Reliable Data Transfer: Go-back-n

In this lesson, we'll study go-back-n: a simple protocol to ensure detection and retransmission of lost packets.

WE'LL COVER THE FOLLOWING

In the last lesson, we discovered that a sending sliding window alone is not enough to ensure **detection and retransmission of lost packets**. In order to do that, we will look at two protocols:

1. **Go-back-n**
2. **Selective Repeat**

Go-back-n

The simplest sliding window protocol uses **go-back-n** recovery.

Go-back-n Receiver

Intuitively, go-back-n receiver operates as follows:

1. It only accepts the segments that arrive in-sequence.
2. It discards any out-of-sequence segment that it receives.
3. When it receives a data segment, it always returns an acknowledgment

3. When it receives a data segment, it always returns an acknowledgment containing the sequence number of the **last in-sequence segment** that it has received.

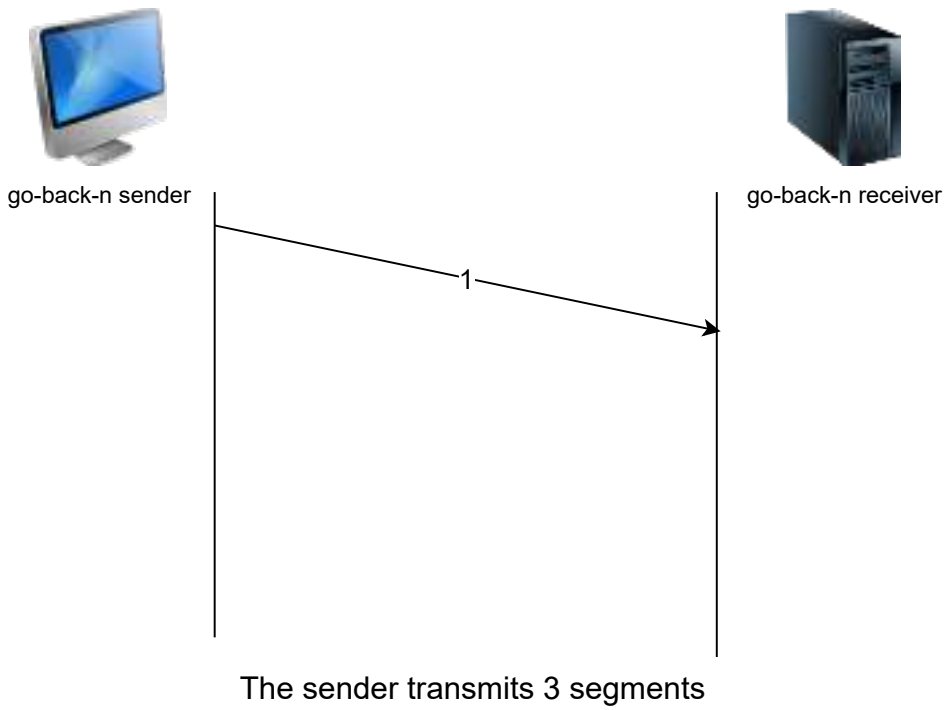
Cumulative Acknowledgements

This acknowledgment is said to be **cumulative**. When a go-back-receiver sends an acknowledgment for sequence number x, it implicitly acknowledges the reception of all segments whose sequence number is smaller than or equal to x.

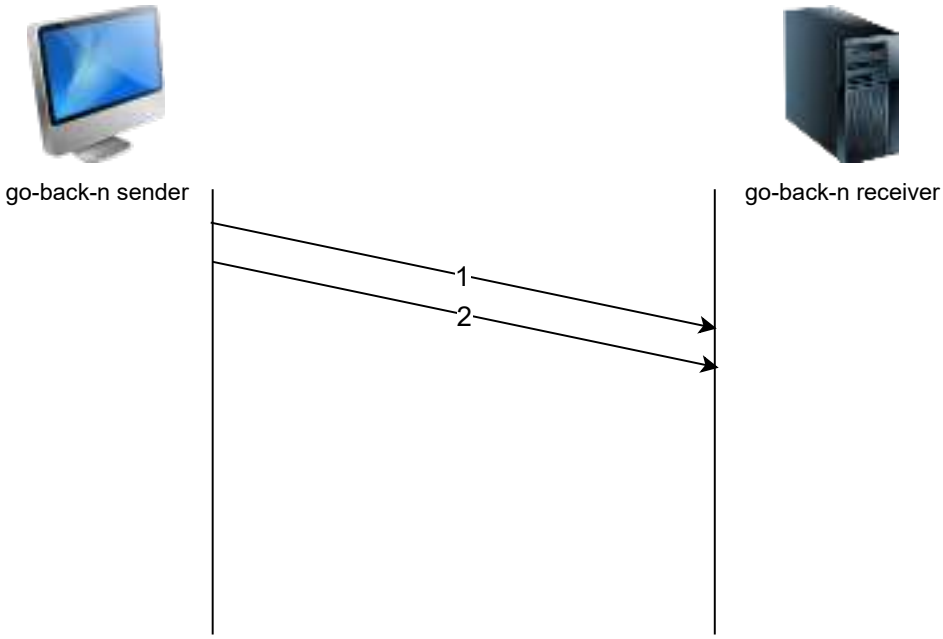
A **key advantage** of these cumulative acknowledgments is that it's **easy to recover from the loss of an acknowledgment**.

Consider for example a go-back-n receiver that received segments 1, 2 and 3.

1. It sent acknowledgments for all three segments.
2. Unfortunately, acknowledgments of the first two were lost.
3. Thanks to the cumulative acknowledgments, the receiver receives the acknowledgment for the last segment and so it knows that all three have been correctly received.

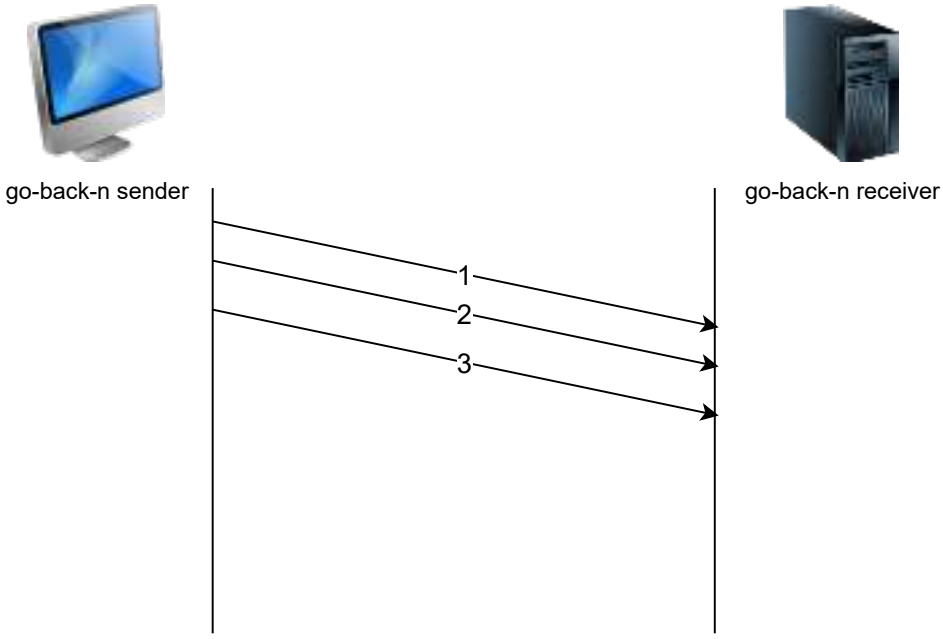


Go-back-n cumulative acknowledgements



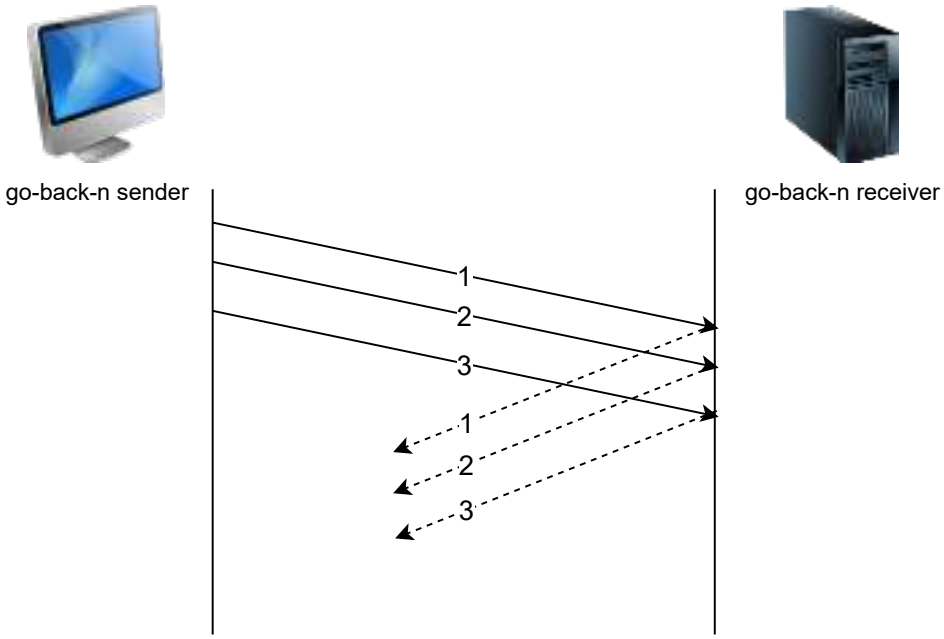
The sender transmits 3 segments

Go-back-n cumulative acknowledgements



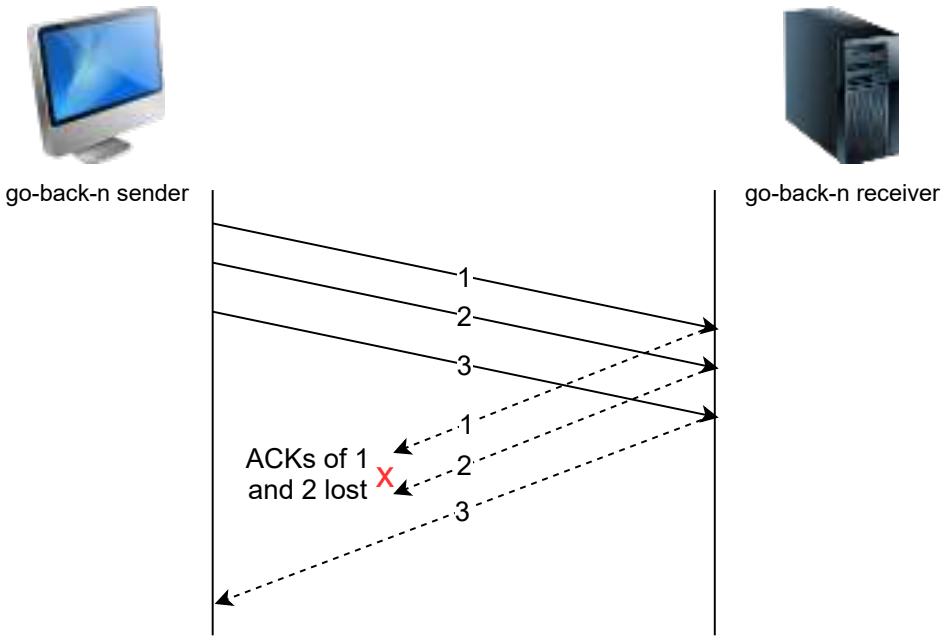
The sender transmits 3 segments

Go-back-n cumulative acknowledgements



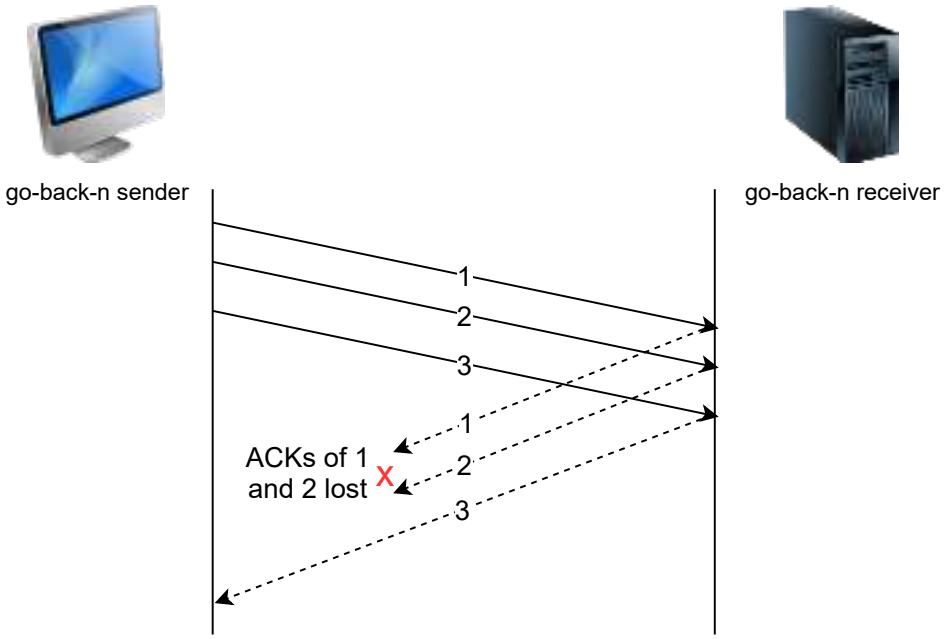
The receiver sends acknowledgements for all 3 segments

Go-back-n cumulative acknowledgements



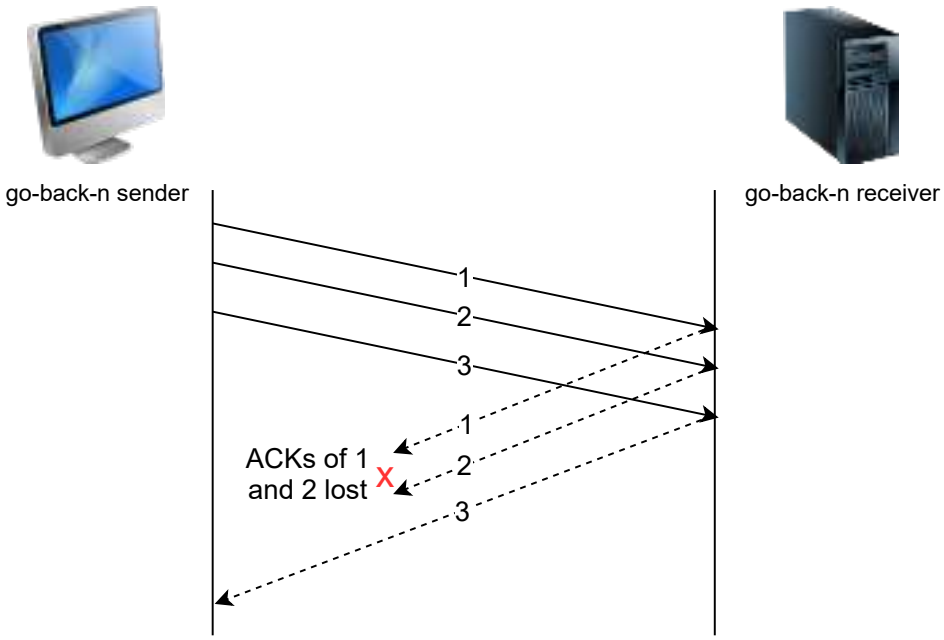
However, the acknlowedgments for the first and second segments are lost

Go-back-n cumulative acknowledgements



The acknowledgment for the third segment gets passed through though

Go-back-n cumulative acknowledgements



Hence the sender knows that the first two were received correctly too and does not retransmit them

Go-back-n cumulative acknowledgements

Go-back-n Sender

A go-back-n sender uses a sending buffer that can store an entire sliding window of segments.

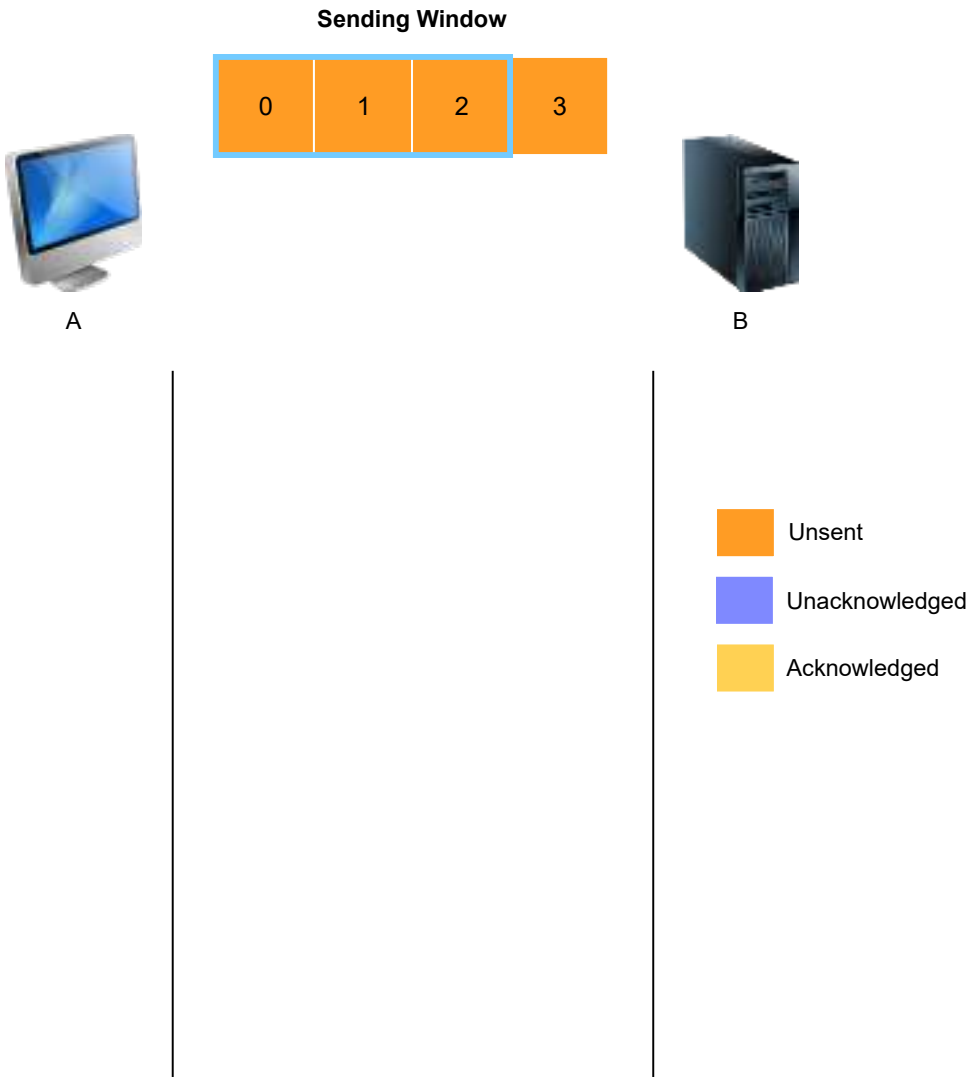
- The segments are sent with a sending sliding window that we looked at in the last lesson.
- The sender must wait for an acknowledgment once its sending buffer is full.
- When a go-back-n sender receives an acknowledgment, it removes all the acknowledged segments from the sending buffer.

Retransmission Timer

A go-back-n sender uses a **retransmission timer** to detect segment losses. It maintains one retransmission timer per connection. Here's how it works:

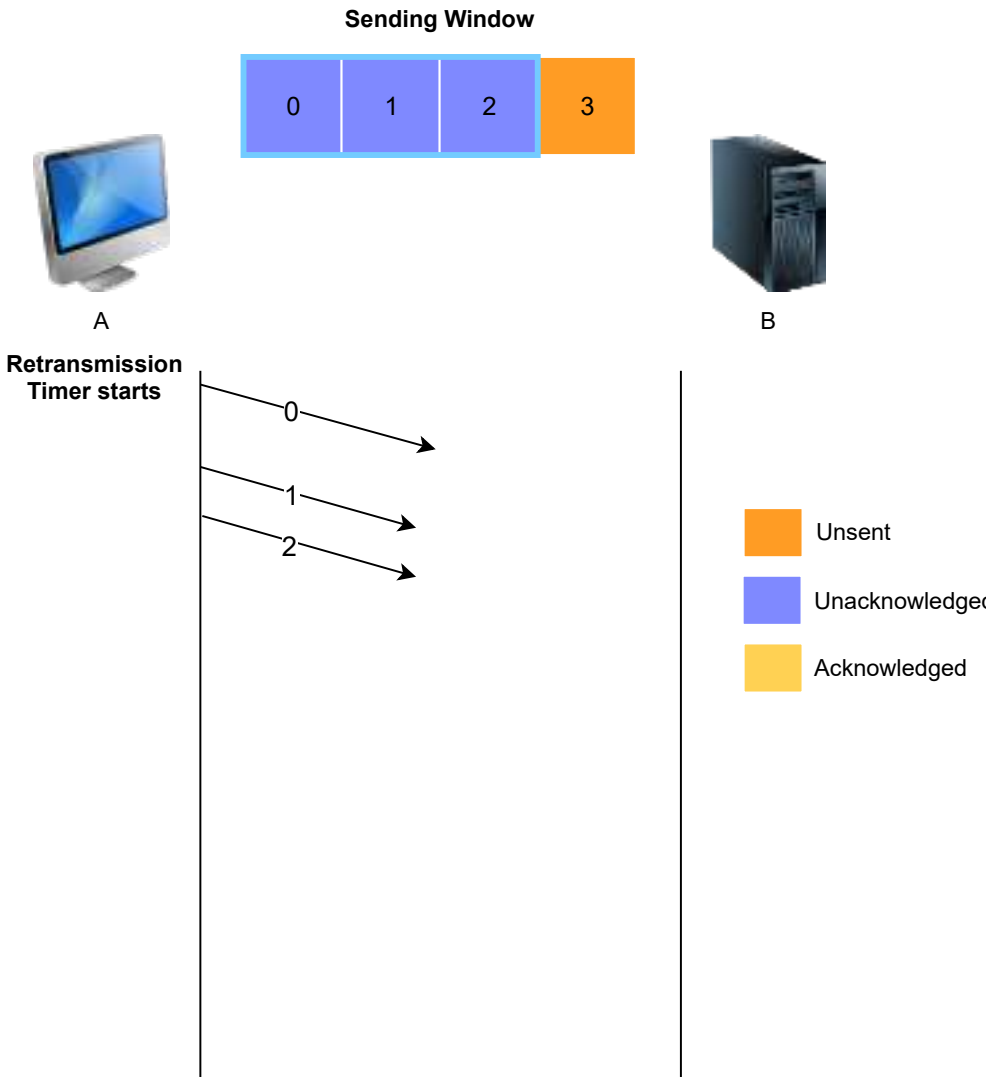
1. This timer is started when the first segment is sent.
2. When the go-back-n sender receives an acknowledgment, it restarts the retransmission timer, but only if any unacknowledged segments exist in its sending window.
3. When the retransmission timer expires, the go-back-n sender assumes that all of the unacknowledged segments currently stored in its sending buffer have been lost. It thus retransmits all the unacknowledged segments in the buffer and restarts its retransmission timer.

The general operation of go-back-n is illustrated in the figure below.



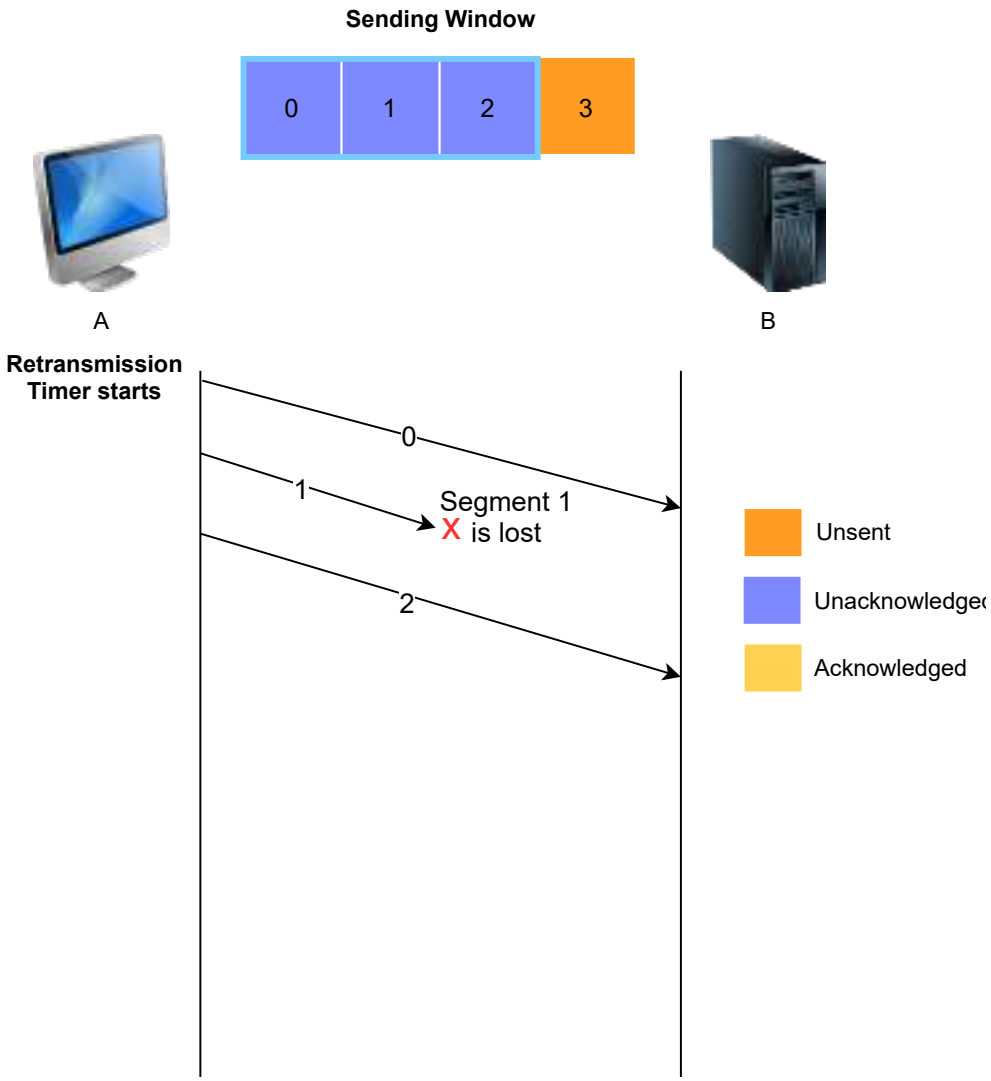
A sending sliding window of size 3 is initiated

Go-back-n: example



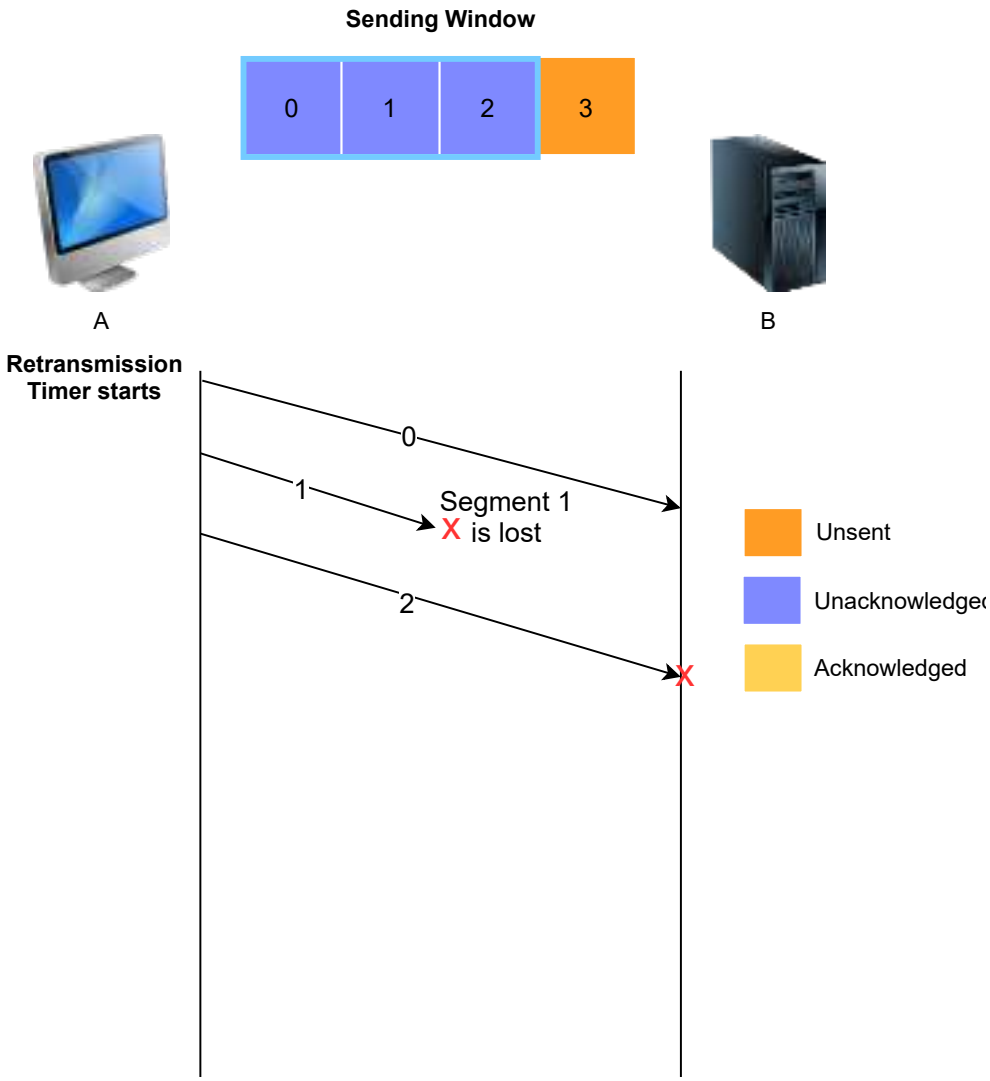
GBN sender sends 3 segments

Go-back-n: example



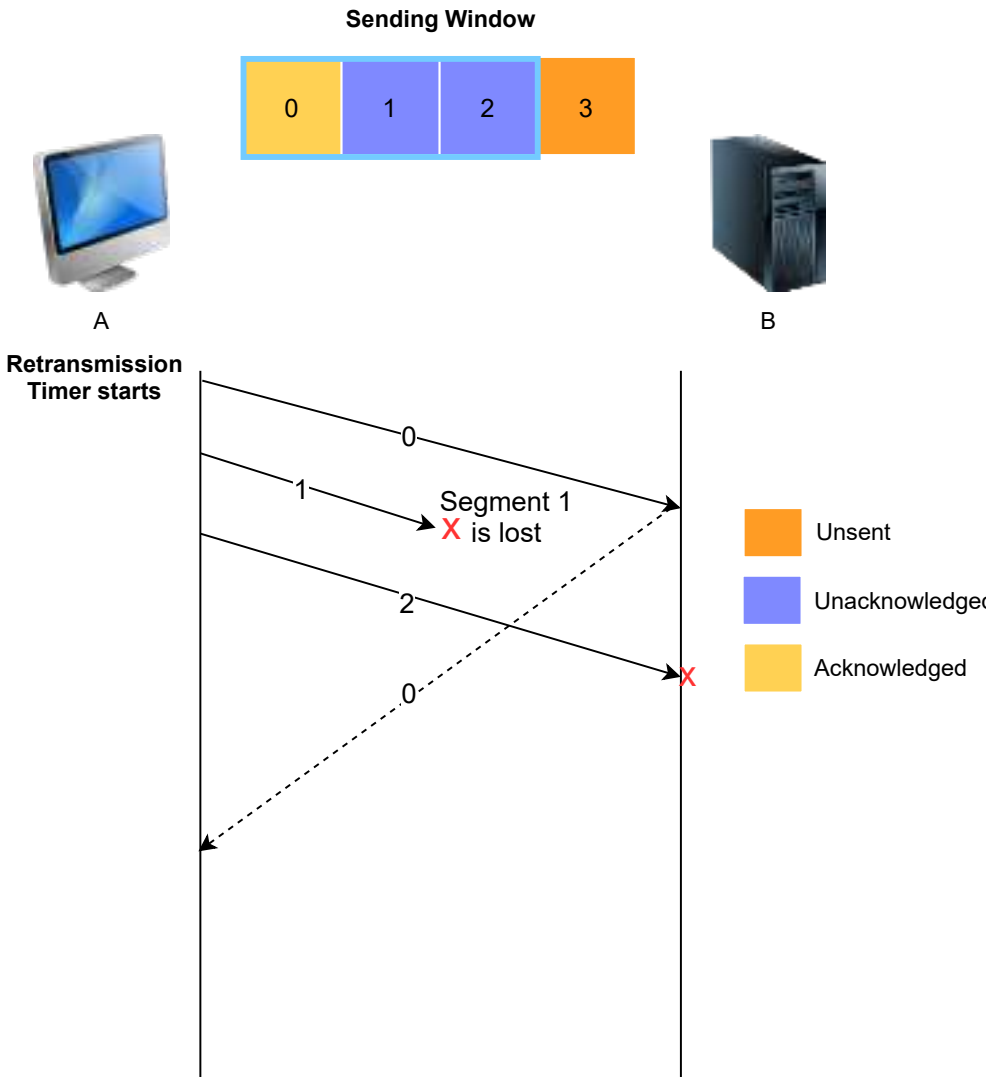
Receiver was expecting to receive segment #1
but it is lost and it receives segment #2 instead

Go-back-n: example



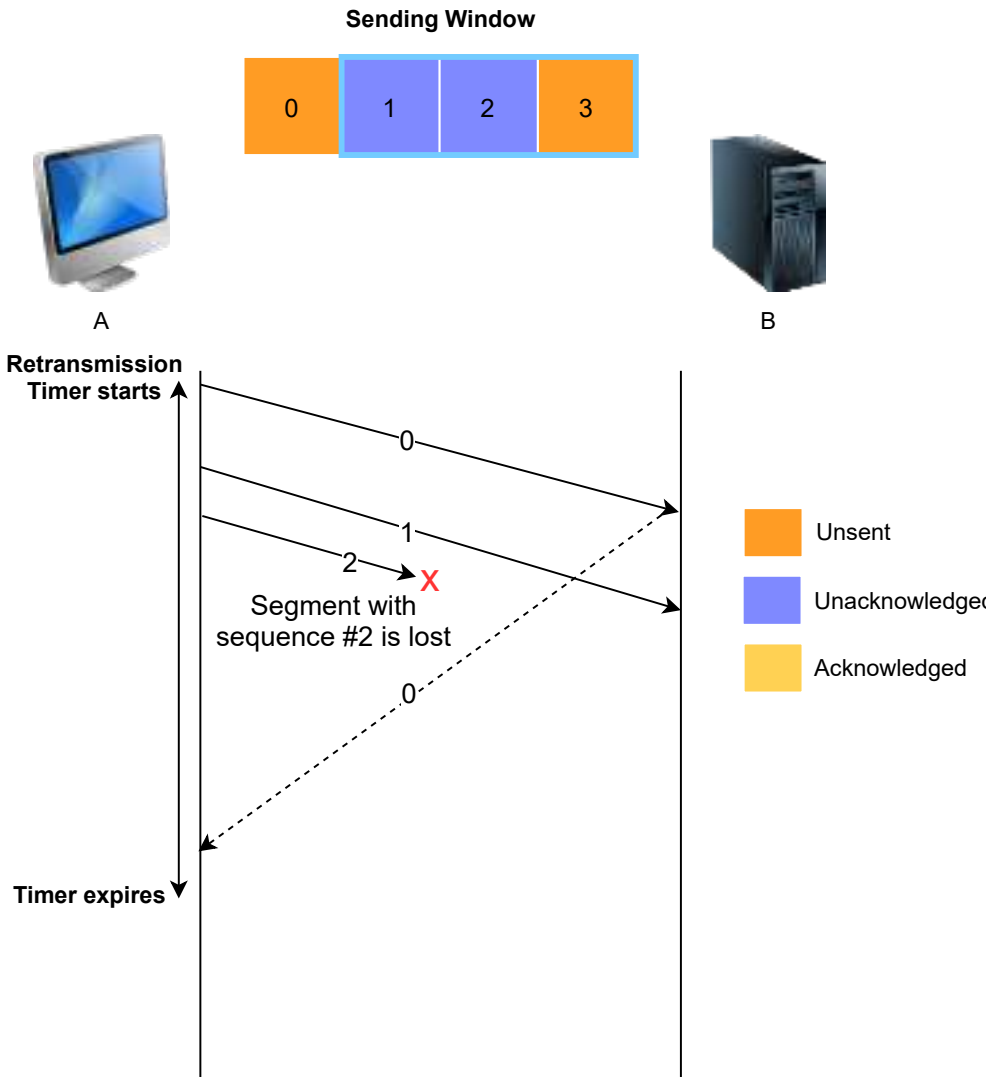
Sement #2 is discarded because that sequence number was not expected. #1 was expected

Go-back-n: example



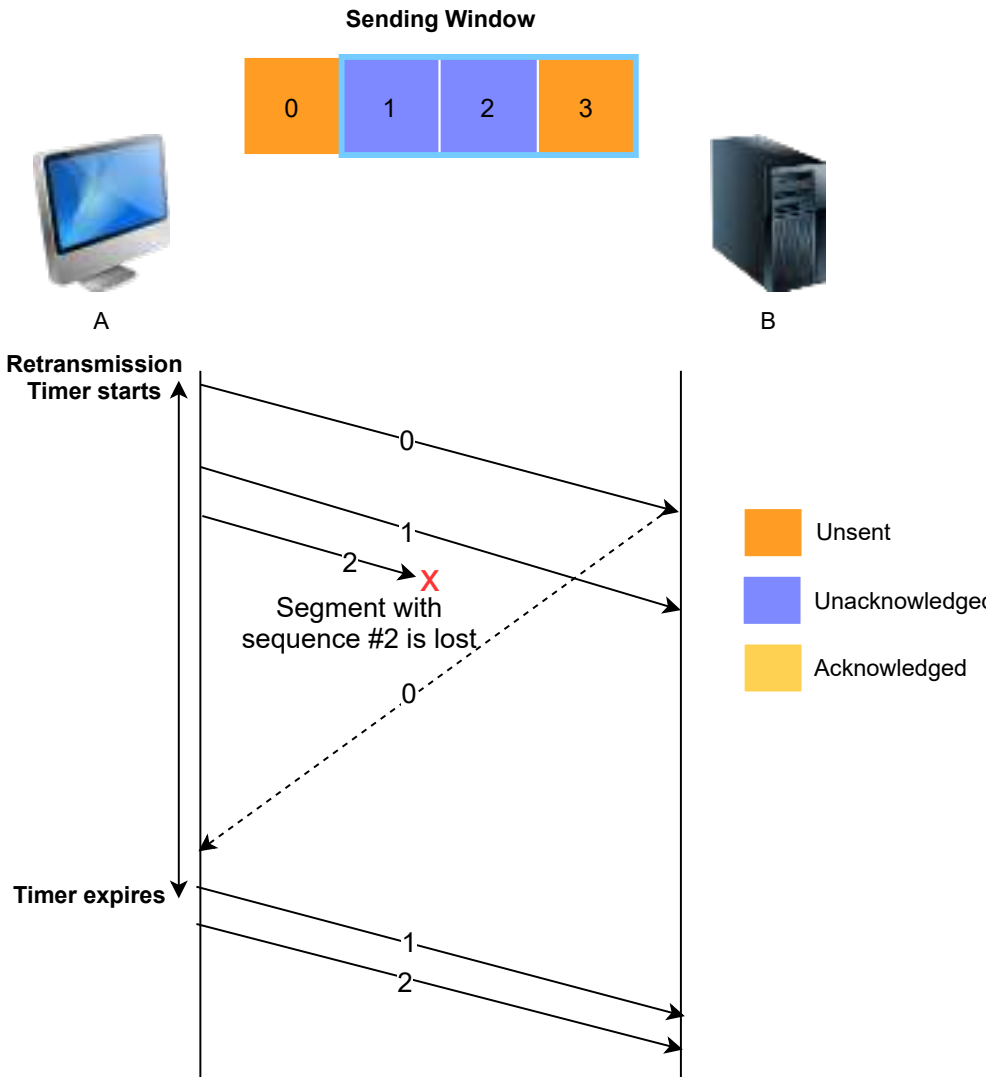
Acknowledgement for segment #0 is sent

Go-back-n: example



Transmission timer expires so the sender has to resent all unacknowledged segments, i.e., #1 and #2

Go-back-n: example



Go-back-n: example

Advantages of Go-back-n

The main advantage of go-back-n is that it can be easily implemented, and it can also provide good performance when only a few segments are lost. But when there are many losses, the performance of go-back-n quickly drops for two reasons:

- The go-back-n receiver does not accept out-of-sequence segments.

- The go-back-n sender retransmits all unacknowledged segments once it has detected a loss.

Since the go-back-n protocol does not accept out of order segments, it can waste a lot of bandwidth if segments are frequently lost.

The Selective Repeat protocol attempts to remedy this by accepting out of order packets and only retransmitting packets that are corrupted or lost in the network.

Selective Repeat

- Uses a sliding window protocol just like go-back-n.
- The window size should be less than or equal to half the sequence numbers available. This avoids packets being identified incorrectly. Here's an **example**: suppose the window size is greater than half the buffer size.
    - Segment '1' is lost, hence the receiver expects a segment with sequence number 1 to be retransmitted.
    - Meanwhile, the window wraps around back to sequence number '1.'
    - The sender sends a new packet with sequence number 1 and the receiver perceives it to be the original one that it was expecting.
- Senders retransmit unacknowledged packets after a timeout or if a *NAK* (*negative acknowledgment/not acknowledged*) is received.
- The receiver acknowledges all correct packets.
- The receiver stores correct packets until they can be delivered in order to the upper application layer.

>  **Note: NAKS** Negative acknowledgements can be realized using positive acknowledgements (ACKs) of the last correctly received segment!

Comparing to go-back-n

In comparison to the go-back-n protocol, selective repeat conserves bandwidth, which is, the rate of data transfer across a path.

Quick Quiz!

1

A cumulative acknowledgement for sequence number n acknowledges the receipt of all sequences numbers upto and including _____.

COMPLETED 0%

1 of 5

In the next lesson, we'll start with UDP, a transport layer protocol.