Project II：Implement the Network

Member：Zhenyu Chen、Shiqi Zhang、Xinxun Zeng

In this project, we simply implemented network layer and transport layer protocols based on a link layer emulator. And we wrote several application by the new socket api to test our protocols. All requirements in document are done. Feature table is below:

IP-like Protocol	generate ICMP packet
	deal with ICMP packet
ICMP-like Protocol	reply ICMP request
Router	NAT
	Routing
	Forwarding
TCP-like Protocol	three-way handshake
	timeout and retransmission
	RST support
	pipeline sending
	out-of-order data storage
	timeout calculation
	fast retransmission
UDP-like Protocol	best effort
Socket API	bind, connect, listen, accept, send(to), recv(from), close, buffer
Testing Application	echo on unreliable link layer by TCP
	chatting room by UDP
	ping request by ICMP

1、Network Layer

a. IP-like Protocol

In the network layer, IP protocol need to act as both sender and receiver. On the one hand, it needs to handle the datagram sent by distinct transport layer

protocols: TCP and UDP. One the other hand, it needs to push the processed datagram to the end router.

i. General tool method:

a). pack:

Add headers(source ip, destination ip, upper protocol, ttl) to given tranport layer segment. After that, we need to calculate the checksum for it and fill the checksum into given position in headers. Finally, it is an appropriate IP datagram and ready for push downwards.

b). unpack:

Unpack the given datagram by IP datagram format. According to the header information ‘upper protocol’ to decide where to send.

ii. Send and push:

a). push:

For pushing upwards, it depends on the unpacked protocol information. For each different transport layer protocol, use their own push method.

b) send:

The general idea of sending is to pack the upper segment into IP datagram and then send to routers. however, for UDP and TCP, we use different send methods. Because UDP is connectionless and TCP is connection-oriented. Thus, we need to tell UDP where to send in parameter, but it has already known the destination ip for TCP.

b. ICMP-like Protocol

There are 4 kinds of ICMP situations in our implement:

1).echo request

2).echo reply

3).destination unreachable (4 different conditions)

4).time exceeded

Each situation has two methods: generate and reaction.

Generate method pack a icmp segment according to the type and code given and send it out.

Reaction method unpack the icmp segment, get the type and code, choose the right method accordingly.

c. Router

Routers can be divided into two parts: end router and core router. For end routers,they need to translate ip address and port number between public and local, which relies on NAT tables. For core routers, they just need to follow the forwarding table to forward the datagram to next one.

i. Preliminary

Because of the lack of hardware devices, we emulate a network situation by a graph read from a file. The node represents the ip address of our testing end device and the edge weight is randomly selected.

a).Dijkstra algorithm:

For a given start node and a network graph, we use Dijkstra algorithm to calculate the shortest path from start node to other nodes. The general idea of this shortest path algorithm is that we keep updating a shortest path

table which contains the current shortest path from start node to this node and its pre node.

Everytime, we set a min cost node in not-visit set as current node and find all its not-visited neighbors, if the min cost between start node and current plus the distance between current node and this neighbor is smaller than the min cost between start node and this neighbor in the table, then we update the min cost table. We keep following this procedure until all nodes are visited.

By using the min cost table, we generate forwarding table for each node by back propagation.

b). Network address translation (NAT):

We use dictionary to store NAT information. NAT_in uses a tuple of public ip address and port number as key and a tuple of local ip address and port number as value, however, NAT_out is versa. The reason why we use two dictionaries is that we want to quickly retrieve values by hashing property. About the update of NAT table, we only monitor NAT_out table, if current pair of local address and port number does not exist in NAT_out key set, we randomly assign an unused port number to it. Meanwhile, update NAT_in table.

ii. Pack and unpack:

Because of the NAT, we need to update not only ip address but also port number. Thus, the pack and unpack procedure are more complicated than IP.

a). Unpack:

In the unpack process, we first unpack the ip datagram into ip header and transport layer segment. Based on the protocol type, we use different transport layer unpack method to get transport layer header information. Notice that when we get the TTL header it may be zero and the protocol is unreachable , in that case we need to send back corresponding ICMP message.

b). Pack:

Before packing, we firstly need to unpack the datagram and update the source ip address and port number according to NAT table. This procedure will be called when both into the LAN and out to WAN.

iii. Send and callback:

a). Send:

In send function, we use the forwarding table of current router to find next hop ip address. After converting the ip into mac address, we can send to link layer.

b). Callback:

The callback function will be used the moment link layer recieving datagram. After unpacking the datagram, if the outside datagram is received by destination end router, it will firstly do NAT and update datagram and then push upwards. If the internal datagram is received by destination, then no need for NAT and push upward directly. It this

datagram is received by core routers, then send to next hop. During the process, if host ip is unreachable, send back corresponding ICMP.

2、Transport Layer

a. TCP-like Protocol

The TCP protocol has two classes, one is TCP object, it's used for pack payload from application layer or unpack data from network layer. Another class is TCP_socket, it's used for conduction all functionalities of TCP protocol.

i. TCP object method

a). from_bytes

This method will load TCP object from bytes, which is used to unpack packet received from network layer.

b). build

This is a metho that build a TCP object with spcial set. Input taken in as flags, bytes, sequance number, acknowledge number and so on.

c). calculate_checksum

This method calculates the checksum of the TCP object, to form a complete one or check whether the TCP object is a good one.

d). pack

This is static method prepared for network layer to use - to pack data.

e). unpack

This is static method for network layer to use - to unpack data.

ii. TCPsocket object

a). listen

This method tells the network layer that this socket is ready to listen. The status of this socket is in LISTEN.

b). accept

If a socket is in LISTEN satus. This method will wait until a SYN packet is retrived. After that, it will conduct the three-way handshake with the client. We also provided a timeout mechisism if there's timeout accured in three-way handshake. The timeout duration is accoring to [RFC 793]. If the handshake compete, it will return another TCPsocket with status of ESTABLISHED for future communication. One thing need to mention is that in the three-way handshake, the first SampleRTT is generated.

c). recv

The socket will wait until it can retrive data from TCPsocket buffer. The length of data maximum is transfered through parameter of this method.

d). send

If the status of this socket is ESTBLISHED, this method will packup a TCP object with correct sequence number generated before and and to

sending_buffer. The data in sending_buffer will be sent through the sending_thread. This way we can make sure that if there is an ACK message and a normal message, it can be send at the same time.

e). _sending_thread

The sending_thread will check every 0.01 seconds to find where there is ACK message or data to be sent. If both are in need, it will packet it to one message. At the same time, the sampleRTT start time will be recorded if not retransmission. And a timer will be started if there is no timer running. The timer will be given a timeout calculated by SampleRTT, dev_RTT. What's more, this method will also check whether there is an timout occured. If so, it will resend the data and cancel the SampleRTT recording for this segment. And the sequence number update, pipeline sending is implemented in this method as well.

f). _add_data

This method is called by network layer when the socket is in ESTABLISHED status. It will check the order of the data. If the data is out of order, it will put it into window_buffer. If not, it will pull out the data in window_buffer in order and add to TCPsocket_buffer. The TCPsocket_buffer is used when recv method is called. And in this exact method, it will check whether the sequence is duplicate or not. If it's duplicate, it will add duplicate counter and resend the last send segment waiting to be acked. If the duplicate counter is 3 or more, it will conduct fast retransmission and restart timer with a doubled timeout. If it's not duplicate, it will add ack number to ack_number_buffer. In the next turn when _sending_thread is performed, the ack number will be sent.

g). connect

This method is used for client to connect server. It will perform three-way handshake and transform status from CLOSED to LISTEN and then to SYN_RECVD and ESTABLISHED. If a timeout occured, it will follow the rule defined by [RFC 793]. The sequence number is generated here randomly for future use. If it tries to connect a server that are not listening, an RST method will return and it will throw an exception.

h). push

This method is for network layer to push data to TCP socket.

i). _get_new_timeout

This method is used for calculate timeout.

j). __check_time

This method is used for timer to check whether there is a timout occured.

k). close

This method is used to close connection.

b. UDP-like Protocol

The UDP protocol has two classes, one is udp object, it's used for managing the data of a udp segment. Another is udp socket, it's used for communicating with socket api.

i.UDP object method

a).pack

Pack method is to add UDP header to a application payload to generate a UDP segment. The header contains source and destination port, length and checksum. When packing, checksum should be calculated.

b).unpack

Unpack method is a reverse of pack. It unpack a segment from network layer and generate a udp object, containing src/dst port number, length, checksum and payload. Checksum will be validated after unpack. If the checksum is incorrect, the packet will be abort.

c).checksum

Partition the data by 16 bytes, sum them and plus 1. return a 16 bytes result.

ii.UDP socket method

a).bind

Do three things: bind local ip and port number to the socket, generate a ip object according to local ip and create a buffer for the port.

b).sendto

Pack the data from application layer to a udp segment. Use the ip object to send it into network layer. If the socket has not bind ip and port, use get_local_ip and get_available_port to get one and bind.

c).recvfrom

Check the buffer according to local port, get a segment. If buffer is empty, wait it.

d).push

This method is called by network layer. If network layer get any udp segment, it call this method and push the segment to buffer according to port number.

e).get_local_ip

Get local ip which can access the internet.

f).get_available_port

Get a avaliable port number that is not used.

3、Application Layer

a. Socket API

bind, connect, listen, accept, send(to), recv(from), close, send/receive buffer

b. Testing Application

i.Echo

A echo app running on tcp, the client send the number 1 to 10 one by one, when server receive message will reply the same number to client . Because of the link layer is unreliable, this app can test the retransmission of tcp.

ii.Chatting room

A chatting room running on udp, a server and multiple clients. Clients can login and chat with each other. The server maintains the user list and forwards the message between clients.

iii.Ping

A ping-like app running on icmp, that can ping a specific ip, get the reply and display the RTT.