TCP SERIAL LINE EFFICIENCY MEASUREMENTS ON CARNET FESB NODE

JULIJE OŽEGOVIĆ, LADA PRAVDICA

FESB Split, tel. 021 563777, fax. 563877
Julije.Ozegovic@fesb.hr, Lada.Pravdica@fesb.hr

Abstract: Measurements in the Internet environment are often conducted on Ethernet LANs, placing interfaces in the promiscuous mode of operation. Various types of frames and protocol data units can be traced and recorded for later analysis. The TCP measurements of purposely initiated FTP sessions are performed to analyze the causes of poor serial line performance. In this work, the measurement environment and network configuration, as well as some basic results, are described.

KEYWORDS: Internet, Ethernet, measurement, TCP, serial line

INTRODUCTION

The serial access users of CARNet FESB node have suffered poor performance, even when using 14400 b/s modems (some students still use 2400 b/s modems). A series of measurements were performed to locate and eliminate the causes of that fact. The measurement possibilities were analyzed, the network configuration was chosen, and actual measurements were performed systematically.

In this work, the procedures of actual measurements are described. In Chapter 1. the measurements in the Internet environment are defined. Chapter 2. deals with various protocol data units (PDUs) and header formats expected on the Ethernet LAN. In Chapter 3. we describe actual network structure and measurement setup. Some basic results are presented in Chapter 4. Conclusions have been derived.

1. MEASUREMENTS IN THE INTERNET ENVIRONMENT

Internet as the network of networks, is structured by connecting various local area networks using direct links, or through backbone networks. In such environment, most LANs are Ethernet CSMA/CD installations. Furthermore, Ethernets are not used only for user equipment connections, but also as an efficient way to implement complex multi-gateway sites.

The main measurement opportunity in Ethernet based installations is the fact that all of the data traffic pass over the same cable, thus allowing simple connection of measurement equipment. In such environment, the access to all of the communication system architecture layers is possible.

On the physical level, various signal quality tests can be performed. In this work, we have been concerned with higher level measurements. This means that we have supposed correct operation of connected stations, and measurement attachment was just another computer connection.

The data link layer measurements on Ethernet LAN involve frame capturing and tracing. Although this activity can gain information on cable load and some statistics (e.g. number of collisions), it is the basic procedure to access protocol data units (PDUs) of higher level protocols. Ethernet interface hardware is turned to “promiscuous” mode of operation, meaning that destination addresses are ignored and all frames are reported to the capturing process.

Higher layer measurements are based on frame analysis and filtering. Selected information is then saved in raw (binary) or interpreted (textual) format for later analysis, or some statistical information is displayed in real time. Filtering is defined by logic equations, dealing with protocol types and host addresses. Only frames that satisfy the equations set, are recorded.

The saved traces are used for network and protocol diagnostics, including host malfunction detection, address problem resolution, bottleneck location and protocol evaluation. To achieve these goals, traces may include frame information (Ethernet addresses) and higher protocol headers, associated with the time of capture. Time can be expressed as relative (to the beginning of measurement) or absolute value with adequate resolution (tenths of milliseconds are satisfactory in most cases). For user security reasons, only packet headers are traced.

Finally, the capturing equipment should be able to process cable traffic, or otherwise frame losses could occur.

2. INTERNET PROTOCOL SUITE

The Ethernet measurement traces include various PDUs, ranging from data link frames to user level service packets. The Internet protocol suite is outlined in Figure 2.1.

APPLICATION LAYER	FTP, TELNET, WWW, SMTP, SNMP
TRANSPORT LAYER	TCP, UDP, GATEWAY PROTOCOLS
INTERNET LAYER	IP, ICMP / ARP, RARP
NETWORK ACCESS LAYER	ETHERNET II, 802.3, 802.2, 802 SNAP

Figure 2.1. - The Internet protocol suite

On the data link level, various Ethernet frame formats are present. On the network level, IP, ICMP, ARP and RARP packets are found. Transport level includes basic TCP and UDP packets, as well as various gateway protocol data. Finally, on the applications level, various services like FTP, TELNET and WEB generate their data. In this work, only data link, network and transport protocols are considered.

The set of four frame formats [2, 11] can be expected on Ethernet installations, depending on network protocols sharing the same media. Software solutions for protocol multiplexing are available from various manufacturers, including ODI (Open Data-Link Interface) specification from Novell and NDIS (Network Driver Interface Specification) from Microsoft/3Com. Frame header length varies from 4 to 6 words.

When standard computer connection is used for Ethernet measurements, the frame recognition capability is the same one included in the network interface driver. The driver is also responsible for the promiscuous mode operation of the hardware, as well as for the efficiency of frame capturing in the heavy load condition. Frame buffering is recommended.

In the Internet network layer, protocols that directly use Ethernet frames are found. The main protocol for user service is Internet Protocol (IP, [8]), while other protocols are used for network operations. Internet Control Messages Protocol (ICMP, [7]), despite using IP addresses, is considered network layer protocol, and is used for various traffic management functions. Address Resolution Protocol (ARP, [6]) and Reverse Address Resolution Protocol (RARP, [1]) are auxiliary protocols in Ethernet environment, utilised to assign IP to Ethernet addresses.

When frame traces are captured, the complete headers of PDUs should be recorded. Neglecting the options, the minimum length of IP header is 5 words (20 octets). Intelligent trace software can store option information adaptively.

The ICMP header is included inside the IP packet. However, because of its purpose this protocol is considered a network layer protocol. A total of 8 words (32 octets, IP header included) should be recorded.

ARP and RARP protocols, although different in purpose, have similar header format. The minimum of 7 words (28 octets) is to be recorded.

The Transmission Control Protocol (TCP, [9]) and User Datagram Protocol (UDP, [10]) are most frequently used protocols in transport layer. TCP provides connection oriented, delivery guaranteed end to end service, while UDP provides connectionless transport without delivery guarantees. The TCP header is included inside the IP packet. The record length should be 10 words (40 octets). UDP header is also included in the IP packet. Total record length is at least 16 words (64 octets).

3. EXPERIMENTAL MEASUREMENTS

Actual measurements were performed using the test configuration outlined in Figure 3.1.

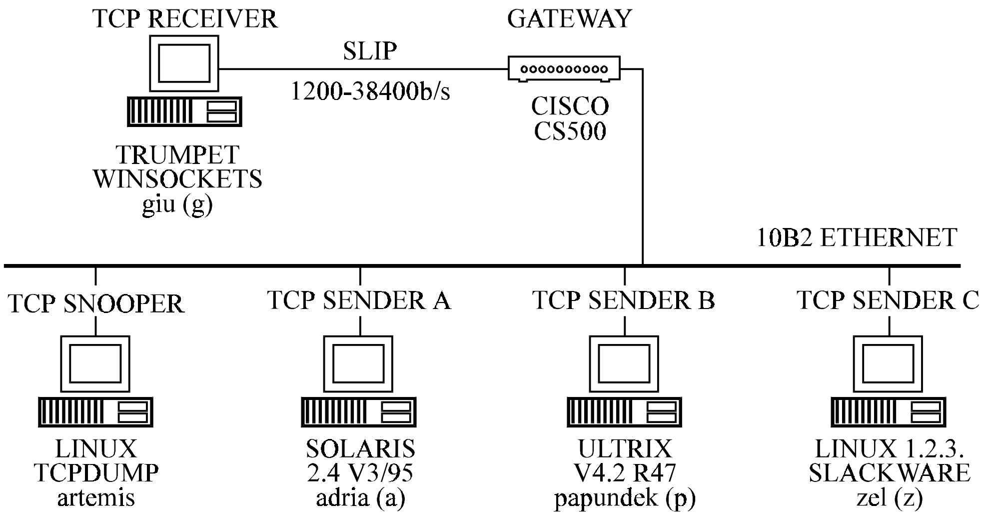

Figure 3.1. - The test network configuration

The main host server, also running the WWW proxy system, is SUN SPARCstation 20 workstation with Solaris 2.4 rev. 3/95 operating system software installed. On the same cable, several DECstation 5000 hosts with ULTRIX V4.2a rev.47 serve as domain name server and the general purpose workstations. Linux Slackware 1.2.3. systems on DEC Venturis PCs are connected as department servers, with WEB, FTP and other services installed. Finally, Cisco CS500 serial line servers are used for remote access. Various PCs are used as user workstations.

The measurement configuration involved one of every TCP brand available. SUN workstation was adria.fesb.hr, signed “a” in measurement results, ULTRIX system was papundek.fesb.hr, signed as “p”, and LINUX host was zel.fesb.hr, referenced as “z”. All of them were activated as TCP senders in remotely initiated FTP sessions.

On the other side of the measurement link, the PC workstation giu.fesb.hr was involved running Windows for Workgroups 3.11 with Trumpet Winsock V2.0 revB and WS-FTP ver 95/07/30 shareware FTP client. EWAN ver 1.052 telnet client was used to control the measurement server. WS-FTP with Trumpet Winsock were used in actual measurements as TCP receiver.

Cisco CS500 modem server with SuperLAT software installed was used for remote access. For speeds up to 14400 b/s, the public switched telephone network access using V series internal modem and V.42bis compression was involved. For higher transmission speeds, the optically isolated current loop interface was installed. The purpose of modem access was to simulate average remote user as much as possible, while the direct connection provided the possibility of higher speed measurements. However, a compressed file was transferred to obtain comparable results on both types of connection. On serial link, the Serial Line Internet Protocol (SLIP) with compressed headers was used.

The TCP measurement system was LINUX installation on artemis.fesb.hr host. The standard TCPDUMP [4] software utility was used to filter out and record the FTP session packets, including control and data TCP connections. The packets were recorded in standard (interpreted) format. We used the text parser to transform the measurement traces to spreadsheet readable format. The imported data were drawn as scattered charts. Figure 3.2. contains few sample traces from TCPDUMP output file.

```
18:48:33.749200 adria.fesb.hr.20 > giu.fesb.hr.1044: P 1:953(952) ack 1 win 24752
18:48:34.051609 giu.fesb.hr.1044 > adria.fesb.hr.20: . ack 953 win 13648
18:48:34.054404 adria.fesb.hr.20 > giu.fesb.hr.1044: . 953:1905(952) ack 1 win
24752
18:48:34.055517 adria.fesb.hr.20 > giu.fesb.hr.1044: P 1905:2857(952) ack 1 win
24752
18:48:34.313676 giu.fesb.hr.1044 > adria.fesb.hr.20: . ack 1905 win 13648
```

Figure 3.2. - TCPDUMP output file format

The measurements were performed systematically with various bit rates on the SLIP line and various initial segment sizes, as listed in Table 3.1.

b/s	MSS			
	212	512	966	1460
1200	xx0102yy	xx0105yy	xx0110yy	xx0115yy
2400	xx0202yy	xx0205yy	xx0210yy	xx0215yy
9600	xx0902yy	xx0905yy	xx0910yy	xx0915yy
14400	xx1402yy	xx1405yy	xx1410yy	xx1415yy
19200	xx1902yy	xx1905yy	xx1910yy	xx1915yy
28800	xx2802yy	xx2805yy	xx2810yy	xx2815yy
38400	xx3802yy	xx3805yy	xx3810yy	xx3815yy

Table 3.1. - The measurement setup parameters

The “xx” entry in Table 3.1. denotes host involved, e.g. “ag” means adria to giu, “pg” means papundek to giu, and “zg” means zel to giu. The “yy” entry is serial number of measurement for given setup. In all experiments, the TCP receiving window was set to 10*MSS.

4. MEASUREMENT RESULTS

The network configuration described was set to provide the background for the TCP performance measurements in the bottleneck link environment. FTP access from giu.fesb.hr was used to initiate download connections with hosts listed, thus transferring data from hosts to workstation. In effect, packets were sent through Ethernet LAN to congested gateway (Cisco CS500), and then by slow serial line to the user's workstation.

The results of measurements have shown significant performance oscillations when comparing different TCP implementations. Characteristic traces from each of sending hosts are plotted on Figures 4.1, 4.2. and 4.3.

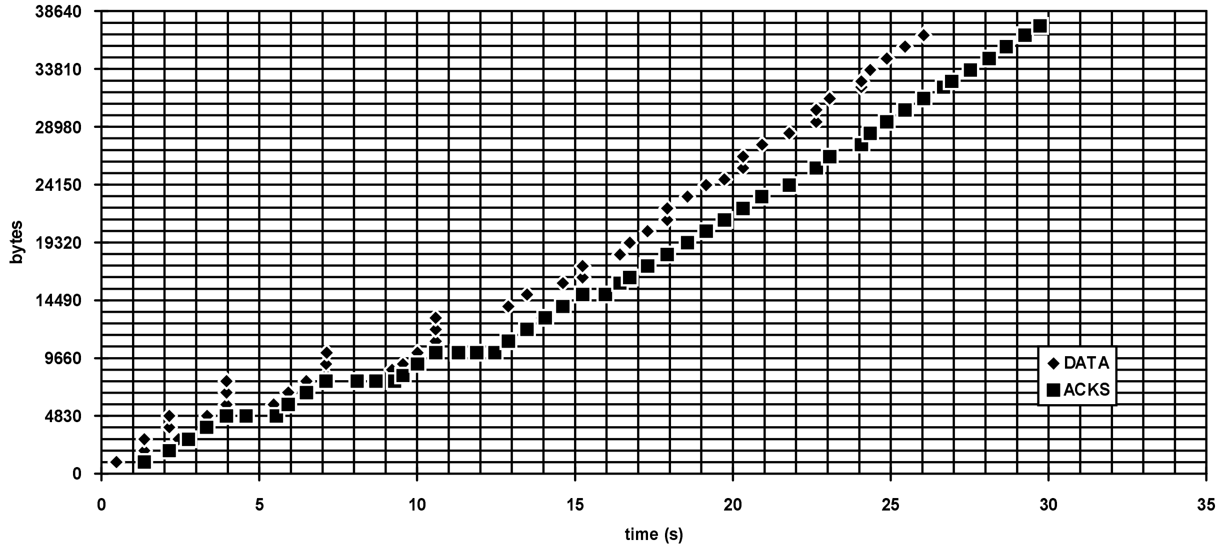

Figure 4.1. - The characteristic trace of adria.fesb.hr (AG141002)

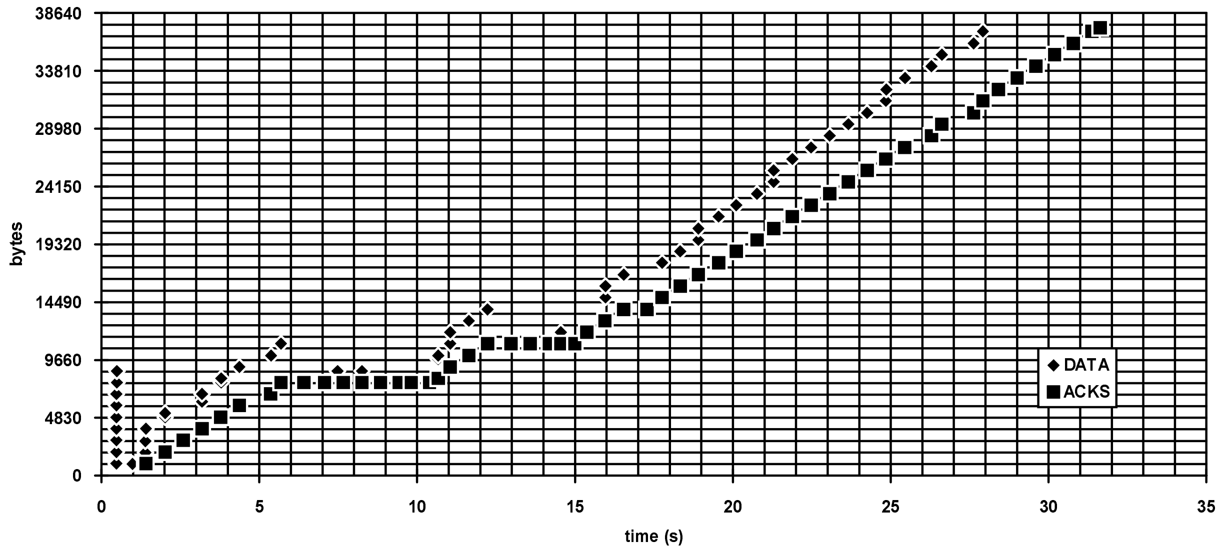

Figure 4.2. - The characteristic trace of papundek.fesb.hr (PG141002)

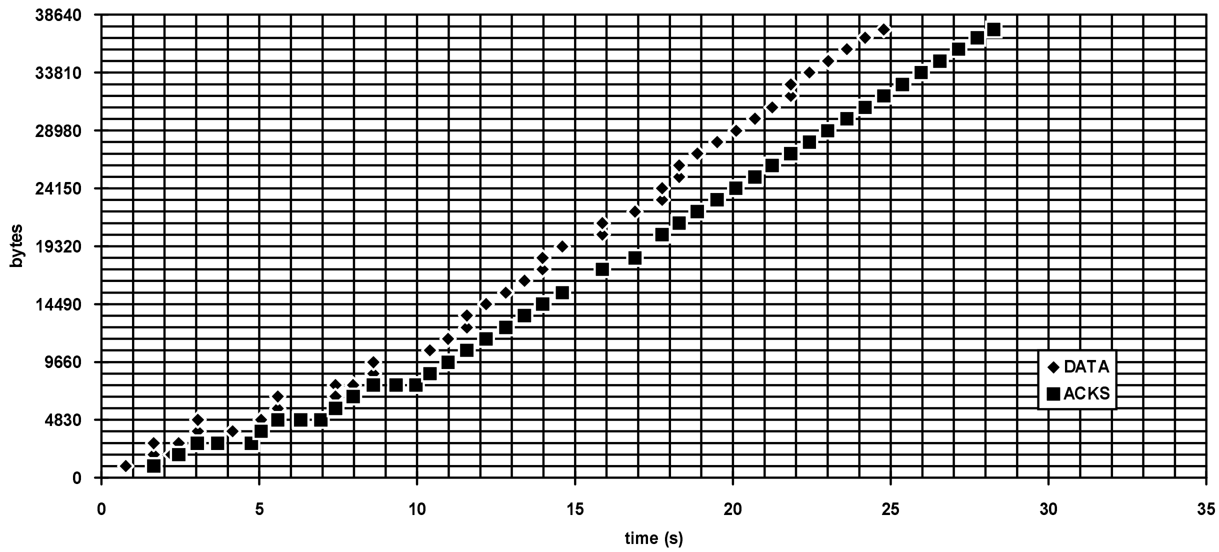

Figure 4.3. - The characteristic trace of zel.fesb.hr (ZG141001)

The adria.fesb.hr trace shows repeated slow start procedures, with relatively late stabilization. Depending on the falling round trip time (RTT), other traces range from continuous oscillations to smooth operation. The papundek.fesb.hr trace is characterized by total absence of slow start mechanism, thus sometimes causing gateway buffer overflow. Again, low RTT setups lead to better performance. Finally, zel.fesb.hr trace shows the behaviour of full standard TCP implementation. Connection stabilizes fast, with better results on lower RTTs.

In the network configuration of Figure 3.1, RTT depends heavily on slow serial line transmission time. The minimal RTT, RTT_0, can be estimated with acceptable precision by dividing total bit count per TCP segment with line bit rate. Total bit count includes user data, TCP and IP headers and SLIP header in data transfer direction, and all those headers only in reverse (acknowledging) direction. Headers are compressed by Van Jacobson compression algorithm [3]. The synchronous transmission of V.42bis modem excludes start and stop bits from the data stream, further compressing the header data in some extent (user data have been already compressed). For simplicity, we assume that header overhead for data and acknowledgment packets is roughly 16 octets.

Table 4.1. shows the measured channel efficiency, calculated for first 5, 10, 20 and 37 thousand octets. The first column is measurement setup as shown on Table 3.1. and the next leftmost column is RTT_0 for the setup as described above.

The calculated efficiencies are plotted as function of RTT_0 for various data lengths, Figure 4.4, 4.5. and 4.6.

	RTT_0	ADRIA					PAPUNDEK			ZEL			
		η05	η10	η20	η37	η05	η10	η20	η37	η05	η10	η20	η37
0102	1.520	60.3	70.7	77.8	81.6	48.6	62.0	71.5	76.9	67.6	79.1	82.2	83.5
0105	3.520	26.6	27.2	36.3	47.1	44.2	50.3	64.3	72.7	52.7	62.2	69.7	77.3
0110	6.547	33.0	22.8	22.1	26.6	84.4	48.5	47.2	59.4	53.6	67.2	77.4	81.9
0115	9.840	25.2	25.0	26.3	24.7	88.9	92.0	38.4	51.7	53.6	71.5	77.4	79.5
0202	0.760	72.8	79.6	82.9	84.8	47.1	57.1	72.9	76.4	66.1	77.7	71.4	83.1
0205	1.760	38.3	37.7	45.5	58.6	47.5	51.6	65.0	74.0	52.2	65.5	75.0	80.1
0210	3.273	26.7	22.6	18.1	26.0	85.5	39.0	54.9	66.5	59.7	71.8	80.9	83.1
0215	4.920	16.9	22.5	20.9	21.4	39.2	44.7	66.3	61.2	55.0	69.6	75.0	81.2
0902	0.190	77.4	84.3	82.6	85.7	78.2	83.9	82.0	84.9	73.1	81.7	86.4	84.0
0905	0.440	37.4	42.0	55.8	65.5	49.0	44.1	57.5	68.4	51.4	66.1	75.4	80.3
0910	0.818	25.7	23.0	29.4	37.5	83.3	50.5	53.8	67.0	51.8	55.4	67.9	75.7
0915	1.230	25.8	29.6	24.1	28.8	92.6	92.6	48.3	59.3	55.6	55.9	70.6	75.9
1402	0.127	74.5	74.8	82.1	85.5	71.8	85.9	79.0	82.7	75.9	79.8	81.9	84.0
1405	0.293	46.1	60.3	72.1	78.4	43.5	57.4	69.5	76.5	49.3	63.6	73.9	79.5
1410	0.546	54.1	45.1	60.6	70.3	78.7	50.4	54.7	66.0	44.6	53.3	66.4	74.7
1415	0.820	36.2	42.2	41.4	49.4	91.1	95.0	90.0	61.6	46.1	42.0	52.8	64.8
1902	0.095	71.1	72.7	73.6	75.1	53.0	62.8	68.1	71.6	73.6	61.9	74.9	73.3
1905	0.220	39.8	53.8	64.4	66.3	49.4	69.2	72.3	75.1	53.8	63.9	69.9	72.7
1910	0.409	32.6	31.6	40.3	51.0	74.9	42.6	48.2	57.6	42.5	55.6	65.6	69.8
1915	0.615	35.6	41.0	36.0	46.4	83.3	80.1	52.4	51.2	50.1	50.4	58.9	66.9
2802	0.063	66.8	72.5	74.3	75.1	65.8	71.0	73.5	74.7	72.7	74.9	75.5	75.8
2805	0.147	76.3	65.1	70.6	73.6	79.4	80.5	72.6	75.9	76.7	78.0	77.6	77.5
2810	0.273	38.8	37.6	48.0	61.3	79.4	79.4	79.9	78.8	43.8	55.9	66.4	70.8
2815	0.410	74.3	62.1	49.7	57.5	86.8	81.7	82.3	78.8	46.9	50.9	61.3	67.8
3802	0.048	68.5	72.6	74.8	73.6	74.9	74.7	75.9	74.9	67.6	69.2	75.6	74.8
3805	0.110	71.8	74.9	74.4	74.8	79.5	76.0	77.3	76.4	75.5	80.7	76.0	76.5
3810	0.205	55.4	51.1	49.7	58.5	80.1	74.9	77.4	77.2	44.0	50.0	60.7	56.8
3815	0.308	89.0	53.8	54.3	62.3	75.5	59.9	66.3	62.8	46.7	42.7	58.9	65.4

Table 4.1. - The measured channel efficiency

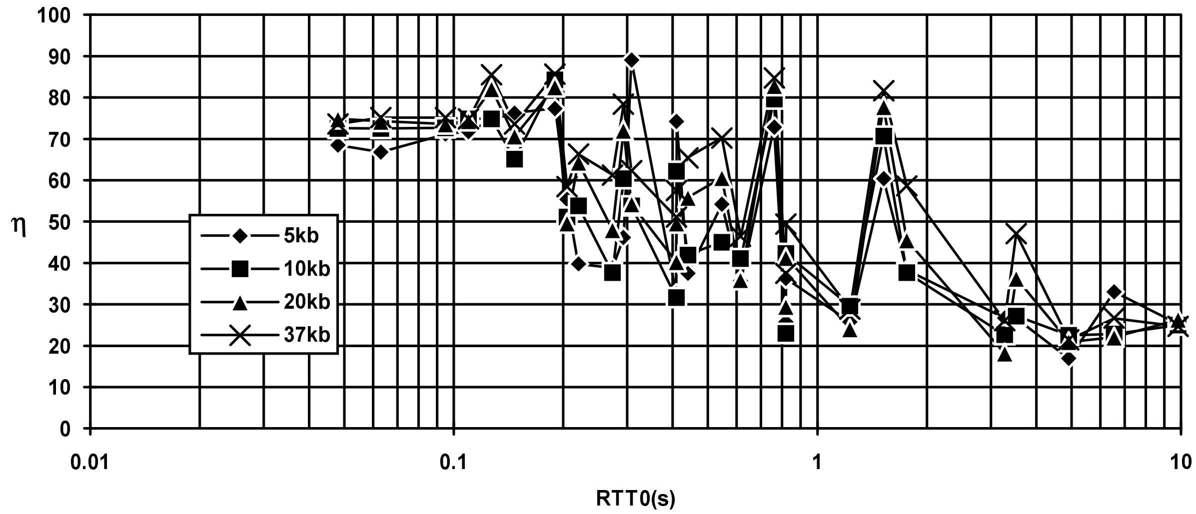

Figure 4.4. - The efficiency for adria.fesb.hr

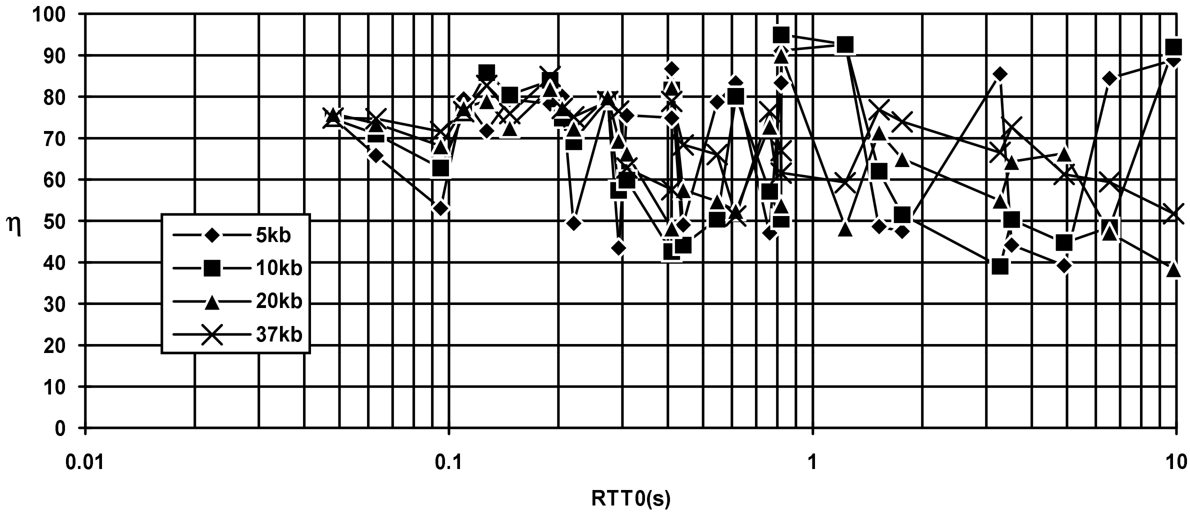

Figure 4.5. - The efficiency for papundek.fesb.hr

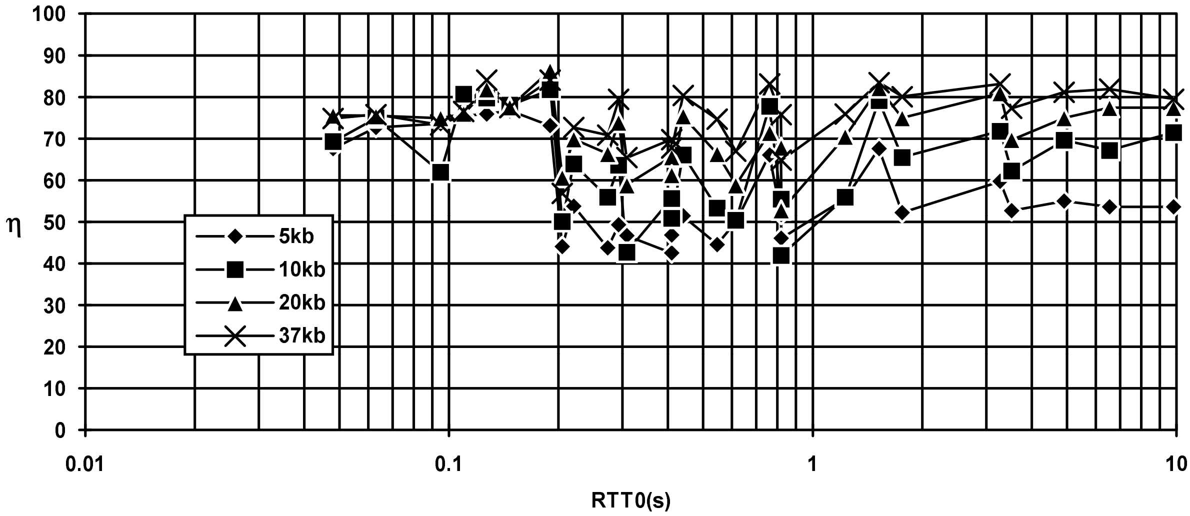

Figure 4.6. - The efficiency for zel.fesb.hr

In the RTT_0 range of up to 0.2s, all TCP implementations show similar and stable satisfactory performance between 60% and 80%. In the RTT_0 range of 0.2 to 1s all senders behave unstably and sometimes results are surprisingly good or bad. Finally, in the RTT_0 range above 1s, we have low performance of adria.fesb.hr, unstable but not so bad performance of papundek.fesb.hr and expectedly stable performance of zel.fesb.hr. In this range, 1200 and 2400 b/s measurements are located, but also some of 9600 and 14400 b/s traces are very close.

The authors tried to establish the correlation between measured results and initial retransmission time-out of TCP senders. Although these times were not the same, namely 0.5s for adria, 1.5s for papundek and 3.0s for zel, it seems that TCP behaviour depends mainly on other sender related algorithms.

5. CONCLUSIONS

The performance measurements of TCP over serial line connections were performed systematically. The measurement network involved three different implementations of TCP. The trace analysis has shown the differences in TCP implementations, e.g. absence of Karn algorithm on Sun adria.fesb.hr and lack of slow start algorithm on DEC papundek.fesb.hr. The Linux zel.fesb.hr seems to follow the actual standards in full extent.

Recorded traces were used to calculate serial line efficiency for 5k, 10k, 20k and 37k file transfers. The results are presented in table and graph forms. Various rate and MSS setups are bound to single measure, the minimal round trip time RTT_0.

All TCP senders are equally efficient in the low RTT_0 range. In the middle range of RTT_0, all senders behave unstably. In the range of high RTT_0 Sun adria.fesb.hr shows worse performance due to excessive retransmissions. Linux zel.fesb.hr shows stable satisfactory results, achieving 80% for file sizes of more than 20k. DEC papundek.fesb.hr shows unstable results indicating that aggressive startup can be of benefit as well as damaging for the user.

The further analysis of recorded traces should include TCP algorithm justification, with conclusions towards parameter trimming. It can be found in [5].

Literature:

[1] Finlayson, R., M.T., M.J., T.M. “A Reverse Address Resolution Protocol”, RFC 903, IETF, June 1984, ftp://ds.internic.net/rfc/rfc903.txt

[2] Hornig, C. “A Standard for the Transmission of IP Datagrams over Ethernet Networks”, RFC 894, IETF, April 1984, ftp://ds.internic.net/rfc/rfc894.txt

[3] Jacobson, V. “Compressing TCP/IP Headers for Low-Speed Serial Links”, RFC 1144, IETF, February 1990, ftp://ds.internic.net/rfc/rfc1144.txt

[4] Jacobson, V., C.L., S.McC. “TCPDUMP manual page”, Linux 1.2.3. manual page, 1993.

[5] Ozegovic, J., M.K. “Functionality of TCP Implementations Installed on the CARNet FESB Node”, accepted for SOFTCOM’96, Split 1996.

[6] Plummer, D.C. “An Ethernet Address Resolution Protocol”, RFC 826, IETF, November 1982, ftp://ds.internic.net/rfc/rfc826.txt

[7] Postel J. “Internet Control Message Protocol”, RFC 792, IETF, September 1981, ftp://ds.internic.net/rfc/rfc792.txt

[8] Postel J. “Internet Protocol”, RFC 791, IETF, September 1981, ftp://ds.internic.net/rfc/rfc791.txt

[9] Postel J. “Transmission Control Protocol”, RFC 793, IETF, September 1981, ftp://ds.internic.net/rfc/rfc793.txt

[10] Postel J. “User Datagram Protocol”, RFC 768, IETF, August 1980, ftp://ds.internic.net/rfc/rfc768.txt

[11] Postel, J., J.R. “A Standard for the Transmission of IP Datagrams over IEEE 802 Networks”, RFC 1042, IETF, February 1988, ftp://ds.internic.net/rfc/rfc1042.txt.