2S−5

An Acceleration of TCP/IP Protocols Processing

Bernady APDUHAN, Yoshimasa OHNISHI, Toshinori SUEYOSHI, Itsujiro ARITA

Kyushu Institute of Technology

1. Introduction

Studies have been conducted to investigate and analyze the standard communications protocols to identify its soft spots, the heavy processing loads and its causes. An investigation was conducted on a 10 Mbps Ethernet of UNIX workstations[1], to study thoroughly in what degree the protocol processing load affects the transfer execution speed, and to grasp the guiding principles necessary to speed up network performance. The experimental results reveals that in a local area network, host(s) having high CPU capability are necessary for high-speed communication, and that the transmit processing is a heavy load. Further, the transmission delay and window management were found to be the factors which greatly influence network performance.

This paper addresses the acceleration of multi-layered communications protocol processing based on standard protocols, i.e., the TCP/IP, on a local area network environment. This paper describes the environment testbed, the parallel processing and pipelining techniques in communications protocol processing.

TCP/IP usually refers to DARPA's Internet overall protocol suite, in this paper we refer to the TCP and IP protocols which are the subjects of our study.

2. Acceleration Strategies in TCP/IP Processing

This section describes the proposed acceleration strategies of TCP/IP protocols processing. We have conducted analysis and identified the stages wherein parallel processing and pipelining can be implemented to enhance the communications protocol processing. We divided the overall protocol processing into different modules and designated each module as an event.

2.1 TCP/IP Output Processing

The TCP/IP output modules' actions and the acceleration strategy are shown in Fig. 1 and 2, respectively.

In TCP, aside from the many events that may trigger the sending of data in a connection, packets must also be sent to communicate acknowledgments and window updates. Most commonly, the tcp_output module is invoked when a user has written new data in the socket, and transmission is realized whenever any of the pre-required conditions is satisfied; otherwise, TCP will start to execute the maximum delay time,

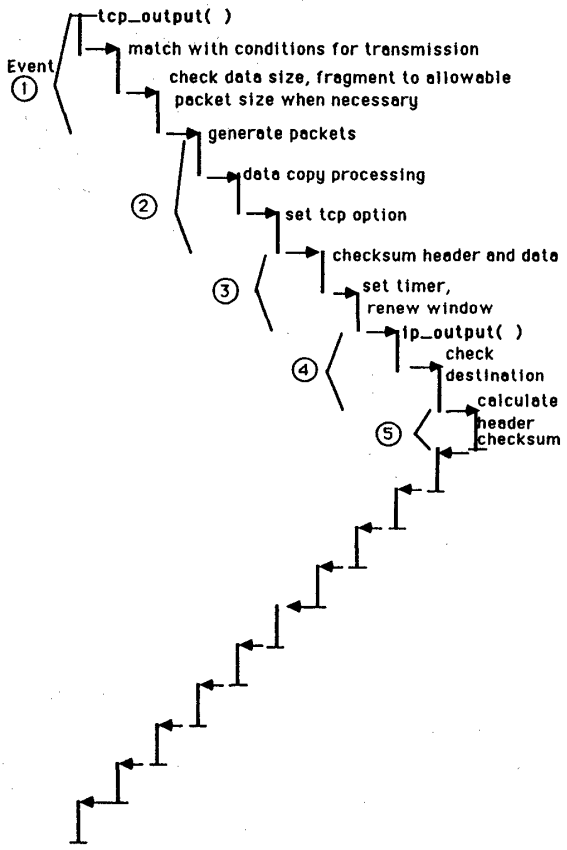

Fig 1. TCP/IP output modules actions.

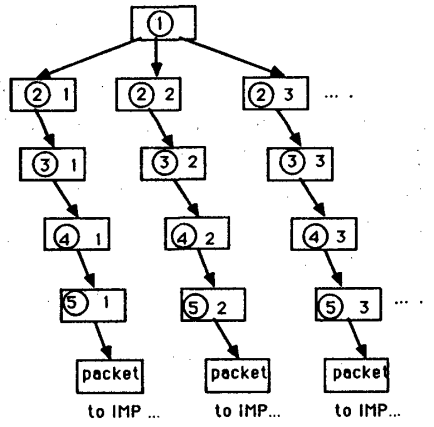

Fig 2. Acceleration strategy of TCP/IP output processing.

until timeout expires, and then commence the transmission processing.

In TCP/IP standard, when a packet has traversed the tcp_output processing module, it is handed down to the

An Acceleration of TCP/IP Protocols Processing

Bernady APDUHAN, Yoshimasa OHNISHI, Toshinori SUEYOSHI, Itsujiro ARITA

Kyushu Institute of Technology

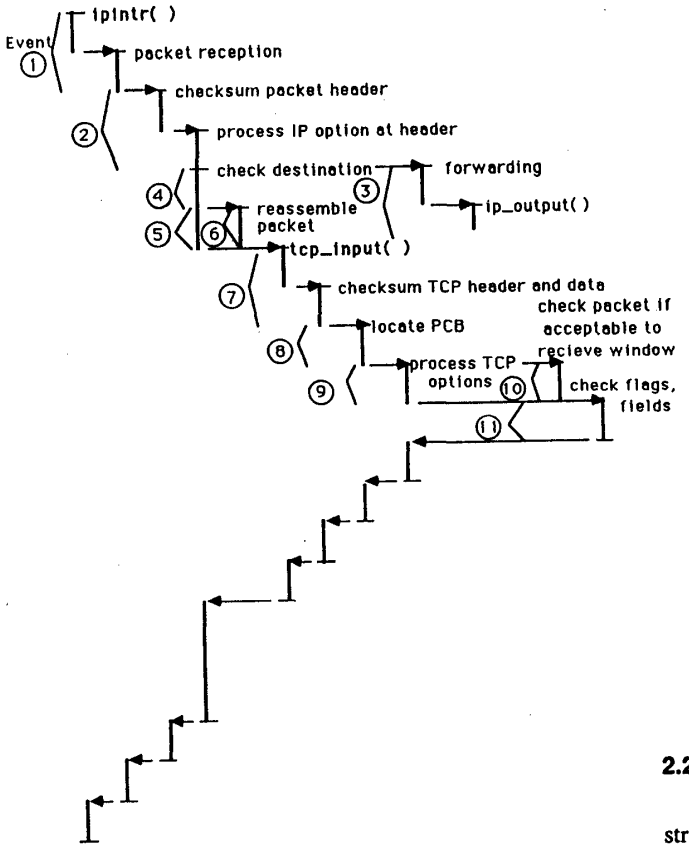

Fig 3. TCP/IP input modules actions.

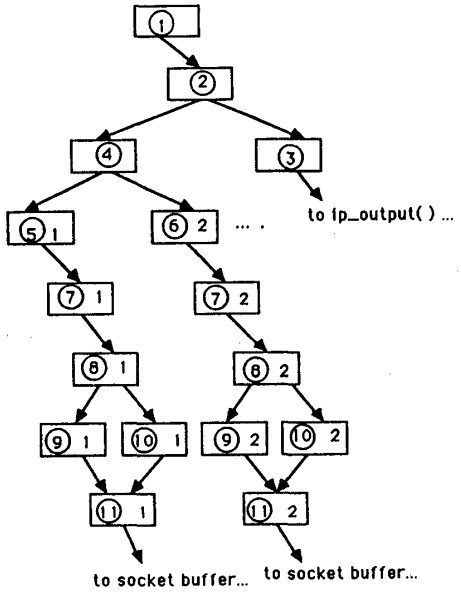

Fig 4. Acceleration strategy of TCP/IP input processing.

ip_output module where the packet's destination is determined. Further, the size of the packet is checked to determine whether it conforms to the allowable packet size at the network layer. If the packet exceeds the maximum allowable size, fragmentation of the packet into small fragments will then take place.

At the IP layer, the maximum allowable size is determined statically(or arbitrarily, depending on its implementation). Knowing this maximum size in the first place, we can then apply the fragmentation at the early stage of the tcp_output module, preferably right after the original packet(the packet which has just entered the tcp_output module) is ready for transmission processing. This packet will be fragmented into small packets and these will determine the number of subprocesses to be created and executed in parallel. The packets' sequence numbers or urgent packet numbers will be provided during the fragmentation process. Pipelining will be imposed on the small packet, i.e., with the other packets in the same flowline, as it is being process and will proceed until IP completes the actions and hands over the packets to the IMP(Interface Message Processor) at appropriate times.

In the process, whenever a failure is experienced by any of the subprocess or events, this will be reported directly to all peer subprocesses or events which will in turn terminates further execution. Such failure will also be reported to event ① , where it hands over a copy of the same data to event ② , and the same procedures are executed.

2.2 TCP/IP Input Processing

The TCP/IP input modules' actions and its acceleration strategy are shown in Fig 4 and 5, respectively. In IP input routine processing, we can let the reception of the next packet start just before the checksum calculation of the previous packet header ends. In other words, we proposed pipelining at these stages. Since ③ and ④ are independent events, they can be executed in parallel. Likewise, the same method can be applied to events ⑤ and ⑥ , ⑨ and ⑩ . Event ⑪ completes the actions of the tcp_input module and hands over the packet to the socket buffer at the application layer. However, since the acknowledgment of all outstanding data or a new window update may be required at this stage, a call at the tcp_output is invoked at the end of tcp_input processing.

3. Concluding Remarks

This study have provided facts and knowledge on the different factors affecting communications protocol processing and motivated further study on the proposed acceleration techniques to increase the effective transmission rate. Experiments on the proposed techniques of accelerating TCP/IP protocols processing are well underway at this time of writing.

References

[1] Apduhan, B., Sueyoshi, T., Yoshida, T. and Arita, I., "An Investigation on the Speed-up of Communications Protocol Processing", in *Proc. 5th International Joint Workshop on Computer Communications*, Korea, pp. 1-10 (July 1990).

[2] Clark, D.D., Jacobson, V., Romkey, J. and Salwen, H., "An Analysis of TCP Processing Overhead", *IEEE Communications Magazine*, pp. 23-29 (June 1989).