DATA TRANSFER PROTOCOL FOR DISTRIBUTED INFORMATION ACQUISITION FOR USE IN PHYSICAL EXPERIMENTS

Alexei Korolkov[1]*, Alexei Moschevikin*[2]*, Alexei Soloviev*[3]

Abstract *— Data Transfer Protocol for Distributed Information Acquisition (DTP/DIA) was designed and proposed to IETF to be an experimental standard (TCP port 3489 have already assigned to DTP/DIA by IANA). DTP/DIA is based on TCP/IP stack and supports such devices as sources of physical information (sensors), transit transmitters and receivers, connected with data base engines. All these devices are usually constructed with embedded microcontrollers (Atmel, Microchips, Intel) with reduced cost. The main features of DTP/DIA are: simple realization, reliability, ability to resolve various sources of information, support up to* 2^{24} *unique devices. Due to implementation of TCP procedures and utilization of world communication channels sources and receivers may be far distanced. Practical realization of DTP/DIA was introduced in net of online air temperature sensors in North-west Russia (http://thermo.karelia.ru/eng/). Also there is an ability to construct measurement and data acquisition nets for any physical quantity including mixed nets (for example for pressure, humidity and temperature sensors).*

Index Terms — data transfer protocol, distributed information acquisition, embedded microcontrollers, online air temperature, sensor net.

INTRODUCTION

The Data Transfer Protocol for Distributed Information Acquisition (DTP/DIA) was developed for application in distributed information measurement systems (IMS). DTP/DIA provides the functionality of the presentation and application layers of the OSI Reference Model for the specified purposes.

IMS stands for a bundle of software and hardware which performs acquisition, processing, storing, and presentation of measured data. The systems with control function are out of scope. The simplest IMS consists of measuring device, connected to computer by means of some instrument interface (such as IEEE 488 (GPIB) or EIA/RS 232). Typical measuring device contains a sensor and a microcontroller, which performs initial data acquisition and processing. In such a system the majority of IMS functions are given to computer.

More complicated systems can be developed on the basis of CAMAC or VXI. But projects with large amount of investigated objects at far distances from each other require to install distributed IMS. Some nodes of distributed IMS should perform data transmission to nodes which process and store data. Sometimes this communication can be done by measuring device itself if it is rather sophisticated, or this function may be performed by a computer. That is we deal with two kinds of data channels: local (1) and network (2) (Fig.1).

In description of IMS we use the term "data source" for the object, which transmits measured data, and "data collector" - for the node which receives measured data. If the node performs both reception and retransmission, this type of nodes will be named "retranslator". Obviously, in this terminology measuring devices are "data sources", but a single computer could be either "end-point data collector" or "retranslator".

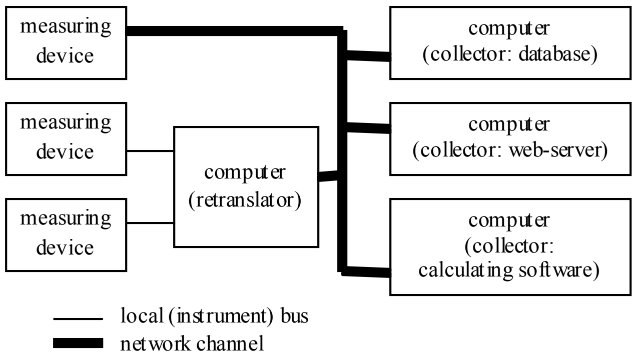

FIGURE. 1
A TYPICAL STRUCTURE OF DISTRIBUTED IMS.

In most cases the nodes of distributed IMS are considered to be the components of the open systems. So OSI Reference Model can be applied to their communication channels. The upper layers of such systems are poorly standardized due to the wide range of IMS's applications. The most of existing standards are vendor-specific. That's

[1] Alexei Korolkov, Petrozavodsk State University, Department of IMS & PhE, 33 Lenin Str., 185640 Petrozavodsk, Russia, phone: +7(8142)711021, e-mail: lehkor@lab127.karelia.ru
[2] Alexei Moschevikin, Ph. D, Petrozavodsk State University, Department of IMS & PhE, alexmou@dfe3300.karelia.ru
[3] Alexei Soloviev, Petrozavodsk State University, Department of IMS & PhE, avsdov@yahoo.com

The research described in this publication was made possible in part by Award No. PZ-013-02 of the U.S. Civilian Research & Development Foundation for the Independent States of the Former Soviet Union (CRDF) and of Ministry of Education of Russian Federation.

why it's necessary to introduce a simple protocol, suitable for both kinds of data channel. DTP/DIA matches these requirements.

DTP/DIA was implemented in both embedded firmware (for Atmel microcontrollers) and personal computer software (Distributed Information Acquisition software – DIA).

MAIN FEATURES OF THE DTP/DIA

The protocol describes data transmission in small stand-alone packets. This feature allows using DTP/DIA over both streamed (connection-oriented) and message-oriented channels. DTP/DIA packet consists of three logical blocks: *Header*, *Measured Data* and *Special Data*. *Header* and *Measured Data* are required. They have a fixed size. *Special Data* block is recommended and its size is variable. The maximal size of DTP/DIA packet is 60 bytes. A simple measuring device must produce packets, contained at least *Header* and one of the forms of *Measured Data* block. More sophisticated device should include *Special Data* block as well.

DTP/DIA provides few data presentation forms. Each form corresponds to a specific packet type. Some of the forms help to avoid the implementation of floating point arithmetic in device firmware.

DTP/DIA allows transmission of measured data accompanied with information about measurement error and its unit measure.

In distributed IMS it is very important to distinguish measuring devices from each other at the application layer. So the protocol includes identification mechanism. This feature helps to transmit data from several data sources over a single communication channel. For example, one can use a device with several sensors (such device is represented as multiple data sources). To distinguish data from various data sources DTP/DIA packet contains Data Source Identifier. To introduce a bit of structurization in distributed IMS the Identifier was divided into two parts. They correspond to high and low levels of 2-level hierarchy. Developer of IMS may use the first part to specify the group of units in distributed IMS and the second part to appoint a device number in this group. The protocol supports up to 16 millions (2^{24}) data sources.

Another important part of the DTP/DIA identification mechanism is an optional feature called the *Data Source Identification Procedure,* which helps sharing a single transport layer connection for data transmission from several data sources. This feature is not supposed to be applied to connectionless (message-oriented) channels. So data source can distinguish every collector by a certain transport layer connection. This mechanism consists of two events: "Offer To Identify" and "Device Request". They represent a some kind of handshaking. Also *Data Source Identification Procedure* may be applied for assigning identifiers.

We suppose to apply DTP/DIA for both network and local channels. Few types of local channels don't provide reliable data delivery. To avoid unreliability of local interfaces DTP/DIA has the following primitive features: detection of packet start (packet leading sequence), check sum and time stamp.

DTP/DIA provides neither authentication mechanism, nor security issues. To produce security based applications developer of IMS should provide authentication and data protection on the transport layer (by means of SSL [2] or TLS [1]).

DIA SOFTWARE

Distributed Information Acquisition Software fully implements the described protocol. It was developed and tested on several platforms: Windows 95/98/Me, Windows NT/2000/XP and Linux. The program was also tested on FreeBSD in Linux-emulator. Linux version can be linked statically and placed with the Linux kernel on a floppy disk. So it can be used in diskless low-performance systems (floppy required only). The main part of the program is provided with open sources under the terms of the GNU Lesser General Public License.

The program deals with objects of two kinds: *Device* and *Channel*. *Device* objects represent a particular data source and store current measuring information (date of measurement, measurement result, accuracy, units etc). *Channel* objects hides the method of data receiving and transmitting from other parts of the program. Such object implements input/output procedures for a specific communication resource (serial port, TCP-socket, UDP-socket, file, database, etc). Although it is not limited by the protocol, the current release of the DIA software doesn't support bidirectional *Channel* objects, i.e. any *Channel* object must either transmit data or receive it. Each *Device* object may be connected with few transmitting or receiving channels. Transmitting channels implements the "retranslator" function in the terms of DTP/DIA. Moreover the data storing function is the transmitting function as well. That is when we suppose to store measuring information in a file or database we assign a certain transmitting channel to corresponding object, which "transmits" data to file or database.

The open-source interface of objects interconnection and the modular structure of the program allow developing a module for any communication resource. The program can use new modules without recompiling. At the moment we have implemented the following communication modules:

- RS232 module performs communication via serial line.
- FILE module allows receiving of data from file. It is supposed that new data arrives when file modification time is changed. This module can be flexibly configured to support different file formats.
- MYSQL module provides the opportunity to store data in MySql database.

- TCP/UDP module supports receiving and transmitting of data over Internet/Intranet channels ([5], [6]). When the program transmits data by means of this module it acts as retranslator in the terms of DTP/DIA. This module can be configured to use SOCKS5 proxy [3].
- SSL module performs secure receiving and transmitting of measuring information. It provides some kind of authentication. At the stage of SSL handshaking both communicating sides should provide certificates, which are signed by trusted certification authority. Otherwise the connection establishment won't be authorized and SSL handshaking will fail.
- CTRL module implements some control interface, which allows creation and removing of both kinds of objects in the run-time. This feature provides the opportunity to configure the distributed IMS dynamically. By means of this module the operator may watch the state of all the objects in the running program.
- SMTP module implements a simple SMTP client according to [4]. It is supposed to inform the operator about long network time-outs. Because of various SMTP gates this information may be delivered to SMS, pager or ICQ.

DTP/DIA IN MICROCONTROLLER FIRMWARE

The realization of DTP/DIA in microcontroller built-in software is rather unsophisticated due to simplicity of the protocol. Binary code of packet formation and transmission procedure does not exceed 100 bytes in case of hardware UART in MCU (microcontroller unit).

One of the examples of DTP/DIA implementation is temperature sensor, based on Atmel MCU AT90S2313, which has the following useful features:

- 2K bytes of in-system programmable flash memory;
- 128 bytes of SRAM;
- programmable watchdog timer;
- full duplex UART;
- low power consumption.

Block scheme of device is presented in Fig.2.

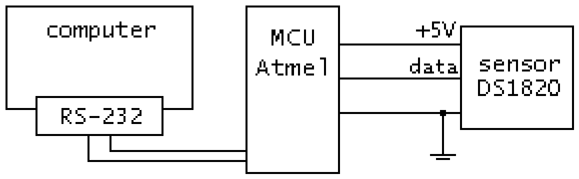

FIGURE. 2
BLOCK SCHEME OF DIGITAL TEMPERATURE SENSOR.

Digital sensor DS1820 (Dallas Semiconductors) is connected to MCU through 1-wire interface in slave mode. MCU puts various commands on data line and sensor replies. The whole cycle of data acquisition includes: sensor reset, temperature registering procedure, another sensor reset, temperature read procedure, processing read data, transmitting the data and sleep period (aimed at power saving). The stage of processing is necessary when digital format of obtained data does not comply with the certain field specification in DTP/DIA. For example, the most common format of DTP/DIA packet assumes the transmitting real physical value multiplied by 10 to escape fractional part (usually, it is sufficient to know value with accuracy of ±0.1).

Lets suppose that we read 19 from scratchpad of sensor (that means temperature of 19°C). Decimal X=19 in binary form is expresses as 00010011. This value should be multiplied by 10 and then be transmitted to computer-collector or retranslator (all calculation must be performed by MCU having no such direct instruction). Multiplication by 10 is appeared to be produced as 8*X+2*X=10*X. Multiplication by 2 and 8 carries out by simple instruction LSL (logical shift left). Once applied, new value in MCU's register is two times greater than previous. Let R0 to be a 8-bit register, then…

command	R0-register (00010011 - initial)	decimal expression in R0-register	action
lsl	00100110	38	*2
lsl	01001100	76	*2
lsl	10011000	152	*2

If the values in the first and third strings are summed the result will be 190.

Each DS1820 has unique 64-bit hardware number. They are partially used in forming Data Source Identifier field in DPT/DIA packets. Due to this feature few sensors may be connected on the 1-wire interface to one MCU. Microcontroller distinguishes various sources, so any distributed network of sensors may be constructed both with use of local and network channels (Fig.1).

ONLINE AIR TEMPERATURE SENSORS NET

On the basis of described protocol the online net of air temperature sensors was created (web version see at http://thermo.karelia.ru/eng/). It covers the territory of Republic of Karelia (North-west Russia). Also one of the sensors is distanced at several thousand kilometers from Petrozavodsk and is situated in Mezhdurechensk (Kemerovsky region). Geographical scheme of Karelian region is presented in Fig.3.

Three letter abbreviations on the map stands for different towns in Karelia. PAA – Paanayarvi, LOU – Louhi, SEG – Segezha, PTZ – Petrozavodsk, SOR – Sortavala. Arrows near the names of towns designate the current rise or fall of air temperature in respect to previous hour value.

All sensors yield data every 5 minutes and transmit them through Internet channels to the collector server situated in Petrozavodsk. Collector, database and web-server software is running on one computer. The main difference between various sensors is the way of data transfer and physical connection of sensors to data transfer devices.

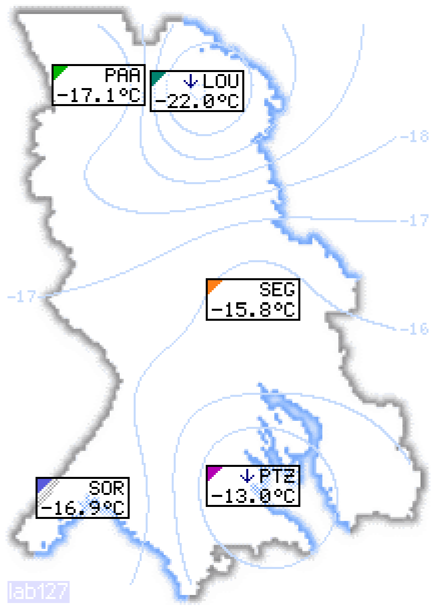

FIGURE. 3
KARELIAN MAP MARKED WITH TEMPERATURE MEASUREMENT POINTS.

Graphs (3-day, 3-week, 3-month and 1-year periods) with time dependence of air temperature are renewed every half hour.

Almost all temperature sensors are connected to network in various ways. LOU sensor is connected to COM port of computer running under Windows NT, SEG – Windows 98. Both are hidden behind firewall. Sortavala's is connected to gate's COM port, which is mapped to certain TCP port (feature of Cisco Systems hardware). All computers except PTZ and Mezhdurechensk (MER) sensors have Internet accessible IP addresses. PTZ and MER have local IP addresses and are translated by NATs (network address translation services). Moreover, PTZ sensor is constructed on not DS1820, but on MAX6577. So data transmission is carried out through FILE channels. To establish more security SSL connection with authentication and certifications was implemented between retranslator and collector computers. PTZ computer runs under Linux, and MER computer runs under FreeBSD.

Collector software stores temperature data from various town in corresponding log files. Web-server software and additional scripts build graphs, banners, and other required for certain web-design objects in HTML-pages.

Due to universality of DTP/DIA the same network channels may be used for transmission of various types of physical quantities. Any packet may include fields for quantity identification, unit measure (for example, °C), confidence interval and accuracy. It is up to collector software to determine each source and physical quantity sent.

ACKNOWLEDGMENT

We'd like to express gratitude to all our colleagues at the department of information measurement systems and physical electronics of Petrozavodsk State University for support in this project.

GLOSSARY OF ACRONYMS

CAMAC	Computer Automated Measurement And Control
EIA/RS	Electronic Industries Association Recommended Standard
GPIB	General Purpose Interface Bus
HTML	HyperText Mark-up Language
IANA	Internet Assigned Numbers Authority
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IMS	Information Measurement System
OSI/RM	Open Systems Interconnection Reference Model
SMTP	Simple Mail Transfer Protocol
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UART	Universal Asynchronous Receiver and Transmitter
UDP	User Datagram Protocol
VXI	VME bus eXtension for Instruments

REFERENCES

[1] Dierks T., Allen C., "The TLS Protocol Version 1.0", *IETF RFC 2246,* 1999.

[2] Frier A., Karlton P., Kocher P., "The SSL 3.0 Protocol", *Netscape Communications Corp.,* 1996.

[3] Leech M., etc., “SOCKS Protocol Version 5”, *IETF RFC 1928*, 1996.

[4] Postel J., “Simple Mail Transfer Protocol”, *IETF RFC 821*, 1982.

[5] Postel J., “Transmission Control Protocol”, *IETF STD 7 (RFC 793)*, 1981.

[6] Postel J., “User Datagram Protocol”, *IETF STD 6 (RFC 768)*, 1980.