Design of a modified RFB protocol and its implementation in an ultra-thin client

Daniel Zinca
Communications Department
Technical University of Cluj-Napoca
Cluj-Napoca, Romania
Daniel.Zinca@com.utcluj.ro

Abstract— **This paper describes a modified RFB (Remote Frame Buffer) protocol intended to be used by LAN ultra-thin clients. The prototype implementation is based on the ATMEL AVR microcontroller family and on the Microchip's Ethernet integrated circuit ENC28J60, showing that a low cost microcontroller can implement the network functions. The main differences consist of the use of UDP as the transport protocol instead of TCP and in additional fields to the RFB header.**

Keywords - thin-client, zero-client, Embedded Ethernet, RFB ptotocol, VNC protcol.

I. INTRODUCTION

Thin clients are used in LANs and WANs to cut costs associated with maintenance and energy consumption. But today's solutions are becoming more complex due to the additional features (support for multiple Remote Access protocols, features needed in WANs, support for multiple USB interfaces, enhanced security, etc.). Usually, thin clients implement downsized versions of desktop operating systems, like Microsoft Windows CE or Windows Thin Client.

Recently, efforts were made in order to develop an architecture called ultra-thin client (or zero-client) where no operating system is running on the client hardware, but a minimal TCP/IP stack. The screen contents and the keyboard/mouse activity are sent between the client and the host. The cost can be substantially lower compared to the one of thin clients because energy consumption is even lower, there is no operating system maintenance nor software cost.

This paper describes a prototype implementation of an ultra-thin client (called also a zero-client) and focuses on the networking aspects.

Generally, the protocols developed for remote access fall in 2 categories:

- image frame buffers
- high level API display functions

In the image frame buffer category we have:

- RFB (Remote Frame Buffer) from Olivetti Research [1] used widely in VNC (Virtual Network Computing)
- SLIM, a stateless protocol developed by SUN [2] that was a precursor of ALP - Appliance Link protocol [9]
- PC-o-IP developed by Teradici [6]
-

Among the most used protocols in the second category are:

- RDP (Remote Desktop Protocol) by Microsoft that uses specific GDI display API calls
- ICA (Independent Computing Architecture) from Citrix, based on the same GDI API calls but optimized for low-speed WAN links.

The image frame buffer category is better suited for ultra-thin clients (sometimes called zero-clients) because it sends an array of pixels, without relying on complex, high layer graphics functions. RFB supports video only while the other protocols support audio also. In our implementation we are not interested in an audio implementation.

Recent advances in the area of zero clients was introduced by Teradici [6] that developed a protocol called PC-o-IP (PC over IP) where the host compresses the stream of bits from the display adapter and sends it to the client where the reverse process is performed. The disadvantage of this protocol is that it requires dedicated processing hardware at the host, increasing the Total Cost of Ownership. This protocol is supported by VMWare and an increasing number of vendors.

The remaining of the paper is organized as follows:

Section II describes a proposal for a modified RFB, optimized for use in LANs. Section III describes the ultra-thin client prototype that was built using SDKs from ATMEL and Microchip. Section IV is dedicated to the conclusions and future work.

II. THE MODIFIED RFB PROTOCOL

An ultra-thin client connects to a server in the LAN and the following assumptions are made:

- LAN with IEEE 802.3u 10/100 Mbps
- Layer 2 switches with VLANs (Virtual LANs) and CoS (Class of Service)
- very low BER

Based on these assumptions, we consider that UDP can be used as the replacement of TCP as the transport protocol in an RFB architecture. The advantages of using UDP (compared with TCP) when implementing a thin client are the following:

- simpler hardware implementation
- decreased redundancy

The drawbacks of using RFB on top of UDP:

- lower security: it is easy to initiate spoofing attacks
- no error control: lost packets can have an important influence on the information that is displayed
- possible lost packets during congestion periods.

We consider that the disadvantages of using UDP can be eliminated by proper design of the LAN. For each drawback we propose a solution that will be described here.

The security problem can be solved by moving the ultra-thin clients into an distinct VLAN along with the host switch port. Layer 2 communication will be permitted only between these computers.

As for the second problem, even if BER is very low in LANs, packets can be lost due to other causes, like congestion. Therefore an additional Sequence number field was defined and used, between the UDP header and the RFB header, for each packet. If a packet contains only data from the video buffer itself, the sequence number is sent.

The solution for the third problem is to use IEEE 802.1p CoS and to classify the traffic to an high-priority queue inside the switches between the ultra-thin clients and the host port.

In order to make RFB packet interpretation easier in hardware, an additional field called RFB Command was added. We encoded each possible message type defined in [1] with a different value. For example, the RBF Command has the value 05 h for sending the mouse and keyboard information from the zero-client. The list of headers starting with the one corresponding to Layer 3 is presented in Fig.1:

IPv4 Header	UDP Header	RFB Command (1 byte)	Sequence Number (1 byte)	RFB header and / or data

Figure 1.List of headers in a modified RFB packet

We set the sequence number size on one byte because is important to notice any packets that are lost. Also, the probability to lose a burst of 256 packets is very low. When the value in the Sequence No reaches 255 it will start from 0 again.

Because of these two additional fields and because of the usage of UDP on the Transport Layer instead of TCP, the existing client and server implementations (VNC) are not compatible with this one.

III. DESCRIPTION OF THE ULTRA THIN CLIENT PROTOTYPE

An ultra-thin client can be implemented using any of the following technologies:

- microprocessor based
- FPGA implementation
- SoC (System on a Chip)
- microcontroller based, using a dedicated video card

We choose a microcontroller-based implementation. For implementing the network communications, the interface with the keyboard and mouse, we choose an RISC microcontroller (less than 16 MIPS). The display interface is not yet implemented, it needs a different microcontroller or FPGA and remains for future study.

The prototype we implemented is based on the ATMEL AVR architecture. For the IEEE 802.3 stack we used Microchip ENC28J60 that can be connected to a MCU using its SPI interface.

For the AVR family a starter kit is available and was used: AVR STK500[3]. It incorporates an ATMega 8515L microcontroller, an SPI interface, an programmer, spare RS-232 port and AVR Studio for developing applications using assembly language. For easier development, a free software development tool called WinAVR [4] was installed, that include the GCC C and C++ compilers.

The Microchip ENC28J60 can be used by different microcontroller families due to its SPI interface. Microchip offers PICtail Ethernet Daughter board [5] that contains the ENC28J60, RJ45 connector and associated magnetics. The interface software is specific to Microchip family MCUs. For programming, source code is available in Microchip assembly language and in C. We modified the C source code to be compiled in a WinAVR GCC compiler. ENC28J60 is a IEEE802.3 10Mbps device, for the purpose of this project it is sufficient.

Based on the pin specifications on both devices we established the connection between AVR STK500 and Microchip Ethernet PICtail as described in Table 1:

Cable No.	SPI name	AT8515L Pin No.	PICTail pin no
1	SS -Slave select	PB4	RB3 (CS)
2	SCK Clock	PB7	RC3 (SCK)
3	MISO	PB6	RC4 (SO)
4	MOSI	PB5	RC5 (SI)

Table 1. Connection of the SPI interface between STK500 and PICtail

Each module has its own power supply. The corresponding pins in Port B of ATMega 8515L must be programmed and the interface must be initialised.

After initializing the SPI interface in the ATMega 8515L at the f/2 speed, each data byte is loaded in the SPDR register of ENC28J60. The transfer status can be checked in SPSR.

When sending a frame, the following commands are issued to the ENC28J60, in the following sequence:

1. WCR (Write Control Register: initializer)
2. WBM (Write Buffer Memory: writes the packet in the TX buffer)
3. BFS (send the packet on the network)

When receiving a frame, the following commands are issued:

1. RCR (pooled, verify if a packet was received)
2. WCR (point to the beginning of the buffer)
3. RBM (read status and the packet)

Packet reception is performed in a loop. When reception is complete, the FCS is eliminated and the data bytes in the received buffer are further processed.

The protocols /headers that must be implemented are:

- IEEE 802.3/Ethernet for the frame structure
- IPv4 header
- UDP header
- RFB modified protocol

The implementation of the Ethernet frame structure offloads the computing of the FCS to ENC28J60. All the other protocols were implemented in C language and embedded in the microcontroller.

The protocols that are useful to be implemented are:

- ARP
- ICMP
- DHCP client

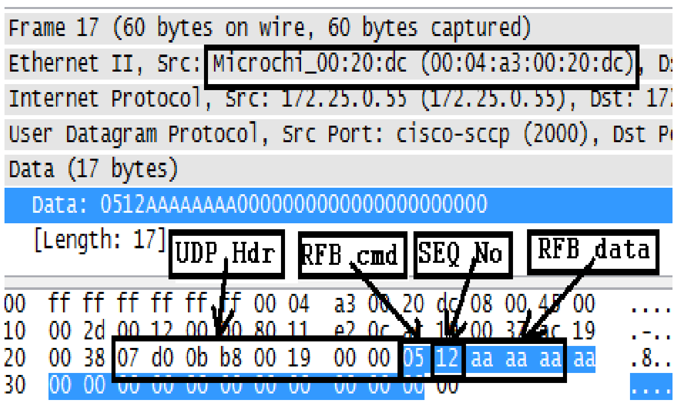
ARP is useful because the ultra-thin client must find the MAC address of a corresponding IP address and must respond to ARP requests made by another computer.

The implementation of ICMP is useful for testing purposes. The Echo request and Echo reply messages are implemented.

The implementation of DHCP is useful in order for the ultra-thin client to obtain an IP address and other necessary parameters. A DHCP option can be added to specify the IP address of the host.

From these 3 protocols we implemented the first two, leaving the DHCP client implementation for future development.

Fig.2 displays a packet sent by our prototype and captured with Wireshark [10]. The highlited fields are those described in Fig.1. The current values are: 05h for the command (send the mouse position and/or the key press) and 12h for the Sequence number.



```
Frame 17 (60 bytes on wire, 60 bytes captured)
Ethernet II, Src: Microchi_00:20:dc (00:04:a3:00:20:dc), D
Internet Protocol, Src: 172.25.0.55 (172.25.0.55), Dst: 17
User Datagram Protocol, Src Port: cisco-sccp (2000), Dst P
Data (17 bytes)
  Data: 0512AAAAAAAA0000000000000000000000
  [Length: 17]

00  ff ff ff ff ff ff 00 04  a3 00 20 dc 08 00 45 00   ....
10  00 2d 00 12 00 00 80 11  e2 0c ac 19 00 37 ac 19   .-..
20  00 38 07 d0 0b b8 00 19  00 00 05 12 aa aa aa aa   .8..
30  00 00 00 00 00 00 00 00  00 00 00 00               ....
```

Figure 2.Packet sent by the ultra-thin client prototype

Because of the incompatibilities with existing versions of RFB, the server source code had to be modified. We used Microsoft Visual Studio 2008 Professional Edition to implement RFB for use by VNC and our modified version that will work with our ultra-thin client. The programming language was C#. The main tasks performed by the server are:

The server starts operating as a service. It will wait for connections from hosts. The number of simultaneous connections depends on the operating system. If the operating system is a desktop one (like Microsoft Windows XP or Windows 7) then one remote connection is allowed. For server operating systems, multiple simultaneous connections are available.

After establishing the connection, the protocol enables the transmission of screen contents from the server. In order to improve the use of the network channel, detection of the modified screen at the server was performed and a packet with that modified screen is sent to the client. The transfer mode is Raw, leaving for further implementation the RRE, HExtile and ZRLE because it requires more processing power in the embedded client.

The client sends the keys (using the KeyEvent message) and the mouse positions (PointerEvent message) back to the server. At the server side, the commands are run and the result is displayed on the screen and then sent to the client. Periodically the client sends FrameBufferUpdateRequest. As a response, the server detects the portions of the image that are modified and send them in an FrameBufferUpdate.

In order to test the communications between the server and the embedded client, all packets that were received over the network connection were sent to the spare serial V.24 interface. From here the characters were captured with a terminal emulation software and saved to a file for later comparison. The buffer contents is limited to 1024 bytes but is sufficient to test the transfer of the FrameBuffer.

IV. Conclusions and future work

Until now, the implemented modules are those related to the network communications (the implementation of the proposed modified RFB protocol and the other protocols below), the keyboard interface and mouse interface.

For future work we intend to design a solution for the display subsystem and to integrate all the modules.

We intend to study the implementation of IPv6 as the network layer protocol.

As a conclusion, the proposed modification to the original RFB protocol is appropriate for use in a ultra thin client, as verified by the microcontroller implementation. The proposed protocol can also be implemented using other hardware families with minimal modifications.

[1] T. Richardson, "The RFB Protocol", Version 3.8, August 2008, http://www.realvnc.com/docs/rfbproto.pdf

[2] B. K. Schmidt, M.S. Lam, J. D. Northcutt, "The interactive performance of SLIM: a stateless, thin client architecture", 17th ACM Symposium on Operating Systems Principles, December 1999, pp 32-47.

[3] ATMEL, "AVR STK 500 Starter Kit and Development System," http://www.atmel.com/atmel/acrobat/doc1939.pdf

[4] WinAVR, "WinAVR project", http://winavr.sourceforge.net/

[5] Microchip ,"Ethernet PICtail daughter board",P/N AC164121

[6] http://www.teradici.com/pcoip/pcoip-technology.php, retrieved at 10th of June, 2010.

[7] http://msdn.microsoft.com/en-us/library/cc240445%28PROT. 10%29.aspx, retrieved at 10th of June, 2010.

[8] http://en.wikipedia.org/wiki/Independent_Computing_Architecture, retrieved at 10th of June, 2010.

[9] http://en.wikipedia.org/wiki/Appliance_Link_Protocol

[10] http://www.wireshark.org, retrieved at 10th of June, 2010.