1. Purpose

The purpose of this lab session is to illustrate three automatic repeat request (ARQ) algorithms: Alternating Bit Protocol (ABP), Selective Repeat Protocol (SRP) and Go-Back-N (GBN). All three algorithms can be described as error control algorithms. The underlying transmission channel is assumed to be noisy and hence unreliable. Any frames transmitted on such a channel may thus be lost or corrupted, i.e. a transmission error occurs. The three algorithms studied in this lab are responsible for the re-transmission of packets in the presence of such a noisy channel. A successfully transmitted frame is acknowledged by transmitting a special acknowledgement frame (ACK). Unsuccessfully transmitted frames cause a timeout by the transmitter and the affected frame is re-transmitted. The transmission is thus automatically repeated whenever a timeout occurs, hence the general label ‘Automatic Repeat reQuest’ – algorithms.

At the end of this lab, you should be clear on
• how each of these algorithms responds to errors in the underlying transmission channel
• the role of the timeout parameter in each of these algorithms
• the differences between the ABP, SR and GBN algorithms
• the functions of a COMNET III response source
• the functions of a COMNET III cloud link
• how to understand simple application sequences using a COMNET III application source.

In this lab we will not be concerned with the related topic of flow control. We will not consider
• the effect of the window size on the throughput
• the role of sequence numbers and ordered delivery of frames
• routing algorithms to overcome link failures

This lab session builds on the theoretical material on ARQ algorithms. To successfully complete this lab session, you are expected to be familiar with the COMNET III network simulator. To execute this lab, please make sure that COMNET III is installed on your machine and that the model files are also installed under the directory ‘C:\CommTut\Lab4’. The following files should appear under this directory:
• TimeoutABP.c3
• ARQComp.c3

In addition, you should find the following subdirectories and files
• Subdirectory: ‘C:\CommTut\Lab4\TimeoutABP’
• File: ‘C:\CommTut\Lab4\TimeoutABP\comments.txt’
• Subdirectory: ‘C:\CommTut\Lab4\ARQComp’
• File: ‘C:\CommTut\Lab4\ARQComp\comments.txt’
2. Theoretical Background

The purpose of Automatic Repeat reQuest (ARQ) algorithms is to overcome the problems introduced by a noisy / erroneous channel. When a station transmits a frame, there is no guarantee that the frame actually arrives at the destination. Transmission channels are often prone to errors, for example because of electromagnetic interference on the channel. The three algorithms presented in this lab overcome such transmission errors and thus guarantee that a frame arrives at the destination. This is done using two basic concepts

i. Acknowledgements  
ii. Timeouts

An acknowledgement\(^1\) (ACK) is a special frame, which is returned to the sender to indicate the proper arrival of a data frame. Thus, upon arrival of a data frame at the destination, a check is made that the contents of the frame are in order and if this is the case, an ACK is returned to the sender. The successful arrival of the ACK at the sender indicates that the data frame has been correctly received.

However, there is no guarantee that an ACK will make it back to the sender, since it too may be corrupted or lost during transmission. This is where timeouts come into play. A timeout is a time variable, indicating the maximum amount of time the sender will wait for an ACK. Upon transmitting a data frame, the sender starts a clock. If the ACK has not arrived at the sender by the time the timeout elapses, then the sender assumes that the transmission of the data frame has failed, e.g. because either the data frame or the ACK were lost, and thus re-transmits the data frame.

So the combination of ACKs and timeouts guarantees that a data frame will eventually reach the destination and that the sender will be informed about this. Of course, the underlying assumption here is that the channel is noisy or only failed for a limited period of time. If a link is failed, no protocol can guarantee transmission of data over this link.

Let us now give you a brief description of the three algorithms in turn.

2.1 Alternating Bit Protocol (ABP)

The alternating bit protocol is the simplest of the three protocols under study in this lab. It operates as follows:

The sender transmits a data frame, but keeps a copy of it in memory. The sender then waits for an ACK to return. If the ACK arrives before the timeout elapses, the old data frame is discarded from memory – the ACK indicates that it has successfully arrived at the destination – and the next data frame is transmitted. Again a copy is kept in memory in case the

---

\(^1\) We will only use positive acknowledgements in this lab. The notion of a negative acknowledgement (NAK) is not considered here.
transmission fails. If, however, the ACK does not arrive before the timeout elapses, the sender re-transmits the data frame and retains the copy in memory. The sender starts a new timer to be able to re-transmit the data frame after the timeout elapses.

The receiver performs the following operations: upon arrival of a data frame, the checksum is computed and the frame is determined as being either valid or corrupted. If the frame is valid, the receiver delivers it to the higher layer protocol. If the frame is corrupted, the receiver simply discards the frame.

Note that we have omitted a detailed description of sequence numbers above. In this lab, we will only comment on the sequence numbers for completeness, but we will not study this issue in detail here. Our focus is on the operation of the ARQ algorithms. We will assume that the receiver has a mechanism to deliver frames to the higher layer protocol in order. Furthermore, we will assume that a frame which has arrived out of order is treated like a corrupted frame, i.e. it is discarded.

Figure 1 illustrates the sequence of events in this protocol. Note that the first data frame arrives without errors. The second data frame is corrupted, and so the receiver does not generate an ACK, which in turn leads to a timeout on the sender’s side. In the third transmission, the data frame arrives in order, but the ACK becomes corrupted. Finally, the fourth transmission (the second re-transmission of the second data frame) is successful.

Note that there are a number of different delays involved in the transmission. These are:

**Transmission delay** ($T_{\text{trans}}$): time to transmit a data frame. This is a function of the bandwidth capacity of the link. It is computed as the ratio of the frame size in bits per frame divided by the link speed in bits per second. The unit is thus seconds per frame.
2. Theoretical Background

Propagation delay ($T_{\text{prop}}$): time for a single bit to propagate from the source to the destination. This is a function of the type of the transmission medium. For terrestrial links, the propagation delay is typically negligible.

Processing delay ($T_{\text{proc}}$): time to process a data frame or an ACK frame at the sender or the receiver. Note that we assume that this processing delay is the same for the sender and the receiver, and therefore it is not a function of the frame size. This delay is also typically assumed to be zero.

Acknowledgement delay ($T_{\text{ack}}$): time to transmit an ACK frame onto the link. This is the delay corresponding to the transmission delay for a data frame. However, since the sizes of a data frame and an ACK frame are different, the resulting transmission delays also differ.

These delays are important for the determination of the timeout interval. The timeout should be set to the minimum time between successive frame transmissions. Hence, the formula for the timeout interval ($TO$) is given by

$$TO = T_{\text{trans}} + T_{\text{prop}} + T_{\text{proc}} + T_{\text{ack}} + T_{\text{prop}} + T_{\text{proc}}$$

$$= T_{\text{trans}} + T_{\text{ack}} + 2T_{\text{prop}}$$

under the assumption that the processing delay is zero. Figure 2 illustrates these different delays. If the timeout interval is set to anything less than $TO$, then the sender would go into re-transmission even though the data frame may arrive without errors. The sender’s timeout would elapse before the valid transmission sequence could finish. On the other hand, if setting the timeout value higher than $TO$ would have an impact on the efficiency of the protocol, as you will see now.

![Figure 2: Delays incurred in the transmission of a data frame under ABP](image)

The efficiency of the protocol is defined as the “fraction of time that new data frames are sent – under the assumption that the sender has an unlimited supply of data frames ready for transmission”. As you can see from figure 2, the fraction of time that a new data frame
is sent is $T_{\text{trans}}$, whereas the entire duration for the ACK to return to the sender is indicated by $T_O$. Hence, the efficiency of ABP under a perfectly functional transmission channel is

$$\eta_{\text{ABP}} = \frac{T_{\text{trans}}}{T_O}$$

To assess the efficiency under a noisy transmission channel, we have to make certain assumptions about the frame error probability. If we assume that the probability that a data frame or an ACK frame are corrupted is denoted by $p$, then we can calculate the expected time to successfully transmit a single frame $E[X]$ as

$$E[X] = (1-p)T_O + p(T_O + E[X])$$

This equation says that with probability $(1-p)$ the frame is transmitted successfully at the first attempt, and that with probability $p$ a timeout occurs, and so the transmission time is $T_O$ plus again the expected value to transmit the frame after the timeout. Solving this equation for $E[X]$ gives us

$$E[X] = \frac{(1-p)T_O + pT_O}{1-p} = \frac{T_O}{1-p}$$

We can now use this value to compute the efficiency as

$$\eta_{\text{ABP}}(p) = \frac{T_{\text{trans}}}{E[X]} = \frac{(1-p)T_{\text{trans}}}{T_O}$$

Note that we assume here that the timeout $T_O$ is set to the minimum time required to complete a successful transmission sequence, as indicated in equation (1). We also assume that only one station at the time is transmitting, i.e. either the sender is transmitting a data frame, or the receiver is transmitting an ACK frame. This assumption is equivalent to assuming a half-duplex channel. For a discussion of these results under without these assumptions, consult one of the books in the bibliography.

### 2.2 Selective Repeat Protocol (SRP)

The selective repeat protocol (SRP) is an extension of the ABP. It introduces a major improvement: allowing the transmission of multiple data frames. If you consider the ABP closely, you should notice that it is quite inefficient for the transmission of data frames. Only one unacknowledged data frame can be outstanding at any time. The sender transmits a data frame, and then has to wait for the ACK to return before transmitting the next data frame. This waiting introduces inefficiency in the protocol.

The selective repeat protocol introduces the concept of a window. A window is basically a counter of the maximum number of unacknowledged frames that the sender is allowed to transmit. The SRP protocol operates as follows:
The sender has a counter, say $W$, indicating the window size. It transmits $W$ data frames and keeps copies of these data frames in memory – just in case the frames get corrupted and have to be re-transmitted. Furthermore, the sender starts a timer for each data frame transmitted. The receiver is obliged to acknowledge every data frame it receives without errors. Thus, upon return of an ACK, the sender transmits the next data frame, keeping a copy of it in memory for possible re-transmissions. It also discards the acknowledged data frame from memory. If the ACK for a data frame does not arrive in time – i.e. if a timeout occurs – then the sender re-transmits that data frame. Notice that because of such selective re-transmissions, the data frames may arrive out of order. We will comment on this below.

As mentioned above, the receiver is obliged to acknowledge every data frame it receives without errors. Data frames received with errors are not acknowledged. The receiver also keeps correctly received data frames in memory. It has to ensure that the data frames are passed to the higher layer protocol in the correct order. The receiver thus keeps all data frames which have arrived without errors in memory until it can pass them up to the higher layer protocol in the correct sequence.

As you may realize, the SRP is more complex than the ABP. The notion of a send window introduces additional complexity at the sender’s side. Both the sender and the receiver have to maintain a block of memory to store data frames. The receiver has to take on the additional task of keeping track how to deliver the data frames to the higher layer protocol in order. The advantage of this additional complexity is an improved efficiency. Before we examine the SRP’s efficiency, we will give you an example of how data frames may arrive out of order.

Example 1: Let us assume that the window size $W$ is set to 4 data frames. The sender transmits all 4 data frames and starts a timer for each frame. The first frame encounters heavy noise on the transmission channel and subsequently becomes corrupted. The remaining 3 frames arrive in order at the receiver. The receiver then puts frames 2-4 in its memory, realizing that the first frame is missing. It therefore cannot deliver these frames to the higher layer protocol. The receiver also acknowledges frames 2-4.

Eventually, the timeout of the first data frame elapses, and the sender realizes that something has gone wrong. The sender takes the copy of the first data frame from its memory and re-transmits the frame. This time,
transmission is successful and the frame arrives at the receiver. The receiver now has the first 4 frames in the right sequence and is able to deliver these to the higher layer protocol.

You should notice the following: if the receiver did not check for the ordered arrival of the data frames and store them in its memory, the higher layer protocol would have received the data frames in the sequence 2, 3, 4, 1. This would have resulted in an error.

Notice furthermore that the sender starts to transmit data frame number 5 as soon as the ACK for data frame 2 returns. Similarly, data frame number 6 is transmitted as soon as the ACK for data frame number 3 returns. The sender thus always has 4 frames outstanding, which could be either data frames in the direction sender-receiver or ACK frames in the direction receiver-sender.

Figure 3 illustrates a typical sequence of events under SRP. Notice that a timer is kept for every data frame, and that every data frame is acknowledged.

As indicated in section 2.1, we are not going to describe the details of sequence numbers. We simply assume that both sender and receiver number the data frames and the ACK
frames with increasing numbers in the range \((0, \infty)\). For a discussion on limited range sequence numbers you should refer to the books in the bibliography.

Let us now consider the efficiency of SRP. To do so, we will retain our assumption that the timeout value is set as in equation (1) above, i.e. to the minimum time required to successfully transmit a single data frame. We will also retain our definition of the efficiency of a protocol.

First of all, we will assume no errors occur during the transmission. In this case, the time to transmit a data frame and receive its ACK is equal to \(T_O\). During the same time interval, the sender transmits \(W\) data frames, under the assumption that \(W^*T_{\text{trans}} < TO\). In this case, the efficiency of SRP is given by

\[ \eta_{\text{SRP}} = \frac{W T_{\text{trans}}}{TO} \]

Note that we already have a condition on the relationship between the window size and the timeout interval. If the timeout interval is set to less than \(W^*T_{\text{trans}}\), then the sender is allowed to transmit another data frame under two conditions: firstly that an ACK has arrived and secondly that a window’s worth of data frames has not yet been transmitted. In any case, the sender would transmit another data frame and the efficiency would be 100\%, again assuming no transmission errors.

To compute the efficiency for a noisy link is more complicated. We will simplify the computation by assuming that \(pW \leq 10\%\), i.e. that the probability that two or more frames (data or ACK) are lost out of \(2W\) successive transmissions is negligible. Under this assumption, the expected number of times that a single frame is transmitted \((N)\) is given by

\[ E[N] = \sum_{i=1}^{\infty} ip^{i-1}(1-p) = \frac{1}{1-p} \]

where the variable \(i\) represents the counter for the number of times that the frame has been transmitted. Therefore, under the above assumptions, \(W\) frames are transmitted \(W/(1-p)\) times. Our derivation for \(E[X]\), the expected time for the successful transmission of a single frame, therefore remains as is equation (2), and we only need to modify the efficiency to be

\[ \eta_{\text{SRP}}(p) = \frac{(1-p)WT_{\text{trans}}}{TO} \]

Again, if you wish to see a more detailed derivation of these calculations, and calculations under relaxed assumptions, consult the books in the bibliography.

2.3 Go-BACK-N (GBN)
The Go-Back-N protocol is similar to SRP, in the sense that it allows multiple outstanding unacknowledged data frames. It is therefore also an improvement over the ABP. However, GBN uses a different algorithm, which ensures that the frames arrive at the receiver in order. The receiver thus does not have to provide the functions to store out-of-sequence data frames and determine in which order to deliver them to the higher layer protocol. The operation of GBN can be described as follows:

The sender has a counter $W$ indicating the window size. It transmits $W$ data frames and keeps copies of these data frames in memory – just in case the frames get corrupted and have to be re-transmitted. Furthermore, the sender starts a timer for each data frame transmitted. The receiver is obliged to acknowledge every data frame it receives without errors. Thus, upon return of an ACK, the sender transmits the next data frame, keeping a copy of it in memory for possible re-transmissions. It also discards the acknowledged data frame from memory. If the ACK for a data frame does not arrive in time – i.e. if a timeout occurs – then the sender re-transmits not just the corrupted data frame. It re-transmits the corrupted data frame and all subsequent data frames. Because of this re-transmission algorithm, the sequence of the data frames is preserved, and so they arrive in order.

As mentioned above, the receiver is obliged to acknowledge every data frame it receives without errors. Data frames received with errors are not acknowledged. Since the order of the data frames is guaranteed by design, the receiver can pass the data frames up to the higher layer protocol in order. However, it still has to discard any data frames that arrive out-of-order.

Figure 4 illustrates a sample sequence of events under GBN. Recall that all frames are acknowledged and that again a separate timer is kept for every data frame.
2. Theoretical Background

As in our previous discussions, we are not going to describe the details of sequence numbers. We simply assume that both sender and receiver number the data frames and the ACK frames with increasing numbers in the range \((0, \infty)\). Furthermore, we assume that the sender is able to identify the data frames which have to be re-transmitted by their implied sequence numbers. For a discussion on limited range sequence numbers you should refer to the books in the bibliography.

Let us now consider the efficiency of GBN. We will again retain our assumption that the timeout value \(T_O\) is set as in equation (1) above, i.e. to the minimum time required to successfully transmit a single data frame. We will also retain our definition of the efficiency of a protocol.

First of all, we will assume no errors occur during the transmission. In this case, the time to transmit a data frame and receive its ACK is equal to \(T_O\). During the same time interval, the sender transmits \(W\) data frames, under the assumption that \(W \times T_{trans} < T_O\). In this case, the efficiency of GBN is given by

\[
\eta_{GBN} = \frac{W T_{trans}}{T_O}
\]

i.e. it is the same efficiency as SRP. Note that we again have a condition on the relationship between the window size and the timeout interval. If the timeout interval is set to less than \(W \times T_{trans}\), then the sender is allowed to transmit data frame on a continuous basis, thus giving a 100% efficiency for the protocol.

To compute the efficiency for a noisy link is more complicated. Like above, we will simplify the computation by assuming that \(pW \leq 10\). Under this assumption, the expected number of transmissions required to successfully transmit one frame, \(F\), is given by

---
2. Theoretical Background

\[ E[F] = (1 - p) + (N + 1)p(1 - p) + (2N + 1)p^2(1 - p) + \cdots \]
\[ = \sum_{i=0}^{\infty} (iN + 1)p^i(1 - p) = \frac{1 - p + Np}{1 - p} \]

where the variable \( i \) represents the counter for the number of times that the frame has been transmitted and the variable \( N \) represents the number of frames that are retransmitted to retain the correct delivery sequence, \( 0 \leq N \leq W \). In the worst case, \( N = W \) and so every time, the first data frame is corrupted, leading to a re-transmission of the entire window. In this case,

Therefore, under the above assumptions, \( W \) frames are transmitted \( W(1 - p + Wp)/(1 - p) \) times. Our derivation for \( E[X] \), the expected time for the successful transmission of a single frame, therefore remains as is equation (2), and we only need to modify the efficiency to be

\[ \eta_{\text{GBN}}(p) = \frac{(1 - p)WT_{\text{trans}}}{TO(1 - p + Wp)} \]

Again, if you wish to see a more detailed derivation of these calculations, and calculations under relaxed assumptions, consult the books in the bibliography.

Figure 5 plots the efficiency for the above three protocols against the probability of a frame loss, as well as the combined plot to allow you to compare the efficiencies better. As you can see, the SRP protocol achieves the highest efficiency – but recall that it has a higher memory overhead than GBN and ABP! The window size for SRP and GBN was set to 8 frames for these plots. The transmission delay \( T_{\text{trans}} \) was set to 1 time unit and the propagation delay \( T_{\text{prop}} \) was set to 3 time units. This gives timeout values of 8 time units for all protocols. Note that the window size is set such that \( W \ast T_{\text{trans}} = TO \). Note also that for high loss probabilities, \( p \approx 0.5 \), the efficiency of ABP approaches the efficiency of GBN. However, any link with such a high frame error probability would definitely not be in operation, and hence the comparison is purely academic!
Figure 5: Plots of the theoretical values of protocol efficiency vs. frame loss probability $p$
3 Experiment 1

The first experiment in this lab is again very simple: it allows you to play with the timeout parameter to investigate the efficiency of the ABP protocol. The model for this experiment, however, is more complicated than anything you have encountered in the previous labs. It is depicted in figure 6. This model introduces two COMNET III building blocks which you have not seen before: Response sources and Application sources. Furthermore, it makes use of point-to-point links, which have been briefly described in the introduction. We will now give you a detailed description of this model and the new building blocks.

Figure 6: Experiment 1 Model Layout

3.1 Nodes

There are only two nodes in the model, as you can see from figure 6. They are labeled ‘Station A’ and ‘Station B’ respectively. The function of these two nodes is simply to act as sources and destinations for messages. Since they do not perform any additional functions, their parameter sets have been left at the default values, which basically make the nodes infinitely fast and equip them with infinite buffer spaces.

3.2 Links

Click on ‘Define / Backbone Properties / Comments’ for a short description of this model.
The two stations in the model are connected by a single point-to-point link. Such a link is used to connect two stations with each other. Since our objective in this experiment is to model the ABP, we are leaving the link parameters at their default values. These model a full-duplex link with 1 channel and a bandwidth capacity of 1536 Kbps. Furthermore, the link is assumed to be terrestrial, and so no propagation delay has been entered in the appropriate dialog box, as shown in figure 7.

Furthermore, the link is assumed to be very reliable. No frame error probability has been specified, and hence all frames transmitted over this link are guaranteed to arrive at the other end. The framing characteristics, which usually describe the segmentation characteristics of the data link protocol, have also been left at their default values. These are all zeros, indicating that any packet arriving from the COMNET III transport protocol is simply put in a frame of the same size as the packet, and no overhead is added. This effectively does not model a COMNET III datalink protocol.

### 3.3 Traffic Sources

The model contains a number of traffic sources. Station A ‘owns’ 4 application sources. Station B ‘owns’ a single response source. Application sources are used in COMNET III to model software applications at a higher level of detail. Recall that the usual message source that you have seen in the previous labs only determines the arrival time of the messages, the size, the destination and the transport protocol used. There is no possibility...
to describe in detail how the packets are generated or to model more complicated event sequences. To overcome this limitation is the function of an application source.

### 3.3.1 Application Sources

An application source is basically a sequence of very basic commands known by COMNET III. The sequence of commands defines the application. Of course, your immediate question is going to be: what are commands, and which commands does COMNET III know? The following list provides you with a brief description of the principal commands, and thus gives you an answer to this question.

<table>
<thead>
<tr>
<th>Command Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transport Command</td>
<td>the transport command acts like a message source. If it is encountered in the list of commands describing the application, a message is generated and sent to the specified destination using a specified transport protocol. You will see that the parameters for this command are almost identical to the parameters of a message source – obvious, since they perform the same function!</td>
</tr>
<tr>
<td>Wait Command</td>
<td>the wait command allows you to model delays. Some applications are interrupted for a specific period of time. They are only resumed after an event happens. Consider for example a timeout: a timeout is a delay that has to occur before the next event happens. The timeout is thus modeled as a wait command.</td>
</tr>
<tr>
<td>Processing Command</td>
<td>the processing command allows you to model the use of the CPU inside a COMNET III processing node. It is like a delay for processing. An example would be some number-crunching application which has to perform a CPU intensive operation. This operation would be modeled as a processing command.</td>
</tr>
<tr>
<td>Answer Message Command</td>
<td>an answer message command is like a transport command, in that it generates network traffic. However, as the name implies, it only answers to incoming messages. This is typically the case for client-server interactions, where the server only responds to a client’s message. The answer message command has parameters describing the size of the message and the transport protocol with which the message is returned to the sender.</td>
</tr>
<tr>
<td>Write-File Command</td>
<td>COMNET III allows you to model memory, such as a hard disk. The write-file command models a write operation to this memory. Its parameters are: the</td>
</tr>
</tbody>
</table>
Read-File Command:

the read-file command is the opposite to the write file command. Instead of writing to memory, it models a read operation from memory. Its parameters are again the number of bytes to read, which file to read and how to modify the file.

This list is not exhaustive, i.e. there are a number of less important commands that COMNET III knows. The application is thus made up of a list of such commands. This list can have any number of commands. Also, any command type may appear as often as required in the command list.

Example: As an example of an application source, let us describe the application called ‘SendPacket’ in our model, consisting of 3 commands. A transport command followed by a wait command followed by another transport command. The first transport command is also called ‘SendPacket’. It is responsible for sending a single packet to Station B using the default transport protocol called ‘Generic’. After this packet has been transmitted, the wait command is executed. It is called ‘timeout’ and specifies the following wait: wait until a message called ‘ACK’ arrives at the node where the application is executed, but wait at most 0.3 seconds. So if the ACK does not arrive in 0.3 seconds, the application is terminated and the third command will not be executed. If, on the other hand, the ACK arrives before 0.3 seconds, the third command is executed. This last command sends a dummy message to Station A itself. The message takes the name ‘SendNextPkt’ and basically triggers the next application source, called ‘SendNextPkt’ in the model.

The sequence of commands describing this application source is shown in figure 8.
Figure 8: Command sequence describing the application ‘SendPacket’

When is this application scheduled for execution at node ‘Station S’? This question is answered by inspecting the parameters under the tab ‘Scheduling’, which is hidden in figure 8. Figure 9 reveals these parameters, and as you can see, the scheduling corresponds to the scheduling of a message source. You should know this mechanism by now, so we will not repeat the details again.
Figure 9: Scheduling of an application source

Let us now briefly describe the other 4 application sources attached to ‘Station A’. The application source ‘SendNextPkt’ executes exactly the same command sequence as the source ‘SendPacket’. The only difference is that this source is scheduled by the trigger ‘SendNextPkt’, whereas the application source ‘SendPacket’ is scheduled to arrive only once in the simulation, namely at time 1.0 seconds. Since the third command in the command list (c.f. figure 8) releases such a trigger, application ‘SendNextPkt’ basically triggers itself upon successful completion of the three commands. These commands generate a continuous stream of packets destined for ‘Station B’, until the simulation time has elapsed.

The two applications ‘ReTransmitPkt1’ and ‘ReTransmitPkt2’ are responsible for handling the re-transmission of data frames upon link failures or elapsed timeouts. When the ACK does not arrive in time, the command ‘timeout’ of applications ‘SendPacket’ and ‘SendNextPkt’ terminates. Associated with this premature termination is another trigger, which you can see by clicking on the tab ‘Advanced’ and the button ‘Triggered Sources’. For application source ‘SendPacket’, the application source ‘ReTransmitPkt1’ is in the triggered-sources list. This indicates to the model that upon premature termination of the wait command, the application source ‘ReTransmitPkt1’ is executed. If you inspect its
command list, you will see that it again executes the same sequence of commands. Similarly for the application source ‘ReTransmitPkt2’.

To summarize the relationships between these application sources: Source ‘SendPacket’ is started at time 1.0 and triggers the source ‘SendNextPkt’ if the transmission is successful. Otherwise, it triggers the application ‘ReTransmitPkt1’. Source ‘SendNextPkt’ is being triggered by source ‘SendPacket’ or by the sources ‘ReTransmitPkt1’ and ‘ReTransmitPkt2’. It executes the commands and triggers itself. This source represents a successful packet transmission. Source ‘ReTransmitPkt1’ is being triggered by the wait command in the command sequence upon premature termination (i.e. the ACK does not arrive in time). It also executes the same command sequence. If it terminates successfully, it triggers source ‘SendNextPkt’. If it terminates prematurely, it triggers source ‘ReTransmitPkt2’. Source ‘ReTransmitPkt2’ is triggered by the source ‘ReTransmitPkt1’ and executes the same command sequence. If it terminates successfully, it again triggers ‘SendNextPkt’, otherwise it triggers ‘ReTransmitPkt1’.

Figure 10 represents the conceptual relationships between these fours application sources connected to ‘Station A’. The arrows labeled ‘Success’ are followed after the successful transmission of a data frame. The arrows labeled ‘Failure’ are followed if the ACK does not arrive in time, i.e. if a timeout occurs.

---

2 Source ‘ReTransmitPkt1’ cannot schedule itself and be scheduled by ‘SendNextPkt’ or ‘SendPacket’. This is the reason for having ‘ReTransmitPkt2’, so effectively the two re-transmission sources trigger each other circularly if the transmission is unsuccessful.
The relationships between those sources are certainly hard to understand at first, but if you execute the simulation model and play around with the tracing facility, you should understand how these sources interact.

### 3.3.2 Response Sources

Finally, we come to the response source called ‘ACK’ in the model. A response source can only be scheduled by an incoming message. It then returns a message to the original sender of the incoming message by which it was triggered. Response sources are thus a standard COMNET III building block to model client-server type interactions between messages.

If you look at the response sources’ parameters, you will find that the tab ‘Scheduling’ only allows you the type ‘Received Messages’. In our model, the message which schedules this source is called ‘DataPacket’, which is the message text sent by the command ‘SendPacket’ (note: this command is part of all application sources. Not to be confused with the application source ‘SendPacket’).

All other tabs are identical to a message source, so you should be familiar with them by now. In particular, our response source sends a 40-byte message back to ‘Station A’,
where all the incoming messages originate. The text ‘ACK’ is transmitted by this message, which in turn determines how the command ‘timeout’ in the command sequence behaves. If this acknowledgement message arrives before the 0.3 seconds, the next data packet is sent by the application source ‘SendNextPkt’. Otherwise the re-transmission sources ‘ReTransmitPkt1’ or ‘ReTransmitPkt2’ are triggered.

Figure 11 shows the message size parameters for this source. You should explore the other tabs in the dialog box.

![Figure 11: Message size parameters for the response source ‘ACK’](image)

Figure 11: Message size parameters for the response source ‘ACK’
To Do: Study the model in detail. You should understand how the sources interact to achieve the desired event sequence for the ABP. Complete the last column of the table on the next page to summarize the model parameters.

Run the simulation for 60 seconds with the given parameters. How many data packets arrive at the destination? How many of these data packets are re-transmissions? What value should the timeout parameter take?

Change the timeout parameter to the value you have just determined and re-run the simulation. How many data packets arrive at the destination now? How many of these packets are re-transmissions?

If you still have re-transmissions of data packets in your simulation, increase the timeout value in steps of 0.05 seconds. At what value do you experience no re-transmissions in your simulation?

Provide a brief (max 1 page, type-written) report, commenting on the main characteristics of the results.

*Hint:* You can find out whether you have re-transmission in the simulation as follows. Inspect the statistical results and focus on the report ‘Global Transport + Answer Commands: Message Delivered’. If you see a positive number under the statistic ‘Messages Assembled’ for the sources ‘ReTransmmitPkt1’ and / or ‘ReTransmitPkt2’, then these sources have been active and thus re-transmissions occurred in your simulation. If the respective values are 0, then no re-transmissions occurred.
<table>
<thead>
<tr>
<th>Building Block Name</th>
<th>Type</th>
<th>Tab</th>
<th>Parameter Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Station A</td>
<td>Processing Node</td>
<td></td>
<td>Parameters</td>
<td></td>
</tr>
<tr>
<td>Station B</td>
<td>Processing Node</td>
<td></td>
<td>Parameters</td>
<td></td>
</tr>
<tr>
<td>Link A-B</td>
<td>Point-to-Point Link</td>
<td></td>
<td>Parameters</td>
<td></td>
</tr>
<tr>
<td>SendPacket</td>
<td>Application Source</td>
<td>Scheduling</td>
<td>First Arrival</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Advanced</td>
<td>Triggered Sources</td>
<td></td>
</tr>
<tr>
<td>SendNextPkt</td>
<td>Application Source</td>
<td>Scheduling</td>
<td>Received Message</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Advanced</td>
<td>Triggered Sources</td>
<td></td>
</tr>
<tr>
<td>ReTransmitPkt1</td>
<td>Application Source</td>
<td>Scheduling</td>
<td>Triggering Event</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Advanced</td>
<td>Triggered Sources</td>
<td></td>
</tr>
<tr>
<td>ACK</td>
<td>Response Source</td>
<td>Scheduling</td>
<td>Received Message</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Messages</td>
<td>Prob. Dist.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Text</td>
<td>Message Text</td>
<td></td>
</tr>
<tr>
<td>SendPacket</td>
<td>Global Command</td>
<td>Messages</td>
<td>Prob. Dist.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Destination</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Text</td>
<td>Message Text</td>
<td></td>
</tr>
<tr>
<td>SendNextPkt</td>
<td>Global Command</td>
<td>Messages</td>
<td>Prob. Dist.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Destination</td>
<td>Random List</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Text</td>
<td>Message Text</td>
<td></td>
</tr>
<tr>
<td>Timeout</td>
<td>Global Command</td>
<td>—</td>
<td>Receive message: Required text</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>—</td>
<td>Stop waiting after…</td>
<td></td>
</tr>
</tbody>
</table>
4 EXPERIMENT 2

The previous example focused on ABP and the determination of the optimal timeout parameter. In this experiment, we will look at the effect of noisy transmission channels and compare the efficiency of the three protocols outlined in the theoretical part. To this end, we will discuss the details of COMNET III transport protocols.

The base model is depicted in figure 12. It again consists of two processing nodes, called ‘Station A’ and ‘Station B’ respectively. Like in the previous models, these nodes only function as source and destination of network traffic and will thus not be discussed further below. The description of the remaining building blocks follows.

![Experiment 2: Comparison of ABP, GBN and SRP under a noisy transmission channel](image)

Figure 12: Experiment 2 Model Layout

4.1 LINKS

The link between stations A and B is modeled here as a COMNET III point-to-point link. You have encountered this link type before in the last experiment, as described under section 3.2. Here, we model the link between ‘Station A’ and ‘Station B’ as a 500 Kbps link. You should verify this by double-clicking on the point-to-point link, clicking on the ‘..’-button next to the field ‘DEFAULT’, then clicking on the ‘Edit’-button and inspecting the tab labeled ‘Physical’.

The framing characteristics in this experiment are left at their default values of zero. This indicates that we are not modeling a data-link protocol. So effectively, the model only has
a single protocol layer: the message is divided into packets (end-to-end protocol data units, PDUs, in COMNET III), as will be explained in the subsequent sections. Note that for our purposes it is not important whether we are modeling layer 2 or layer 3. Our model only consists of a single link between two stations, and the messages are only segmented once.

The important parameter set for this experiment is located behind the tab ‘Link Specifics’, which you should find when you double-click on the point-to-point link. If you click on the button ‘More..’, the following dialog box will appear:

![Figure 13: Link Specific parameters](image)

This dialog box model different propagation delays and link errors. The link is specified to have a propagation delay of 24 ms and a positive frame error rate. The probability for such a frame error in figure 13 is set to 0.3, but this is only an example. You will be required to change this parameter to different values in order to examine the efficiency of the different ARQ-protocols under different error probabilities.

### 4.2 Message Sources

In this model, there is only one message source: ‘PacketStream’. It models a continuous stream of packets, since the parameter ‘Msg size calc.’ is set to 10,000 packets. These packets could never be transmitted during the set simulation time of 60 seconds.

The message is set to arrive at the start of the simulation, i.e. at time 0.0. The destination is set to ‘Random neighbor’, which can only be ‘Station B’ in this model. The main novelty in this source is that we do not use the ‘Generic’ transport protocol, but that we use our own transport protocols, defined as ‘ABP’, ‘GBN’ and ‘SRP’. You can see that these transport protocols have been defined for you by clicking on the tab ‘Packets’ and then on the ‘▼’-arrow. The following dialog box appears:
Figure 14: List of transport protocols defined for this model

If you click on the ‘..’-button next to the field ‘Protocol’, you will be presented with the standard two lists: the permanent parameter sets in the COMNET III database, and the parameter sets currently defined in the model. Selecting ‘ABP’ from the right list and clicking on ‘Edit’ will show you the following dialog box:
Figure 15: Protocol parameters for ABP, SRP and GBN

For all three protocols (ABP, GBN and SRP), the packet size is set to 1000 bytes and the size of the acknowledgement is also set to 1000 bytes. The protocols differ in how their error control and flow control parameters are set, defined under the respective tabs.

The ABP models a timeout value of 80 ms (can you verify this using a timing diagram?). The flow control parameters are set to ‘Sliding window’ with a window size of 1. This flow control algorithm automatically sends an ACK for every packet received at the destination. With a window size of 1, we get the desired effect of only one packet being in transmission at any time – either the data packet or the corresponding ACK.

The SRP algorithm is also modeled using the ‘Sliding Window’ flow control method of COMNET III. However, in this case, the window size is set to 5, as depicted in figure 16. Thus at any point in time, 5 packets are in transmission. These may be data packets flowing from station A to station B. They may be ACKs flowing from station B to station A. They may also be less than 5 data packets flowing from station A to station B, and the remaining ACKs flowing from station B to station A such that the total number of packets in transmission is 5. Every time a data packet arrives at the receiver, an ACK is generated. Every time an ACK arrives at the sender, the next data packet is transmitted. If at any point in time the data packet of its ACK becomes corrupted, then the sender will time out and only the affected data packet will be re-transmitted after the timeout.
Figure 16: SRP Parameters

The GBN algorithm is modeled in COMNET III again as a sliding window, but now we select the enhanced version of this algorithm. This enables an additional parameter, labeled ‘Parameters’. In figure 16 you can see that this parameter is not active – it is dimmed out. Double-clicking on the ‘..’-button next to this parameter reveals the settings for the GBN parameters (after you clicked on the ‘Edit’-button). The following dialog box should appear on your screen:

Figure 17: Enhanced Sliding Window parameters for GBN
As you can see from figure 17, the error control method is set to ‘Go Back N’. COMNET III automatically associates the value of ‘N’ with the window size. The other parameters in this set are not important. They are used to model more complicated error control algorithms, as used by TCP/IP or other commercially used protocols.

![Sliding Window Flow Control](image)

Figure 18: Adaptive timeout parameters

You should verify that the retransmission timeout under the tab ‘Adaptive timeout’ is set to ‘Constant’, as shown in figure 18. In this case, the timeout interval will be constant, and not be changed by adaptively by COMNET III. In particular, the timeout interval will be determined by the parameter ‘Re-transmission time (ms)’ which can be found under the tab ‘Error control’ under the main protocol parameters. This parameter set is shown in figure 19. Make sure that you do not confuse the two tabs labeled ‘Error control’! The main error control tab is found under the protocol parameters and looks like in figure 19. It determines the timeout interval. The other error control tab looks like in figure 17, and it overrides the error control algorithm.

As you can see from figure 19, we have set the timeout interval to 128 ms. This corresponds to the maximum round-trip delay that a packet may occur in this model. Note that the transmission and propagation delays for the packet amount to 80 ms. However, with a window size of 5, a data packet may also incur a buffering delay. More than one packet may be created at the sender, for example, and thus some packets have to wait until the other data packets are transmitted. For this reason, the timeout interval is set to 128 ms, thus accounting for an average queuing delay of 48 ms, or 3 packets.
A further note on the enhanced sliding window algorithm. COMNET III implements this algorithm using inclusive ACKs. This means the following: Assume that packet number \( x \) successfully arrives at the receiver, but that its acknowledgement, \( \text{ACK}_x \), becomes corrupted. The subsequent packet, \( x+1 \), also arrives successfully at the receiver, and thus generates another acknowledgement, \( \text{ACK}_{x+1} \). If \( \text{ACK}_{x+1} \) manages to get through to the sender, then the sender will realize that it is missing \( \text{ACK}_x \). However, it will assume that \( \text{ACK}_{x+1} \) indicates that ‘all packets up to and including data packet \( x+1 \) been successfully received’. This is a matter of convention, and thus perfectly legal. However, it means that the timer for packet \( x \) will not time out, and thus that packet \( x \) will not be re-transmitted, even though its acknowledgement has been corrupted. This algorithm thus reduces the number of re-transmissions.

Table 2 summarizes the different settings for these three transport protocols:

<table>
<thead>
<tr>
<th>Transport Protocol</th>
<th>Parameter Set</th>
<th>Tab</th>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABP</td>
<td>Protocol Parameters</td>
<td>Packets</td>
<td>Data Bytes</td>
<td>1000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Ack Size</td>
<td>1000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Error Control</td>
<td>Retransmission time (ms)</td>
<td>80</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Flow Control</td>
<td>Window Packets</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Flow Control Window</td>
<td>Sliding Window</td>
</tr>
</tbody>
</table>
4. Experiment 2

<table>
<thead>
<tr>
<th>SRP</th>
<th>Protocol Parameters</th>
<th>Packets</th>
<th>Data Bytes</th>
<th>1000</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Ack Size</td>
<td>1000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Error Control</td>
<td>Retransmission time (ms)</td>
<td>80</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Flow Control</td>
<td>Window Packets</td>
<td>5</td>
<td>Sliding Window</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Flow Control Window</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>GBN</th>
<th>Protocol Parameters</th>
<th>Packets</th>
<th>Data Bytes</th>
<th>1000</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Ack Size</td>
<td>1000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Error Control</td>
<td>Retransmission time (ms)</td>
<td>128</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Flow Control</td>
<td>Window Packets</td>
<td>5</td>
<td>Sliding Window (Enhanced)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Flow Control Window</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Parameters</td>
<td>GBN</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GBN Sliding Window (Enhanced)</td>
<td>Error Control</td>
<td>Error Control Method</td>
<td>Go Back N</td>
</tr>
<tr>
<td></td>
<td>Adaptive Timeout</td>
<td>Retransmission Timeout</td>
<td>Constant</td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Summary of the protocol parameters for experiment 2

To Do: For each of the protocols discussed above (ABP, SRP and GBN), simulate the performance of the protocol under the following link error probabilities: 0.00, 0.02, 0.04, 0.06, 0.08, 0.10, 0.20 and 0.30.

Do the results match the theoretical results you would expect from section 2? If not, why?

Are the assumptions required for the efficiency formulae derived in section 2 valid in your simulation?

Submit a brief report (max. 2 pages, type-written), commenting on the main results of this experiment.
5. **Further Experiments**

Re-consider Experiment 1 with a timeout parameter of 0.33355 seconds. We now introduce a processing delay at Station B to simulate the following scenario: whenever a data packet arrives at Station B, a probability processing delay is encountered due to the overload of processes at Station B. With probability 0.9, no processing delay is incurred. With probability 0.1, the delay is exponentially distributed with mean 1 second.

You should enter this delay by following these steps:
- Double-click on the response source ‘ACK’
- Identify the parameter ‘Rec message delay (sec)’
- Click on the ‘▼’-button next to this parameter
- Select the mixed distribution ‘Mix(0.995,0,Exp,1.0,Exp,100.0)’
- Click on the ‘.’-button
- Enter the values as in the following dialog box

![Mixed Distribution Dialog Box](image)

Figure 20: Dialog box to specify a mixed distribution

- Click on the ‘OK’-button in all open dialog boxes.

**To Do:** Simulate the modified model for 60 seconds. What percentage of data packets have to be re-transmitted?

Summarize your findings in a brief report (max. 1 page, type-written).
6 Deliverables

6 Deliverables:

Experiment 1:
• Print-out the reports generated by the simulation
• Provide a screen-shot of your model
• Table with parameters filled out
• Report

Experiment 2:
• Print-out of the reports
• Provide a screen-shot of your model
• Report

Experiment 3:
• Print-out of the reports
• Provide a screen-shot of your model
• Report

Note: All reports have to be in letter format. All interpretations have to be typed.

Marking Scheme:

Interpretations: Experiment 1 - 35%
Experiment 2 - 35%
Experiment 3 - 15%

Format: 15% (this includes completeness, clarity, form)
END OF LABORATORY 1