Reliable Data Transfer – (A concise summary)

Source / Credits - https://www.d.umn.edu/~gshute/net/reliable-data-transfer.xhtml

1. Reliable Data Transfer

The internet network layer provides only best effort service with no guarantee that packets arrive at their destination. Also, since each packet is routed individually it is possible that packets are received out of order. For connection-oriented service provided by TCP, it is necessary to have a reliable data transfer (RDT) protocol to ensure delivery of all packets and to enable the receiver to deliver the packets in order to its application layer.

A simple alternating bit RDT protocol can be designed using some basic tools. This protocol is also known as a stop-and-wait protocol: after sending each packet the sender stops and waits for feedback from the receiver indicating that the packet has been received.

Stop-and-wait RDT protocols have poor performance in a long-distance connection. At best, the sender can only transmit one packet per round-trip time. For a 1000 mile connection this amounts to approximately 1 packet (about 1500 bytes) every 20 ms. That results in a pathetic 75 KB per second rate.

To improve transmission rates, a realistic RDT protocol must use pipelining. This allows the sender to have a large number of packets "in the pipeline". This phrase refers to packets that have been sent but whose receipt has not yet verified by the receiver.

2. Basic Tools Summary

Error Detection

The data link and network layers have error detection for detecting bit errors in packets. However, error detection schemes can never detect all errors so it is helpful to have additional error detection in the transport layer to reduce the frequency of undetected errors.

Lower layer protocols with error detection usually have a simple policy for dealing with errors: discard the packet. A transport layer RDT protocol typically just does the same thing, letting its algorithm for dealing with missing packets deal with the problem. Since the lower layers may discard packets an RDT protocol must allow for the possibility that a receiver is not even aware of an attempted transmission.

Sequence Numbering

Packets in the network layer are routed individually. This makes it possible that they are received in a different order than they are transmitted. Sequence numbering is essential for restoring the transmitted order.

Feedback

Feedback involves information sent by the receiver back to the sender about reception of sent packets. This is essential for recovery of missing packets. The feedback takes the form of acknowledgments (ACKs) with one of three forms:

- Negative acknowledgment - "I did not receive the packet with sequence number "*sn*."
- Positive individual acknowledgment - "I received the packet with sequence number "*sn*."
- Positive cumulative acknowledgment - "I have received all packets with sequence numbers up to but not including "*sn*."

Most reliable data transfer protocols use only one of these types of acknowledgment. Negative acknowledgments are useful in human communication, but only because the acknowledgment is not lost, though it may be garbled. Since negative acknowledgment packets can be lost in the internet, they are not useful. They will not be considered in this presentation.

Cumulative acknowledgments allow acknowledgment of numerous packets at a time. They can be useful in pipelined protocols.

Timers

Packet loss in the network layer does not discriminate between data packets and acknowledgment packets. A sender in a reliable data transfer protocol needs to set a timer for transmitted packets. Generally the sender does one of two things:

- Resends a packet after a timer fires.
- Sends a new packet after an acknowledgment (positive) arrives.

If an acknowledgment arrives before the timer fires the sender stops the timer so that it will not fire.

3. Alternating Bit Protocol

The section on timers describes most of the sender policy involved in the alternating bit protocol. The only thing required to complete the protocol is handling of sequence numbers and the responsibilities of the receiver.

The alternating bit protocol uses a 1-bit sequence number. That is, the sequence number alternates between 0 and 1. The protocol only uses positive acknowledgments.

Both sender and receiver maintain state information about an expected sequence number.

- For the receiver it is the sequence number of the packet that is expected next.
- For the sender it is the sequence number of the acknowledgment that is expected next.

The protocol is best described in terms of event handling by the receiver and sender for different kinds of events.

Alternating Bit Receiver

The receiver initially expects a packet with sequence number 0. The receiver has only two kinds of events to respond to.

The receiver has one state variable:

- `expected` (initial value 0) is the next expected sequence number.

The receiver has two kinds of events to respond to.

- **A packet arrives that has the expected sequence number:**

  Send an acknowledgment for the packet to the network layer and toggle the expected sequence number.

  Forward the packet to the application layer.

- An packet arrives that does not have the expected sequence number:

  Send an acknowledgment for the packet to the network layer. The sender must not have received the previous acknowledgment.

The sequence number in an acknowledgment is always the same as the sequence number of the received packet that triggered the acknowledgment.

Alternating Bit Sender

The sender begins with expected sequence number 0. It starts by accepting data coming in from the application layer. Then the sender has four types of events to respond to.

- **A packet arrives from the application layer:**

  Stop accepting input from the application layer.

  Save the packet in case it needs to be resent.

  Send the packet to the network layer and start the timer.

- **The timer fires:**

  Send the saved packet to the network layer and restart the timer.

- **An acknowledgment arrives that has the expected sequence number:**

  Stop the timer so that it will not fire.

  Toggle the expected sequence number.

  Accept data from the application layer.

- **An acknowledgment arrives that does not have the expected sequence number:**

  Ignore it. It is an acknowledgment of a resend due to a late acknowledgment.

4. Pipelining

Pipelining allows a sender to send out multiple packets without waiting for acknowledgment. But all packets that have been sent must be retained until they are acknowledged in case they need to be resent. To limit the number of packets that need to be retained, pipelined protocols need to set a bound on the number of packets in the pipeline (sent but not yet acknowledged). This bound imposes a "window" on the sequence numbers for packets in the pipeline.

Pipelined protocols can be distinguished based on how they use acknowledgments and timers. Two basic pipelined protocols, "Go back *n*" and "Selective Repeat" acknowledge individual packets, but differ in the number of timers that they use. The RDT protocol in TCP uses cumulative acknowledgments

Windowing

The window size is the limit on the number of packets in the pipeline; that is, the number of packets that have been sent but not yet acknowledged. Protocols that use packet numbers as sequence numbers limit sequence numbers to a range twice the size of the window. If the window size is `n` then sequence numbers range from 0 to 2*`n` - 1.

Senders and receivers typically set up arrays for sent or received packets. The sequence number is used as an index into the array so the array size is twice the window size. Circular indexing is used for array access.

Acknowledgments

Basic pipelined protocols use individual acknowledgments. An acknowledgment with sequence number *i* only acknowledges receipt of the packet with sequence number *i*. The pipelined protocol in TCP uses cumulative acknowledgments. An acknowledgment with sequence number *i* acknowledges receipt of all packets with sequence numbers up to and including *i*.

Timers

Pipelined protocols that use individual acknowledgments can either use a single timer or they can use a separate timer for each packet in the pipeline. With a single timer (go back *n*), the protocol does not know what packet the timer is firing for so it resends all unacknowledged packets. With a timer for each packet in the pipeline (selective repeat), the protocol can just resend the packet associated with the timer that fired.

Pipelined protocols that use cumulative acknowledgments just use a single timer, but still only resend a single packet. With cumulative acknowledgments the sender just resends the packet just beyond the most recent acknowledgment. When the receiver gets the resend it will send another acknowledgment to update the sender.

5. Go Back n Protocol

The go back *n* protocol uses individual acknowledgments. A go back *n* sender uses a single timer. When this timer fires the sender does not know which packet failed to be received so the sender resends all packets in the window.

Go Back n Receiver

The receiver has one state variable:

- `windowStart` (initial value 0) is the start index of the receive window.

  The receiver has two kinds of events to respond to.

- **A packet arrives and its sequence number is in the window:**

  Save it and send an acknowledgment for the packet to the network layer.

  Forward acknowledged packets at the beginning of the window to the application layer and advance `windowStart` past these packets.

- **A packet arrives and its sequence number is not in the window:**

  Just send an acknowledgment for the packet to the network layer. The sender must not have received the previous acknowledgment.

The sequence number in an acknowledgment is always the same as the sequence number of the received packet that triggered the acknowledgment.

Go Back n Sender

The sender has two state variables:

- `windowStart` (initial value 0) is the start index of the send window.
- `nextSequenceNumber` (initial value 0) is the sequence number to be used for the next outgoing packet.

The sender has four kinds of events to respond to.

- **A packet arrives from the application layer:**

Give the packet sequence number `nextSequenceNumber` and send it to the network layer. Save the packet in case it needs to be resent. Start or restart the timer.

Increment `nextSequenceNumber` and stop accepting input from the application layer if the window is full.

- **The timer fires:**

  Send all unacknowledged saved packets in the window to the network layer and restart the timer.

- **An acknowledgment arrives that has a sequence number in the window:**

  Stop the timer so that it will not fire.

  Advance the window to begin at the first unacknowledged packet.

  Accept data from the application layer if the window has advanced.

- **An acknowledgment arrives and its sequence number is not in the window:**

  Ignore it. It is an acknowledgment of a resend due to a late acknowledgment.

6. Selective Repeat Protocol

The selective repeat protocol, like the go back *n* protocol, uses individual acknowledgments. Unlike the go back *n* protocol, a selective repeat sender uses a timer for each packet in the pipeline. This makes it possible to be selective about handling timer firing. The selective repeat sender only resends the packet associated with the timer that fired.

Selective Repeat Receiver

The receiver has one state variable:

- `windowStart` (initial value 0) is the start index of the receive window.

The receiver has two kinds of events to respond to.

- **A packet arrives and its sequence number is in the window:**

  Save it and send an acknowledgment for the packet to the network layer.

  Forward acknowledged packets at the beginning of the window to the application layer and advance `windowStart` past these packets.

- **A packet arrives and its sequence number is not in the window:**

  Just send an acknowledgment for the packet to the network layer. The sender must not have received the previous acknowledgment.

The sequence number in an acknowledgment is always the same as the sequence number of the received packet that triggered the acknowledgment.

Selective Repeat Sender

The sender has two state variables:

- `windowStart` (initial value 0) is the start index of the send window.
- `nextSequenceNumber` (initial value 0) is the sequence number to be used for the next outgoing packet.

The sender has four kinds of events to respond to.

- **A packet arrives from the application layer:**

Give the packet sequence number `nextSequenceNumber` and send it to the network layer. Save the packet in case it needs to be resent. Start timer `nextSequenceNumber`.

Increment `nextSequenceNumber` and stop accepting input from the application layer if the window is full.

- **Timer ”I” fires:**

  Send saved packet “*I*”to the network layer and restart timer *i*.

- **An acknowledgment arrives that has a sequence number, *i*, in the window:**

  Stop timer *“I”* so that it will not fire.

  Advance the window to begin at the first unacknowledged packet.

  Accept data from the application layer if the window has advanced.

- **An acknowledgment arrives and its sequence number is not in the window:**

  Ignore it. It is an acknowledgment of a resend due to a late acknowledgment.

7. Cumulative Acknowledgment

When cumulative acknowledgments are used the sequence number in the receiver acknowledgment always indicates the highest sequence number that has been received along with all with all of its predecessors.

The sender uses a single timer that is restarted either when a packet is sent and all of its predecessors have been acknowledged or when an acknowledgment arrives for a previously unacknowledged packet. There are some variations on this policy. i.e., there can be implementations that combines concepts of GBN and selective repeat, in terms of receiver window, timers, and acknowledgements.

.

Reliable Data Transfer Receiver- Summary

	Alternating Bit	**Go Back *n***	**Selective Repeat**	**Cumulative Acknowledgment**
Receiver initialization	The receiver has one state variable: • `expected` (initial value 0) is the next expected sequence number.	The receiver has one state variable: • `windowStart` (initial value 0) is the start index of the receive window.	The receiver has one state variable: • `windowStart` (initial value 0) is the start index of the receive window.	The receiver has one state variable: • `windowStart` (initial value 0) is the start index of the receive window.
Response to an incoming packet with the expected sequence number	Send an acknowledgment for the packet and toggle `expected`. Forward the packet to the application layer.	Save it and send an acknowledgment for the packet. Forward acknowledged packets at the beginning of the window to the application layer and advance `windowStart` past these packets.	Save it and send an acknowledgment for the packet. Forward acknowledged packets at the beginning of the window to the application layer and advance `windowStart` past these packets.	Save it and send an acknowledgment for the packet. Forward acknowledged packets at the beginning of the window to the application layer and advance `windowStart` past these packets.
Response to an incoming packet with an unexpected sequence number	Just send an acknowledgment for the packet. The sender must not have received the previous acknowledgment.	Just send an acknowledgment for the packet. The sender must not have received the previous acknowledgment.	Just send an acknowledgment for the packet. The sender must not have received the previous acknowledgment.	Just send an acknowledgment for the packet. The sender must not have received the previous acknowledgment.
Receiver acknowledgment sequence number	The same as the sequence number of the received packet.	The same as the sequence number of the received packet.	The same as the sequence number of the received packet.	The lowest sequence number that has not yet been received.

Reliable Data Transfer Sender - Summary

	Alternating Bit	**Go Back *n***	**Selective Repeat**	**Cumulative Acknowledgment**
Sender initialization	The sender has one state variable: • `nextSequenceNumber` (initial value 0) is the sequence number to be used for the next outgoing packet.	The sender has two state variables: • `windowStart` (initial value 0) is the start index of the send window. • `nextSequenceNumber` (initial value 0) is the sequence number to be used for the next outgoing packet.	The sender has two state variables: • `windowStart` (initial value 0) is the start index of the send window. • `nextSequenceNumber` (initial value 0) is the sequence number to be used for the next outgoing packet.	The sender has two state variables: • `windowStart` (initial value 0) is the start index of the send window. • `nextSequenceNumber` (initial value 0) is the sequence number to be used for the next outgoing packet.
Response to a packet from the application layer	Give the packet sequence number `nextSequenceNumber` and send it. Save the packet in case it needs to be resent. Start the timer. Increment `nextSequenceNumber` and stop accepting input from the application layer.	Give the packet sequence number `nextSequenceNumber` and send it. Save the packet in case it needs to be resent. Start the timer. Increment `nextSequenceNumber` and stop accepting input from the application layer if the window is full.	Give the packet sequence number `nextSequenceNumber` and send it. Save the packet in case it needs to be resent. Start timer `nextSequenceNumber`. Increment `nextSequenceNumber` and stop accepting input from the application layer if the window is full.	Give the packet sequence number `nextSequenceNumber` and send it. Save the packet in case it needs to be resent. Start the timer. Increment `nextSequenceNumber` and stop accepting input from the application layer if the window is full.
Response to an acknowledgment with an expected	Advance the window to begin at the first unacknowledged packet.	Advance the window to begin at the first unacknowledged packet.	Advance the window to begin at the first unacknowledged packet.	Advance the window to begin at the first unacknowledged packet.

	Alternating Bit	Go Back *n*	Selective Repeat	Cumulative Acknowledgment
sequence number	Accept data from the application layer.	Accept data from the application layer if the window has advanced.	Accept data from the application layer if the window has advanced.	Accept data from the application layer if the window has advanced.
Response to an acknowledgment with an unexpected sequence number	Ignore it. It is an acknowledgment of a resend due to a late acknowledgment.	Ignore it. It is an acknowledgment of a resend due to a late acknowledgment.	Ignore it. It is an acknowledgment of a resend due to a late acknowledgment.	Ignore it. It is an acknowledgment of a resend due to a late acknowledgment.
Response to a timer firing	Send the saved packet and restart the timer.	Send all unacknowledged saved packets and restart the timer.	Send the saved packet associated with the timer and restart the timer.	Send the first unacknowledged saved packet and restart the timer.