A Hands-on Approach to Teaching Basic OSI Reference Model

Tadej Tuma, Iztok Fajfar, Martin Perko, Franc Bratkovič, and Janez Puhan
University of Ljubljana
Faculty of Electrical Engineering,
Tržaška 25, 1000 Ljubljana, Slovenia

Abstract

In this paper we present a new LAN (local area network) concept especially designed for educational purposes. Our LAN is based upon the standard RS232 communication protocol, which is utilized in a bit unconventional but quite interesting manner. The main objective was to develop a platform which would enable students to gain firsthand experience of all seven OSI (open system interconnection) layers.
From our teaching practice we have learned that it is quite impossible for the students to absorb every detail of complex professional networks, especially on the lower layers. The proposed approach to teaching computer network has proved very popular with students, in spite of the additional work the approach imposes upon them. The students' comprehension of basic OSI concepts is considerably improved by the experience they gain through the practical work in the laboratory.

1. Introduction

Almost any electrical engineering curriculum includes some basic course on computer networks. The course will no doubt discuss basic reference model of open system interconnection (OSI) [1, 6]. The reference model contains seven layers:

a) the application layer (layer 7),
b) the presentation layer (layer 6),
c) the session layer (layer 5),
d) the transport layer (layer 4),
e) the network layer (layer 3),
f) the data link layer (layer 2), and
g) the physical layer (layer 1).

After a thorough discussion of the seven OSI layers, the teacher will present some different network concepts in a comparative approach. This is a rough picture of the network theory; however, any modern course will include some practical laboratory work, where students can get realistic experience in developing and using computer networks. The problem with real world networks is that their implementation is far too complex on the lower four layers to be useful for educational purposes. Layers 1 to 4 are usually presented to the students in form of simulations or visual animations.

A few years ago, we decided to develop our own network, especially designed for educational purposes. Our goal was to set up a simple and transparent structure, where the students themselves could build all the OSI layers. We have based our local area network (LAN) [1, 4] on the standard RS232 interface, so there was no need for any special hardware. Unfortunately, the RS232 interface has been designed for connecting only two devices, but we have overcome this inconvenience simply by connecting the devices in slightly unconventional way.

In this paper we concentrate on the architecture and protocols that are used in our so called training LAN. The description follows the OSI layering from bottom to top, in which order the topic is presented to the students.

2. The Physical Layer

This layer provides the mechanical and electrical means to activate, maintain, and de-activate physical connections for bit transmission between terminals. Not much can be altered here as we decided to use the RS232 interface, which transmits data within this layer according to its own protocol.

In theory, RS232 allows duplex connection at a speed up to 19,200 BPS with terminals at most 15 meters apart. In practice, speed can be increased up to 115,200 BPS without modifying hardware and virtually without any data loss. For our purpose, we divide the full duplex into two simplex (unilateral) connections. The RS232 interface employs more signal lines, but our network will only use 4 of them: RTS (Ready To Send), CTS (Clear To Send), RX (Receive) and TX (Transmit). Figure 1 shows how a ring-shaped network can be set up using those 4 lines.

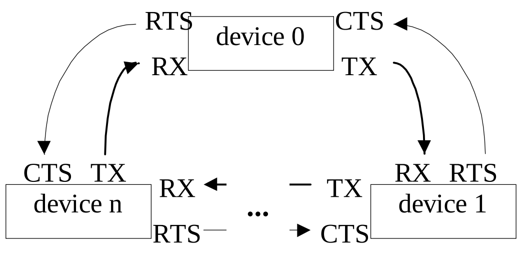

Figure1: Connecting *n* devices into the ring shaped-network

The ring-shape implies some drawbacks, which are briefly studied at the end of the article. Since RTS and CTS lines are intended solely for data flow control, useful data are transmitted only via the RX - TX link. And since data can be sent only through the TX and received through the RX pin, the RX - TX link is unidirectional. A given device can receive data only from the first of its two neighbours and send it only to the second one. We will call the two neighbours sending and receiving neighbour, respectively. If the device wants to send data to device other than its receiving neighbour, the data has to travel clockwise through all devices between the source and destination device. The devices in between just retransmit the data. Only the target device intercepts the data and passes it on to its user.

This type of ring-shaped network is known as *register insertion ring* and is based on a *slotted ring*. An example of such a network is the *Cambridge Ring* [2]. Some similarities can also be noted between our network and the *Token Ring* [3].

Devices in our network are interconnected with a point-to-point type of connection. Therefore only two devices share the use of each connection line (channel). Another type of channel, more commonly found in LANs, is a multi-access or broadcast channel, where all devices communicate through a single channel. The latter type has an advantage over the former due to robustness of the network (if one of devices stops working, the whole network remains active). On the other hand, the usage of broadcast channels implies the need for complex collision detection mechanisms that might be too complex for our teaching purposes. Data from several devices can not collide at point-to-point channel based networks. However, the network crashes when any one device connected to it goes down.

There are two basic structures of networks using point-to-point channels: centralized (e.g. star shaped) and decentralized (e.g. ring shaped) networks. Since host computers in star shaped networks must have one RS232 interface per client, we chose a ring topology where only one interface is needed per computer. Also, in ring shaped LANs, all computers can be equipped with the same software.

3. The Data Link Layer

Up to this point we have physically defined our network, which now provides to the data link layer services for sending electrical signals (bits) via the connections. At the data link layer, however, some data formatting has to be implemented in order to provide required performances of the network [2, 3].

The RS232 interface provides only a rudimentary error detection method through a parity bit. Data is packed in 5 - 8 bit words with one stop, start and parity bit respectively. To decrease the data overhead, we have chosen 8 bit words with no parity bit. Parity checking is not necessary, since data is packed to larger structures on higher levels where we can use more elaborate error detecting/correcting methods.

Since the network consists of more than two devices, at least source and destination device names have to be added to the header of each message. Therefore, the message packet should contain at least 3 words, but we have chosen 16 words per packet to reduce data overhead. This size has also been chosen according to standard UART buffering capability [5]. The packet structure is shown in Figure 2.

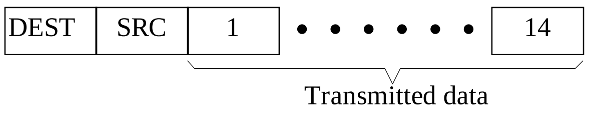

Figure 2: Data packet (a frame)

This packet is the smallest message which can be sent and it is called a frame. Although theoretically 256 devices can be addressed, only 64 are allowed in our construction. Devices are automatically enumerated during a network initialization. The device which starts the initialization, gets the number 0 and becomes the supervising device. The device on its TX channel gets number 1, the next one number 2, and so on. Only the supervising device holds all the information about the network (e.g. device numbers and names), which is important mostly for the network layer. On higher layers, however, the special status of the device 0 is transparent and all devices are therefore equal.

Usually, only one device is the destination of a frame. However, in some cases such as sending network initialization or some other system message, several devices can be the destinations of a single frame. The sender of such messages can be only a supervisor. If a device is a destination of the received frame (that is, if the destination number in the frame is the number of the device), then the frame is passed to the user. If the frame has any other destination, it is forwarded to the device's receiving neighbour. Note that every device first receives all 16 words of a frame, then checks if it is the target device and, if that is not the case, forwards it. As a result of this procedure significant data delays occur. These delays are not a serious drawback per se, but they can complicate implementation of some higher level protocols (those which require intensive interactive bi-directional communication). The data delay in one device can be easily calculated as

$$t_D = N/B, \qquad (1)$$

where *N* is the number of bits in a frame and *B* is the channel capacity in bits per second. As an illustration, if a 10 bit word (8 data plus one start and stop bit) is used and 64 devices are connected via 9600 BPS links, the maximum delay would be

$$64 \cdot \frac{16 \cdot 10\text{bit}}{9600\text{bit / s}} = 1.06s\ .$$

4. The Network Layer

On this layer, we will connect more than two devices together using the services provided by the data link layer. Here, we have to cope with several specific problems. The first one is common to both, point to point and broadcast, network types. Communication channels, connecting all devices together, have to transfer data from several devices virtually at the same time. Their capacities have therefore to be sliced. Moreover, in contrast to more common networks, our physical limits lie very low, and the network is likely to become saturated. Let us imagine a situation, described in Figure 3.

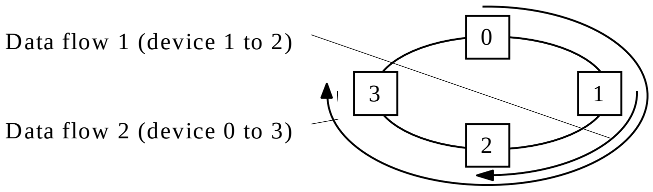

Figure 3: An example of a multiple channel request

Both devices, 0 and 1, want to communicate with their respective target devices continuously. The channel between devices 1 and 2 must therefore transfer data send from device 0 as well as data from device 1. Since the channel capacity equals to the RS232 transfer rate, both devices cannot send data at full speed. According to the network topology, each device can determine whether its sending neighbour wants to send data or not, while its receiving neighbour does not have that knowledge. A given device could therefore stop sending data if its sending neighbour is transmitting data to any other than the given device (if the given device is a target, then its output channel remains unused). That means the sending neighbour would have priority over the given device. Clearly, this is bad since we want as much of equality between devices in our network as possible.

To overcome this problem, an additional RTS-CTS feedback line has to be utilized. Its purpose is clear: through this line each device can tell its sending neighbour whether it can receive any new data or not. If any device wants to send more data that can be received by its receiving neighbour, it can a) store the data temporarily to some buffer or b) block its preceding device. The latter action is necessary in order to prevent any data loss, while buffering is implemented for further enhancements. A block scheme of such buffering is shown in Figure 4.

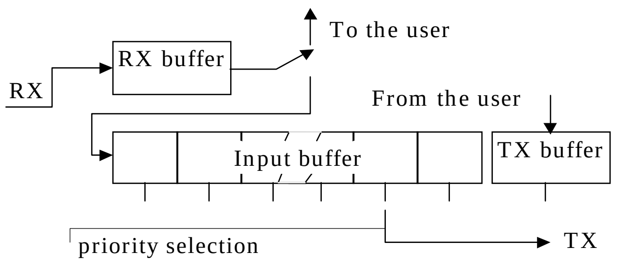

Figure 4: Buffering scheme

The usage of buffers can increase data delays, which can be as high as $k \cdot t_D$ (see equation 1), where k is the number of packets the input buffer can hold. The exact value depends on the specific situation and the type of buffers used.

Every device using a channel must share its capacity with the other devices that are trying to communicate through the same channel. Therefore, every device requests an average bandwidth b_R of every channel between itself and the destination device. Naturally, all requests cannot be granted if the sum of the individual bandwidths exceeds the available bandwidth B. The bandwidth granted to one device can differ from channel to channel according to the priority of the device. Because a device's granted bandwidth on one channel influences the device's requested bandwidth on the next channel, the device's requested bandwidth can vary, too. We define both requested and granted bandwidths for every device on every channel respectively. The nth device would request a bandwidth $b_{Rn,r}$ and get a bandwidth $b_{Gn,r}$ on the rth channel.

Clearly, the sum of the granted bandwidths on the rth channel cannot exceed the channel capacity:

$$\sum_{n=1}^{N} b_{Gn,r} \leq B, r \in [1, N] \tag{2}$$

N is the number of devices in the network. The sum should be equal to *B* if the sum of all requested bandwidths of this channel is greater than *B*.

Every device gets a portion of every channel in its route. The actual bandwidth of the connection between devices *m* and *n*, as seen by the user of the device, is given by smallest granted bandwidth along the route.

$$b_{Gm} = \left| b_{Gm,r} \right|_{\min}, r \in [m, n-1] \tag{3}$$

If we want optimal usage of capacities, then the granted bandwidths for a given device should be the same on all channels along the route:

$$b_{Gm} = b_{Gm,m} = b_{Gm,m+1} = \ldots = b_{Gm,n-1} \tag{4}$$

The bandwidth granted to a device is generally smaller than the requested bandwidth and an average time for sending a message is longer. Thus, every device should keep track of priorities of all sending devices to keep an average relative time increase for all devices as low as possible. Let us examine one channel, through which two devices are sending their data with requested bandwidths b_{R1} and b_{R2}. Their sum is larger than *B*, so they get granted bandwidths b_{G1} and b_{G2}, where $b_{G1} + b_{G2} = B$. The relative time increase is therefore:

$$g_1 = \frac{b_{R1} - b_{G1}}{b_{R1}} \text{ and } g_2 = \frac{b_{R2} - b_{G2}}{b_{R2}} \tag{5}$$

It can easily be verified that for minimal time increase the granted bandwidths should be

$$\frac{b_{G1}}{b_{G2}} = \frac{b_{R1}}{b_{R2}} \tag{6}$$

Since we cannot predict what the requested bandwidths will be in future, we must rely on past requests. Although the most widely acknowledged method is “packet aging”, we have implemented more appropriate method. Every device counts all requests and all granted requests respectively for all devices in the network. This numbers somewhat represent requested and granted bandwidths, respectively. Let the number of the *n*th device’s requests be r_n, and the number of *n*th device’s granted requests g_n. We can express its requested and granted bandwidth by

$$b_{Gn} = \frac{g_n}{\sum_{r=1}^{N} g_r} \text{ and } b_{Rn} = \frac{r_n}{\sum_{r=1}^{N} r_r}, \tag{7}$$

where *N* is the number of all devices in the network. Now the priority selection is clear: if the total of the requested bandwidths is smaller than the bandwidth *B*, all of the requests can find their way out at the same rate as they have arrived, and the transfer buffer remains empty. However, if the total of the requests exceeds the bandwidth *B*, the transfer buffer starts to fill up with yet to be sent messages. In that case the device selects the message that has the smallest granted bandwidth to requested bandwidth ratio, *γ*, calculated using equation 7 as

$$\gamma_n = \frac{b_{Gn}}{b_{Rn}} = \frac{g_n}{r_n} \frac{\sum_{r=1}^{N} r_r}{\sum_{r=1}^{N} g_r} \tag{8}$$

Since both sums in equation 8 are equal for all devices, they do not have to be calculated. Taking that into account, a condition defined by equation 6 is met relatively rapidly. To eliminate the influence of too early events all counters should be reset when a transfer buffer is empty. That way, the priority selecting is strictly limited to events in critical time. A device must calculate priorities only if its user wants to transmit data or if it has a restricted outgoing channel (its receiving device can no longer accept data). Otherwise, it simply transfers received data to the outgoing channel.

This method is very appropriate because infrequent and short messages (system messages, short notes) are usually more important than the long ones (file transfers). The former ones gain full priority at first request because the γ ratio of its sender is 0 (1 requested and 0 granted messages). When packet aging is used, such messages can be considerably delayed, as they are the youngest in the buffer. On the other hand, when more than one device request full channel, its capacity is divided into two equal parts. The RTS-CTS feedback line implicitly transfers the information about the smallest priority on the communication route to the devices preceding the weak point (the channel with the lowest priority). Thus, equation 4 is satisfied.

Note that the γ ratio refers to a device and not to a communication route. Therefore, only the sender's number of requests is taken into account for calculating γ.

Transfer buffer size can vary. If it is large, the priority selection can run with less compromise, but the time delays become larger. With larger buffers, there is also a smaller chance that priority selecting will have to be engaged as they can accumulate larger data peaks. We have chosen a 256 bytes long transfer buffer, which is enough for 64 frames.

Note that this method is proper just for the networks with limited number of connected devices. For large networks, priority selecting would require some heavy computing and time delays would raise. Practical experiments have shown that this method is most effective in case of a saturated network..

5. The Transport Layer

On this layer, we finally take care of data and message transfer. It is somehow obvious that we need to implement some more mechanisms in order to get the network working. Two types of messages are possible on the network: user and control messages. Each of them can consist of more than one data frame. Therefore, we need an additional header attached to every message as we did on the data link layer. We have chosen a 3-byte header in the first data packet of every message. The first two bytes contain the total length, in bytes, of the message, whereas the last byte is the information about the message type (e.g. data, diagnostics, control, error, etc.). The structure of such a message is shown in figure 5.

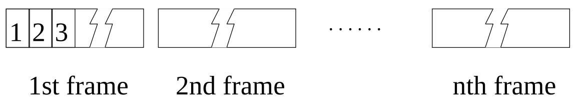

Figure 5: The structure of a message

On the transport layer, we can employ some error correcting method such as CRC. Unfortunately, due to large time delays, the usage of more effective methods (e.g. variable share of error correcting data, as used in Zmodem) would be obstructive. For the same reason, the usage of error detecting mechanisms (such as parity) is not effective, as it would take too long to re-send the data in case they were erroneous. This is the reason why our network does not utilize parity offered by RS232.

6. The Upper Layers

Higher OSI layers (session, presentation, and application) are mainly hardware independent. Indeed, if two applications reside in the same computer, it generally is not necessary to invoke the services of the lower four layers.

The session layer

The session layer is very rich in functions and somewhat complex. This layer provides for the organized means to exchange data between end-user applications. One of its most important functions is the provision for a graceful close between the user applications. Since in our model network this functions are not crucial for its operation, we do not use this layer in our laboratory sessions. Another reason for this decision is the fact that even in practice this layer is not always used. For example, the Internet standards do not use the session layer.

The presentation layer

The presentation layer is used to assure that user applications can communicate with each other, even though they may use different representations for their PDUs (packets or messages). This layer can accept various data types (eg. integer, floating point, character) from the application layer. It is used to encode accepted data from an internal format of the sending machine into a common transfer format and then decode this format to a required representation at the receiving machine. We derived our encoding from ASN.1 Basic Encoding Rules [7]. Structure of our (simplified) encoding is straightforward and can be seen in Figure 6.

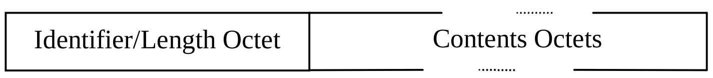

Figure 6: The structure of an encoding

Identifier/Length Octet contains information about the type of the data value and its length (in octets). Actual data are placed in one or more Contents Octets that follow.

The application layer

Finally, at application layer, students, in groups of two or three, develop their own simple protocols such as file transfer, electronic mail, and electronic chat. This is the only layer where they are not given exact protocol specifications but rather they are encouraged to "invent" their own protocols based on theoretical knowledge gained at lectures. At the end, after the projects have been finished and are working, the students present their work to their classmates, and we discuss benefits and draw-backs of different application-level protocols they invented, as well as the limitations that were posed upon the system using our simplified network model.

7. Conclusion

We have set up a relatively effective and simple network architecture, enabling students to design the entire network software all by themselves. In spite of extreme simplicity, the proposed architecture is still complex enough to demonstrate many of the basic problems one has to face when designing network systems. Apart from that, the nature of the problem is quite convenient for team work, since the software development can be easily separated into designing individual services according to the OSI reference model. Our network may be implemented using many different hardware platforms; even embedded microcontroller systems may be included in our LAN ring.

The students get hands-on experience at every network layer since our LAN is extremely transparent. It is easy to demonstrate an overloaded network, the students may study the impact of the buffer size on the network throughput, some error detection mechanism might be considered on the transport layer, a statistic analysis of the network’s traffic can be evaluated, even a bridge device between two separate token rings might be attempted. On higher layers, especially on application layer, there are even more possibilities. The students develop different applications for file transfer, a mail and message exchange and electronic chat. These are all applications they are already quite familiar with, which raises their motivation for work significantly.

In conclusion, we feel that our approach to teaching computer networks is very popular with students, in spite of the amount of additional work they have to do, and students' comprehension of basic network concepts is considerably improved by the experience they gain through the practical work in the laboratory.

References

[1] Andrew S. Tanenbaum C*omputer networks,* 2nd edition , Prentice-Hall, 1989.
[2] Michael E. Woodward, *Communication and computer networks modeling with discrete-time queues*, Pentech Press, 1993.
[3] Joseph L. Hammond, *Performance analysis of local computer networks*, Addison-Wesley, 1986.
[4] IEEE 802.3 *Token ring access method and physical layer specifications*, IEEE, 1985.
[5] Intel Corporation, M*icroprocessors and peripheral handbook*, Intel Corp.,1989.
[6] ITU-T Recommendation X.200 (1994) | ISO/IEC 7498-1:1994, *Information technology – Open Systems Interconnection – Basic Reference Model: The Basic Model.*
[7] ITU-T Recommendation X.690 (1994) | ISO/IEC 8825-1:1994, *Information Technology – ASN.1 Encoding Rules: Specification Of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) And Distinguished Encoding Rules (DER).*