Indian Institute of Technology, Kanpur
Computer Science and Engineering

Mid Semester Exam
CS425: Computer Networks

Instructor: Adithya Vadapalli

21/02/2025

Name: ______________________________

Roll Number: ______________________________

This exam contains 19 pages (including this cover page) and 8 questions. Total of marks is 150. You are allowed to bring one A4-sized cheat sheet, which is handwritten (by you!!) notes to the exam. Please write your name in ALL CAPS. You have **120 minutes** to solve the exam. Good luck!

Distribution of Marks

Question	Points	Score
1	10	
2	14	
3	12	
4	6	
5	15	
6	24	
7	41	
8	28	
Total:	150	

1. This question has 10 parts. Each of them is a true or false question.

   (a) (1 point) When a new host enters a network, it uses DHCP to obtain an IP address.

   Solution: True

   (b) (1 point) Since UDP does not guarantee reliable data transfer, there is no checksum in the UDP header.

   Solution: False

   (c) (1 point) A student in IIT D and another student in IIT K claim that their laptop has the same IP address. It is possible that both of them are telling the truth.

   Solution: True

   (d) (1 point) The primary goal of the transport layer is to transport packets from one host to another.

   Solution: False

   (e) (1 point) Recursive DNS is the most preferred type of DNS resolution.

   Solution: False

   (f) (1 point) While doing socket programming in UDP, it is not required to specify the send function's IP address.

   Solution: False

   (g) (1 point) The only purpose of cookies is to track user behaviour.

   Solution: False

(h) (1 point) Since NAT uses port numbers, it resides in the transport layer.

Solution: False

(i) (1 point) Network Layer ensures that packets are transported reliably over an unreliable channel.

Solution: False

(j) (1 point) While doing socket programming in TCP, specifying the IP address in the send function is not required.

Solution: True

2. Suppose you click on a link and obtain a web page within a web browser. Also, suppose that the IP address for the associated URL is not yet cached in your localhost. Therefore, a DNS lookup becomes necessary to obtain the IP address. Suppose that only one DNS server, the local DNS cache, is visited with an an RTT delay of $\texttt{RTT}_0 = 2$ ms. Initially, let's suppose that the Web page associated with the link contains exactly one object, consisting of a small amount of HTML text. Suppose the RTT between the local host and the Web server containing the object is $\texttt{RTT}_{\texttt{HTTP}} = 12$ ms.

   (a) (3 points) Assuming zero transmission time for the HTML object, how much time (in msec) elapses from when the client clicks on the link until the client receives the object?

   Solution: The time from when the Web request is made in the browser until the page is displayed in the browser is: $\texttt{RTT}_0 + 2 \cdot \texttt{RTT}_{HTTP} = 2 + 2 \cdot 12 = 26$ msecs. Note that 2 RTTHTTP are needed to fetch the HTML object - one RTTHTTP to establish the TCP connection, and then one RTTHTTP to perform the HTTP GET/response over that TCP connection

   (b) (3 points) Now suppose the HTML object references 8 very small objects on the same server. Neglecting transmission times, how much time (in msec) elapses from when the client clicks on the link until the base object and all 8 additional objects are received from web server at the client, assuming non-persistent HTTP and no parallel TCP connections?

   Solution: The time from when the Web request is made in the browser until the page is displayed in the browser is: $RTT0+2\cdot\texttt{RTT}_{HTTP}+2\cdot8\cdot\texttt{RTT}_{HTTP} = 2+2\cdot12+2\cdot8\cdot12 = 218$ msecs. Note that two RTTHTTP delays are needed to fetch the base HTML object - one $\texttt{RTT}_{HTTP}$ to establish the TCP connection, and one $\texttt{RTT}_{HTTP}$ to send the HTTP request, and receive the HTTP reply. Then, serially, for each of the 8 embedded objects, a delay of $2 \cdot \texttt{RTT}_{HTTP}$ is needed - one $\texttt{RTT}_{\texttt{HTTP}}$ to establish the TCP connection and then one $\texttt{RTT}_{HTTP}$ to perform the HTTP GET/response over that TCP connection.

(c) (3 points) Suppose the HTML object references 8 very small objects on the same server, but assume that the client is configured to support a maximum of 5 parallel TCP connections, with non-persistent HTTP.

Solution: Since there are 8 objects, there's a delay of 2 msec for the DNS query, two $\texttt{RTT}_{HTTP}$ for the base page, and $4 \cdot \texttt{RTT}_{HTTP}$ for the objects since the requests for 5 of these objects can be run in parallel (2 $\texttt{RTT}_{HTTP}$) and the rest can be done after (2 RTTHTTP). The total is $2 + 24 + 24 + 24 = 74$ msec. As in 2 above, 2 $\texttt{RTT}_{HTTP}$ are needed to fetch the base HTML object - one $\texttt{RTT}_{HTTP}$ to establish the TCP connection, and one $\texttt{RTT}_{HTTP}$ to send the HTTP request and receive the HTTP reply containing the base HTML object. Once the base object is received at the client, the 8 HTTP GETS for the embedded objects can proceed in parallel. Each (in parallel) requires two $\texttt{RTT}_{HTTP}$ delays - one $\texttt{RTT}_{HTTP}$ to set up the TCP connection, and one $\texttt{RTT}_{HTTP}$ to perform the HTTP GET/response for an embedded object.

(d) (3 points) Suppose the HTML object references 8 very small objects on the same server, but assume that the client is configured to support a maximum of 5 parallel TCP connections, with persistent HTTP.

Solution: Since there are 8 objects, there's a delay of 2 msec for the DNS query. There's also a delay of two $\texttt{RTT}_{HTTP}$ for the base page, and 2 $\texttt{RTT}_{HTTP}$ for the objects. The total is $2 + 24 + 24 = 50$ msec.As in 2 and 3 above, two $\texttt{RTT}_{HTTP}$ delays are needed to fetch the base HTML object - one $\texttt{RTT}_{HTTP}$ to establish the TCP connection, and one $\texttt{RTT}_{HTTP}$ to send the HTTP request, and receive the HTTP reply containing the base HTML object. However, with persistent HTTP, this TCP connection will remain open for future HTTP requests, which will therefore not incur a TCP establishment delay. Once the base object is received at the client, the maximum of five requests can proceed in parallel, each retrieving one of the 8 embedded objects. Each (in parallel) requires only one $\texttt{RTT}_{HTTP}$ delay to perform the HTTP GET/response for an embedded object. Once these first five objects have been retrieved, (if necessary) the remaining embedded objects can be retrieved (in parallel). This second round of HTTP GET/response to retreive the remaining embedded objects takes only one more $\texttt{RTT}_{HTTP}$, since the TCP connection has remained open

(e) (2 points) What is the fastest method we have explored: Nonpersistent-serial, Nonpersistent-parallel, or Persistent-parallel?

Solution: The delay when using persistent parallel connections is faster than using nonpersistent parallel connections, which is faster than using nonpersistent serial connections.

3. In this question, we will explore the differences between UDP socket programming and TCP socket programming.

   (a) (3 points) Recall that in TCP server there is a function call to `listen()` , while in UDP there is no such call. Why?

   (b) (3 points) While sending messages in the UDP, IP address is a parameter. However, that is not the case in TCP. Why?

   (c) (3 points) Can a single socket be used to receive messages from multiple IP addresses in TCP? Why?

   (d) (3 points) Can a single socket be used to receive messages from multiple IP addresses in UDP? Why?

4. In this problem, you will compare the time needed to distribute a file that is initially located at a server to clients via either client-server download or peer-to-peer download. The problem is to distribute a file of size F = 9 Gbits to each of these 6 peers. Suppose the server has an upload rate of u = 79 Mbps. The 6 peers have upload and download rates of:

   1. $u_1 = 18$ Mbps, $d_1 = 29$ Mbps (Client 1)
   2. $u_2 = 11$ Mbps, $d_2 = 27$ Mbps (Client 2)
   3. $u_3 = 14$ Mbps, $d_3 = 16$ Mbps (Client 3)
   4. $u_4 = 19$ Mbps, $d_4 = 39$ Mbps (Client 4)
   5. $u_5 = 18$ Mbps, $d_5 = 18$ Mbps (Client 5)
   6. $u_6 = 12$ Mbps, $d_6 = 23$ Mbps (Client 6)

   (a) (2 points) What is the minimum time needed to distribute this file from the central server to the 6 peers using the client-server model?

   Solution: The minimum time needed to distribute the file = max of: N*F / US and F / dmin = 683.54 seconds.

   (b) (2 points) For the previous question, which of the following is the root cause of this specific minimum time? Server, Client 1, Client 2, Client 3, Client 4, Client 5, or Client 6?

   Solution: The root cause of the minimum time was Server.

   (c) (2 points) What is the minimum time needed to distribute this file using peer-to-peer download?

Solution: The minimum time needed to distribute the file = max of: F / US, F / dmin, and N * F / sum of ui for all i + uS = 562.5 seconds.

5. Consider the figure (in the next page) in which a TCP sender and receiver communicate over a connection in which the segments can be lost. The TCP sender wants to send a total of 10 segments to the receiver and sends an initial window of 5 segments at t = 1, 2, 3, 4, and 5, respectively. Suppose the initial value of the sequence number is 180 and every segment sent to the receiver each contains 408 bytes. The delay between the sender and receiver is 7 time units, and so the first segment arrives at the receiver at t = 8, and an ACK for this segment arrives at t = 15. As shown in the figure, 1 of the 5 segments is lost between the sender and the receiver, but none of the ACKs are lost. Assume there are no timeouts and any out-of-order segments received are thrown out.

   (a) (5 points) What is the sequence number of the segment sent at T1, T2, T3, T4, and T5?

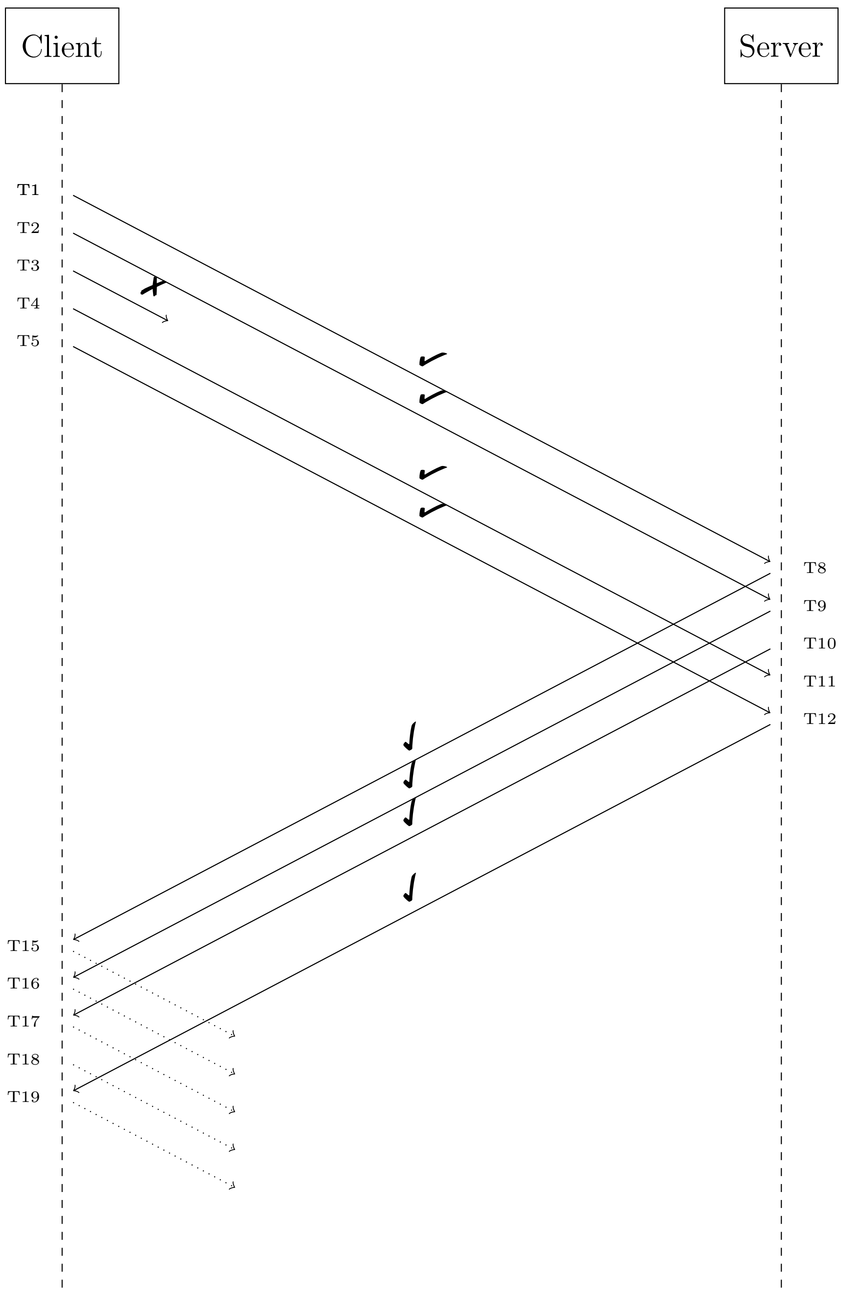
   Solution: 180, 588, 996, 1404, 1812

   (b) (5 points) What is the value of the ACK sent at T8, T9, T10, T11, and T12? (If segment lost, write 'x')

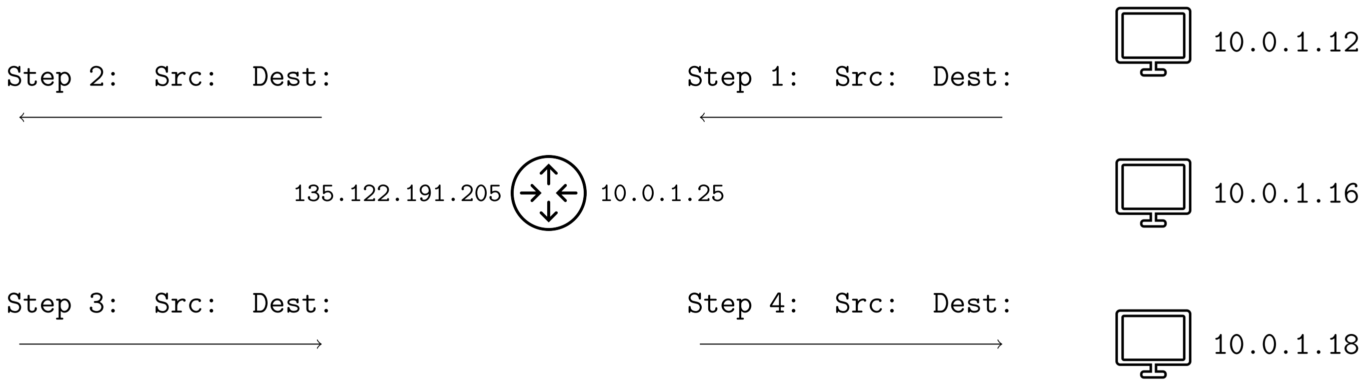
   Solution: 588, 996, 1404, x, 1404

   (c) (5 points) What is the sequence number of the segment sent at T15, T16, T17, T18, and T19? (If segment lost, write 'x')

   Solution: 2220, 2628, 3036, x, x


Client
Server
T1
T2
T3
T4
T5
T8
T9
T10
T11
T12
T15
T16
T17
T18
T19

6. (24 points) Consider the scenario below in which three hosts, with private IP addresses 10.0.1.12, 10.0.1.16, 10.0.1.18 are in a local network behind a NAT'd router that sits between these three hosts and the larger Internet. IP datagrams being sent from, or destined to, these three hosts must pass through this NAT router. The router's interface on the LAN side has IP address 10.0.1.25, while the router's address on the Internet side has IP address 135.122.191.205. Suppose that the host with IP address 10.0.1.12 sends an IP datagram destined to host 128.119.165.183. The source port is 3416, and the destination port is 80.

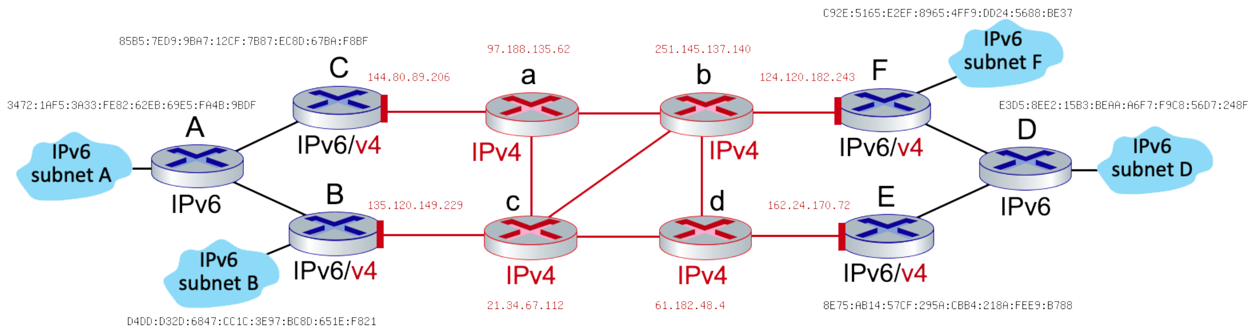


Complete the following table:

Step #	Source IP Address	Destination IP Address	Src Port Change? (Yes / No)
Step 1			
Step 2			
Step 3			
Step 4			

Solution: The table is filled as follows:

Step #	Source Address	Destination Address	Src Port Change? (Yes / No)
Step 1	10.0.1.12	128.119.165.183	
Step 2	135.122.191.205	128.119.165.183	
Step 3	128.119.165.183	135.122.191.205	
Step 4	128.119.165.183	10.0.1.12	


85B5:7ED9:9BA7:12CF:7B87:EC8D:67BA:F8BF
C
144.80.89.206
97.188.135.62
a
251.145.137.140
b
124.120.182.243
F
C92E:5165:E2EF:8965:4FF9:DD24:5688:BE37
IPv6
subnet F
3472:1AF5:3A33:FE82:62EB:69E5:FA4B:9BDF
A
IPv6
subnet A
IPv6
IPv6/v4
IPv4
IPv4
IPv6/v4
E3D5:8EE2:15B3:BEAA:A6F7:F9C8:56D7:248F
D
IPv6
subnet D
IPv6
B
135.120.149.229
c
d
162.24.170.72
E
IPv6
subnet B
IPv6/v4
IPv4
IPv4
IPv6/v4
D4DD:D32D:6847:CC1C:3E97:BC8D:651E:F821
21.34.67.112
61.182.48.4
8E75:AB14:57CF:295A:CBB4:218A:FEE9:B788

7. Consider the network shown below which contains four IPv6 subnets, connected by a mix of IPv6-only routers, IPv4-only routers and dual-capable IPv6/IPv4 routers. Suppose that a host of subnet A wants to send an IPv6 datagram to a host on subnet D. Assume that the forwarding between these two hosts goes along the path: $A \rightarrow C \rightarrow a \rightarrow c \rightarrow b \rightarrow F \rightarrow D$.

   (a) (28 points) Fill in the following table:

Datagram	IPv4 or v6	Encapsulating (Y/N)	Source IP Address	Destination IP Address
A to C				
from C				
c to b				
b to F				
F to D				
a to c				
F to D				

Solution: The table is filled as below:

			Source	Destination
A to C	IPv6	NO	3472:1AF5:3A33:FE82:62EB:69E5:FA4B:9BDF	E3D5:8EE2:15B3:BEAA:A6F7:F9C8:56D7:248F
from C	IPv4	YES	144.80.89.206	124.120.182.243
c to b	IPv4	YES	144.80.89.206	124.120.182.243
b to F	IPv4	YES	144.80.89.206	124.120.182.243
F to D	IPv6	NO	3472:1AF5:3A33:FE82:62EB:69E5:FA4B:9BDF	E3D5:8EE2:15B3:BEAA:A6F7:F9C8:56D7:248F
a to c	IPv4	YES	144.80.89.206	124.120.182.243
F to D	IPv6	NO	3472:1AF5:3A33:FE82:62EB:69E5:FA4B:9BDF	E3D5:8EE2:15B3:BEAA:A6F7:F9C8:56D7:248F

(b) (7 points) In the above question, for all the encapsulated packets is encapsulated, what are the Source and Destination Addresses (of the encapsulated packets)? If there is no encapsulation in any of the below datagrams in the table, just write "NA".

Datagram	Source IP Address	Destination IP Address
A to C		
from C		
c to b		
b to F		
F to D		
a to c		
F to D		

(c) (2 points) What router is the *tunnel entrance*? Give the router's letter

Solution: The tunnel entrance is router C

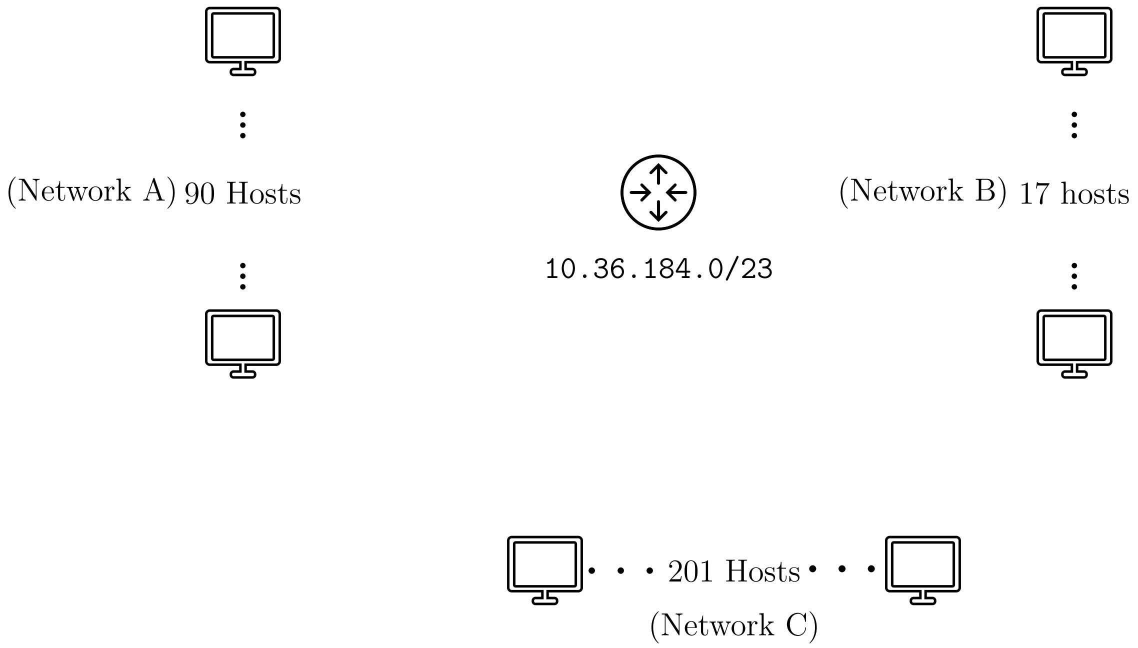
(d) (2 points) What router is the *tunnel exit*? Give the router's letter

Solution: The tunnel entrance is router F

(e) (2 points) Which protocol encapsulates the other, IPv4 or IPv6?

Solution: IPv4, in order to maintain compatibility with existing IPv4 infrastructure, IPv6 datagrams are put in the payload of an IPv4 datagram. These IPv4 datagrams are passed on until it reaches a router which supports IPv6, where the IPv6 datagram is decapsulated and passed on.

8. Consider the router and the three attached subnets below (A, B, and C). The number of hosts is also shown below. The subnets share the 23 high-order bits of the address space: 10.36.184.0/23. Assign subnet addresses to each of the subnets (A, B, and C) so that the amount of address space assigned is minimal, and at the same time leaving the largest possible contiguous address space available for assignment if a new subnet were to be added. Then answer the questions below. *(Hint: The last address in the subnet is the broadcast address and the first address is the subnet ID!)*



(a) (2 points) Is the address space public or private?

Solution: Private

(b) (2 points) How many hosts can there be in this address space?

Solution: Maximum number of hosts = $2^x - 2 = 2^9 - 2 = 510$. The reason we have to subtract 2 from the final number is because there are always 2 addresses allocated for each address block: the subnet ID (the first address) and the broadcast address (the last address); for example, if you have 5 bits for hosts, you can have 30 hosts, because 2 of the addresses are for the subnet ID and the broadcast address which when added equals 32, which is 2^5.

(c) (24 points) Fill in the following table:

Network	Subnet Addr (CIDR)	Broadcast Address	Starting Address	Ending Address
A				
B				
C				

Solution: The table is filled as follows:

Network	Subnet Addr	Broadcast Addr	Starting Addr	Ending Addr
A	10.36.185.0/25	10.36.185.127	10.36.185.1	10.36.185.126
B	10.36.185.128/27	10.36.185.159	10.36.185.129	10.36.185.158
C	10.36.184.0/24	10.36.184.255	10.36.184.1	10.36.184.254