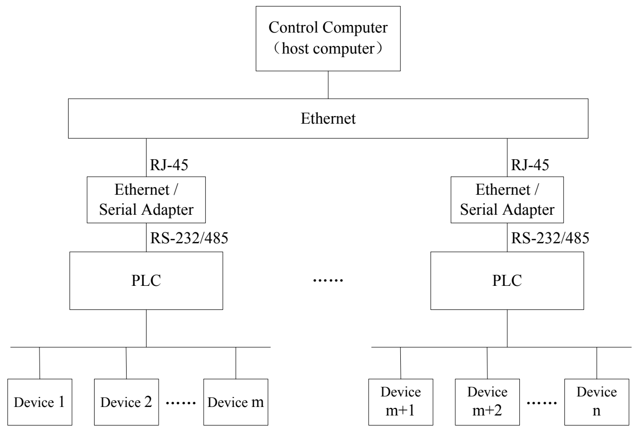

Fig. 1. system structure

A distributed data communication structure is used in Figure 1. Each PLC detects and controls a set of devices and the data collected by it is transferred to the Ethernet and serial adapters through serial port. The adapter converts the data into TCP datagram and delivers them to Ethernet. The communication between these nodes is TCP connection-oriented reliable data transmission. Although this architecture provides the system with a distributed processing environment, communication model and algorithm will largely affect the real-time of data acquisition. Therefore, the design of communications algorithm is also important.

III. COMMUNICATION MODEL DESIGN

A. *Design Principles*

For the industrial automatic control system, the real-time is very important. In the control system shown in Figure 1, the host computer sends instructions to PLCs via Ethernet and serial adapters, and PLCs control devices of system. If data collection is performed, PLCs transfer the data to host computer via the Ethernet and serial adapter. When the number of PLC and devices is large, it is difficult to guarantee real-time if polling communication is still used. Taking data collection as an example, the obstruction of any communication channel will interfere with sampling period and affect real-time. Although the real-time of data acquisition can be improved by using multiple threads concurrently, the scalability of system is unable to be ensured, if there is no reasonable synchronization scheduling mechanism. Queue-buffer communication model describes a way to achieve host computer communicating with PLCs in multi-threaded by synchronization and mutual exclusion of critical resources, and improve the real-time of system, the reliability of data acquisition and archiving as far as possible.

According to the Bernstein condition for concurrent execution [6], the following formula must be satisfied for process P1 and p2 can execute concurrently and has a reproducible operating environment.

$$R(p_1)\cap W(p_2)\cup R(p_2)\cap W(p_1)\cup W(p_1)\cap W(p_2)=\Phi$$

Where R represents a set of resource which some process will read, and W represents a set of resource which some process will write. The following conclusion can be drawn. We assume that T is a set of threads for communication in the host computer, if the following formula is established, the communication threads can execute concurrently.

$$\forall t_i,\forall t_j,t_i\in T,t_j\in T,t_i\neq t_j,\ R(t_i)\cap W(t_j)\cup R(t_j)\cap W(t_i)\cup W(t_i)\cap W(t_j)=\Phi$$

In order to improve the concurrent performance of system, we used two kinds of independent thread (ie, communication thread and archive thread) to achieve the communication with PLCs and archive of collected data. Meanwhile, in order to reduce the critical region and competition of threads for critical resources, we used private queues and public queue for data buffer and buffering threads to schedule the data. Queue-Buffer communication model is shown in Figure 2:

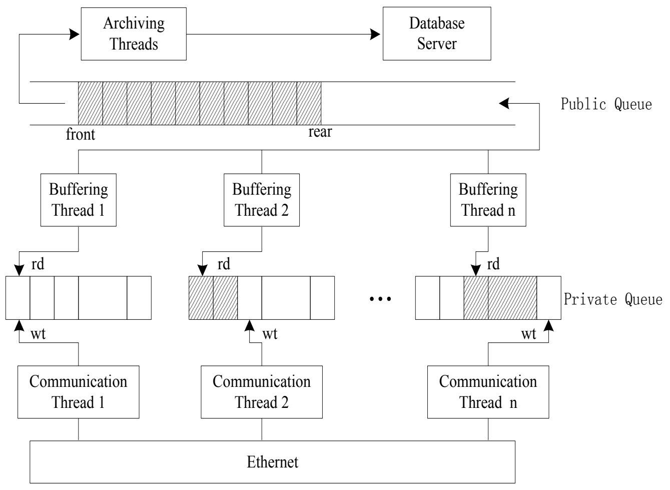

Fig. 2. Queue-Buffer Communication Model

B. *Feasibility Analysis*

In the communication model shown in Figure 2, each communication thread is assigned with a buffer queue for storing the data collected from the PLC. Because each thread has a buffer queue and the queue is not shared, so we call the queue a "private queue". In the private queue is not the full case, communication threads will not be blocked for delay of data saving. Concurrent operation can be guaranteed. Communication thread writes data to the private queue, and buffering thread reads data from the private queue. What they affect is the movement of the rd pointer and wt pointer, so synchronization will occur only when the private queue is empty or full. When the private queue is empty, the buffering thread will be blocked, but the communication thread will not be affected, and real-time can still be guaranteed. Only when the private queue is full, the communication will be blocked, and what causes this phenomenon is that the data archive is not fast enough. In addition to archiving the cycle too quickly, what impacts the speed of archive are the following two reasons. (1) The buffer size of private queue is too small, and the buffer overflows before the thread which is responsible for data archiving gets the time slice. (2) The concurrent processing capacity of database is not enough and a large number of connections is occupied, or archiving thread switches between connecting and disconnecting too frequently that degrades the performance. In the first case, we can resolve the problem by increasing the buffer size appropriately. But for the second case, the database concurrency is determined by the database performance, and therefore the number of connections can not be increased arbitrarily. To solve this problem, another queue buffer, namely the public queue, is added between the archiving thread and the buffering thread. The number of database connections is controlled by the archiving threads which maintain database connection stably to reduce the switch of connection state.

In the above communication model, the data of private queue is not archived in the database directly, but is stored into a public queue by buffering threads. So that not only data can be copied from private queues to public queue quickly that reduce the possibility of private queue to overflow, and the number of threads can be isolated and the database communication correlation between the degree of concurrency, but also the correlation between communication threads number and concurrency of database is isolated. Thus the competition for the database connection will be changed into the competition of buffering threads for rear pointer of public queue buffer, and the number of threads which access front pointer of public queue buffer can be determined according to the performance of the database. This scheme effectively improves the universality of database for archiving, and when a lot of concurrent operations are required, so a distributed database can be easily used.

C. Algorithm Description

An important step forward in this area of concurrent programming occurred when Edsger Dijkstra introduced the concept of the semaphore. A semaphore is a special variable that takes only whole positive numbers and upon which programs can only act atomically [7]. The following algorithm is described by primitive P and V. Because concurrency and synchronization between threads is happening in the queue for private queues and public queue, algorithm based on the type of queue buffer will be described separately. Read operation and write operation of private queues is described below:

```
int wt, rd, pvt_queue_len; // Initial values of wt and rd are 0, and value of pvt_queue_len equals to the length of private queue
sem_t pvt_queue_full, pvt_queue_empty; // Initial values for the two semaphores are 0 and pvt_queue_len respectively
struct data_item pvt_queue[pvt_queue_len];  // pvt_queue is private queue
write_pvt_queue()  // Write collected data to private queue
{
  P(pvt_queue_empty);  // If private queue buffer is not full, then write the data, or block
  Write data to pvt_queue(wt);
  wt = (wt+1)% pvt_queue_len;// Move the write pointer of private queue backward
  V(pvt_queue_full);
}
read_pvt_queue()  // Read data from private queue
{
  P(pvt_queue_full);   // If buffer is not full, then read data, or block
  dat = pvt_queue[rd];  //Save data to variable dat
  rd = (rd+1) % pvt_queue_len;// Move the read pointer of private queue backward
  V(pvt_queue_empty);
}
```

In the algorithm above, sem_t is a data type of semaphore in linux and struct data_item is a custom data type for the data item stored in queue buffer. On the other hand, In the read_pvt_queue() function which is to read data from private queue, the data is copied to a temporary storage, variable dat, in order to prevent the data from being overwritten by communication thread before it is saved into public queue buffer. Since each communication thread and buffering thread require the data members and functions above, the readability of the program can be improved by encapsulating them into a class.

Read operation and write operation of public queues is described below:

```
int front, rear, pub_queue_len; //Initial values of front and rear are 0, and pub_queue_len means the length of public queue
sem_t pub_queue_full, pub_queue_empty; // Initial values for the two semaphores are 0 and pub_queue_len respectively
struct data_item pub_queue[pub_queue_len]; // pub_queue is public queue
pthread_mutex_t  mtx_front, mtx_rear; // Mutex operation for the front and rear of public queue, Initial values of them are 1
write_pub_queue()  // Write data to public queue
{
  P(pub_queue_empty); // If public queue buffer is not full, program continue, or block
  P(mtx_rear);// Mutex operation for rear of public queue
  Buffering thread write data into pub_queue[rear];
  rear=(rear+1) % pub_queue_len;// Move the rear pointer of public queue backward
  V(mtx_rear);
  V(pub_queue_full);
}
read_pub_queue()   // Read data from public queue, executed by archiving thread
{
  P(pub_queue_full); // If public queue buffer is not empty, program continue, or block
```

```
        P(mtx_front); // Mutex operation for front of public queue
        Copy data of queue[front] into local variable;
        front=(front+1)%Q2_size; // Move the front pointer of public queue backward
        V(mtx_head);
        V(pub_queue_empty);
        Data archiving;
    }
};
```

In the algorithm above, pthread_mutex_t is a data type of mutex in linux, and mtx_front and mtx_rear are used to achieve mutex operation for the front and rear pointer of public queue buffer. The data members and functions can be encapsulated into a class, while just one object of this class is need.

In linux, the P and V operations for semaphore can be achieve by sem_wait and sem_post function, while for mutex pthread_mutex_lock and pthread_mutex_unlock functions should be used [8]. The executing function can be specified when a thread is created by pthread_create function. Taking buffering thread as an example, read_pvt_queue and write_pub_queue functions should be called cyclically.

IV. APPLICATION EXAMPLES

To transport bulk powder and granular materials by pneumatic conveying pipeline is a two-phase flow technology, and there is not yet fully quantified theory to define it precisely at home and abroad, but semi-qualitative and semi-quantitative method is used. So to learn from past experiences and history data to guide pneumatic conveying becomes quite important. Meanwhile, the requirement of industry users for the pneumatic conveying varies, some materials should be kept hygiene and not broken during transportation, some should be reduced wear as far as possible, and some should be transported by inert gas and kept from explosion and static. Therefore, it is a key for using suitable pneumatic conveying technology and equipment and the method to control and operate depending on the process characteristics.

Queue-buffer communication model has been applied to pipe pneumatic conveying data collection and equipment monitoring. In the pneumatic conveying graph shown in Figure 3, program gets equipment operating status and records real-time valve of pressure and temperature, etc, and stores these data in database which will provide favorable conditions for the subsequent analysis and mining.

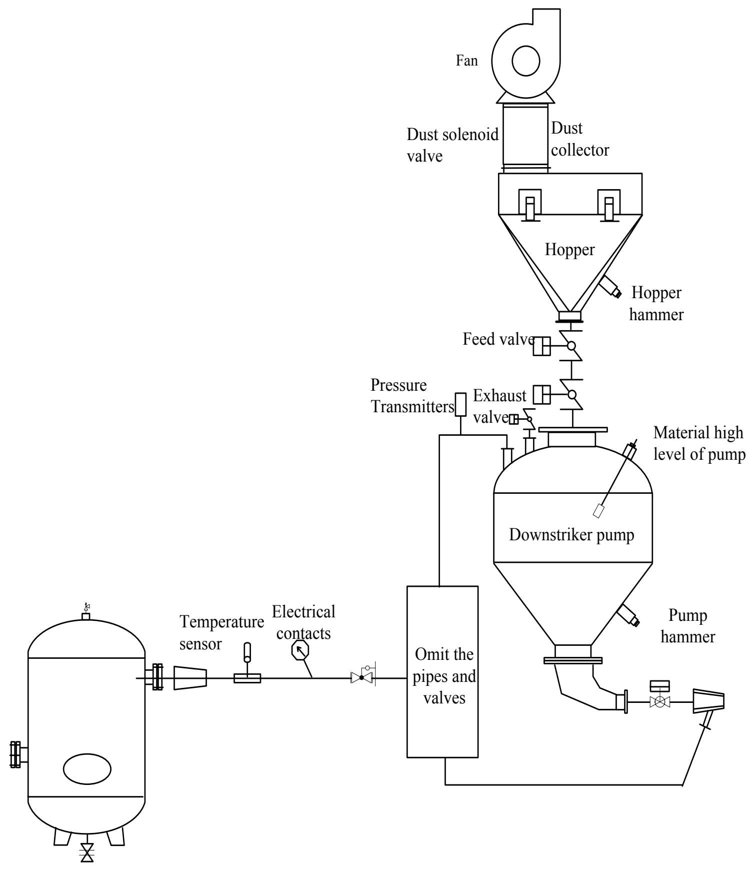

Fig. 3. Application in Pneumatic Conveying

V. CONCLUSION

Queue-Buffer communication model as a kind of automatic control architecture and software algorithm based on PLC's free port is to achieve flexibility and freedom of communication with PLC. This model switches the communication port of computer and PLC by Ethernet and serial adapter's conversion, and provides the possibility of increasing the number of PLCs in automatic control network will not be limited by the configuration software and compatibility of the communication hardware. This model has the following characteristics: (1) The model has good scalability and communication nodes are convenient to increase. (2) The communication distance of model is unrestricted. (3) The model is independent of operating system platform and has good portability. The application in Pneumatic conveying shows that the model is feasible and effective. The idea of it also provides a reference for data communication in internet of things.

ACKNOWLEDGMENT

The projects have been supported by the Innovation Program of Shanghai Municipal Education Commission "Vehicle Collision Avoidance System based on Vehicle Wireless Communication" (No.12YZ151), and "Discipline Comprehension Construction on Control Science and Engineering" (No.13XKCZ07), and 'First Class Discipline Cultivation - Mechanical Engineering" (No.YLJX12-7).

Liping Zhang acknowledges the guidance & support of Cheng-Chew Lim of the University of Adelaide during her academic visit in 2013/2014.

REFERENCES

[1] Gong Yunxing, Zhao Houuu, Qi Benzhi, PLC technology and application – Based on Siemens s7-200 series. BeiJing: Tsinghua University Press, 2009, pp.68–73.

[2] Liu Ning, Design and Implementation of the Control and Monitoring subsystems in Small Scale Water Treatment Engineering. Jilin University [D] pp.12, April 2012.

[3] Xiang Xiaohan, Industrial Communication Tutorial for Siemens PLC Specialist. BeiJing: Chemical Industry Press, March 2013.

[4] Li Jiangquan, Yan Haijuan, Liu Jiaodi, Deng Hongtao, Application Examples of Siemens PLC communication and Control Programming. BeiJing: China Electric Power Press, January 2012.

[5] Xu Qiyi, Wu Yuqiang, Jiang Xiuping, Li Kun, Communication between S7-200 PLC and Computer under C++ Builder Environment. Process Automation Instrumentation, vol. 1, pp.57-60, 2007.

[6] Tang Xiaodan, Liang Hongbin, Zhe Fengping, Tang ZiYing, Computer Operating System. ShangXi : Xi'an Electronic and Science University Press, August 2011.

[7] Neil Matthew, Richard Stones. Beginning Linux Programming 4th Edition. Wrox, 2007, pp.578-580.

[8] Yang Zongde, Lu Guanghong, Liu Yong, Linux Advanced Programming, 3rd Edition. BeiJing: Posts & Telecommunications Press. November 2012, pp.305-307.