Zion Transfer Protocol

CS3251 Fall 2015
BumJun Park, YongHui Cho
11/4/2015

ZTP (Zion Transport Protocol)

- **Is your protocol non-pipelined (such as Stop-and-Wait) or pipelined (such as Selective Repeat)?**
  Our protocol is pipelined using Cumulative ACK and GBN

- **How does your protocol handle lost packets?**
  If the received cumulative ACK does not match our expected ACK we set our window's first element to be the at the received cumulative ACK and start sending from that packet.

- **How does your protocol handle corrupted packets?**
  Using 16-bit checksum, ZTP can check whether packets are corrupted before notifying the application.

- **How does your protocol handle duplicate packets?**
  When duplicate packets are delivered, we check our receiving buffer and if the buffer contains the packet with the same SYN number then it is discarded.

- **How does your protocol handle out-of-order packets?**
  The private buffer stored received packets in order so as long as all packets within the window arrives it is okay. However if our private buffer element count is not equal to window size after the timeout(assuming all packets have been sent by the sender) we send our cumulative ACK with that was tracking how many packets we have received in order that acknowledges the sender to send packets again from the cumulative ACK.

- **How does your protocol provide bi-directional data transfers?**
  Bidirectional data transfer is provided in our protocol by its design. The sender is capable of sending messages and excepting the host to be receiving. The host that initiates connection would be the client. Also, the host can send when the client is listening.

- **Does your protocol use any non-trivial checksum algorithm (i.e., anything more sophisticated than the IP checksum)?**

No

Cumulative ACK Method

- Cumulative ACK is used to keep track of most successfully received data and acknowledge the sender that receiver has received all data preceding the cumulative ACK number. Also, this helps in controlling traffic that is unnecessary by stopping packets beyond window size before receiving lost packets.

Acknowledgement Method

- ZTP considers next expected sequence number to be the ACK.

Handling Lost Packets

- ZTP handles lost packets using TCP's method of calculating Timeout Interval (Estimated RTT + 4*DevRTT) and set the timeout dynamically. When Acknowledgement for the packet does not arrive in time within the timer's calculated time, the packet is sent again.

Handling Duplicate Packets

- ZTP accepts all packets that are either new or out-of-order and saves in its private buffer. Once it is known to be duplicate packets, the receiver side discards the packets.

Handling Out-Of-Order Packets

- ZTP has a private buffer with a given window size. Since an offset of SYN number and index SYN number is the index number for the private buffer, ZTP won't have problem with out-of-order packets problem.

Handling Delayed Packets

- Delayed packets are considered lost packets if they do not arrive within the time. If they do arrive in time but in different order it is considered to be out-of-order packet.

Handling Corrupted Packets

- Corrupted packets are detected by checksum. If the packet is deemed corrupted then it is immediately discarded and is asked for resend like the lost packets.

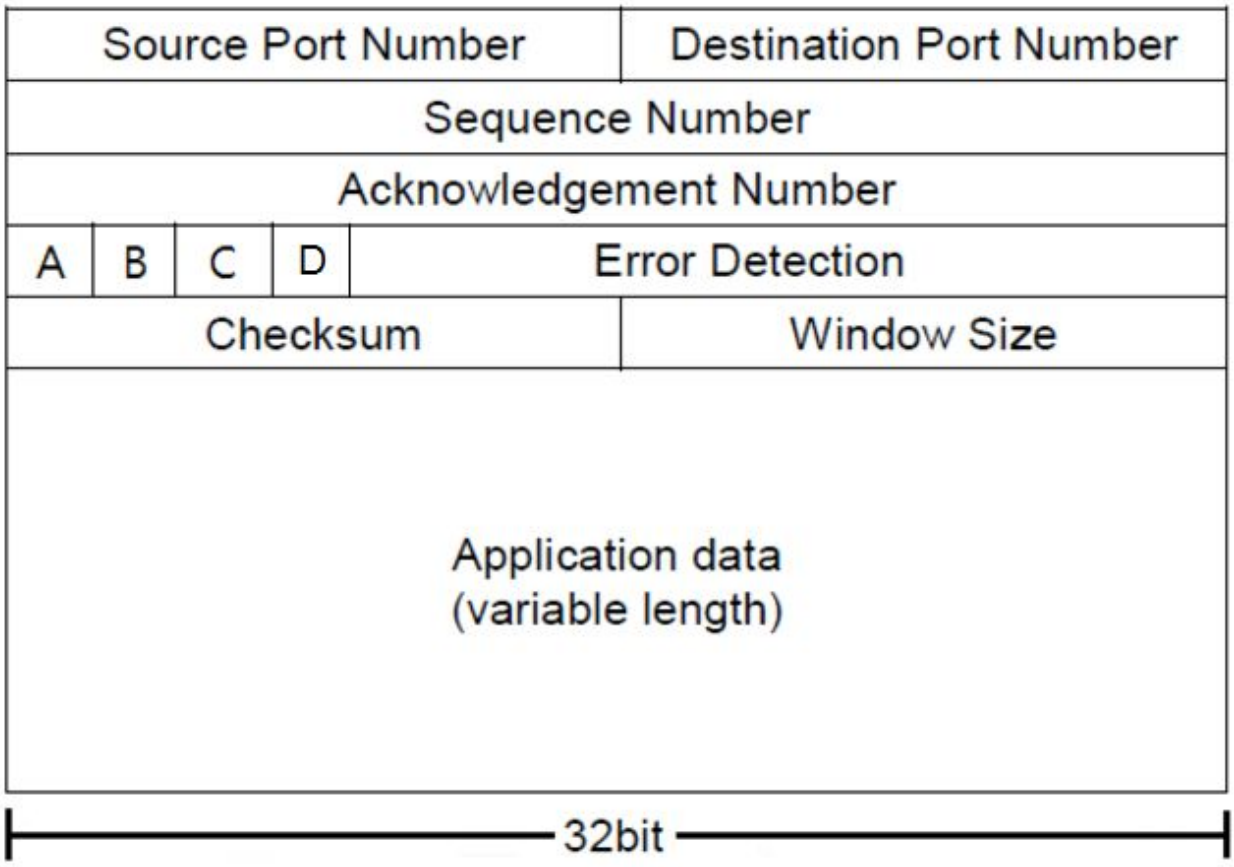
Handling Checksum

- ZTP uses CRC16 (Cyclic redundancy check) Algorithm to make checksum.

Connection

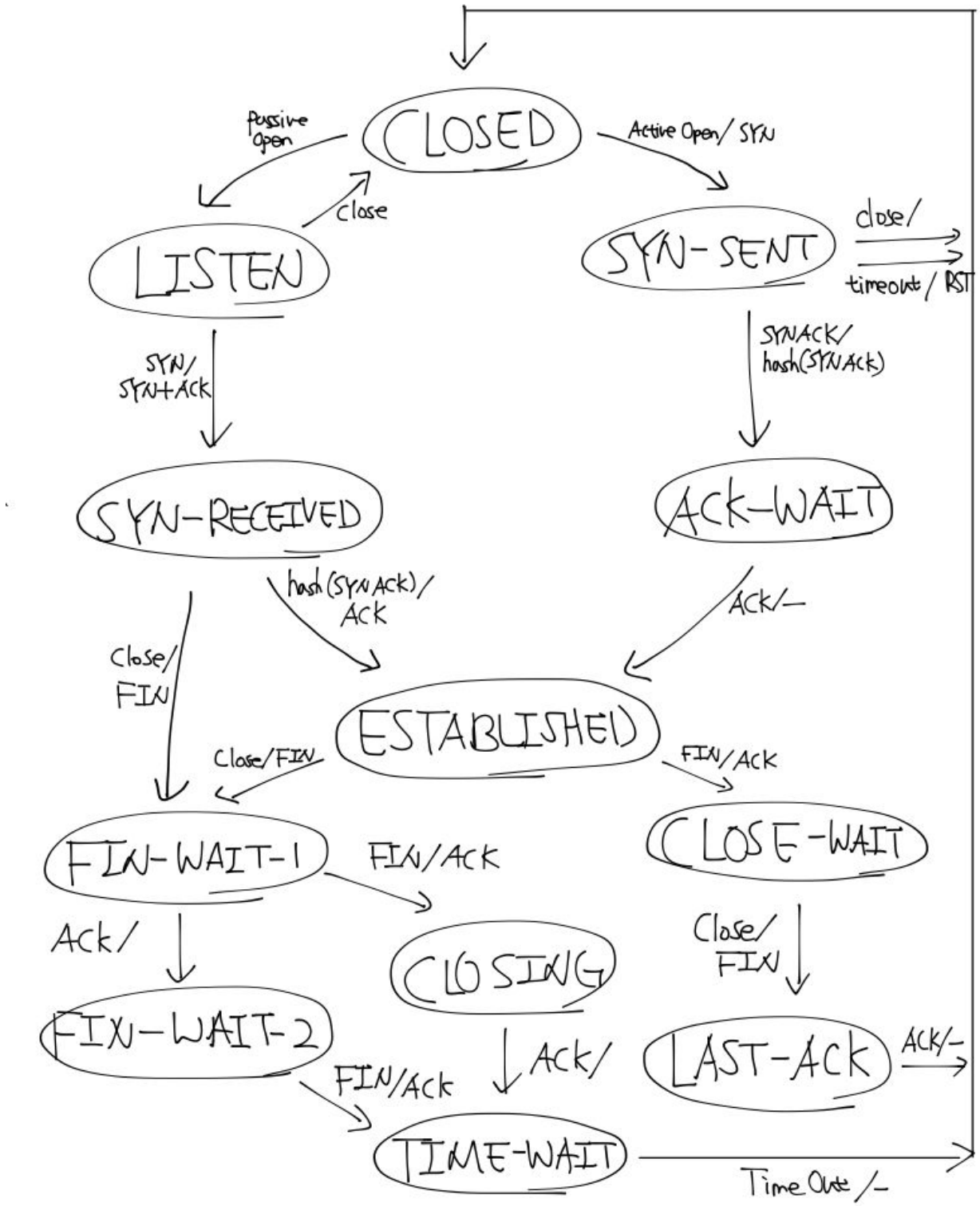
In our ZTP connect() it uses ‘4 way’ handshake. The handshake includes 1. Client sends SYN 2.Host receives SYN and sends SYN+ACK 3.Client sends hashed SYN+ACK 4. Host receives hashed SYN+ACK and if verified it sends ACK and then goes to established state. 5.Client goes to established state as it receives ACK. Please refer to the FSM down below for visual aid.

ZTP Header Structure



A: SYN
B: ACK
C: hash SYN/ACK
D: FIN

ZTP FSM


CLOSED
Passive Open
Active Open/ SYN
close
LISTEN
SYN-SENT
close/
timeout / RST
SYN/ SYN+ACK
SYNACK/ hash(SYNACK)
SYN-RECEIVED
ACK-WAIT
hash(SYNACK)/ ACK
ACK/-
Close/ FIN
ESTABLISHED
Close/FIN
FIN/ACK
FIN-WAIT-1
CLOSE-WAIT
FIN/ACK
Ack/
Close/ FIN
CLOSING
FIN-WAIT-2
LAST-ACK
ACK/-
FIN/ACK
Ack/
TIME-WAIT
Time Out /-

Application Programming Interface

Zocket object is the main object that is used for RxP API. Zocket object is implemented using Python 3. It is used to connect, enable bidirectional communication, manage connection status.

class Zocket

Methods	Description
bind(address)	returns socket instance bound to the address and port number from the parameter
connect(destAddr)	returns True if connection is successful. Connects with IP addr from destAdrr[0] and port number from destAddr[1]
listen()	Listens for connection request on previously set port
accept()	accepts incoming connections.
send(msg)	sends input parameter to receiver
recv()	receives message.
close()	closes connection and frees port. closing one socket allows connected socket to close().

Properties	
Socket.srcAddr	IP address and Port number of the client
Socket.destAddr	IP address and Port number of the host
Socket.connection	Current status of the connection
Socket.sendWind	The size of window for send
Socket.recvWind	The size of buffer for recv
Socket.timeout	Timeout of connection