A LAN protocol for efficient file transfer on Linux based system

Ajit Ambekar[1], Pravin Dilpak[2], Pravinkumar Malviya[3] and Sameer Mahant[4]

V.J.T.I /Computer Technology,Mumbai, India.
[1]Email: ajitsa_bes@yahoo.com [2]Email: pravindilpak@rediff.com .
[3]Email: malviyapravin2010@gmail.com [4]Email: shm.mtech@gmail.com

***Abstract*— Data sharing has always been an cruicial task over a LAN. Considering the existing TCP/IP model only network and transport layers are of immense importance. As we know that, TCP defines the actual port numbers on which connections are made by the applications and of course, the local IP address of a machine, it stands as a candidate for data communication between applications over the internet and also in a LAN .On the other hand Internet protocol (IP) just takes care to deliver the packets to its destination. In this paper we are proposing a scheme to bypass the IP in a LAN environment**.

***Index Terms*— LAN, protocol, efficient, file transfer.**

I. INTRODUCTION

Gigabyte Ethernet has become a well known standard used for data transfer or communication. Lot of researches are being carried out to match the make as much use as of the capability of the Ethernet to transmit data. With the technical advance of link transmission and ASIC hardware processing the mature commercial Ethernet link interface speed has developed from 1G b/s to 10G b/s, laboratory 100G b/s Ethernet interface transmission technique has been thoroughly investigated [1].

In this paper we propose the design, of a new connection-oriented transport level protocol, which was designed specifically for local area networks (LANs). Our idea is to bypass the Network layer for a LAN (LAN is supposed to be within the limits such that no router comes in the scenario.) This will reduce the overhead of IP Encapsulation and decapsulation which will also result in less CPU utilization and faster transmission of data. The study is divided into two parts one of which is the Address resolution and the second is the design approach of the protocol.

II. BACKGROUND

As we know Internet is a public network which is a collection on private networks all over the world. These private networks belongs to enterprises, large institutions etc. So we can say that not only Internet but LAN is a widely used feature throughout the world for file sharing and data transfer. Hence we focus on LAN , so as to improve the performance on the basis that the data is to be transferred to a host within a LAN.

Address resolution

ARP protocol is required for actual communication between the network nodes at the physical layer. In prevailing ARP implementations, traditional methods of sending ARP query for each of required sending frame is rarely used; instead, they usually use ARP cache table (short for ARP table in the following) to store part of <MAC address, IP address> corresponding pairs so as to reduce overhead time for at least RTT each time. When failing to find an associated ARP entry, it would perform normal ARP query/reply procedure. However, the aim of dynamically updating and linear lookup of ARP table in high speed puts forward high requirements to system design [2].

The main issue Ethernet faces when forwarding packets is what to do when the entries are not found in the ARP cache table. Hence there should be a mechanism which takes care that the entries are present in the table and also the resolved Physical address has not been changed. Some implementations overcome this problem by adding an entry AGE in the ARP table which is basically a decrement counter. It defines the time limit for which the AGE entry in valid after which the entry must be marked as invalid and if required an lookup has to be done again just to confirm that there is no change in the Physical address.

In our scenario since we are working on a pure Local Area Network where no routing is required as the numbers of nodes in the network are going to be the same (depending upon the class of the IP address used). Hence as Routing is avoided the need to update the ARP table as fast as that of the earlier case become less, we have the option for static ARP table but because some physical damages may cause the machines to be replaced it isn't a good idea. So lets continue with dynamic ARP for our protocol.

The only extension to the ARP methodology carried out by the OS would be to increase the expiry time for every local entry in the ARP table; so as to avoid unnecessary congestion with ARP reply and request packets over the network.

III. PROTOCOL DESIGN

We will explain this part in two difference sections. The first section will contain a view about how the

protocol would actually work as compared to the TCP/IP stack and the second part would define the design specification that would be taken into consideration.

A. *Comparison with TCP/IP stack*

The diagram below show the traditional TCP/IP protocol model alongside which we have a basic diagram which depicts where our protocol would be placed.

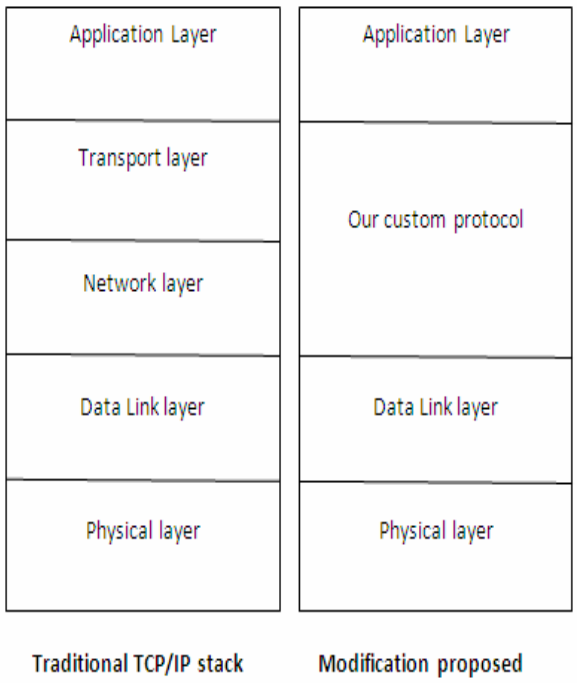

Figure 1 : Comparison with TCP/IP stack

In Fig. 1 we can see how the Transport layer and the Network layer should be mergred together. The important fact to keep in mind is, this advancement will work only at the OS level.

B. *Design Principles*

The primary design philosophy of our protocol is efficiency. Thus, it would not include any protocol features which are not strictly relevant to a LAN environment. Also, those functions which are only rarely needed, especially if they can be achieved by the application program, are excluded.

Another design approach is to exploit the properties of the underlying network and the application environment as much as possible. Some of the main features of our protocol are given below.

i. Unsupported Features

Our protocol would make no provision for routing, packet fragmentation/reassembly, packet self-destruction, or service options. These functions are typically irrelevant to a LAN environment and are thus unnecessary. Because of the much higher bandwidth, congestion is less of a problem in LAN's and not considered by us. There would be, however, a flow control mechanism which should indirectly help to alleviate any congestion problem[3].

ii. Error Control

The simplest error control scheme for our proposal would be retransmission of packets. The main purpose of error control in a LAN environment is to handle packet loss due to buffer overflows. With respect to the philosophy of simplicity, for our protocol, a go back n strategy seems to be a better option. Since the bit error rate in a LAN is extremely low, of the order of 1 in 10^{11} to 10^{12} [CLAR78 and COTTSO], more than likely errors are a result of buffer overflow rather than due to a lost or damaged packet. Also, since once a buffer overflows it is quite likely that more than one packet will have to be rejected and resent, the go back n strategy represents an uncomplicated scheme which does not result in an undue loss of throughput.

iii .Checksum

Check summing is specified as an option as the possibility exists that the network interface performs this function in hardware.

iv. Addressing

The addressing would be kept as it is, so as to enable the machine interact with the outside networks with the same logical address. Deploying a separate address may lead to an extra overhead.

IV. IMPLEMENTATION OVERVIEW

Raw Ethernet Frames are the one which are going to serve our purpose. An raw Ethernet frame can be injected into the network by bypassing the Linux kernel.

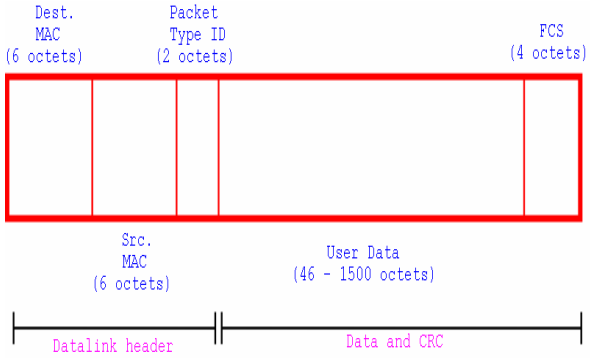

Figure 2 : An Ethernet Frame

The Fig 2 shows a Ethernet frame. As we know that the "user data" part consisting of 46-1500 Bytes, is

basically, IP over TCP/UDP. Hence the fact that a minimum of 40 Bytes, excluding the options, in the higher level protocols are consumed.

Our Idea is to bypass the upper protocols whenever data is to be shared in a LAN. We are mainly concentrating on the Physical address of a machine. The type filed in the above diagram is considered as a Length filed is the values lies between 0000-05DC. If we design a preamble for our frames to be recognized it would result in a very efficient procedure to transfer files in a LAN. For the following reasons,

1) The amount of data carried in the Raw Ethernet frames will be more as compared to the existing Frames which pass through the OS protocol stack.
2) The amount of overhead will be reduced on both the machines to encapsulate and then encapsulate the packets.
3) The processing rate will be increased as an result of less number of headers.

The new protocol would have an header which combines the features of TCP/IP (excluding the feature mentioned above) and would ultimately result in the reduction of the header size.

V. CONCLUSIONS

Hence we conclude that Raw Ethernet frames can be used for efficient file transfer in a LAN and we look forward to implement the same.

VI. REFERENCES

[1] J.S. Turner, Terabit Burst Switching, Journal of High Speed Networks, 1999.

[2] Yun Qin, A High Performance ARP Lookup System for Gigabit Ethernet.

[3] Samuel T. Chanson,The Design And Tuning Of A Transport Protocol For Local Area Networks