APAS NETWORK COMMUNICATION SOLUTION

[1]Sorin-Robertino Sintea, [2]Dan Grebenisan, [3]George Nicolae

[1]Software Dept., Software Integration Technology Ltd, 33, 1 Decembrie 1918, 900158 Constanta, Romania, phone: +40-241-549479, sinteasorin@rdslink.ro

[2]Automation Dept., General Motors, 532 Arnhem Dr., L1G2J5 Oshawa, Canada, dan.grebenisan@gm.com

[3]Electronics and Computers Department, "Transilvania" University of Brasov, Politehnicii Street, no.1, 500174, Brasov, Romania, phone/fax: +40 0268 471893, nicolaeg@vega.unitbv.ro

This paper describes communication solution for fast data acquisition systems in a large distributed network, with data recovery, who using a central communication server. In welding plants, the automation systems send data with a large frame rate. The data acquisition controllers send data at small time intervals, and the size of data blocks can be very large. In large welding plants, with a big numbers of controllers (starting from about 100 controllers) this can be a problem. In their network (in conformity with IEEE-802.3) can appear a data packet storm or/and the destination server can be locked. In this case we can have loosing of data packets. The solution proposed describes a data revival mechanism, for prevent loosing of packets. This solution is implemented at application level of TCP/IP stack, on both sides (server and controller). The solution is implemented in Marathon Weld's APAS (Arc Process Analysis System) data acquisition system.

Keywords: data communication software, network protocols

1. INTRODUCTION

APAS (Arc Process Analysis System) is a high reliable solution for welding analysis systems. The system is consists of one server (Fig. 1) and APAS controllers connected with welding signals through sensors (current, voltage, gas pressure i.e.). In small work plants data communication is not a problem, in large work plants (about hundreds of controllers) where controllers working simultaneous factor is greater than 0.8 (80%) data transfer between controllers and server can be a problem. In intervals when controllers send data at every 0.1 second data traffic can be overflowed.

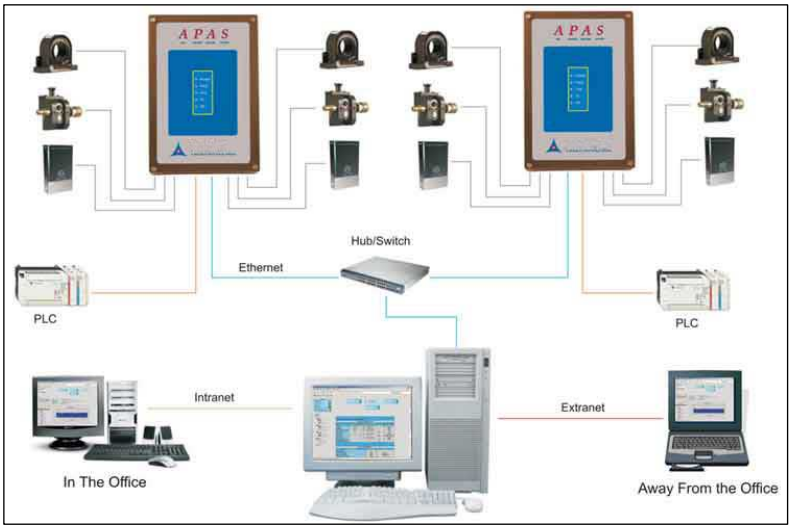

Fig. 1. APAS Server

2. NETWORK PROTOCOL

For obtaining a fast data transfer between controller and server we choose UDP (User Datagram Protocol) packets in place of TCP (Transmission Control Protocol) packets. The UDP packet will be delivered immediately in place of TCP packets (TCP packets must have a TCP connection before data communication begin). Because we haven't acknowledged of UDP packet receiving on transport layer level, this protocol isn't so reliable than TCP packet protocol. Because the network layer isn't locked when UDP packet is sent to an unknown destination, or destination isn't answer consists another advantage for this solution.

To solve the great disadvantage of missing data packet, which not arrived on server, we choose to solve this problem on this intermediate layer. Solution is simple: we create intermediate communication layer between the application and the TCP/IP communication stack – APAS Net Layer - (Fig. 2). This layer will create a reliable data communication solution.

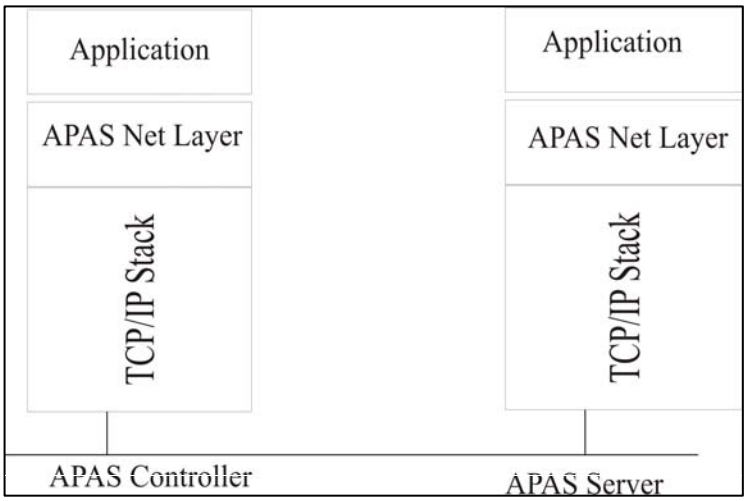

Fig. 2. APAS Net Layer

2.1 APAS Network Layer. Controller's Side.

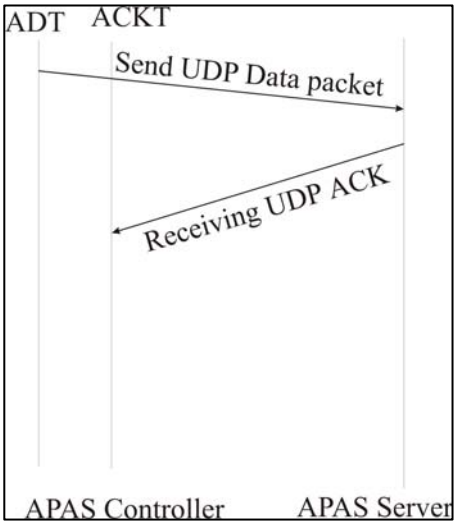

Fig. 3. APAS Acknowledge Thread

This layer is responsible with testing packet communication delivery and also this layer will resend loosed packets.

For implementing of APAS Net Layer we choose a three thread solution in a single task (see Fig. 3). The first thread (ADT - APAS Delivery Thread, see Fig. 3) is responsible with data sending to APAS Server, through TCP/IP stack. Also all sending packets are stored in a local database. The other thread (ACKT – APAS Acknowledge Thread, Fig. 3) will listen on a UDP port; acknowledge received packets, sent by APAS Server. If acknowledge is received the sent UDP packet from database will be removed.

For implementing of this protocol, each UDP data packet sent by ADT thread has attached an APAS_PACK_ID identifier. On receiving, each UDP sent by APAS server contains only the APAS_PACK_ID identifier.

If UDP sent packets by ADT doesn't receive acknowledge they remain in database, as unsent packets.

One simple solution is to trying to resend packets using the same solution via UDP communication port. This solution isn't so good because loading too much data communication layout and can create a UDP packet storm in local network.

Our solution is to create a third thread (ARUPT – APAS Recovery Unsent Packet Thread). This thread is TCP listener. He listens on TCP port server's request. If server is request a TCP connection this thread will be activated and will manage sending unsent data packets to the APAS server, over TCP connection. After data was sending, the TCP connection will be released, and local unsent packets database is empty.

2.2 APAS Network Layer. Server's Side

On server this layer is create from a dual thread task. The first thread (ADR – APAS Data Receiver) is responsible for receiving of UDP data packets sent by APAS controller. If packet is received corectly, the thread inform the sending controller that data are received, by sending a acknowledge packet (this packet contains ACK_PACK_ID). And in the same time the packet will be delivered to server application layer.

The other thread is a TCP thread (ARUD – APAS Received Unsed Data Thread). This thread is activated by scheduler event. When data comunication thread ADR is in suspended state, no data are delivered, this thread is activated and will check every controller about their unsent database packets (Fig. 4). The thread will receive the packets and will deliver them to application layer. The activation scheduler is depending by:

1. local database size;
2. the maximum frame rate determined by the welding unit;
3. the number of APAS controllers in the local network, who can send data simultaneous.

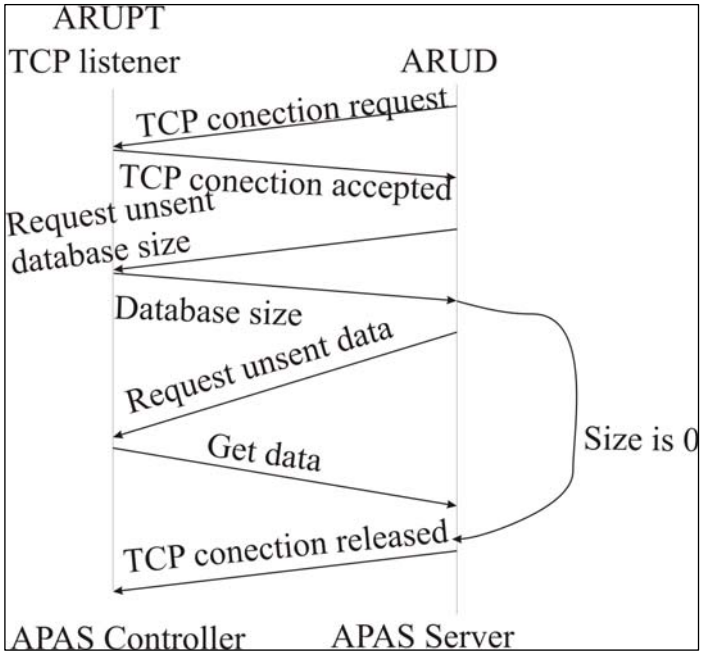

Fig. 3. APAS Received Unsed Data Thread

3. UNSENT PACKETS DATABASE

This database is created on APAS controller side, at APAS Net Layer level. The database contains each undelivered packet and his APAS_PACK_ID identifier.

When a packet is send the same package is saved in this database by ADT thread. If acknowledge is received for a packet, by ACKT thread, the packet is removed from database. If all is ok the database must be empty.

Periodic (at APAS server request) the ARUPT thread is check the database and empty this database by sending all records to the server.

If database is filled with packets, the older packets will be removed, and newer packets are stored in database.

The status of this database is monitored by class methods. When database achieved 85% from maximum size, a UDP signal will be sent, from APAS Controller to APAS Server, and announce server that unsent packet database achieve warning level. The APAS server must force staring of ARUD thread, for retrieving unsent packets. If database reach top level (it is full) another UDP signal will be sent to server announce that top level is rich and the unsent message database will start overwriting older records. From this time we can consider the system can loose packets.

There are a lot of solutions to don't reach at this point:

1. increase database maximum records count;
2. decrease the APAS server scheduler time constant;
3. analyze congestions in our network.

4. APPLICATION LAYER

This layer is remaining unchanged for main application and data connection between APAS server and controllers.

5. CONCLUSIONS

In welding plants, the automation systems send data with a large frame rate. The data acquisition controllers send data at small time intervals, and the size of data blocks can be very large. In large welding plants, with a big numbers of controllers (starting from about 100 controllers) this can be a problem. In their network (in conformity with IEEE-802.3) can appear a data packet storm or/and the destination server can be locked. In this case we can have loosing of data packets.

The solution proposed describes a data revival mechanism, for prevent loosing of packets. This solution is implemented at application level of TCP/IP stack, on both sides (server and controller). The solution is implemented in Marathon Weld's APAS (Arc Process Analysis System) data acquisition system.

6. REFERENCES

[1] Tanenbaum A.S., *Computers Network*, Computer Press Agora, 1997 (in Romanian).

[2] Boian Lorian M., *Internet Distributed Programming, Method and Apllications*, Blue Ed, 1997 (in Romanian).

[3] Joni N., N. Trif, *Automation Welding*, Lux Libris Ed., 2005, pp 35-68 (in Romanian).