# Chapter 3: ARQ Protocols

— Reference 1, sections 2.1, 2.4, 2.6-2.9

## 1. ARQ retransmission strategies

Automatic Repeat reQuest - The receiver detects a transmission unit ( a frame or a packet or a message ) with an error and asks the transmitter to retransmit that unit.

Problems:

1. Correctness: Does the receiver accept every transmission unit once and only once, and place them in the same order as the source.
2. Efficiency: How many extra transmissions are made and how much time is spent in an idle state

Channel model

In a layered architecture, the channel model is the service provided by the lower layers of the architecture to the layer with the protocol

The services provided by the protocol are provided to the layers above the protocol

— variable delay

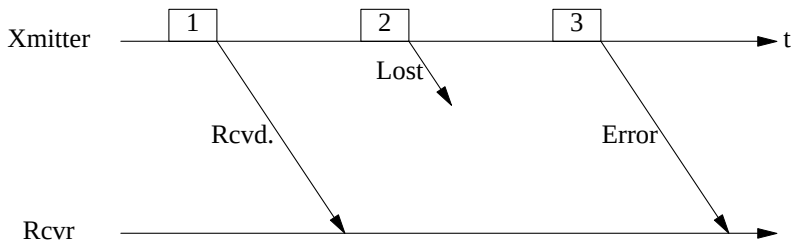
— packets are received in the same order that they are transmitted.

- This is not the case on the Internet
- A lower layer of the architecture may re-order the packets when they are received out of order, or may simply discard a packet if its sequence number is less than that of a previously received packet.
- Exactly how the sequencing is obtained, depends on the specific protocol that is used at the lower layer.
- However, the guarantees are obtained does not concern the protocol at this layer.

— On a half-duplex channel, a lower layer coordinates transmissions between the source and receiver so that only one transmits at a time.

— 2 types of losses

1. Unit 2 never arrives at the receiver
2. Unit 4 arrives, but an error is detected.



## 2. Stop and wait ARQ

— Full duplex channel - the source and receiver can transmit messages to each other at the same time.

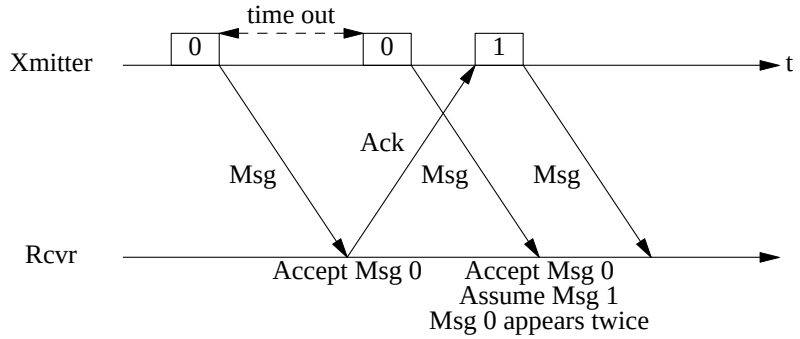
— The transmitter periodically transmits a message until it receives an indication that the receiver has correctly received the message, then starts transmitting the next message.

— Indications from the receiver

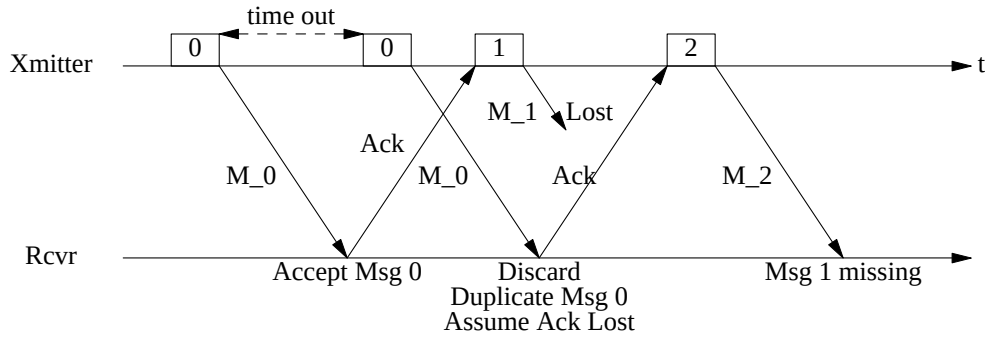
- ACK - Positive acknowledgement.
  Message indicating that the receiver has correctly received a packet
- Timeout - no response from receiver
  Assume that the receiver did not receive the message
- NACK - Negative acknowledgement
  Message indicating that the receiver has received a message with a transmission error.
  - Can reduce the time until a message is retransmitted, since we do not have to wait for a timeout.
  - Cannot detect lost messages
  - We will not use NACK's in this part of the work.
  - Nack's are used in some versions of the go-back-N and selective repeat protocols to reduce delays,
    And, in RMP - the reliable multicast protocol - we will show how NACK's can reduce the number of control messages.

— Importance of numbering

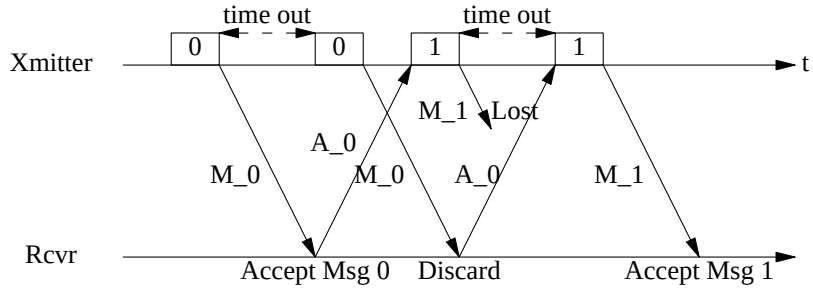
- Unnumbered messages
  — round trip delay > the time between retransmissions
  — Receiver does not know if it received message 0 or 1



- Unnumbered ACKS
  - round trip delay > the time between retransmissions
  - Transmitter does not know if ACK is a repeat for message 0 or for message 1




- Story of HDLC, DCCP-1 -- Unnumbered ACK's
  - — Used for 10 years before it was realized that the protocol had this error.
  - — Originally designed for IBM 2780 remote card reader
  - — 2 cards stuck together much more often than error occurred and the effect was the same - a card is missing
    Programmers started numbering the cards so that they could distinguish a missing card from a programming error
  - — Need for formal methods for protocol verification
  - — Lead to probabilistic verification
- Operation with Numbered messages and acks
  - — In reference [1] the ack contains the number of the next message expected by the receiver.
  - — Here the ack has the number of the message being acknowledged.
  - — The two schemes are equivalent



## 2.1 Correctness of stop and wait

— **Safety:** The algorithm never produces incorrect result.

In this case:

- The receiver never accepts a packet out of correct order.
- The receiver only accepts a packet once

Proof by rule of operation:

- Initially, the receiver awaits packet ’0’ and only releases that packet.
- At each step the receiver waits for packet RN and only releases that packet.

— **Liveness:** The algorithm does not dead lock.

In this case

- The receiver eventually receives packet i, and will then accept packet i+1
- The source will eventually receive ack i, and will then send packet i+1

Proof :

- As long as the receiver is waiting for packet i, it hasn’t sent ack i, and the source will send packet i
  The source periodically sends packet i until it receives ack i
  As long as the probability of receiving a packet is > 0, the receiver eventually receives packet i, and sends ack i.
  Once the receiver receives packet i, it expects packet i+1, but continues to send ack i every time it receives packet i.
- As long as the source is waiting for ack i, the receiver will send ack i
  The source periodically sends packet i until it receives ack i
  Eventually, the receiver receives packet i and sends ack i
  Eventually, the source will receive ack i and start sending packet i+1

## 2.2 How large do the packet numbers have to be?

— We can't have an infinite packet number.
*We send the packet number modulo N.*

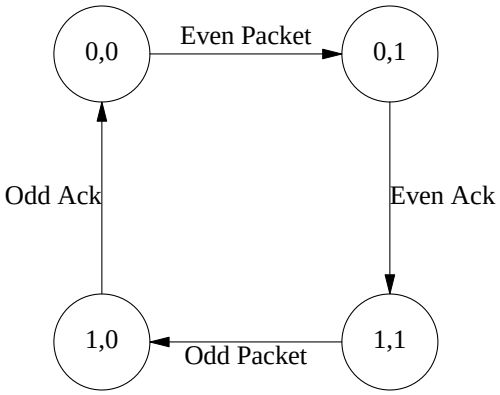
— When packets are received in the same order that they are transmitted,
*Numbering the packets and acknowledgements Mod 2 is sufficient.*

— Proof:

1. Show that knowing packet *i mod* 2 at the receiver is sufficient to identify packet i
    a. When the receiver is expecting packet i, the source will not transmit packets i+1 or greater
        - The source does not send packet i+1 until after ack i is received
        - When the receiver sends ack i, it is no longer expecting packet i
        - Therefore, the source does not send a larger packet number than the receiver expects.
    b. When the receiver is expecting packet i, the source will not send a packet less than packet i-1
        - When the receiver expects packet i, the source has sent packet i-1.
        - When the source sends packet i-1, it has received ack i-1, and will not resend packet i-2.
    c. Therefore, when the receiver expects packet i, it cannot receive $j < i - 1$ or $j > i$
    d. i mod 2 is sufficient to distinguish between packet i and packet i-1
    one is odd, and the other is even.
2. Show that knowing ack *i mod* 2 at the source is sufficient to identify acknowledgement i that is received
    a. When the source is waiting for acknowledgement i, ack $j > i$ cannot be transmitted,
        - The source does not send packet i+1 until it receives ack i.
        - The receiver does not send ack i+1 until the source sends the packet i+1
    b. When the source is expecting ack i, it does not receive ack $j < i - 1$
        - When the source expects ack i, it has received ack i-1
        - The receiver does not transmit ack i-2 once it has transmitted ack i-1
        - The acks are received in the order that they are transmitted
        - Therefore, ack i-2 will not be received.
    c. Therefore, when the source expects ack i, it cannot receive ack $j < i - 1$ or $j > i$
    d. i mod 2 is sufficient to distinguish between ack i and ack i-1

— Operation

- 4 state machine:
- State = (Source transmitting, Receiver waiting for)
- This is an incomplete FSM, as will be shown in the next section.



## 2.3 Channel Efficiency

efficiency = $\eta$ = fraction of the time channel is used to transfer bits

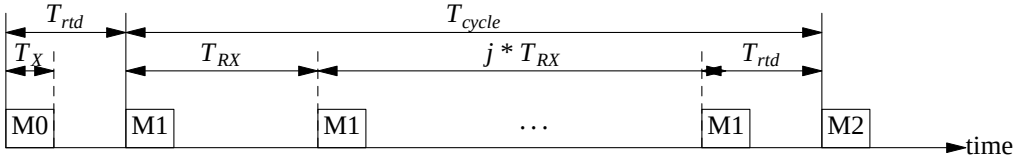
$T_X$ = time to transmit a packet.

$T_{rtd}$ = round trip delay
the time from the start of transmission until an acknowledgement is received

$T_{RX}$ = timeout between retransmissions

$T_{cycle}$ time to successfully transmit a packet

$P_L$ = probability that message or ack is lost




$$\begin{aligned}
\bar{T}_{cycle} &= T_{rtd} + T_{RX}P_L(1-P_L) + 2T_{RX}P_L^2(1-P_L) + \cdots \\
&= T_{rtd} + T_{RX}(1-P_L)\sum_{i=1}^{\infty} i * P_L^i \\
&= T_{rtd} + T_{RX}(1-P_L)P_L\,\frac{d}{dP_L}\left[\sum_{i=0}^{\infty} P_L^i\right] \\
&= T_{rtd} + T_{RX}(1-P_L)P_L\,\frac{d}{dP_L}\left[\frac{1}{1-P_L}\right] \\
&= T_{rtd} + T_{RX}\,\frac{P_L}{1-P_L}
\end{aligned}$$

$$\eta = \frac{T_X}{T_{cycle}} = \frac{T_X}{T_{rtd} + T_{RX}\,\dfrac{P_L}{1-P_L}}$$

$$\lim_{P_L \to 0} \eta = \frac{T_X}{T_{rtd}}$$

It is important to learn how to analyze the performance of simple protocols using infinte series in order to appreciate the power of Z-Transforms to analyze more complicated systems. This will be discussed in a later section.

## 3. Go back N ARQ

Objectives:

1. Allow multiple packets in transmission so that line isn't idle in networks with long delays
2. Accept all packets in the proper order

Implementation:

— packets are numbered

— The xmitter may xmit up to $N$ unacknowledged packets. $N$ is the "window size".

- If the window size is greater than the round trip delay, the channel is kept busy continuously
- If the window size is less than the round trip delay there are bursts of transmissions - up to the window size - followed by idle intervals

— The receiver sends an ACK with the number of the last packet in the sequence that it has received.

— When the xmitter receives an ack for packet i, it advances its window to i+N

- If the xmitter receives an ack for a packet with a higher number than it expects, it knows that the receiver must have received that packet and that the ack for that packet was lost
- There are several chances to receive an ack before the protocol times out
  We don't have to retransmit every missing ack
- When the window size is significantly larger than the RTD, the probability of retransmitting a packet is reduced from the probability that the message or the ack is lost - in stop and wait - to the probability that the message is lost
- $P_{L,gbn} = P_E$ instead of $P_{L,sw} = 1-(1-P_E)^2 \approx 2P_E$, where $P_E$ is the probability of error on either the forward or reverse channel
- The probability of retransmitting is cut in half for small error rates

— When the receiver receives a packet with a larger number than it expects, it discards that packet.

— If the xmitter does not receive an acknowledgement for packet $i$ by a time-out, it goes back and retransmits that packet and all of the subsequent packets that have also been unacknowledged.

### 3.1 Lower Bound on the Efficiency of GBN

- The window size is large enough to keep the channel busy continuously
- The probability that a sequence of i packets are received correctly, before an error is $X_i = P_L(1-P_L)^i$
- The average length of a correct sequence is

$$\begin{aligned}\bar{X} &= \sum_{i=0}^{\infty} iP_L(1-P_L)^i \\ &= P_L(1-P_L)\frac{d}{d(1-P_L)}\sum_{i=0}^{\infty}(1-P_L)^i \\ &= P_L(1-P_L)\frac{d}{d(1-P_L)}\left[\frac{1}{1-(1-P_L)}\right] \\ &= \frac{1-P_L}{P_L}\end{aligned}$$

- Following each correct sequence, there are at most N packets, the window size, that will be retransmitted.

- The efficiency is $\eta = \frac{\bar{X}}{\bar{X} + N} = \frac{1 - P_L}{1 + P_L(N-1)}$

## 3.2 Correctness of go-back-N

Safety: packets are received in the order transmitted. The receiver only releases the packets to the upper layer in the correct order.

Liveness: Eventually a packet is received, and eventually the ack for that packet is returned. Therefore, it continues to make progress.

## 3.3 How large a number field do we have to use?

The correctness is maintained as long as the modulus m > N

- At the Xmitter: I-1 = last acknowledgement number received from the receiver.
  - The xmitter will only transmit packets with numbers X where $I \le X \le I + N - 1$.
- Therefore, the receiver can only transmit acknowledgement numbers R between $I - 1 \le R \le I + N - 1$
  The receiver may transmit ack(i-1) if it receives another copy of packet(i-1), but once it receives packet(i-1) and sends ack(i-1) and will never send ack(i-2) again.
- And $|R - X| \le N$.

Operation:

1. Instead of putting X in the packet, the transmitter puts $x = X \ mod \ m$, $X = x + km$.
   - The rule at the receiver is that it accepts the packet if $X = R$, but it only knows $x$.
   - The receiver constructs $r = R \ mod \ m$, so that $R = r + lm$.
   - If $x - r = 0$, $X - R = (k - l)m$.

     Since $|R - X| \le N$, and $m > N$, $k = l$ and $X = R$.
   - Therefore, the receiver can correctly receive the packet if $x = r$, even though it doesn't know $X$.

2. Instead of putting $R$ in the ack, the receiver puts $r = R \ mod \ m$.

   - The rule at the transmitter is that it increases $X$ to $R + 1$ if $R \geq X$
   - The transmitter doesn't know $R$, but only knows $r$.
   - The transmitter calculates $x = X \ mod \ m$, where $X$ is the number of the next ack that the transmitter expects to receive.
   - The transmitter checks $x - r$, where $x = X \ mod \ m$
   - If $x - r = 0$, then $R - X = (l - j)m = 0$.

     since $|R - X| \leq N$ and $m > N$.

     — The transmitter increases the number of the next ack it expects to $X' = X + 1$

     — The transmitter sends packets from $X'$ to $X' + N - 1$.

   - If $x - r \neq 0$,

     A. If $|x - r| < N$, $R > X$, and the ack is for message $X + |x - r|$

        - The next ack that the source expects is $X' = X + |x - r| + 1$, and
        - The window is from $X'$ to $X' + N - 1$.

     B. If $|x - r| = N$

        — This cannot be an ack for $X + N$, because the source only transmits up to packet $X + N - 1$.

        — Therefore, $R < X$, and the acknowledgement is a repeat of the acknowledgement for packet $X - 1$.

     C. $|x - r|$ can't be $> N$, even if $m > N$, because the source can not transmitted a packet $> X + N - 1$, and the receiver will not ack a packet $< X - 1$ after acking $X - 1$.

## 4. Selective repeat ARQ

— The objective of selective repeat is to only retransmit the packets that are in error, then re-order the packets at the receiver.

— Used for TCP

— Operation: There are many ways to implement selective repeat.

- The objective is to create several independent stop and wait protocols, that can keep the channel busy
- Each stop-and-wait protocol is responsible for delivering its own messages, then the receiver re-orders the messages.
  - — When a stop-and-wait protocol successfully transmits its message, it get the next message from the queue of messages that are waiting to be transmitted.
  - — If one message is lost and must be retransmitted many times, the messages being transmitted by the other stop-and-wait protocols may get far ahead of that message.
  - — In order to control how far ahead the other protocols may get, we assign a window size W that is the maximum distance that we allow the other protocols to get before waiting.
- In order to reduce the number of messages that are retransmitted because of lost acknowledgements, Each acknowledgement includes a list of the lower numbered messages that are missing.
  - — In effect, when an acknowledgement is lost, and the message is not in the list of missing messages, we can assume that the acknowledgment was transmitted.
  - — When the transmitter learns that an lower numbered message is missing, if that message was transmitted before the acknowledged message it is retransmitted without waiting for a time-out
  - — The lower numbered message may have been transmitted after the acknowledged message if it being retransmitted by its ARQ protocol.

— One way to implement selective repeat is to:

- At the transmitter,
  - — All packets with numbers less than $X$ have been received.
  - — The transmitter sends packets from $X$ to $X + W - 1$
    Window size W
  - — Each transmitted packet has its own timer, and is retransmitted when the timer expires.
  - — A packet is also retransmitted, and the timer is reset, if a NAK is received.
- At the receiver.
  - — All packets less than $R$ have been received.
  - — The receiver is willing to receive packets from $R$ to $R + W - 1$.
  - — Unlike go back N, when a packet is received, the window at the receiver may be advanced by more than one message.
    If $R$ is received and packets $R + 1, R + 2, \ldots, R + i$ have already been received and acknowledged, then the window is advanced to $R + i + 1$
  - — When a receiver acknowledges a packet $J$, if the packet is $J > R$, it also sends a negative acknowledgment for the packets between $R$ to $J$ that are missing.

- When acks are received at the transmitter for $J$:
  - If $J < X$, the ack has already been received and is discarded.
  - If $J = X$, $X$ is advanced to $X + 1$
  - If $J > X$,
    - and there are no missing messages at the receiver, then $X$ is advanced to $J + 1$, since the receiver has all of the packets prior to $J$.
    - If there are missing packets, and those packets were last transmitted prior to $X$, those packets are retransmitted before the next transmit packet.
    - $X$ is set to $J_{missing,min}$
    - The timers for all of the packets less than $J$ that are not missing are turned off, since these packets have been received
  - If an acknowledgement is lost for message $i$, but an acknowledgement for $k > i$ is received, message $i$ is noted as being received, if it is not in the list of missing messages with a smaller number.

- Efficiency of Ideal Selective Repeat
  - The window size is large enough that xmission never stops because the transmit window has been exceeded.
    - The window size should be greater than the round trip delay or the timeout
    - By making the window size a multiple $m$ of the timeout the source can retransmit the packet m-1 times before the window is exceeded
    - The window is only exceeded if there are m consecutive errors in the packet
  - Lost acks can be recovered by later ACKS, so that the probability of loss is $P_L$, as in go back n, rather than $1 - (1 - P_L)^2 = 2P_L - P_L^2$, as in stop and wait.
  - Ideally, only a lost packet is retransmitted
  - $\eta = \dfrac{1}{1/(1 - P_L)} = 1 - P_L$
- The modulus for message numbering in Selective ARQ is 2W, twice as large as in go back N
  - When the receiver acknowledges a packet it may advance from R to R+W, and be willing to accept packets R+2W-1
  - If the transmitter misses this Ack, it may retransmit packet R
  - The receiver must distinguish packets from R to R+2W-1

**Homework**

A telephone modem is used to connect a personal computer to a host computer. The speed of the modem is 56 kbps, the one-way propagation delay is 100 ms, the packet size is 256 bytes, and the probability of an error in a packet is $10^{-4}$.

1. Find the efficiency of stop-and-wait ARQ.
2. What window size $N$ is needed to keep the channel busy when there are no transmission errors. For this window size find the efficiency of the go-back-N and selective repeat protocols.

# 5. Buffer Management

## 5.1 Beeforth's protocol [2]

Objective: To retain packet sequence in a virtual circuit network when packets are lost on a path, and to limit the buffering at intermediate nodes

A protocol mechanism, rather than a routing discipline, such as hot-potato routing, is used to limit the amount of buffering in the network

This protocol is particularly useful in all optical networks. Some optical switches have been proposed that transfer signals similar to the double acks.

Implementation

1. first packet is the scout and reserves 1 buffer at each node
2. 2 - acks per link
3. A message is periodically retransmitted on a link until ACK 1 is received

   Ack 1 says that the packet was successfully transferred to the next node

   The node can stop transmitting the packet

   The transmitting node can discard the packet, since it has been successfully forwarded, and make the buffer available to receive another packet.
4. When the buffer becomes available for the next packet, a node sends the node that transmitted the packet ACK 2

   ACK 2 is an invitation to send the next packet in the message

   If ACK 2 is lost the previous node will not transmit the next packet

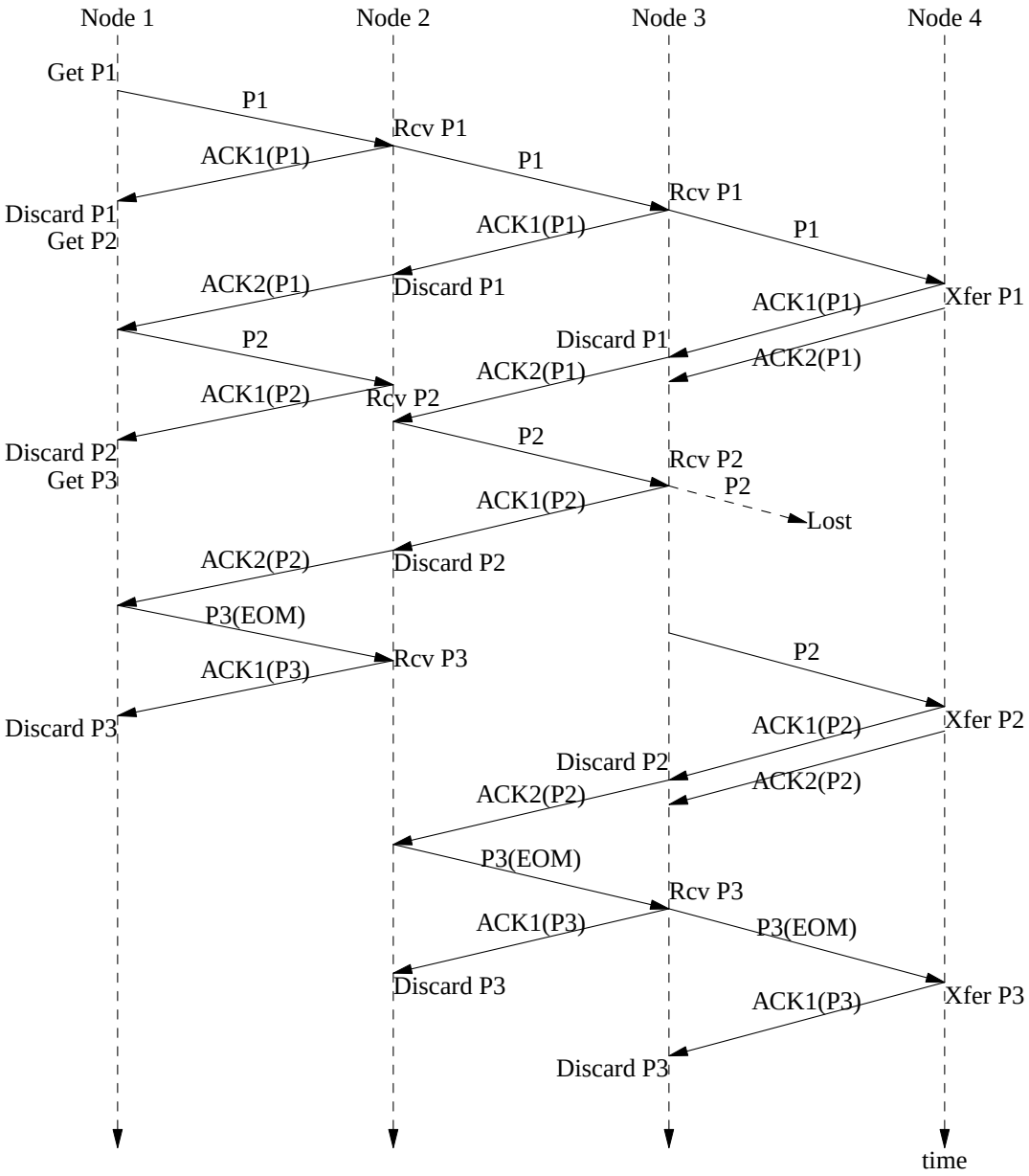
   ACK 2 is periodically retransmitted until the next packet is received

   The retransmit time of ACK 2 is longer than the retransmit time for messages, so that ACK 2's are not retranmitted when a message is lost
5. The last packet in a message contains an EOM

   When a node receives ACK 1 for the EOM, it does not send ACK2, since there are no more packets

   When ACK 1 is received, the buffer is deallocated

   If the final ACK 1 is lost, the node continues transmitting the packet until it is received




*REFERENCES*

[1] D. Bertsekas, R. G. Gallager, **Data Networks**, Prentice-Hall Inc, 1992.

[2] T. H. Beeforth, R. L. Grimsdale, F. Halsall, D. J. Woolons, "Proposed Organisation for Packet-Switched Data-Communication Network," Proc. IEE, vol. 119, no. 12, Dec 1972, pp. 1677-1682.