Towards Multi-port Modbus Gateway

Claudiu Chiculita
Ene Liviu
Dunarea de Jos University of Galati, Romania
Galati, Romania

Mihai Lucian Cristea
REDANS SRL, Romania
Galati, Romania

***Abstract*—Integration of various equipments into a distributed automation project becomes a necessity for building large automation systems and rises problems of networking of heterogeneous nodes in a complex communication network. In this paper, we investigate the interconnection problems of nodes into an industrial network using Modbus protocol, and propose solutions to address the problems optimally.**

***Keywords*—industrial networks, modbus, embedded systems.**

I. INTRODUCTION

Automatic control systems for current and future industrial processes grow in complexity as they address an increasing range of problems. For example, a typical process automation was aimed at optimizing the productivity in the past. Now, besides productivity, the automation process takes into account also other factors such as efficiency (cost of energy, materials) or security. The increasing complexity of automation control system leads to an interconnection of a growing number of components. These components (e.g., engines, valves, energy consumption monitoring, safety devices) are distributed in the automation field at various distances (from a few meters to kilometers). Despite the increased performance of each component (due to technology enhancements), ensuring the automation system performance depends also on the interconnection of distributed components and hence, on the communication network that becomes more and more complex.

Although an industrial network uses the same principles of packet switching/routing over wires in an 'office' network, it has a few technical differences as follows. For example, the nodes use various wire connections (e.g., 2 wires, 4 wires) to support slow serial communications (RS232, RS485, RS422, CAN) over long distances in a noisy environment (strong electromagnetic fields, random perturbations). Besides the speed, variety of layer 2 communication protocols and the strong perturbation, there is a need for a protocol that ensures a deterministic behavior as required by real-time industrial automation processes. Therefore, these technical characteristics become a challenge for building gateways (routers/switches) that handle the data traffic in a distributed automation system (see Fig. 1).

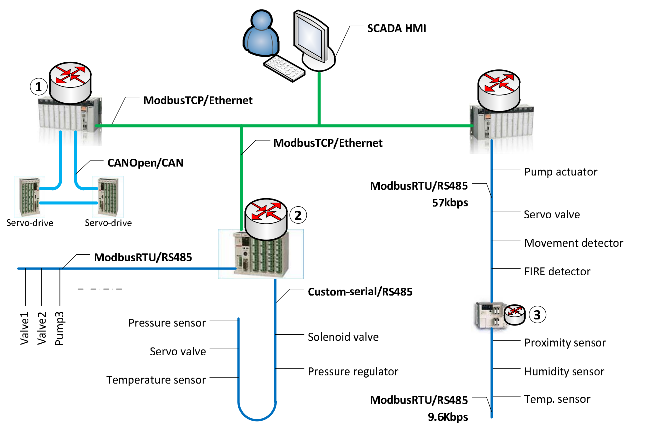

Fig. 1. Industrial Network

Although there is a need for gateways that offer multiple network ports in order to integrate more different networks easier and at lower costs, we notice that the market offers gateways that provide limited number of ports (mostly offers only two ports with one to one connectivity such as www.hms.se, www.icpdas.com).

Another typical way of integrating various industrial equipments into a network is using Programmable Logic Controllers (PLCs) with multiple interfaces. However, these PLCs were designed for data acquisition and processing of data, not for packet processing and protocol handling. Therefore, the cost of implementing a distributed automation systems using PLCs as gateways is higher than that of using dedicated systems (routers, switches, gateways) due to the price of PLCs with communication interfaces and the cost of man-hours to program the PLC functionality as gateway.

We believe that dedicated systems for packet processing such as multi-port gateways will provide the required features (integration of heterogeneous nodes, lower man-hours need to deploy, etc.) to build large scale distributed automation systems.

In the next section, we present the implementation and evaluation of our multi-port gateway for the Modbus industrial protocol. The article concludes with future directions to enable multi-protocol network gateways.

II. Multi-port Modbus Gateway

Modbus is a serial communication protocol that became a de facto standard for interconnection of devices in a industrial network. Modbus is openly published and royalty free and was designed with industrial applications in mind [1,2].

The *MobGate* made possible the communication between devices that are placed in different networks. It is designed to pass the messages from one port to another and if necessary to convert the messages from one protocol to another.

The prototype hardware of the *MobGate* was built around the PIC32 microcontroller from Microchip, that is capable of running up to 80MHz, has 512kB flash and 128kB RAM, and has a very large set of peripherals. The main peripherals used are the Ethernet and four UARTs (see Fig. 2) through which the Modbus communication will take place.

The prototype hardware is realised on three stacked boards.

The first two, from chipkit platform have the pic32 microcontroller and ethernet; the third has the UART/RS485 communication ports (see Fig. 3).

III. Software Implementation

A. Modbus Stack

The *MobGate* uses freemodbus library that is an implementation of the popular Modbus protocol specially targeted for embedded systems [3]. It is divided into three layers (User Application, Modbus Application, Porting Layer) and supports RTU/ASCII [4] and TCP [5] transmission/reception modes.

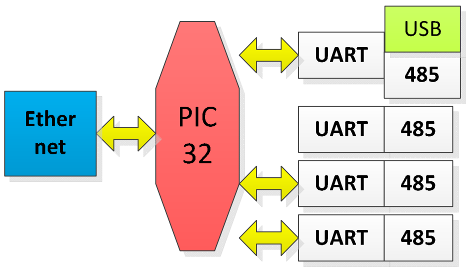

Fig. 2. Hardware architecture

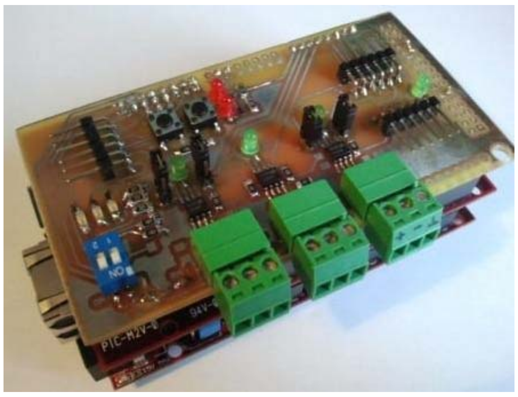

Fig. 3. Router architecture.

The following Modbus functions are currently supported: Read Input Register (0x04), Read Holding Registers (0x03), Write Single Register (0x06), Write Multiple Registers (0x10), Read/Write Multiple Registers (0x17), Read Coils (0x01), Write Single Coil (0x05), Write Multiple Coils (0x0F), Read Discrete Inputs (0x02), Report Slave ID (0x11).

1) User Application Layer:

Consists in application specific functions and structures. For a Master device this layer contains the Modbus Slave requests and the data response. For a Slave device this layer contains the callback functions that generate the Modbus response [6].

2) Modbus Application Layer:

It is implemented by the stack author and makes the connection between User Application and Porting Layer. Here are included all the functions and structures that assemble and prepare to send a Modbus request and prepare to receive and disassemble a Modbus response. Also this layer includes specific structures for every Modbus function.

3) Porting Layer:

It is hardware specific and contains functions that physical receive and transmit Modbus frames.

For Modbus TCP/IP it contains the Microchip TCP/IP Stack-a suite of programs that provides services to standard TCP/IP applications). Microchip's TCP/IP stack [7] includes the following key features: Supported Protocols-ARP, IP, ICMP, UDP, TCP, DHCP, SNMP, HTTP, FTP, TFTP, Socket support for TCP and UDP, Secure Sockets Layer (SSL), DNS–Domain Name System, Ethernet Device Discovery.

B. Free RTOS

Free RTOS (Real-Time Operating System) is a class of RTOS that is designed to be small enough to run on a microcontroller - although its use is not limited to microcontroller applications [8,9]. Features: it provides methods for multiple tasks, mutexes, semaphores and software timers, tasks can be created having different priorities, the scheduler can be configured for both preemptive or cooperative operation, the thread tick method switches tasks depending on priority and a round-robin scheduling scheme, queue operations are supported, a tick-less mode is provided for low power applications, generic trace macros that provide trace support, tools such as Free RTOS+Trace (provided by the Free RTOS partner Percepio) can thereby record and visualize the runtime behavior of FreeRTOS-based systems (this includes task scheduling, user events and kernel calls for semaphore and queue operations).

The *MobGate* uses Free RTOS because it provides task management and task communication using queues. For every *MobGate* active port is created a task that will manage the port. Also is created a Routing task which according to a routing rule will make a virtual link inside the *MobGate* between two ports. The messages are passed between tasks using queues so for every task is created a queue.

Fig. 4 represents the *MobGate* architecture and it works as follows:

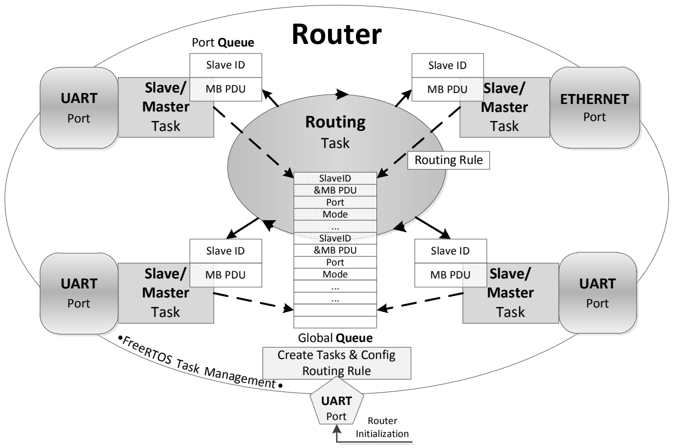

Fig. 4. Router architecture

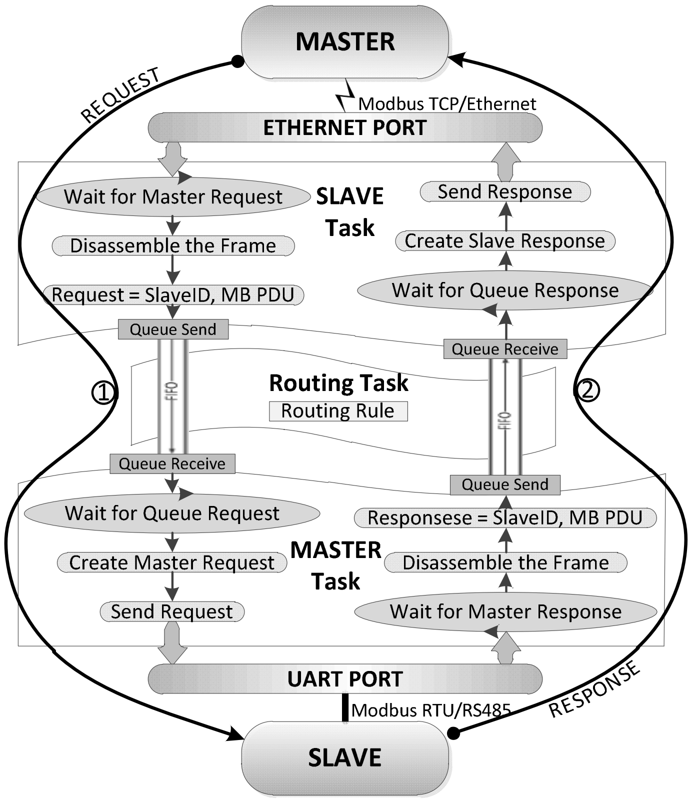

Fig. 5. Routing example between two ports

Initialization:

On startup the *MobGate* waits the user to enter some settings depending on which it will configure its operation. These settings will specify:

- What kind of task should be created for active ports:
  The type of the task – master/slave;
  The type of the protocol – Modbus RTU/ASCII/TCP;
  The number of the port;
  In the case of Modbus RTU/ASCII the baud rate.
- The Routing Rule depending on which the messages will be passed from one port to another.

After the settings have been entered tasks and queues will be created:

- First will be created the Routing task which according to the routing rule will pass the messages from one port to another using queues. For this task a Global Queue will be created that will be read by this task and will be written by the other tasks.
- For every active port a task will be created that will meet the specifications. Also, for every task will be created an own queue which will be read by the specific task and it will be written only by the routing task.

Every task is waiting for a new message: Slave tasks are waiting for requests from the network and Routing task and Master tasks are waiting for messages from the queues.

As shown in Fig. 5, when a request is received to a Slave port the specific task will take the message and disassemble it. The Slave Address, MB PDU (Protocol Data Unit), Port Number and Mode (master/slave) will be written to the Global Queue. Task will enter to the blocked state waiting for the response.

The Routing task will receive the message from the Global Queue and will check the Routing Rule to establish the destination port. The message will be written to the specific destination task queue.

After the message is received from the Master task queue the message fields will be used to assemble a request. This request will be sent on the network and then the task will wait for the response.

When the Master port receives the response from the network the cycle repeats:

- The Master task write the message to the Global Queue;
- The Routing task read the message, check the Routing Rule to establish the destination port and writes the message to the destination task queue;
- The Slave task receives the message, leaves the blocked state, read the message, assemble the request and sent it on the network.

Routing Rule:

To pass the messages from one port to another the Routing task uses a routing table. This table is checked when a message is coming to a port to establish his destination according to some parameters.

Considering the Table 1 the messages are routed as follows:

- The messages that are coming to P1, P2 ports with Slave Address 1-64 are transferred to P2, P1 ports;
- All the messages that are coming to P3, P4 ports are transferred to P4, P3 ports;

TABLE 1 ROUTING TABLE

Source	Slave ID	IP	TAG	Destination
P1	1-64	*	*	P2
P2	1-64	*	*	P1
P3	*	*	*	P4
P4	*	*	*	P3
P502	65-128	192.168.1.101	*	P2
P2	65-128	*	*	502

- The messages that are coming to P502 port (TCP/IP port) with Slave Address 65-128 and IP 192.168.1.101 are transferred to P2 port;
- The messages that are coming to P2 port with Slave Address 65-128 are transferred to P502 port (Ethernet port).

The "TAG" field will be used in a future implementation where Masters will address Slaves with the same ID that are placed in different networks.

IV. EVALUATION

When using a router for delivering Modbus packets, it is important to determine the delay introduced in transmitting packets. Experiments were performed with a fixed packet size, and different communication speeds, in the following three cases:

- Direct master and slave communication, without the gateway (Tstd).
- Master and slave communication through gateway without other traffic going through gateway ("Trouter").
- Master and slave communication through gateway while 2 additional ports of the gateway are used ("Trouter_incarcat").

The measurements for three typical baud rates are presented in Fig. 6.

It can be observed that the propagation time is double when using the gateway; this is due to the followings:

- The most part of the round trip time is due to the transmission delay using relative slow speeds;
- The processing done in router introduces a small delay;
- When the traffic goes through the router there are twice as much transmissions taking place (Fig. 7).
- When the traffic through the gateway is doubled (by using 2 more ports for communication) only a slight increase of about 1% is observed in delay.

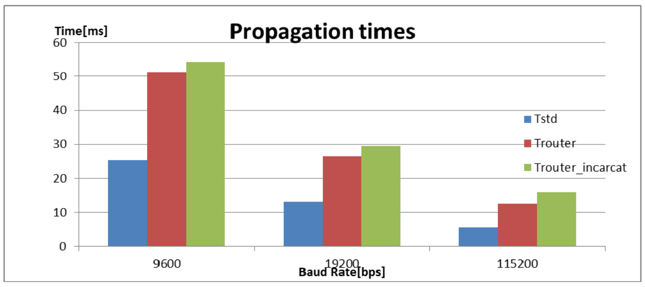

Fig. 6. Propagation times at different speeds

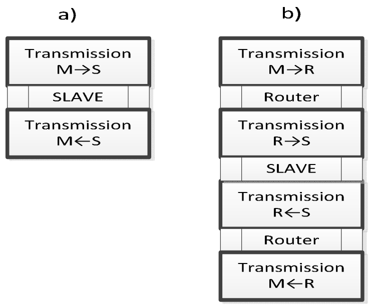

Fig. 7. Propagation times a) direct connection b) through router

V. CONCLUSION AND FUTURE WORK

This paper presented the *MobGate* multi-port router implementation and evaluation for Modbus traffic. Multiple usecases of master-slave configuration of ports were studied to prove the flexibility of the MobGate to integrate various automation equipments.

We are working on an enhancement to our gateway towards supporting other protocols and hence, other networks at the same time. Such a multi-protocol gateway will facilitate the integration of equipments into a distributed automation system even better.

REFERENCES

[1] http://www.modbus.org/
[2] http://en.wikipedia.org/wiki/Modbus
[3] http://www.freemodbus.org/
[4] Modbus Organization, "Modbus over Serial Line - Specification and Implementation Guide V1.02", 2006
[5] Modbus Organization, "Modbus Messaging on TCP/IP Implementation Guide V1.0b", 2006
[6] Modbus Organization, "Modbus Application Protocol Specification V1.1b", 2006
[7] Nilesh Rajbharti, Microchip Technology Inc., "The Microchip TCP/IP Stack", 2002
[8] http://www.freertos.org/
[9] Richard Barry, "Using The FreeRTOS Real Time Kernel, PIC32 Edition",2009