IP-Shield Machine(IPSM): An Ethernet Interface for High Speed Packet Radio

Satoshi Funada, 7M3LCG
Packet Radio Users Group of Japan
P.O. Box 66, Tamagawa, Setagaya-ku, Tokyo 158-0094 JAPAN
sfunada@cs.titech.ac.jp

Abstract

In PRUG96 project, we developed an Ethernet interface for high-speed digital transceiver, called IPSM-ZZ. IPSM deals with a only Media Access Control layer protocol. With its partner, Protocol Server deals with upper layer protocols. It allows us to develop upper layer protocols flexibly in common operating systems.

Keywords: IPSM, PRUG96, High-Speed Packet Radio, Ethernet interface

1. PRUG96 system and IPSM

The PRUG96 project, consists of PRUG members, aims to realizing the practical High-Speed Digital Network. The area, we develop, lies from the physical layer to the network protocols.

Figure 1 shows the schematic image at which we set our goal. A small transceiver and a small computer are settled in a lunch-box-size box. We mount this box on the roof, just below the antenna, hence coaxial cable should be short as possible; we use GHz order microwave in our system.

On the other hand, we aim at Mbps order data transfer rate, which is too fast to catch up with for conventional RS-232C I/F. We, the PRUG96 project, decided to use Ethernet I/F, which is popular in

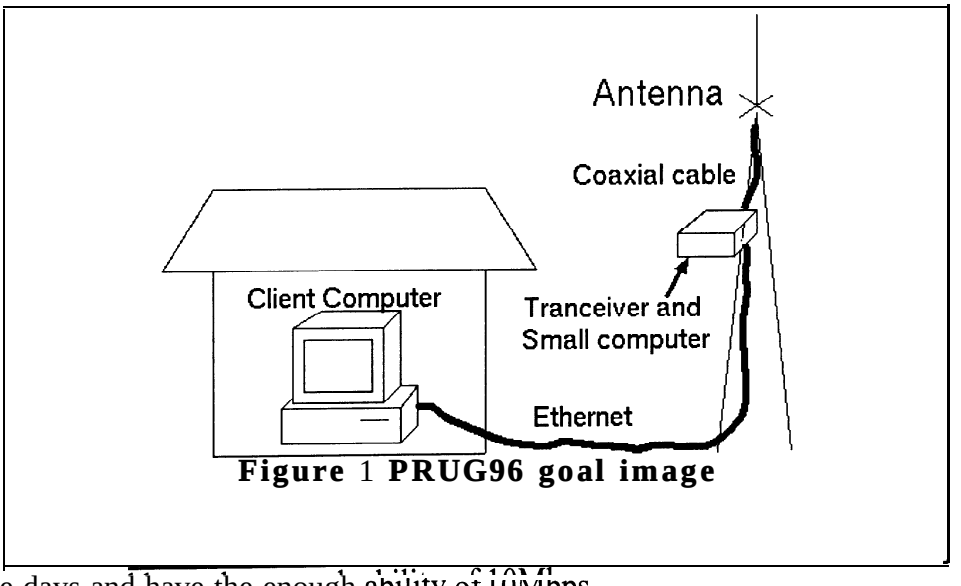

Figure 1 PRUG96 goal image

PCs these days and have the enough ability of 10Mbps.

However, it was very difficult to design physical layer to network layer and to implement them into such a small size device shown in figure 1. In addition, we thought that our system should have a flexibility to make the best use of the resource of amateur radio, such as radio spectra, output power and antennas.

We decided to divide the function of the box into two parts. One part, packet assembling and routing,

upper region above link layer are managed by a PC. Another part remains in the box.

This division makes it possible not only to reduce the cost of the microprocessor in the box, but also to examine and to estimate the routing performance easily, which should be examined again and again.

We designated this method “Protocol Server Method.” At first, The data streams to be transmitted processed in the PC called Protocol Server(describe as PS below). Routings and Packet-assembling are done in the Protocol Server and then handled over to the transceiver via the Ethernet, and finally packets are emitted. When the transceiver receives packets, completely reversed procedure is done.

The functions of the box are: To transmit/receive packets; To control the transceiver; To communicate with the Protocol Server(describe as PS below) via the Ethernet. The box contains two pieces; one is the transceiver and another is the small computer, the IPSM (Internet Protocol Shield Machine) which will be described later.

2. PRUG96 mechanism

Before explaining how does the IPSM work, let me talk about the mechanism --- how to transmit/receive a packet in our PRUG96 system.

The PS and the IPSM are connected by the Ethernet. The computer to be accessed with the system, the client, is connected with the PS by the Ethernet or other ways.(See figure 2) Only IPv4 is supported currently. Of course, the client can be connected to the same Ethernet, where the PS and the IPSM are connected. The PS can be regarded as a ' IP router ' from the viewpoint of the client.

Think, the client wants to emit data streams. The client sends a IP packet to the PS. The PS receives a packet and: Adds callsign, comparing inside routing table; Adds Forward Error Collection codes;

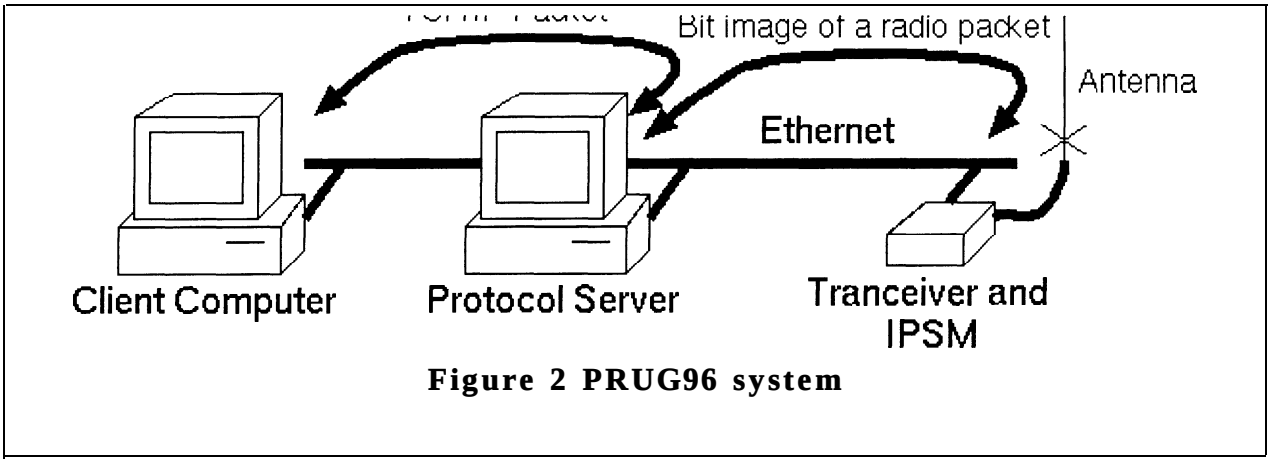

Figure 2 PRUG96 system

Creates bit images of the packet to be emitted. Then, the PS encapsulates the bit images in a UDP packet and sends to the IPSM. After the IPSM receives a UDP packet, the IPSM removes IP/UDP header from the UDP packet and hands them to the transceiver. Finally, the transceiver emits the packets on the air.

When the transceiver receives a packet, completely reversed procedure is done. The IPSM encapsulates a data image of the received packet in a UDP packet and sends it to the PS, without any consideration whether the packet is addressed to its site or not. The PS analyzed the packet, and if the packet is addressed to its site, transmits it to the client, otherwise throw away the packet.

The IPSM has nothing to do with the TX/RX data streams. The IPSM hand the TX packet to the transceiver and the RX packet to the PS. This method makes the IPSM independent from upper

protocols and therefor we can develop the IPSM alone.

3. PRUG96 MAC layer Protocol

Even though the IPSM has nothing to do with any protocols, it must select the real data from the RX packet. In this chapter, the author will explain about MAC(Media Access Control) layer that the IPSM has to treat.

3.1 Frame structure

In the current version of our PRUG96 system, the TX/RX frame has 1408byte(octet) length, as shown in figure 3. The 32bit length PN(Pseudo Noise) code is added in front of the 1404byte length data stream, in order to synchronize. The reason why the length of a packet is fixed is that we want make it easier to treat the data streams.

If the frame length is flexible, the code which indicates the end of the data and data length are required. The IPSM needs to work considering those code and length. On the other hand, if the frame length is fixed, the IPSM is only required to get the 1404 byte length streams after the synchronization. The IPSM has nothing to do with the data stream in a frame. That is the role of the PS.

3.2 Multiple Access

We use CSMA(Carrier Sense Multiple Access) method.

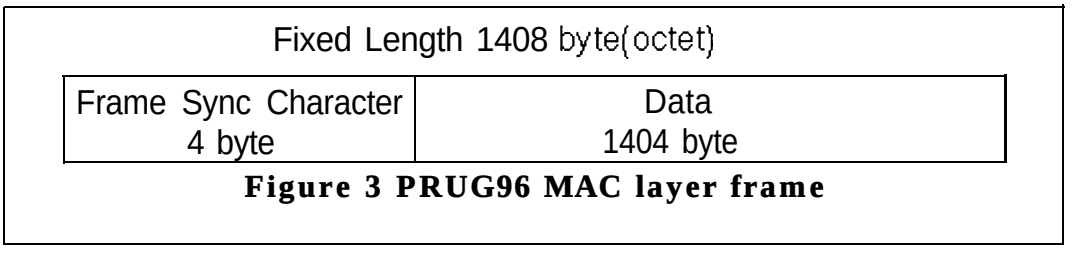

Figure 3 PRUG96 MAC layer frame

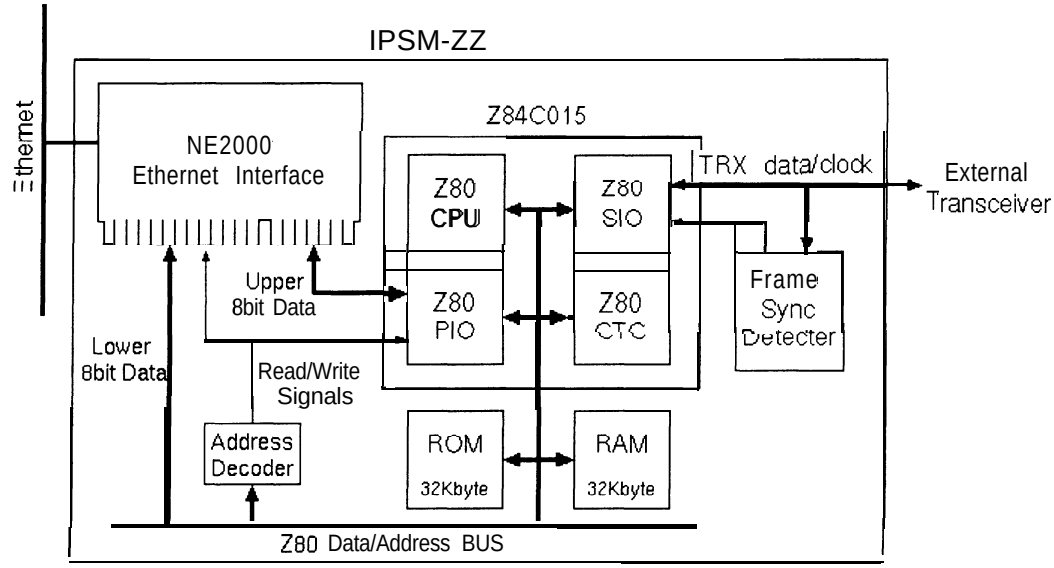

Figure 4 IPSM-ZZ block diagram

4. Hardware Formation

The IPSM-ZZ is one of implementation of the IPSM. The IPSM-ZZ consist of the parts, those are easy to buy. As shown in figure 4, TMPZ84C015BF-1 O(describe as Z84C015 below) and NE2000 compatible Ethernet card(describe as NE2000 below) are used as the CPU and the Network I/F, respectively. NE2000s are widely used in PC/AT compatibles and the price is reasonable. In addition, NE2000s are easy to install because a NE2000 use only IO ports to communicate with a CPU, where other network cards use both **IO** ports and shared memories.

Moreover, using table-comparing technique with ROM makes the frame synchronization circuit simple.

4.1 Ethernet interface

An NE2000, 16bit data bus, is connected to 8bit Z80_data_bus via data with converter. A PIO in Z84C0 15 set to mode3, is used for this converting function, and control signals ASTB and BSTB are connected to NE2000 IO READ and IO_WRITE inputs. (See figure 5)

The idea is that processing 8bit data from Z80 is latched on PIO port A and it will be passed to the NE2000 at the time following 8bit data is come out from ZSO to be written in NE2000 data port.

The lower 5bit of address signal on NE2000 is directly connected to CPU address bus. Upper 5bit **is** connected with PIO port B to search and find preset **NE2000** I/O **address.**

Using built-in peripherals reduces circuit **complex.**

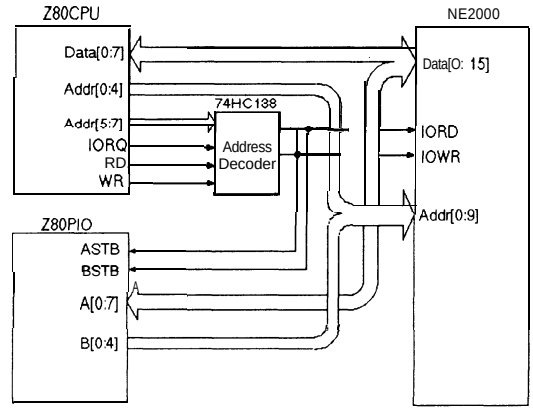

Figure 5 Bus width converter

4.2 Frame synchronization block

Figure 3 shows frame structure on the air interface. It consists of 32bit(4byte) frame synchronization header and 1404byte fixed length data. Data block includes FEC data, PS header block, and the upper layer data. The synchronization bit pattern is 0x08f1f059. This synchronization pattern will be used to find start of a frame.

In order to let FEC(Forward Error Collection) function perform correction on occasion of single bit error of twelve bit in data block, synchronization pattern matching block should work on exist of 3bit error in 32bit synchronization pattern. A volume of electronics circuit realizes these functions using logic gates will be larger.

To avoid this matter, EPROM based table searching circuit is selected. (See Figure 6) Received 32bit serial data is converted to parallel data using shift registers to address a cell data of the table in EPROMs. Data shows result of comparison the received 32bit data detected as intended sync pattern or not.

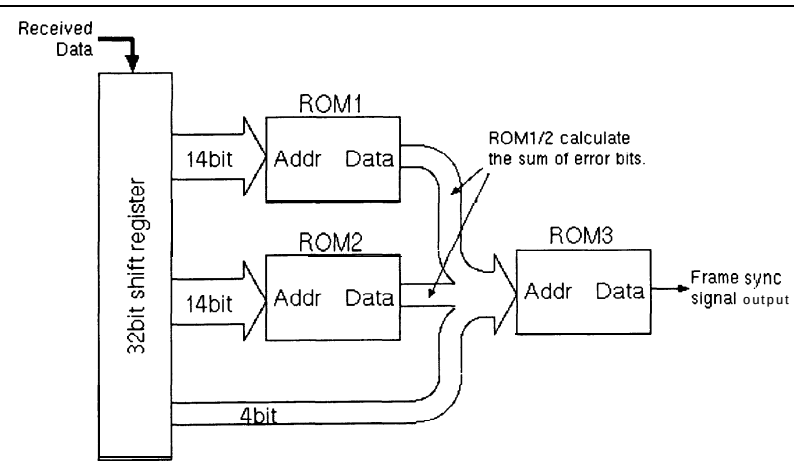

Figure 6 Frame synchronization block

5. Performance

The performances of the IPSM-ZZ are as follows.

(1) Maximum radio speed

Even though the Z80SIO has the ability of 2Mbps, the software can't catch up with it. So that the upper limit of serial transfer speed is about 800kbps. This is due to the defect of RX data just after synchronization. When transmitting, it seems there is no problem.

(2) Delay

It takes about 15ms, to emit a packet since the IPSM-ZZ received a packet from the Ethernet. This is the time to transfer data from the buffer of the NE2000 to the SRAM of the Z80. It's impossible to reduce the time at this moment.

Using 8bit mode of the NE2000s can reduce the delay to almost 2ms. (Implementation is under going, however, not all of NE2000s support 8bit mode.)

(3) Throughput

The transceiver used, when the measurement was done, was the SS data transceiver, produced by root inc., with 808kbps mode. The throughput of the whole PRUG96 system, including the PS and the IPSM-ZZ, using FTP command is as follows: 130kbps(maximum); 80kbps(average).

6. Further work

The current IPSM-ZZ can't make the best use of the transceiver because of insufficient CPU speed. The new model of IPSM is under developing now, aiming to add some functions of the PS into the box.