BULLETIN OF THE POLISH ACADEMY OF SCIENCES
TECHNICAL SCIENCES, Vol. 61, No. 3, 2013
DOI: 10.2478/bpasts-2013-0068

Effective throughput of AX.25 protocol

B. ZIELIŃSKI*

Institute of Computer Science, Silesian University of Technology, 16 Akademicka St., 44-100 Gliwice, Poland

Abstract. AX.25 protocol belongs to the HDLC protocols family and is used, among others, as a data link layer in an amateur Packet Radio network. The paper presents a simple analytical model developed on the basis of protocol frame exchange rules. Then, we compare analytical results to the experimental ones achieved using various different types of TNC controllers that act as network interfaces for Packet Radio network. In the experimental part we focus on both hardware and software properties of TNC that have influence on the effective throughput of the network.

Key words: wireless LANs, AX.25 protocol, TNC controller, effective throughput, throughput upper limit.

1. Introduction

AX.25 protocol [1] is a member of HDLC [2] protocols family. It was developed on the basis of LAP-B [3] in early 1980's [4] in order to provide communication between multiple stations over a broadcast radio channel. Although the protocol has been mainly used in the amateur Packet Radio network, it can be used in other applications as well. Examples are telemetry and remote control networks with stations distributed over a relatively large areas. The protocol can be software-implemented in any microprocessor, however, some microcontrollers are especially attractive for this task as they are equipped with HDLC controller which supports some low-level operations like bit sequencing and bit stuffing. Thus, AX.25 may be successfully used in embedded applications. It is acknowledged by the existence of TNC (*Terminal Node Controller*).

TNC is an autonomous, microprocessor-based device, used as a network interface in the amateur Packet Radio network. Therefore, TNC may be viewed as an interesting implementation of the protocol. TNC controllers, however, are not all the same; in other words, they differ in terms of both hardware and software. Thus, it is interesting how a given protocol implementation influences on its performance.

Moreover, TNC can also be an example of a solution that integrates wired and wireless network segments. Using TNC controllers for experiments seems especially attractive because it allows observe behaviour of data link layer protocol without regard to higher and lower level network layers, unlike in all modern highly integrated network interfaces. Thus, experimental results can be easily compared to those obtained in theoretical analysis.

Despite the long existence of AX.25 protocol, there are very few widely available papers covering this subject. Probably the most commonly known are [4] and [5]. The first one discusses a brief history of hardware, software, and protocol development activities together with a description of amateur packet radio operations, while the latter concentrates on the satellite-based packet radio activities. More information can be found via the TAPR (*Tucson Amateur Packet Radio*) web site, however, there are hardly any papers regarding protocol performance.

Recently, Ronan et al [6] have developed a simple model which allows to estimate AX.25 protocol performance, e.g., frame transmission times or effective throughput, under perfect conditions. In this paper, we present a similar analytical model which slightly differs from the one presented in [6] and seems more accurate. The analysis itself is based on the approach presented in [7] and [8] with the difference that in our considerations the transmission rate is of a real value, not infinitely high. The similar approach has also been shown in [9].

The rest of the paper is organised as follows. First, we discuss the operation of AX.25 protocol over half-duplex and full-duplex links. Then, we introduce an analytic model that allows estimate frame transmission times and effective throughput under perfect conditions. Next, we present TNC controllers and their influence on network achievements. Then we compare theoretical results to the experimental measurements done in various transmission hardware and software configurations. Finally, we discuss the factors that have influence on effective throughput of AX.25 protocol.

2. Analysis of AX.25 protocol

AX.25 protocol may operate over half-duplex or full-duplex links. In both cases, data transmission performs in repetitive frame exchanges. Further in the paper, we will refer such a frame exchange to as transmission cycle.

In the case of half-duplex link, the cycle contains up to k I (*Information*) frames which are commonly acknowledged upon proper reception by an RR (*Receiver Ready*) frame [1]. Prior to sending the first I frame of the cycle, the sender checks channel status using p-persistent CSMA (*Carrier Sense Multiple Access*) mechanism, and – if the link is sensed free – it turns on the transmitter and waits for T_{103} before sending

*e-mail: Bartlomiej.Zielinski@polsl.pl

data. After reception of the latest I frame, the recipient may wait for T_2 to make sure no more I frames are being sent. The RR frame is also preceded by the T_{103}. Transmission cycle of AX.25 protocol operating over a half-duplex link radio is presented in Fig. 1.

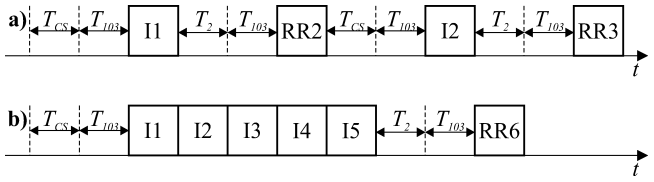

Fig. 1. Transmission cycle of AX.25 protocol using half-duplex link for a) $k = 1$ and b) $k = 5$

In the case of full-duplex link, each I frame is acknowledged separately, however, the RR frames are transmitted over a separate frequency channel [1]. Thus, they do not alter transmission time. Therefore, k parameter is not important, because all the I frames are sent, if possible, consecutively in a single long sequence. A fragment of such frame exchange is presented in Fig. 2.

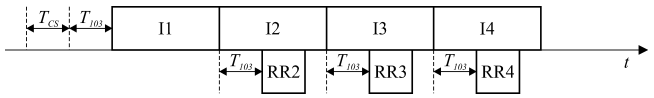

Fig. 2. Transmission cycle of AX.25 protocol using full-duplex link

2.1. Analytical model of AX.25 protocol. Consider a network that consists of two stations, communicating over a perfect channel. In such a case, there are no collisions or transmission errors, also, the sender successfully wins CSMA contention in the first attempt. Bearing in mind the explanations given in [8],

$$T_{CS} = \frac{256 T_{102}}{2(p+1)}, \tag{1}$$

where T_{102} – CSMA slot time [s], p – persistence parameter (transmission probability in a given slot equals to $(p+1)/256$).

Assuming constant I frame length, duration of the transmission cycle for the half-duplex link (see Fig. 1) may be expressed as follows:

$$T_p = T_{CS} + 2T_{103} + kT_I + T_2 + T_{RR}, \tag{2}$$

where T_I and T_{RR} – transmission times of I and RR frames, respectively [s], while k – window size (or maximum number of commonly acknowledged frames). Bearing in mind their formats [1],

$$T_I = \frac{63}{62}\frac{160 + 8N_1}{R_{wl}}, \tag{3}$$

and

$$T_{RR} = \frac{63}{62}\frac{160}{R_{wl}}, \tag{4}$$

where R_{wl} – wireless link transmission rate [bps], N_1 – maximum capacity of the data field in an I frame [bytes]. Control (e.g., RR) frame length equals to 20 bytes, while I frame is N_1 bytes longer, hence the value of 160 [bits] in (3) and (4). The factor 63/62 results from bit stuffing [10] required for protocol transparency.

The model presented above differs from Ronan's one [6] in two points. First, Ronan takes into account protocol transparency as a factor of 8.004 applied to only the data field of the I frame. In our model, the factor of 63/62 applies to every frame entirely, including address and control fields, as suggested by [10]. The second difference is that Ronan's model accounts the T_{txtail} parameter which is similar to T_{103} except it occurs after the frame. We decided not to consider this parameter, because it seems not defined by the AX.25 protocol specification and we found its default value equal to 0 in all AX.25 implementations we knew. Nevertheless, inclusion of T_{txtail} into our model, if necessary, is possible and straightforward.

Some comment on T_2 delay is necessary. Depending on the AX.25 protocol implementation details, the sender may request immediate acknowledge by setting of P/F bit in the control field of the latest frame in a window. In such a case, the receiver knows there will be no more frames and responds with RR immediately, not wasting T_2 for unnecessary waiting for more frames. AX.25 protocol specification defines a response to the I frame with P/F bit set, however, it does not require that the latest I frame in a window is marked. Thus, protocol behaviour remains implementation-specific.

In the case of full-duplex link, RR frame is transmitted during the transmission of the I frame that follows the I frame being acknowledged. Thus, we may assume that $T_p = T_I$, especially if the number of transmission cycles is sufficiently large.

2.2. TNC controllers. When TNC controllers are present in a network, the transmission proceeds in three stages, as explained in Fig. 3. The delays between begins and ends of the stages result from data buffering in TNC memory and processing of AX.25 protocol frames.

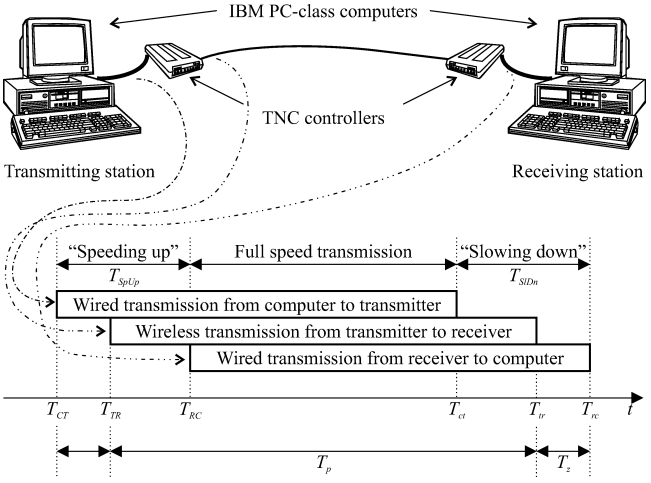

Fig. 3. Transmission stages when using TNC controllers

TNC controller collects the characters received from the serial link. Once there are at least N_1 characters collected, it forms an I frame of AX.25 protocol and sends it over the

wireless link. During this transmission, it continues collecting characters until the buffer is full, and it can prepare consecutive I frames and place them in a queue to be transmitted later. Obviously, TNC may send up to k frames in a transmission cycle. At the receiver side, TNC acknowledges received frames and sends the content of their data field to the serial link. Hence, the presence of TNC controllers causes additional delays at the beginning and the end of transmission.

Assuming that the serial link effective throughput is higher than that of the wireless link, TNC controller extends transmission process by the transmission time of $2N_1$ characters over the serial link:

$$T_{TNC} = 2\frac{10N_1}{R_w}, \tag{5}$$

where R_w – serial link transmission rate [bps].

When the middle transmission stage lasts long enough, i.e., when transmitted information is large, or when wireless link effective throughput is much lower than that of the serial port, T_{TNC} is negligible.

Having calculated T_{TNC} and T_p, we can calculate theoretical effective throughput, dividing capacity of data transmitted during the transmission cycle by its duration.

3. Experimental results

Comparison of analytical and experimental results is one of possible ways to verify the model accuracy. Thus, we verified the model presented in the previous section in an experimental Packet Radio network with several types of TNC controllers. The set of controllers is limited to the models available on the market. Nevertheless, it covers many solutions that differ from each other in hardware and software.

On the other hand, theoretical results achieved using presented formulas – assuming they are correct – represent possible achievements of a "perfect" TNC controller. Thus, we can use them as a reference point in an analysis of experimental results.

It must be noted, however, that the theoretical analysis presented in the previous section assumes perfect transmission conditions. Thus, in order to make the analytical and experimental results comparable, we had to provide transmission conditions in the network as perfect as possible. Therefore, we decided to make a wired connection between TNC controllers instead of a radio link. It was necessary to avoid possible interferences resulting in transmission errors and retransmissions that would visibly affect the measurement results. This approach allows find out how protocol implementation details influence on network achievements. Besides, using no radio, we could use wider range of AX.25 protocol parameters, e.g., R_{wl} or T_{103}.

As aforementioned, TNC is a microprocessor-based device. As such, it may be built using various types of microprocessors, of various processing power. On the other hand, in TNC controller, AX.25 protocol is usually software-implemented, with the exception of lowest-level protocol functions – e.g., frame delimiters and bit stuffing – that are typically supported by a HDLC controller. Nevertheless, details of software protocol implementation may also influence on network performance. Therefore, we decided to check both hardware and software influence on AX.25 protocol effective throughput.

3.1. Hardware influence on effective throughput. We tested AX.25 protocol effective throughput in an experimental network that consisted of two PC-class computers and two TNC controllers. Receiving and transmitting TNC were identical. Network configuration was identical to shown in Fig. 3. Some construction parameters of TNC controllers used for tests are collected in Tables 1 and 2.

Table 1
Microprocessors in TNC controllers

Controller	Vendor	Processor	f_{clk} [MHz]
TNC2	–	Z80	2.4576
TNC2D	Muel	Z80	4.9152
TNC2H	Symek	Z80	9.8304
Spirit-2 Std.	Paccomm	Z80	9.8304
Spirit-2 H.S.	Paccomm	Z80	19.6608
KPC-9612+	Kantronics	68HC11	16.0000
PK-96	Timevawe	Z180	12.2880
KAM-XL	Kantronics	68HC902	9.8304
DSP-232	Timevawe	68340	3.6864
PTC-11	SCS	68360	25.0000
TNC3S	Symek	68302	14.7456
TNC31S	Symek	68302	14.7456
TNC4e	HBTron	68EN302	19.6608
TNC7multi	NtG	LPC2106	58.9824
DLC7	NtG	S3C4530	49.1520

Table 2
Memory and transmission rates in TNC controllers

Controller	ROM [KB]	RAM [KB]	R_w [kbps]	R_{wl} [kbps]
TNC2	32	16-32	9.6	1.2
TNC2D	2×32	32	19.2	1.2
TNC2H	2×32	32	38.4	9.6
Spirit-2 Std.	2×32	32	57.6	57.6
Spirit-2 H.S.	2×32	32	57.6	57.6
KPC-9612+	128	128-512	38.4	38.4
PK-96	64	128	38.4	38.4
KAM-XL	512	512	38.4	9.6
DSP-232	128	256	19.2	9.6
PTC-11	256-512	512-2048	115.2	19.2
TNC3S	256-1024	64-2048	115.2	614.4
TNC31S	128-512	128-512	115.2	614.4
TNC4e	1024	4096	115.2	1228.8
TNC7multi	128	64	115.2	115.2
DLC7	4096	32768	115.2	1536.0

Measured throughput. Measurements results of effective transmission speed for few selected TNC controllers, operating at various window sizes (k) and maximum I-frame data

field capacity ($N_1 = 256$ bytes) are shown in Fig. 4. Transmission rates were set up to: $R_w = 19.2$ kbps, $R_{wl} = 1.2$ kbps. For comparison purposes, the graph contains also the curves showing theoretical achievements of AX.25 protocol obtained using (2); namely, *AX.25 imm* corresponds to protocol implementation with immediate RR frame generation, while *AX.25 T2* – with awaiting for T_2 time prior to sending RR frame.

On the graph, we can see that the results do not differ very much. Some controllers (e.g., Z80-based TNC2 and TNC2D) can't make use of window size 4 and above – increasing this parameter does not practically increase transmission speed. KPC-9612+ behaves similarly. Faster TNC3 and TNC7 controllers, unexpectedly, behave worse than the others for $k < 7$. A more detailed analysis conducted in monitoring mode shows that these controllers do not request immediate acknowledgement by setting P/F bit in AX.25 protocol control field. Thus, the recipient waits for T_2 time for possible consecutive frames and sends the acknowledgement only afterwards. Nevertheless, when $k = 7$, TNC3, TNC7 and DLC7 achieve higher throughput than other controllers, close to the theoretical values. It is possible, because, for a maximum window size allowed by protocol definition, the recipient does not wait for the T_2 time before sending the RR acknowledge.

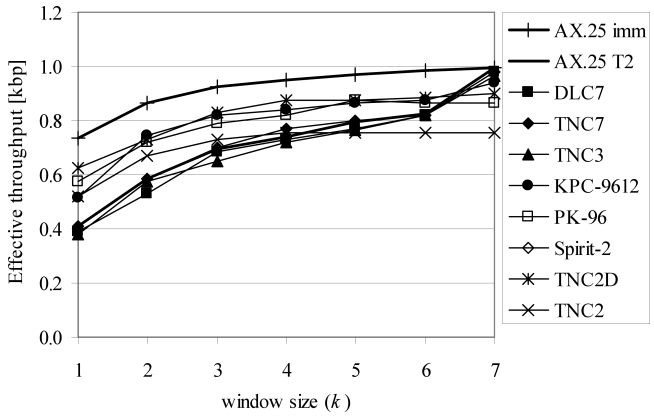

Fig. 4. Effective throughput using 1.2 kbps radio link

Results of similar measurements, conducted for $R_{wl} = 9.6$ kbps, are presented in Fig. 5. For comparison, similarly to Fig. 3., the graph also presents two curves showing theoretical capabilities of AX.25 protocol. In this case, difference between various TNC controllers is much more visible than for lower transmission rates. Depending on controller type, maximum effective throughput varies from about 1.5 kbps (TNC2D) to almost 5.5 kbps (DLC7), while theoretical maximum is about 5.9 kbps. The difference between results for $k = 6$ and $k = 7$ are also visible for TNC3, TNC7 and DLC7 controllers.

Figure 6 presents measurement results for $R_{wl} = 38.4$ kbps radio link. It is the highest transmission rate that allows for comparison of most of TNC controllers – only multimode and Z80-based (except Spirit-2) controllers do not provide this rate. It is also worth noting that the difference between Spirit-2, KPC-9612+ and PK-96[1] is not large. Nevertheless, they all limit the effective throughput to about 5 kbps, while TNC3 allows achieve almost 8 kbps, TNC7 and DLC7 – 10 and 11 kbps, respectively. It can be easily seen that the processing power of the microprocessor used in TNC is essential for the effective throughput. Its significance grows up with increasing transmission rate.

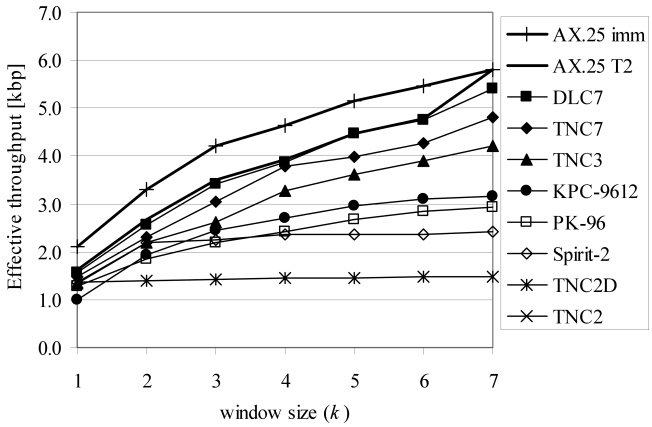

Fig. 5. Effective throughput using 9.6 kbps radio link

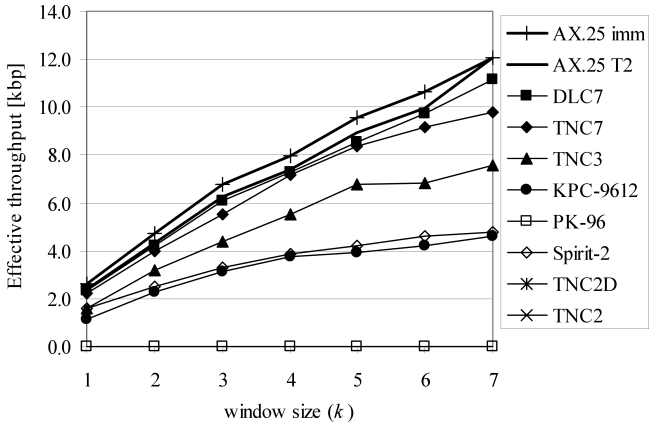

Fig. 6. Effective throughput using 38.4 kbps radio link

Real window size. In order to determine the exact reason of low efficiency of Z80-based TNC controllers, we transmitted a relatively large file (64 KB), for various data field capacities in the I frames (N_1) and window size (k) set to 7. All transmitted frames were logged by the receiving TNC working in the monitor mode. Basing on such a transmission report, we determined real window sizes that occurred during the transmission. Histograms, presenting distribution of real window sizes for few selected controllers and transmission rates, are presented in Figs. 7–9.

[1]PK-96 results are not shown because at 38.4 kbps it works unstable and only few measurements were successful; this rate can be set up in software, but communication proceeds with frequent errors – probably analogue filters are set for lower rates and do not allow for reliable transmission.

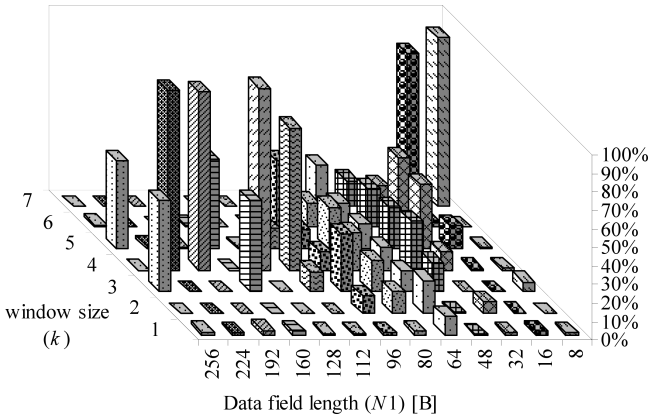

Fig. 7. Distribution of real window size for TNC2D controller at 1.2 kbps

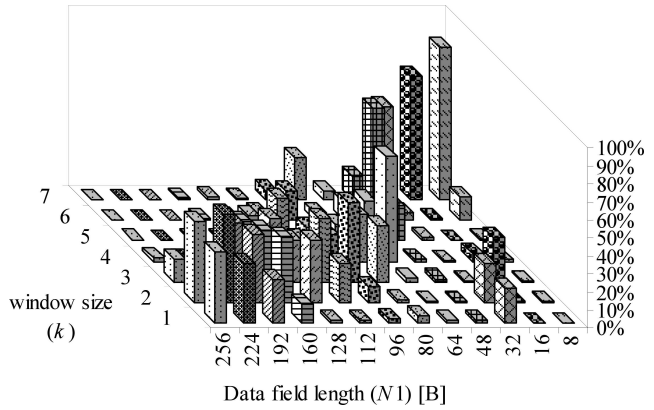

Fig. 8. Distribution of real window size for TNC2H controller at 9.6 kbps

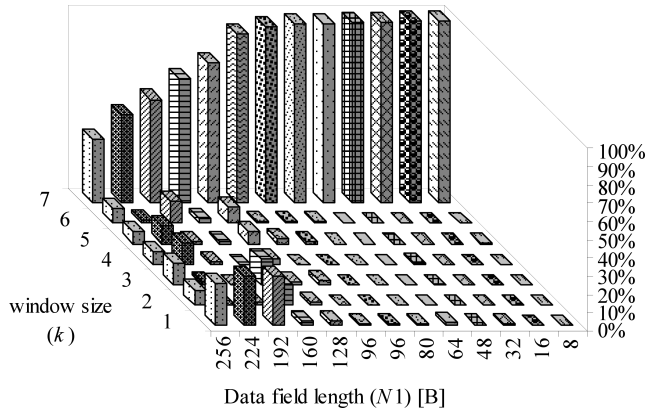

Fig. 9. Distribution of real window size for KPC-9612+ controller at 9.6 kbps

Presented results show that Z80-based TNC controllers are not able to utilize maximum window size, practically regardless of radio link transmission rate. Nevertheless, with decreasing I frame length, real window sizes increase. We may therefore assume that the controller is not able to process sufficiently large amount of data in a sufficiently short time. Possible reasons of such behaviour are: too low processing power of the microprocessor, low efficiency of control software (lack of optimization), too low capacity of memory used as transmit and receive buffers or special limitations introduced in software.

TNC controllers based on microprocessors others than Z80, e.g., KPC-9612+, are much more capable of full utilisation of maximum window size. We may observe efficiency decrease for $N_1 \geq$100 bytes, however, even for maximum-length I frames ($N_1 = 256$ bytes), window size of $k = 7$ dominates. Even better performance is shown by TNC3 and TNC7 controllers – efficiency decrease caused by (not full) utilisation of maximum window size may be observed only for $R_{wl} \geq$150 kbps.

Comparing measured effective throughput with real window size, we can say that ability to utilise maximum window size is essential property of TNC. No matter whether real window size is limited by TNC properties or k parameter value, decreasing of it always increases protocol overhead, thus reducing effective throughput.

3.2. Software influence on effective throughput. Results presented in the previous section clearly show that the microprocessor type and its processing power has great influence on effective throughput. Indeed, even for relatively low transmission rates, TNC controllers containing slow microprocessors present much poorer performance than estimated by the theoretical model. However, during the tests, we observed some software-dependent protocol implementation issues that might affect measured effective throughput. Thus, we decided to compare various TNC control programs using a common hardware platform.

For Z-80-based controllers, there are multiple software types and versions available which allows compare different AX.25 protocol implementations using a common hardware platform. This, in turn, allows achieve clearer results, not distorted by the influence of the TNC hardware.

The experimental test were conducted using four types of Zilog Z80-based TNC controllers running at various clock frequencies (f_{clk}). They also differ in terms of maximum transmission rates on wired and wireless links (R_w and R_{wl}, respectively). The availability of particular wireless link transmission rates is further limited by capabilities of built-in modems. Selected construction parameters of the TNC controllers are collected in Tables 1 and 2.

During the tests, the controllers ran under the control of several types of software, namely:

- MFJ (initials of the author, Martin F. Jue) software (1.1.4, 1.1.9 and Muel versions), as delivered with TNC2, TNC2D and similar TNC2H controllers;
- TF (*The Firmware*) software (2.1d, 2.3b and 2.7b versions, all in 10 connections variant), as delivered with TNC2D and similar TNC2H controllers;
- Spirit-2 TNC software (5.0 version), as delivered with Spirit-2 controllers.

As all the aforementioned controllers are hardware compatible with each other, the software can be easily interchanged by EPROM memory replacement.

The tested controllers acted as transmitters or receivers. In both cases, tested controller was connected in pair with either TNC3 or TNC7 controller. Both TNC3 and TNC7 are much faster than any Z80-based TNC (see Tables 1 and 2 and the results presented in Subsec. 3.1), thus, they should not significantly decrease measured effective throughput.

During the tests, an 8 KB file was transmitted. The size was chosen as a compromise between transmission time and measurement accuracy. AX.25 protocol was configured for maximum theoretical throughput (window size $k = 7$, data field capacity $N_1 = 256$ bytes). Transmission time was measured from the transmission start at the sender side (T_{CT} in Fig. 3) to the transmission end at the recipient (T_{rc} in Fig. 3). Although it takes into account transmission between computer and TNC, the influence of these times is negligible when total transmission time is sufficiently long [11]. Because of transmission rate ranges of TNC built-in modems (see Table 2), tests for different transmission rates had to be performed separately using different controllers. Thus, two different configurations could be set up. In the "slower" one, controllers running at about $f_{clk} = 2.5$ MHz or 4.9 MHz were used and configured for $R_{wl} = 1.2$ kbps. In turn, in the "faster" one, controllers running at about $f_{clk} = 10$ MHz or 20 MHz were used and configured for $R_{wl} = 9.6$ kbps.

Results for "slower" configuration. In the "slower" configuration, TNC2 and TNC2D controllers were configured for $R_{wl} = 1.2$ kbps and $R_w = 9.6$ kbps. Measurements results for TNC's acting as a sender and a recipient are presented in Fig. 10 and Fig. 11, respectively. For comparison, the graphs contain also curves representing theoretical throughput of AX.25 protocol with immediate acknowledge generation (AX.25) or with T_2 delay (AX.25 T2), calculated according to [12].

In the case of tested TNC acting as a sender, one can see clearly that the effective throughput depends on the software used. All tested versions of MFJ and Spirit-2 software are very close to each other (to preserve clarity, only Spirit curve is presented on the graph). Effective throughput grows rapidly when window size (k) increases from 1 to 4. However, further increasing of k does not bring significant improvement of throughput. Probably there are some limitations in the software, e.g., buffer capacity, that do not allow to transmit, in average, more than 4 maximum-length I frames. Maximum achievable effective transmission speed varies from about 750 to 800 bps for slower TNC and from 800 to 900 bps for the faster one. Both results are visibly below theoretical throughput of AX.25 protocol of about 1000 bps. TF software behaves completely different. Version 2.1 seems the most ineffective in the role of the sender and the achievable throughput is independent of window size. Similar is version 2.3, however, only when run on the slower TNC2; on the faster one, it behaves similarly to version 2.7. Thus, one might conclude that it requires more processing power than TNC2 can offer. Version 2.7 is the best one regardless of TNC speed. However, when run on the faster TNC2D, when $k = 7$, it achieves the same results as MFJ software.

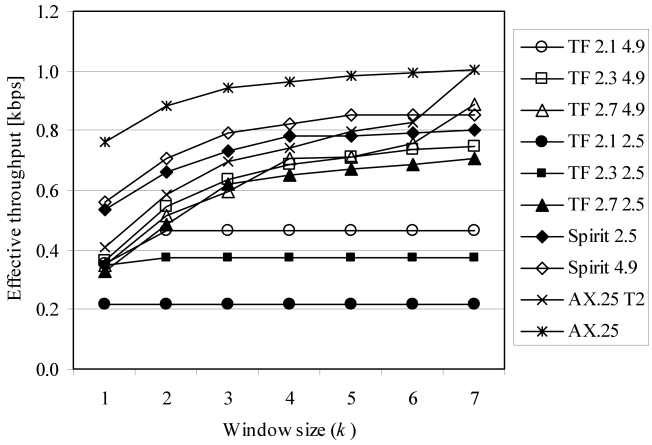

Fig. 10. Effective throughput for TNC2 (2.5 MHz) and TNC2D (4.9 MHz) as a sender

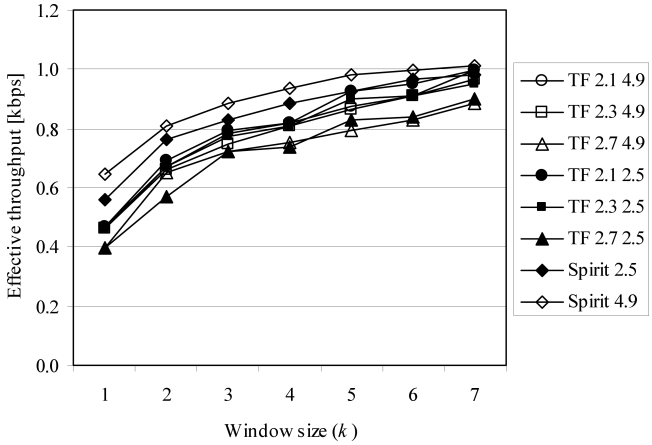

Fig. 11. Effective throughput for TNC2 (2.5 MHz) and TNC2D (4.9 MHz) as a recipient

When the tested TNC acts as a recipient, the difference between the fastest and the slowest software is much smaller. It can be found out, however, that the fastest reception proceeds under control of Spirit software, especially when run on a faster TNC2D. The slowest reception is for TF 2.7, regardless of microprocessor clock. It is also worth noting that these results are much closer to the theoretical throughput and vary from 900 to 1000 bps regardless of TNC clock frequency. Possible reason of this behaviour is such that much faster TNC3 or TNC7 acts as a sender. It allows conclude that the sender processing power is much more important from the point of view of protocol efficiency than that of the recipient. Nevertheless, recipient software has still some influence on effective throughput.

Results for "faster" configuration. In the "faster" configuration, Spirit-2 in Standard and High Speed versions were configured for $R_{wl} = 9.6$ kbps and $R_w = 57.6$ kbps. Measurements results for TNC's acting as a sender and a recipient are presented in Figs. 12 and 13, respectively. The graphs contain also curves representing theoretical throughput of AX.25 protocol with immediate acknowledge generation (AX.25) or with T_2 delay (AX.25 T2).

If the tested TNC acts as a sender, the results are somewhat similar to those obtained in the slower configuration. Again, the slowest sender is TF 2.1, regardless of microprocessor clock frequency. TF 2.3 is a little better. Both versions work faster at the faster microprocessor. However, they are both slower than Spirit and MFJ software in any version. These programs achieve similar efficiency regardless of clock frequency. This may lead to the conclusion that the transmission procedures are well optimised, however, some limitations exist in the software that does not allow reach higher efficiency when microprocessor's processing power would allow for it. It is especially visible for $k \geq 4$, where – similarly to the slower configuration – increase of window size does not bring visible improvement in effective throughput. TF 2.7 achieves speeds similar to MFJ and Spirit. Nevertheless, when microprocessor works with faster clock (20 MHz), it outperforms all other software types, although not significantly. Maximum effective throughput measured in this test is about 4000 bps, while theoretically it could be as high as about 6000 bps. Thus, one can conclude that Z80-based TNC controllers are too slow to obtain performance that is high enough to use 9600 bps radio link efficiently, or the software – especially TF 2.7 – has not been sufficiently optimised.

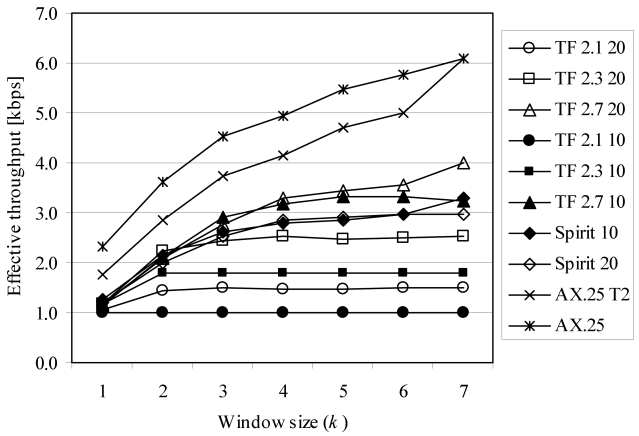

Fig. 12. Effective throughput for Spirit Standard (10 MHz) and High Speed (20 MHz) as a sender

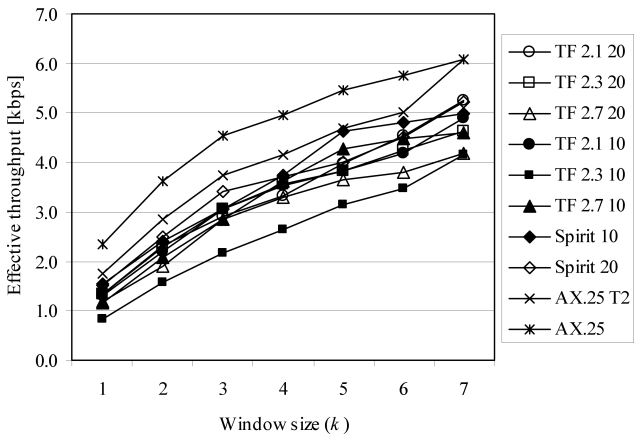

Fig. 13. Effective throughput for Spirit Standard (10 MHz) and High Speed (20 MHz) as a recipient

If the tested TNC acts as a recipient, the results are also similar to the respective ones obtained in the slower configuration. The difference between the slowest and the fastest recipient is not very big – effective throughput ranges from about 4100 bps to about 5200 bps. Both results are visibly below theoretical estimation, which differs from the "slower" configuration where some measurement results were comparable to the calculated limit. It may lead to the conclusion that the processing power of a Z80-based TNC controller is sufficient for $R_{wl} = 1.2$ kbps, but not for $R_{wl} = 9.6$ kbps or more. The fastest recipients are TF 2.1 and Spirit, both at 20 MHz clock frequency. What seems surprising, TF 2.7 performs better for slower clock than for the faster one.

Real window size. In order to find out a more detailed reason of difference in effective throughput achieved for various software types and versions, transmission reports were collected in a monitoring mode that is available in all TNC control software. The monitor output contains not only user data, but also decoded frame headers with addresses and decoded control fields. Browsing such report allows analyse real frame exchange process. From the data gathered this way, we can obtain real window size distribution. The distribution of real window size for both "slower" and "faster" configurations are presented in Figs. 14 and 15, respectively.

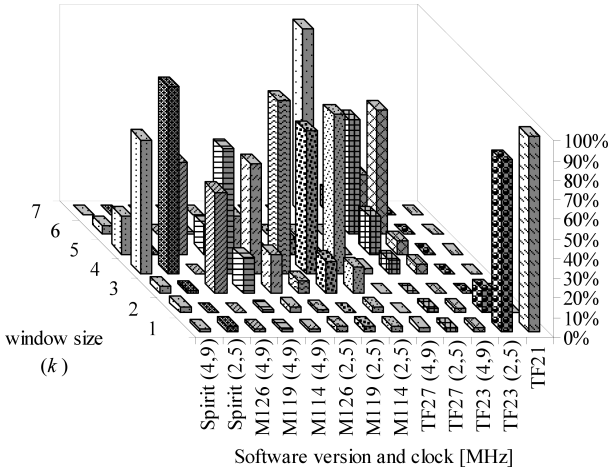

Fig. 14. Real window size for "slower" configurations

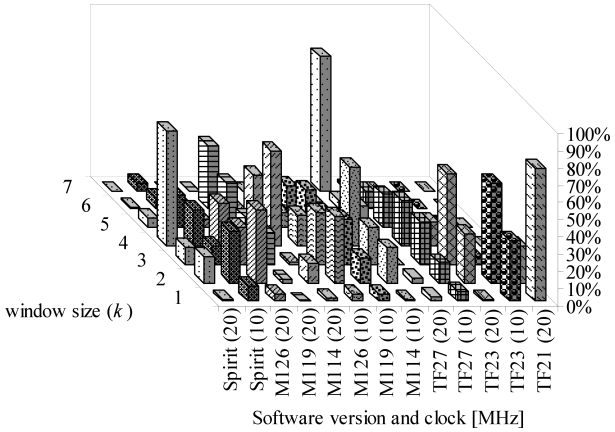

Fig. 15. Real window size for "faster" configurations

In the presented histograms, we can easily see that TF software behaves completely different than MFJ and Spirit ones. For "slower" configuration and MFJ or Spirit software, real window size oscillates around a value of 4. During transmission, depending on version, there is either always window size of 4, or 3 and 5 interleaving. For faster configuration, dominance of $k = 4$ is not so obvious. Indeed, smaller window sizes occur during transmission.

TF software behaviour, in turn, depends more on clock frequency. If it is too low – say, 2.5 MHz – TF 2.1 and 2.3 use always window size of 1 or 2. The situation gets better for TF 2.3 when clock is at least 4.9 MHz – the real window size grows to about 6 if R_{wl}=1.2 kbps and 3 if $R_{wl} = 9.6$ kbps. Nevertheless, TF 2.7 is even more effective and can use window size of 7, but only if the microprocessor has enough processing power. For example, if $R_{wl} = 1.2$ kbps, TF 2.7 achieves average real window size of about 6 when run at 2.5 MHz microprocessor, and almost 7 at 4.9 MHz one. Similarly, if $R_{wl} = 9.6$ kbps, it obtains about 4 at 10 MHz and almost 7 at 20 MHz. Unfortunately, it does not result in improvement of effective throughput, probably because the TF 2.7 software needs more time to process required amount of information than, for example, MFJ or Spirit ones.

Surprisingly, when TF 2.1 or 2.3 operate on 10 or 20 MHz Z80, they use larger window size despite higher transmission rates. TF 2.7, running at 20 MHz, can still use window size of 7. However, this ability doesn't make it much more effective than MFJ or Spirit software, which can't use window larger, in average, than 4 maximum-length frames. This observation allows conclude that TF 2.7 capabilities are achieved at a cost of decreased frame processing speed.

4. Summary and conclusions

During described tests it has been shown that effective throughput achieved in a given configuration depends on not only TNC hardware – particularly microprocessor type and its clock frequency – but also properties of software that controls TNC operation. The following factors, depending exclusively on software, may have influence over circuit performance:

- full utilisation of window size for every data field capacity of a frame,
- sufficiently high processing speed of AX.25 protocol frames,
- immediate generation of acknowledgement of error-free information frame reception,
- immediate acknowledgement request by setting of P/F bit in the latest information frame within a window.

Inability of full utilisation of window size is especially annoying in Z80-based TNC controllers, practically regardless of its clock frequency and memory capacity. However, type and version of software used in TNC has some influence upon its performance. For example, versions supporting TAPR command set rarely utilise window size – the controller can not transmit more than 5 maximum-length I frames consecutively. A little better is TF software, which, especially in most up-to-date 2.7 version, can send up to 7 maximum-length I frames consecutively. It seems however, that such capability is achieved at a cost of longer frame preparation for transmission.

Controllers, supporting TAPR command set, but based on other microprocessor types, can utilise window size much better, even at higher transmission rates. Unfortunately, there is no alternative software for these controllers; it is thus hard to say if this capability results from higher processing power of a microprocessor, or better software quality in terms of both protocol implementation and code optimisation.

Additional factor that influences the effective transmission speed is the way the recipient treats the window size less than 7. In general, if the sender does not mark the latest frame within a window with P/F bit, TF software sends the acknowledgement only after T_2 time elapses, while MFJ – immediately; however, some implementations based on MFJ software, e.g., in TimeWave and Kantronics controllers, behave similarly to TF. In some versions of TF software, T_2 time may be set manually to any value, in others – e.g., TNC3 – it is calculated automatically and cannot be changed. This parameter can also be set up in some versions of TAPR software.

Some software versions – e.g., in Kantronics controllers – at the beginning of transmission, initially limit window size, and later gradually increase it up to maximum value set. Such behaviour may be reasonable, because it allows recognise capabilities of a receiving station. However, when the transmitted information is relatively short, the transmission efficiency decreases.

Effective throughput is not the only network quality measure. In some applications, e.g., control and time-bounded ones, transmission delay is more important than throughput. Currently we investigate this issue.

Presented results are achieved for perfect transmission conditions, because our main goal was to show implementation influence on effective throughput. In real conditions, possible transmission errors could make the experimental results less clear and thus more difficult to analyze. Thus, it could lead to misinterpretation of the results and false conclusions, so we decided such analysis was beyond the scope of this paper. Nevertheless, such results could also be interesting, especially in comparison with the perfect-conditions case.

The presented results are limited to the AX.25 protocol implementations that are commercially available, which limits its set of tested microprocessors and software versions. Thus, we plan to make our own experimental implementations. We hope that results presented in this paper and experience with different protocol operation in various software implementations may help achieve better results.

REFERENCES

[1] W.A. Beech, D.E. Nielsen, and J. Taylor, *AX.25 Link Access Protocol for Amateur Packet Radio*, Tucson Amateur Packet Radio Corporation, Tucson, 1997.

[2] ISO/IEC 13239:2002: *Information technology – Telecommunications And Information Exchange Between Systems – High-Level Data Link Control (HDLC) Procedures, Third Edition*, ISO, Geneva, 2002.

[3] ISO/IEC 7776:1995: *Information Technology – Telecommunications and Information Exchange Between Systems – High-Level Data Link Control Procedures – Description of the X.25 LAPB-Compatible DTE Data Link Procedures, Second Edition*, ISO, Geneva, 1995.

[4] P.R. Karn, H.E. Price, and R.J. Diersing, "Packet radio in the amateur service", *IEEE J. Select. Areas Commun.* 3 (3), 431–439 (1985).

[5] R. Diersing and J. Ward, "Packet radio in the amateur satellite service", *IEEE J. Select. Areas Commun.* 7 (2), 226–234 (1989).

[6] J. Ronan, K. Walsh, and D. Long, "Evaluation of a DTN convergence layer for the AX.25 network protocol", *Proc. Second Int. Workshop on Mobile Opportunistic Networking MobiOpp* 10 ACM, 72–78 (2010).

[7] Y. Xiao and J. Rosdahl, "Throughput and delay limits of IEEE 802.11", *IEEE Commun. Lett.* 6 (8), 355–357 (2002).

[8] D. Qiao, S. Choi, and K.G. Shin, "Goodput analysis and link adaptation for IEEE 802.11a wireless LANs", *IEEE Trans. Mobile Comput.* 1 (4), 278–292 (2002).

[9] B. Zieliński, "Efficiency analysis of IEEE 802.11 protocol with block acknowledge and frame aggregation", *Bull. Pol. Ac.: Tech.* 59 (2), 235–243 (2011).

[10] J.S. Ma, "On the impact of HDLC zero insertion and deletion on link utilization and reliability", *IEEE Trans. Commun.* 30 (2), 375–381 (1982).

[11] B. Zieliński, "An analytical model of TNC controller", *Theoretical and Applied Informatics* 21 (1), 7–22 (2009).

[12] B. Zieliński, "Efficiency estimation of AX.25 protocol", *Theoretical and Applied Informatics* 20 (3), 199–214 (2008).