Detailed Design

Requested by: Mr. Ken Swarner
Systems Administrator
Computer Science Department of Siena College

TCP/IP Packet Descriptor

Mirage Incorporated

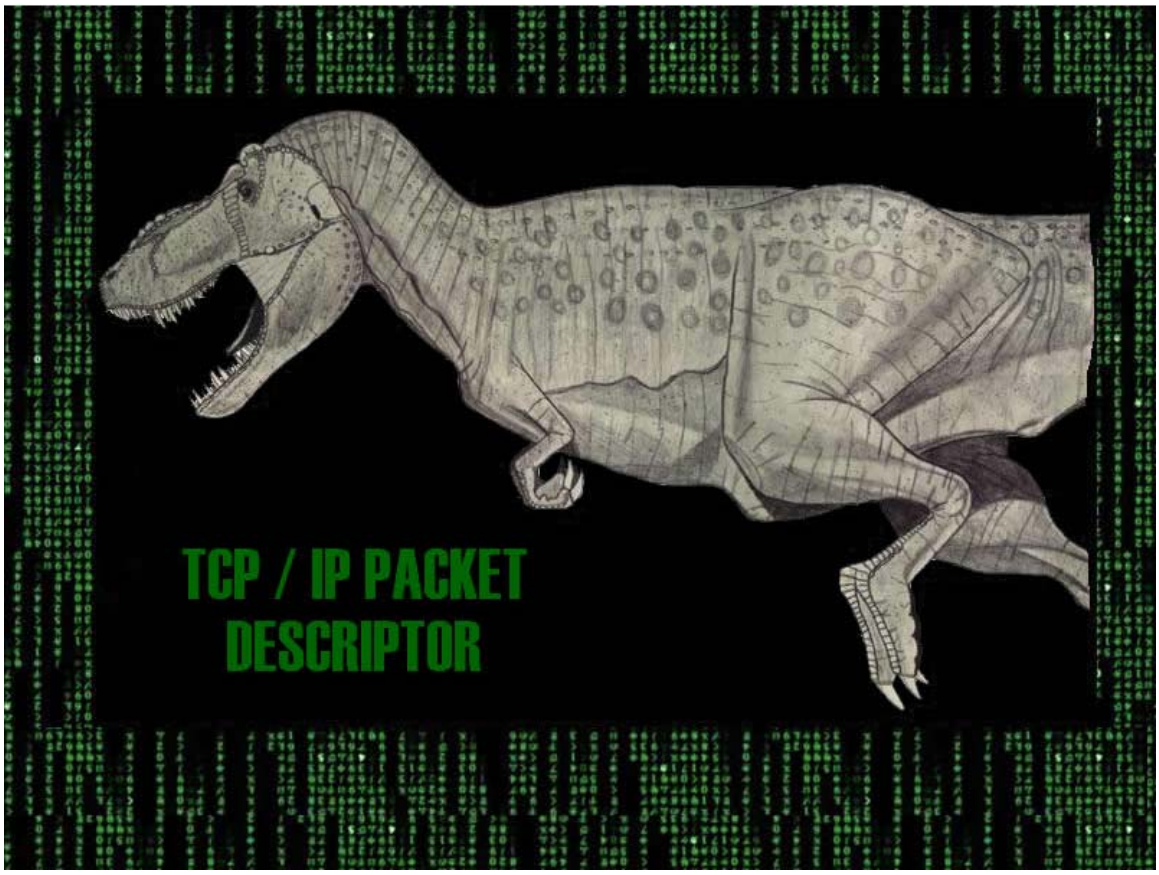
"We are there…even if you cannot see us"
Mirageinc2003@yahoo.com

Prepared by: Paul Aiuto
Richard Connell
Lauren Englisbe
Jayme Gresen, Team Leader
Jeffrey Habiniak

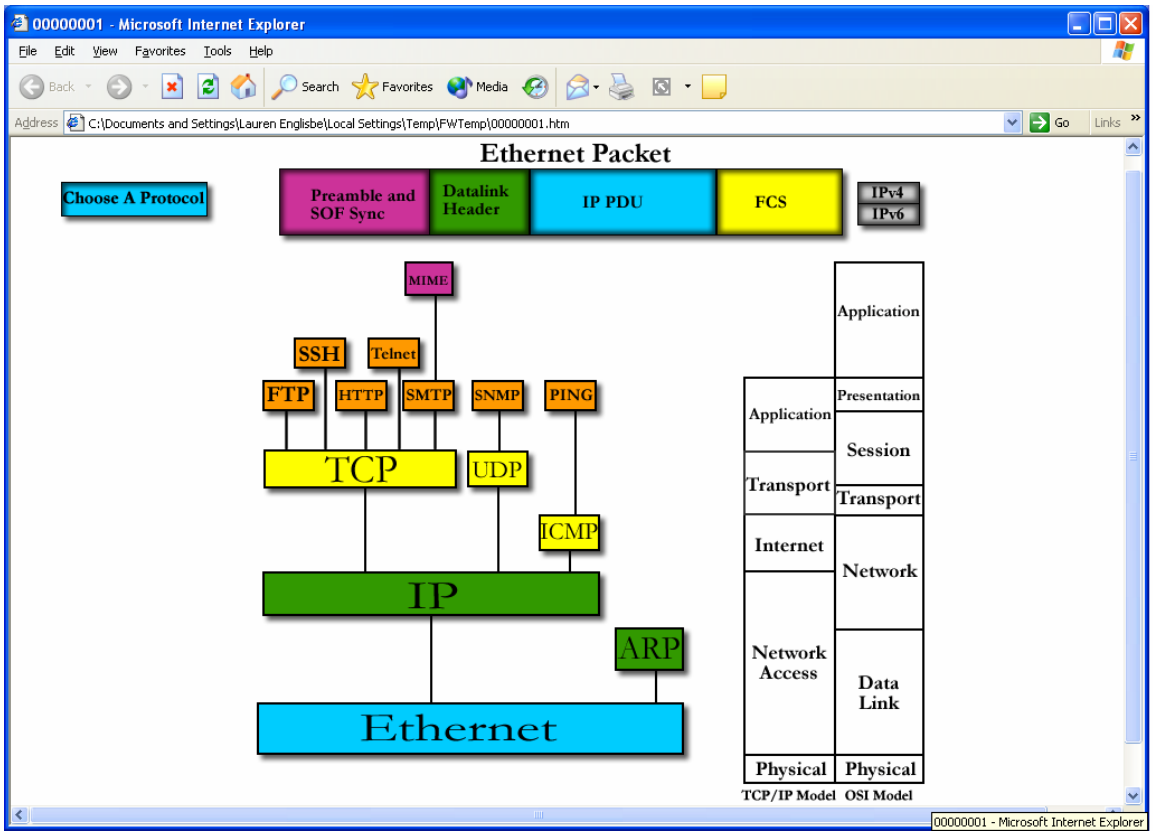
Preliminary Design
Table of Contents

1.0 External Design Specifications

1.1 User Displays

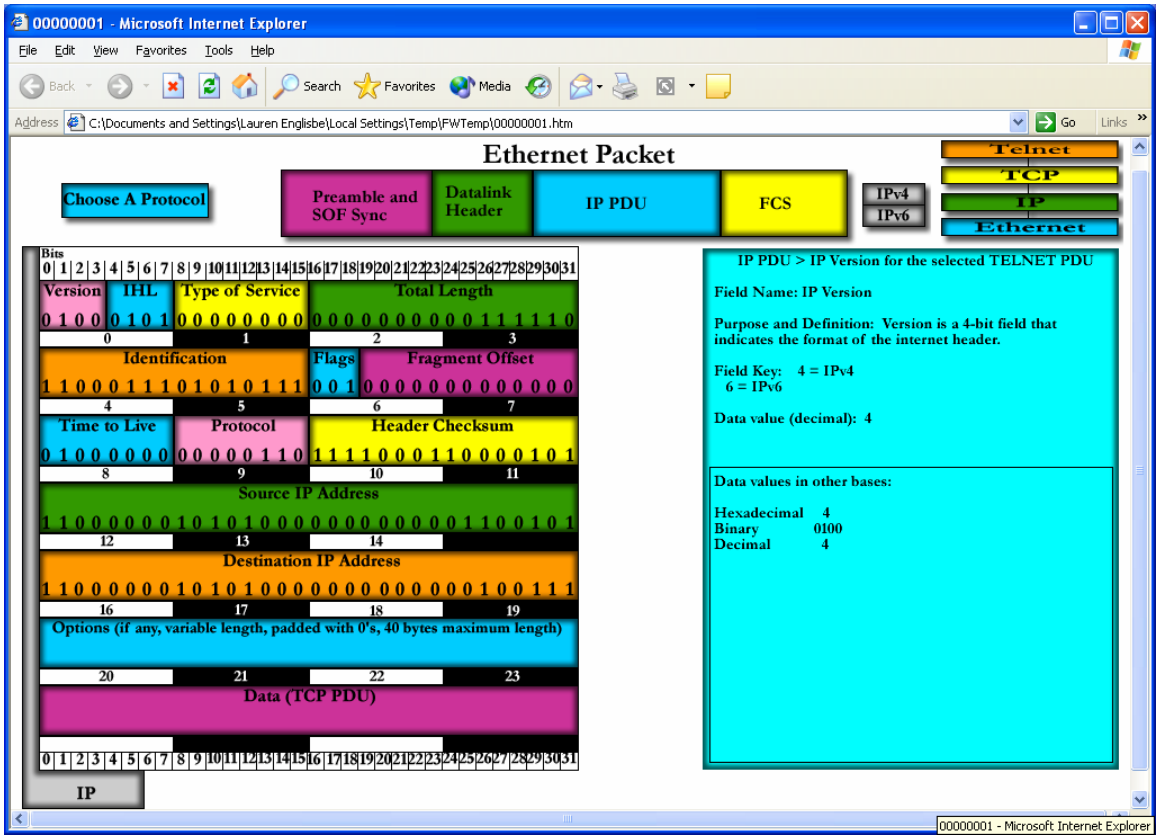


This will be the first screen the user sees. It is the introduction screen to our software, and presents our mascot, “The Descript-roar”.



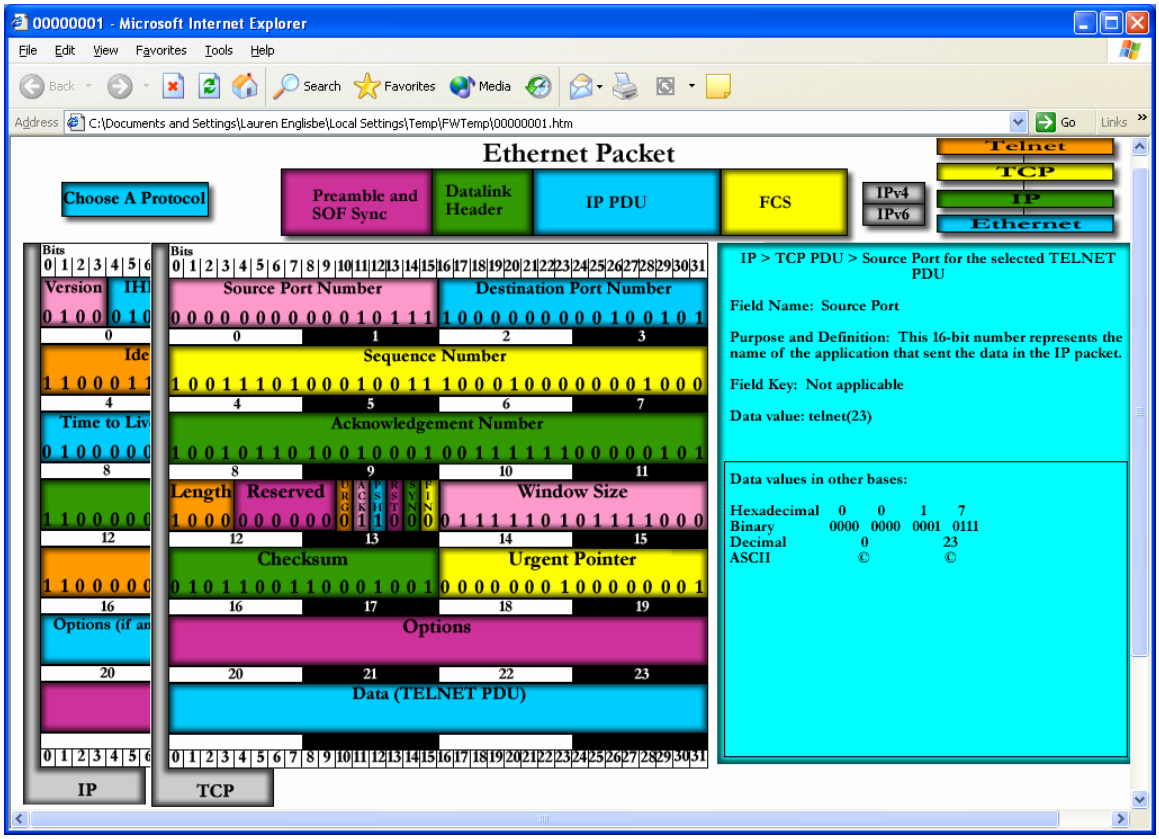
This is the first functional screen the user will see. It allows the user to see what an Ethernet Packet looks like, and eventually each frame within that packet will be clickable and able to display information about that frame.

The “Choose a Protocol” menu gives the user a graphical representation of how protocols are interrelated, and allows the desired protocol to be selected.

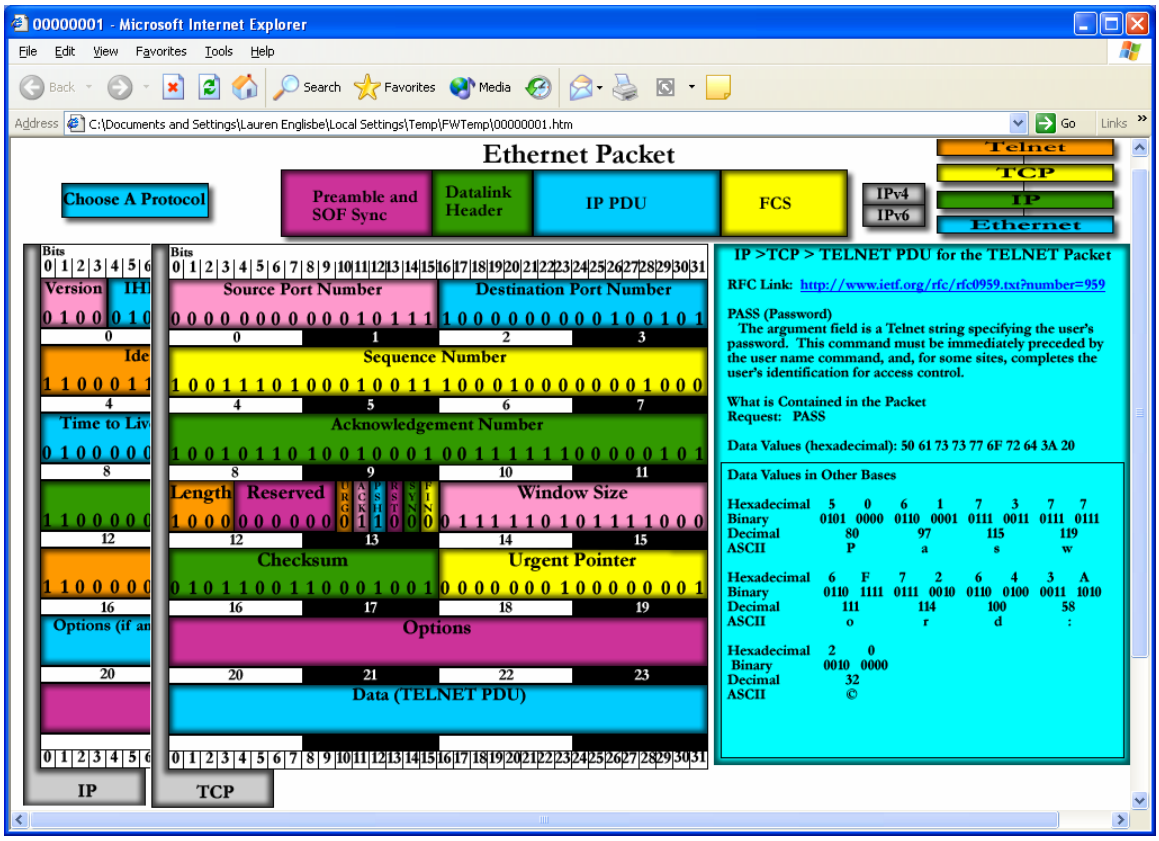


Once a protocol has been selected by the user, this screen will display. The basis for our protocol suite, the IP PDU is displayed on the left, filled in with the given data for the selected protocol. The user is able to click on any field in the IP PDU, and an information box will display on the right, describing that field. In the IP PDU Data field, "TCP PDU" is written – this indicates that the entire TCP PDU is contained within the IP Data field. If the user clicks on this field, the TCP PDU will be displayed.

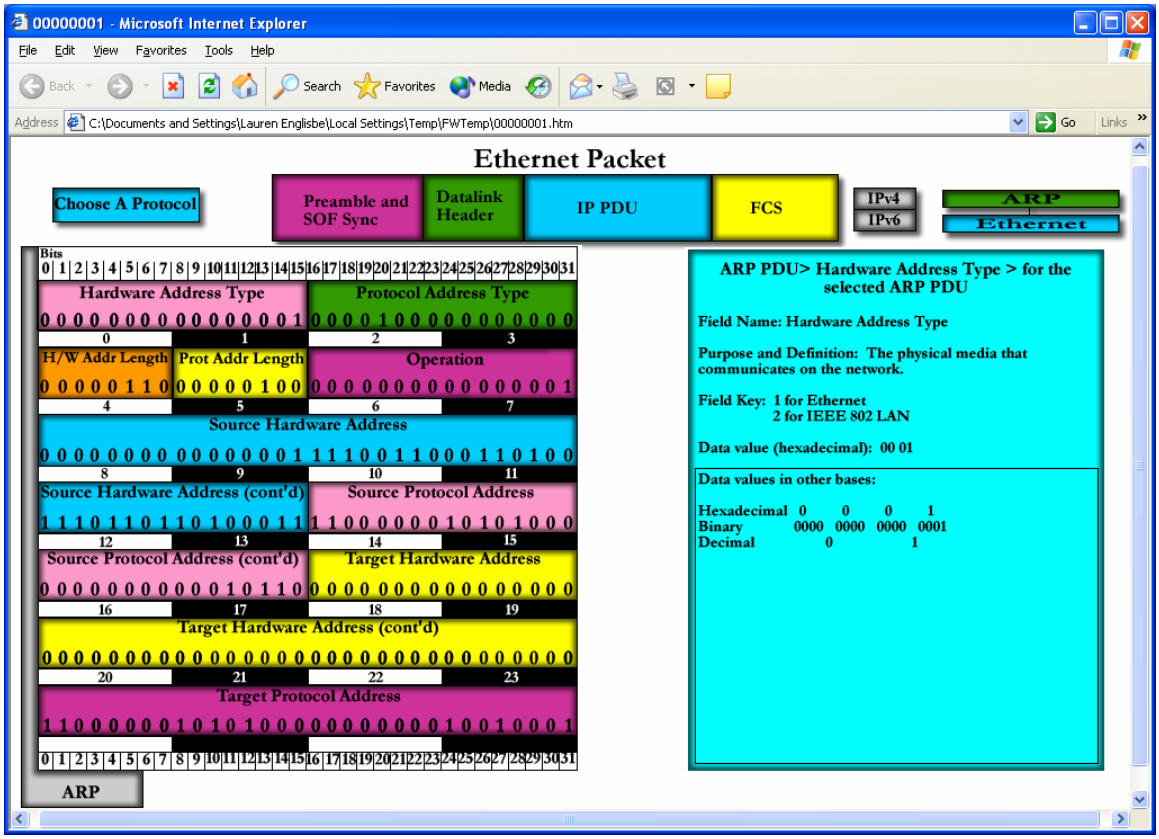
The PDU's are tabbed on their lower left corners: this allows the user to navigate between them. Additionally, there is a key at the top right corner to display what PDUs make up the chosen protocol. The user may also click on any PDU in this map to navigate. If at any point, the user wants to choose a different protocol to view, he or she may click on the "Choose a Protocol" button in the top left corner.



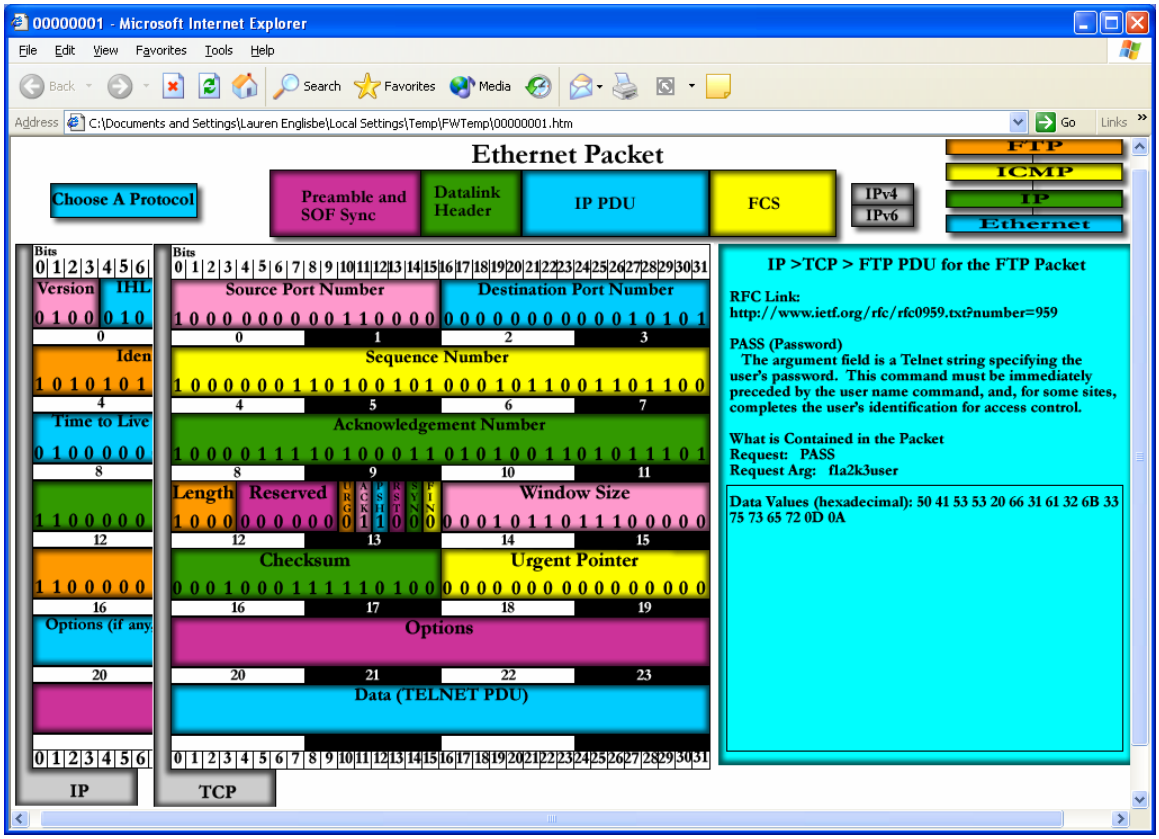
The next screen shows the TCP PDU on top of the IP PDU. This is meant to show that the TCP PDU is contained within the IP PDU. Additionally, the Telnet PDU is contained within the TCP PDU Data field, so clicking on that field will display the Telnet PDU. As with IP, if a TCP field is clicked, the information for that field will be displayed in the information box at the right.



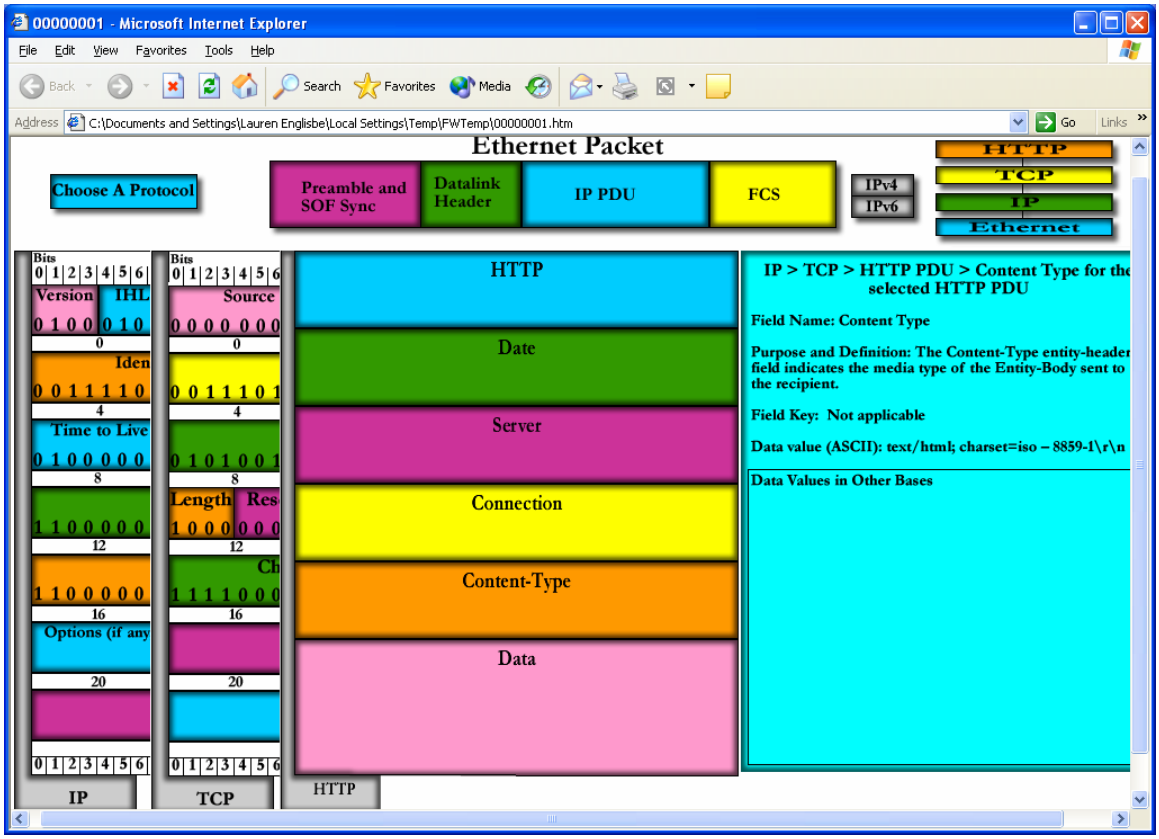
This is the software with all the information displayed for the selected Telnet PDU. The IP PDU gives rise to the TCP PDU, and the TCP Data field has been clicked, which allows us to see the information field for the Telnet PDU on the right.



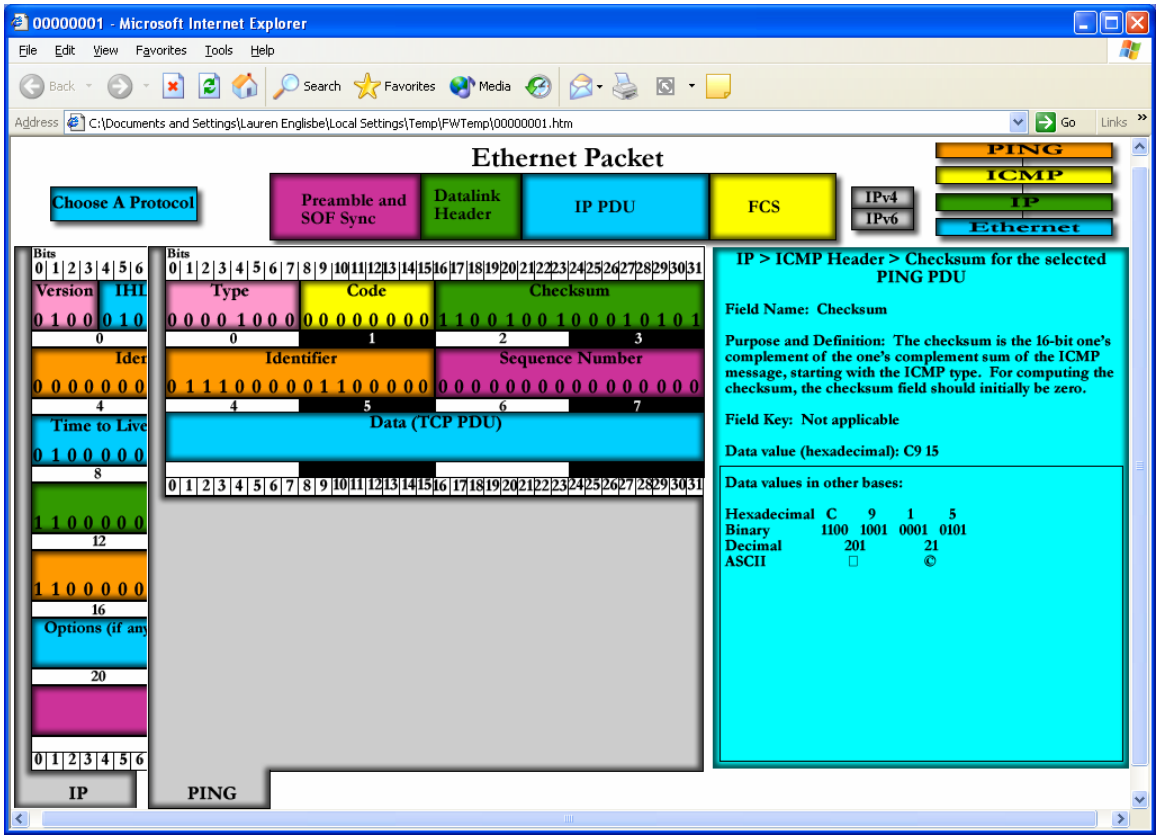
This is a screenshot of the ARP protocol and its information field.



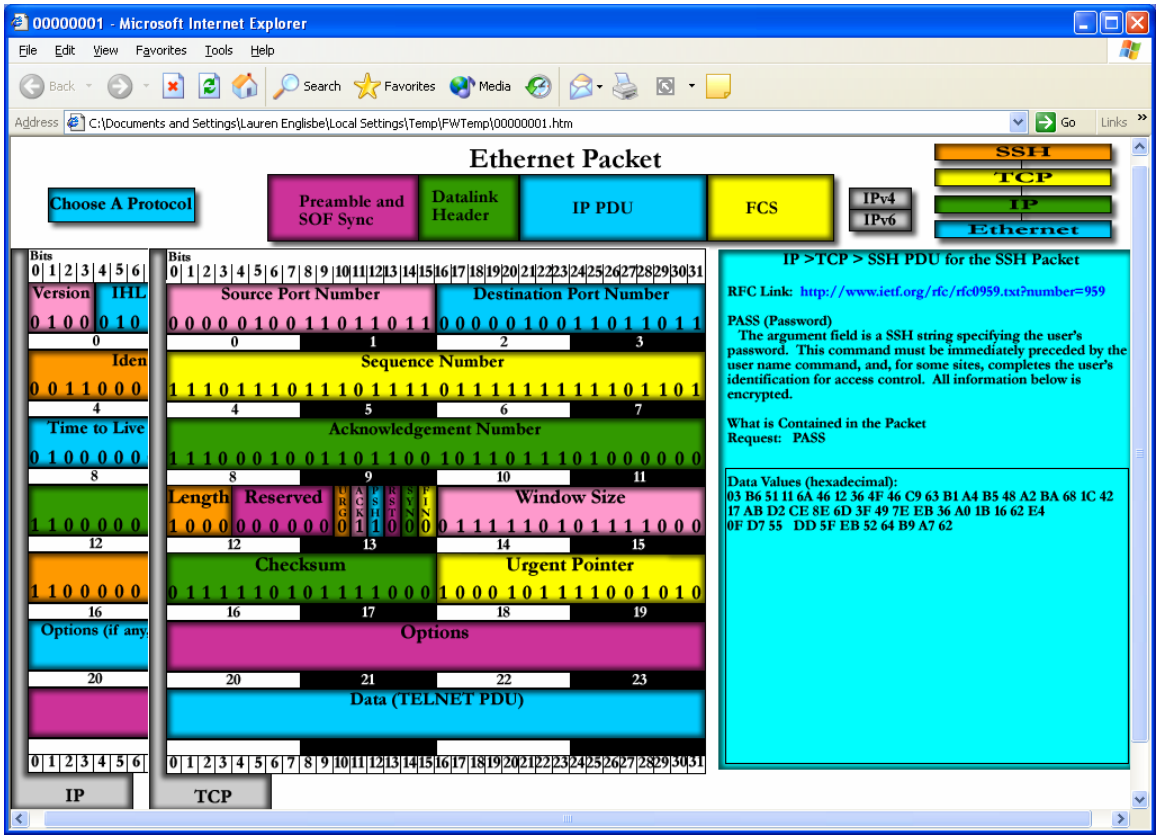
FTP Protocol



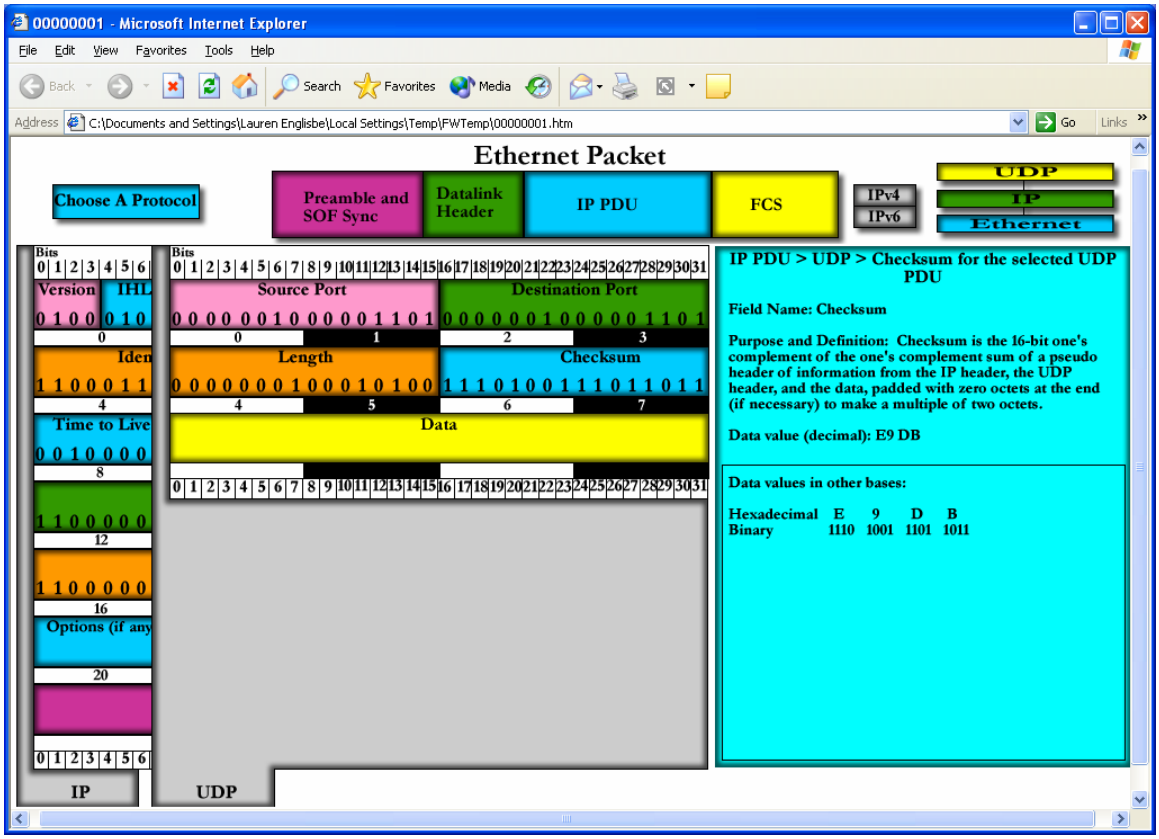
HTTP Protocol



PING Protocol



SSH Protocol



UDP Protocol

1.2 User Command Summary

Main Screen
This is the screen giving the user the option to choose many different packets.

PDU Hierarchy Tree
Allows the user to see their progression through the many packets, and also able to choose their desired packet.

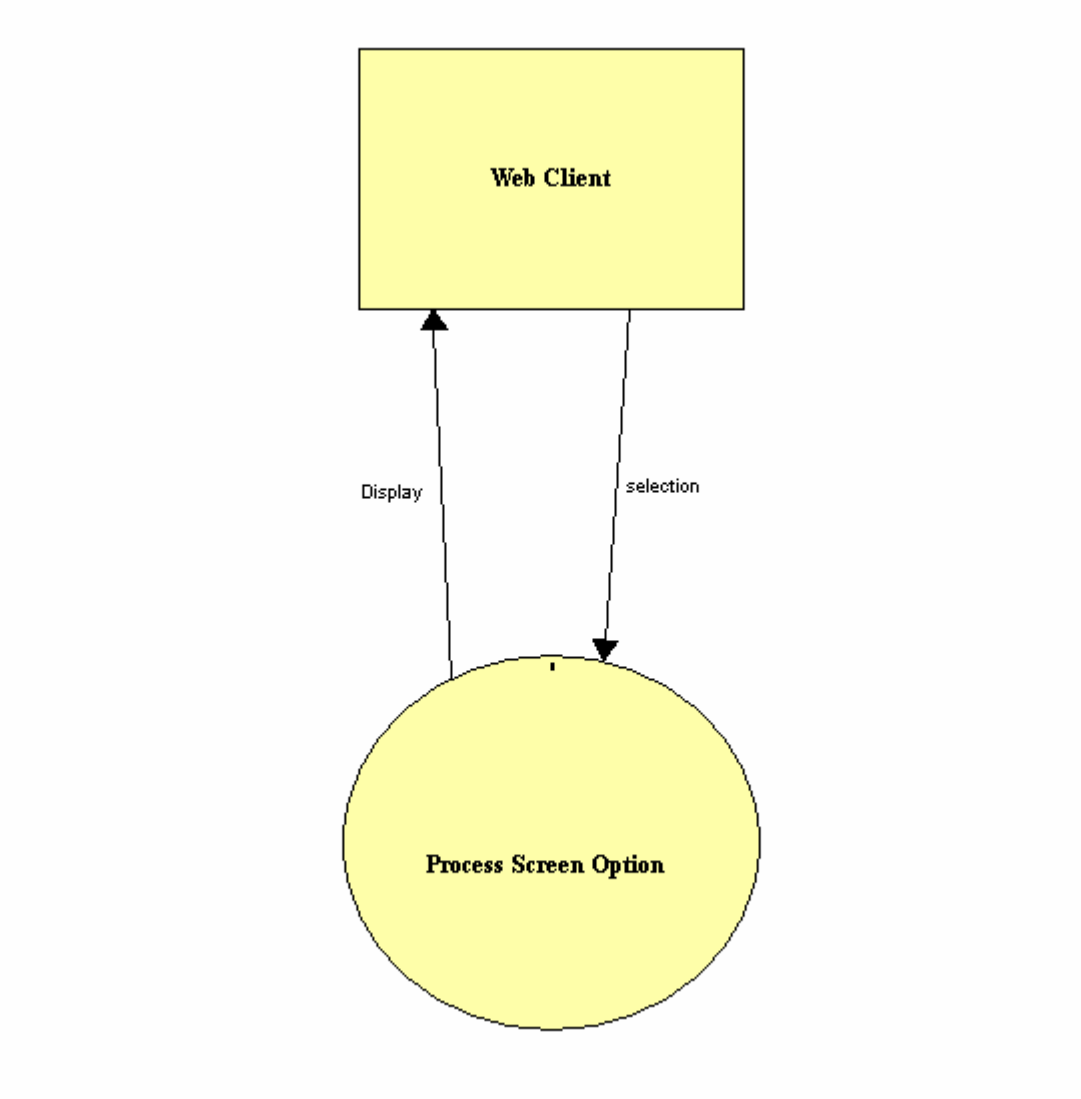
IP Version
Allows the user two chooses of two different IP Versions.

Radix (Base) Selection
Allows the user to select a radix to display.

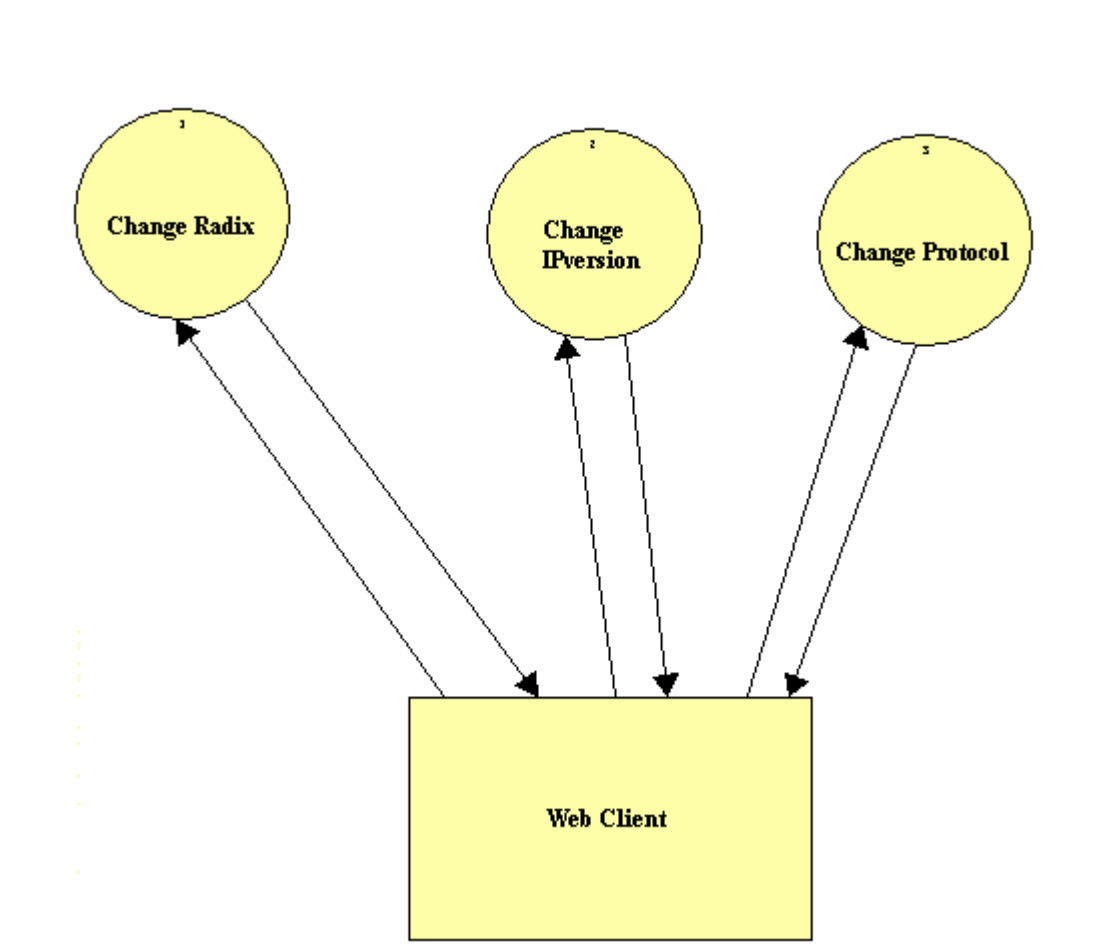
Information Box
Allows the user to see the given information for a selected field.

1.3 Detailed Data Flow Diagrams

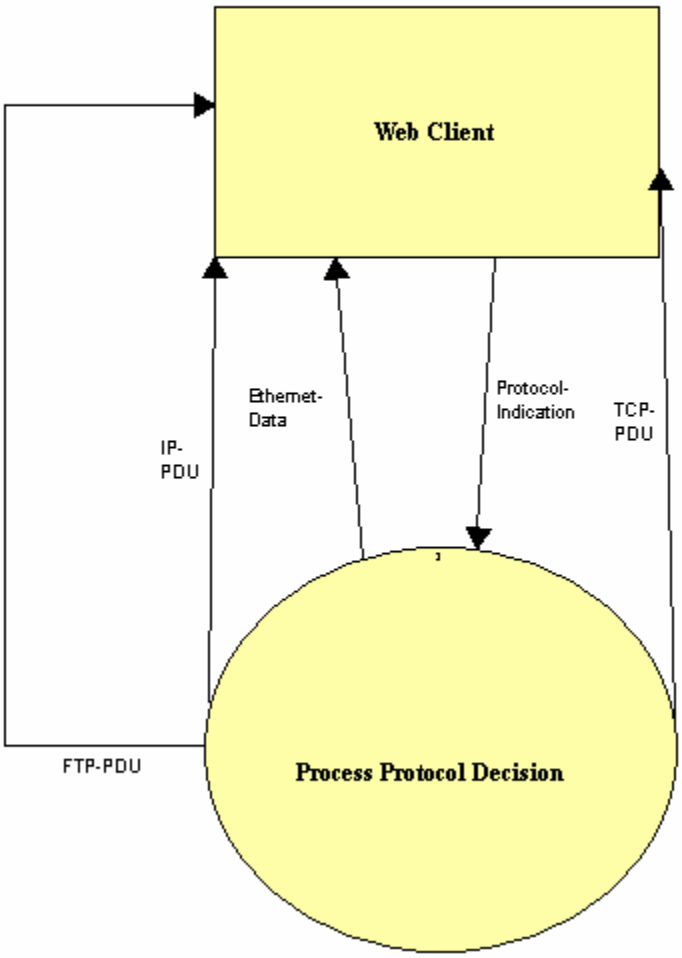
Level 0 Diagram:



Context Diagram:



Detailed Diagram:



2.0 Architectural Design Specifications

2.1 User Commands (AKA “Clickable Buttons”)

IP PDU

IP Version
Internet Header Length
Type of Service
Total Length of Ethernet Frame
Identification
Flags
Fragment Offset
Time to Live
Protocol
Header Checksum
Source IP Address
Destination IP Address
Options
Data

TCP PDU

Source Port Number
Destination Port Number
Sequence Number
Acknowledgement Number
Header Lengths
Reserved
Window Size
TCP Checksum
Urgent Pointer
Options
Data

2.2 Functional Descriptions

2.2.1 IP PDU for the selected FTP PDU

IP PDU > *IP Version* for the selected FTP PDU

Field Name: *IP Version*

Purpose and Definition: Version is a 4-bit field that indicates the format of the internet header.

Field Key: 4 = IPv4
6 = IPv6

Data value (decimal): 4

Data values in other bases:

Hexadecimal	4
Binary	0100
Decimal	4

Programming Hint: The name for this variable in code will be IP_IPVersion_FTP.

IP PDU > *Internet Header Length* for the selected FTP PDU

Field Name: *Internet Header Length*

Purpose and Definition: The IHL field is a 4-bit field indicating the length of the internet header in 32 bit words, and thus points to the beginning of the data. The minimum value of a correct header is 5.

Field Key: *Not applicable*

Data value: The value contained in our field is 20 bytes. This is the hexadecimal and decimal value of 5 multiplied by 4 bits.

Data values in other bases:

Hexadecimal	0	5
Binary	0000	0101
Decimal	5	

Programming Hint: The name for this variable in code will be IP_IHL_FTP.

IP PDU > *Type of Service* for the selected FTP PDU

Field Name: *Type of Service*

Purpose and Definition: Type of Service is an 8-bit field that provides and indication of the abstract parameters of the quality of service desired. These parameters guide the selection of the actual service parameters when transmitting a datagram through a particular network.

Field Key: The major choice is a three-way tradeoff between low-delay, high-reliability, and high-throughput.

0 1 2	3	4	5	6	7
Precedence	D	T	R	0	0

Bits 0-2: Precedence
Bit 3: (D) 0 = Normal Delay 1 = Low Delay
Bit 4: (T) 0 = Normal Throughput 1 = High Throughput
Bit 5: (R) 0 = Normal Reliability 1 = High Reliability

Precedence:

111 = Network Control	011 = Flash
110 = Inter-network Control	010 = Immediate
101 = CRITIC/ECP	001 = Priority
100 = Flash Overrided	000 = Routine

Data value (hexadecimal): 10

Data values in other bases:

Hexadecimal	1	0
Binary	0001	0000
Decimal	16	

Programming Hint: The name for this variable in code will be IP_TypeOfService_FTP.

IP PDU > *Total Length of Ethernet Frame* for the selected FTP PDU

Field Name: *Total Length of Ethernet Frame*

Purpose and Definition: Total Length is a 16-bit field that indicates the length of the frame, measured in octets, including internet header and data. The maximum size is 2^{16}-1 or 65,535 octets; however, the recommended maximum size is 576 octets.

Field Key: *Not applicable*

Data values (hexadecimal): 69

Data values in other bases:

Hexadecimal	0	0	6	9
Binary	0000	0000	0110	1001
Decimal	0		105	
ASCII	©		i	

Programming Hint: The name for this variable in code will be IP_TotalLength_FTP.

IP PDU > *Identification* for the selected FTP PDU

Field Name: *Identification*

Purpose and Definition: Identification is a 16-bit field. An identifying value is assigned by the sender to aid in assembling the fragments of a datagram. The identifier is chosen based on the need to provide a way to uniquely identify the fragments and protocol for the time the datagram or any fragment could be alive in the internet.

Field Key: *Not applicable*

Data value (hexadecimal): AA 41

Data values in other bases:

Hexadecimal	A	A	4	1
Binary	1010	1010	0100	0001

Programming Hint: The name for this variable in code will be IP_Idenfification_FTP.

IP PDU > *Flags* for the selected FTP PDU

Field Name: *Flags*

Purpose and Definition: Flags is a 3-bit field that indicates directions for fragmentation.

Field Key:
Bit 0: reserved, must be 0
Bit 1: (DF) 0 = May Fragment 1 = Don't Fragment
Bit 2: (MF) 0 = Last Fragment 1 = More Fragment

Data value (binary): 010

Data values in other bases: *Not applicable*

Programming Hint: The name for this variable in code will be IP_Flags_FTP.

IP PDU > *Fragment Offset* for the selected FTP PDU

Field Name: *Fragment Offset*

Purpose and Definition: The Fragment Offset is a 13- bit field indicating where in the Ethernet frame this fragment begins. The Fragment Offset is measured in units of 8 octets, and the first fragment has offset 0.

Field Key: *Not applicable*

Data value (decimal): 0

Data values in other bases:
Binary: 0 0000 0000 0000

Programming Hint: The name for this variable in code will be IP_FragmentOffset_FTP.

IP PDU > *Time to Live* for the selected FTP PDU

Field Name: *Time to Live*

Purpose and Definition: Time to Live is an 8-bit field that indicates the maximum time the datagram is allowed to remain in the internet. If this field contains the value 0, then the datagram must be destroyed. This field is modified in internet header processing. The time is measure in units of seconds, and is set by the sender to the maximum time the datagram is allowed to be in the internet. This field is decreased at each point that the internet header is processed. The intention is to cause undeliverable packets to be discarded, and to bind the maximum datagram lifetime.

Field Key: *Not applicable*

Data value (decimal): 64

Data values in other bases:

Hexadecimal	4	0
Binary	0100	0000
Decimal	64	

Programming Hint: The name for this variable in code will be IP_TimeToLive_FTP.

IP PDU > *Protocol* for the selected FTP PDU

Field Name: *Protocol*

Purpose and Definition: Protocol is an 8-bit field that indicates the next level protocol that is used in the data portion of the internet diagram.

Field Key:

Dec	Hex	Protocol	Dec	Hex	Protocol
0	00	Reserved	22	16	Multiplexing
1	01	ICMP	23	17	DCN
2	02	Unassigned	24	18	TAC Monitoring
3	03	Gateway-to-Gateway	25-76	19-4C	Unassigned
4	04	CMCC Gateway Monitoring Message	77	4D	Any local network
5	05	ST	100	64	SATNET and Backroom EXPAK
6	06	TCP	101	65	MIT Subnet Support
7	07	UCL	102-104	66-68	Unassigned
10	0A	Unassigned	105	69	SATNET Monitoring
11	0B	Secure	106	6A	Unassigned
12	0C	BBN RCC Monitoring	107	6B	Internet Packet Core Utility
13	0D	NVP	110-113	6E-71	Unassigned
14	0E	PUP	114	72	Backroom SATNET Monitoring
15	0F	Pluribus	115	73	Unassigned
16	10	Telnet	116	74	WIDEBAND Monitoring
17	11	XNET	117	75	WIDEBAND EXPAK
20	14	Chaos	120-376	78-0178	Unassigned
21	15	User Datagram	377	0179	Reserved

Data value (hexadecimal): 06

Data values in other bases:

Hexadecimal	0	6
Binary	0000	0110
Decimal	6	

RFC Link: http://www.faqs.org/rfcs/rfc790.html

Programming Hint: The name for this variable in code will be IP_Protocol_FTP.

IP PDU > *Header Checksum* for the selected FTP PDU

Field Name: *Header Checksum*

Purpose and Definition: The Header Checksum is a 16-bit field. The Checksum is the 16-bit one's complement sum of all 16-bit words in the header. For purposes of computing the checksum, the initial value of its field is zero. When both header checksums are equal, then the header bits are correct. If either checksums vary, then a new, correct packet will need to be sent.

This is a simple way to compute the checksum and experimental evidence indicates that it is adequate, but it is provisional and may be replaced by a CRC procedure, depending on further experience.

Field Key: *Not applicable*

Data value (hexadecimal): 0E 85

Data values in other bases:

Hexadecimal	0	E	8	5
Binary	0000	1110	1000	0101

Programming Hint: The name for this variable in code will be IP_HeaderChecksum_FTP.

IP PDU > *Source Address* for the selected FTP PDU

Field Name: *Source Address*

Purpose and Definition: The Source Address is a 32-bit field that contains the IP address of the host that sent the IP Packet.

Field Key: *Not applicable*

Data value (decimal): 192.168.0.39

Data values in other bases:

Hexadecimal	C	0	A	8	0	0	2	7
Binary	1100	0000	1010	1000	0000	0000	0010	0111
Decimal	192		168		0		39	

Programming Hint: The name for this variable in code will be IP_SourceAddress_FTP.

IP PDU > *Destination Address* for the selected FTP PDU

Field Name: *Destination Address*

Purpose and Definition: The Destination Address is a 32-bit field that contains the address of the host that is to receive the data contained within the IP packet.

Field Key: *Not applicable*

Data value (decimal): 192.168.0.101

Data values in other bases:

Hexadecimal	C	0	A	8	0	0	6	5
Binary	1100	0000	1010	1000	0000	0000	0110	0101
Decimal	192		168		0		101	

Programming Hint: The name for this variable in code will be IP_DestinationAddress_FTP.

IP PDU > *Options and Padding* for the selected FTP PDU

Field Name: *Options and Padding*

Purpose and Definition: The options may or may not appear in Ethernet packets. They must be implemented by all IP modules (host and gateways). What is optional is their transmission in any particular packet, not their implementation.

The option field is variable in length. There may be zero or more options. There are two cases for the format of an option.

Case 1: A single octet of option type

Case 2: An option-type octet, an option-length octet, and the actual option-data octets.

Field Key: *Not applicable*

Data values: *Not applicable*

Data values in other bases: *Not applicable*

Programming Hint: The name for this variable in code will be IP_OptionsPadding_FTP.

IP PDU > *Data* for the selected FTP PDU

Field Name: *Data*

Purpose and Definition: The Data is a variable length field which contains the actual data that is being sent from one host to another. The data field may start with a Layer 4 header, which will give additional instructions to the application that will be receiving the data; alternately, it may be an ICMP header and not contain any user data at all.

Field Key: *Not applicable*

Data values (hexadecimal) : (TCP) 80 30 00 15 81 A5 16 6C 87 A3 53 5D 80 18 16 D0 11 F4 00 00 01 01 08 0A 1B 25 F3 A1 0b DD 73 58
(FTP) 50 41 53 53 20 66 31 61 32 6B 33 75 73 65 72 0D 0A

Data values in other bases:
Hexadecimal: (TCP) 0 x 80 30 00 15 81 A5 16 6C 87 A3 53 5D 80 18 16 D0 11 F4 00 00 01 01 08 0A 1B 25 F3 A1 0B DD 73 58
(FTP) 50 41 53 53 20 66 31 61 32 6B 33 75 73 65 72 0D 0A

ASCII: (TCP) ↑ 0 © © ↑ ↑ © ↑ ↑ S] ↑ © © ↑ © ↑ © © © © © © © % ↑ ↑ © ↑ s X
(FTP) P A S S S © f 1 a 2 k 3 u s e r © ©

Programming Hint: The name for this variable in code will be IP_Data_FTP.

2.2.2 TCP PDU for the selected FTP PDU

IP > TCP PDU > *Source Port* for the selected FTP PDU

Field Name: *Source Port*

Purpose and Definition:
This 16-bit number represents the name of the application that sent the data in the IP packet.

Field Key: *Not applicable*

Data value (decimal): 32816

Data values in other bases:

Hexadecimal	8	0	3	0
Binary	1000	0000	0011	0000
Decimal	128		48	
ASCII	↑		0	

Programming Hint: The name for this variable in code will be IP_TCP_SourcePort_FTP.

IP > TCP PDU > *Destination Port* for the selected FTP PDU

Field Name: *Destination Port*

Purpose and Definition:
This 16-bit number represents the name of the application that is to receive the data contained within the IP packet. This is one of the major differences between a Layer 3 and a Layer 4 header: the Layer 3 header contains the IP address of the computer that is to receive the IP packet; once that packet has been received, the port address in the Layer 4 header ensures that the data contained within that IP packet is passed to the correct application on that computer.

Field Key:
This key indicates assigned port number values:

Dec	**Port Numbers**
0	Reserved
1-32767	Internet registered ("well-known") protocols
32768-98303	Reserved, to allow TCPv7-TCPv4 conversion
98304 & up	Dynamic assignment

Data value (decimal): 21 (indicates FTP)

Data values in other bases:

Hexadecimal	0	0	1	5
Binary	0000	0000	0001	0101
Decimal	0		21	
ASCII	©		©	

Source: http://www.zvon.org/tmRFC/RFC1475/Output/chapter4.html

Programming Hint: The name for this variable in code will be IP_TCP_DestinationPort_FTP.

IP > TCP PDU > *Sequence Number* for the selected FTP PDU

Field Name: *Sequence Number*

Purpose and Definition:
TCP is responsible for ensuring that all IP packets sent are actually received. When an application's data is packaged into IP packets, TCP will give each IP packet a sequence number. Once all the packets have arrived at the receiving computer, TCP uses the number in this 32-bit field to ensure that all of the packets actually arrived and are in the correct sequence.

Field Key: *Not applicable*

Data value (decimal): 2175080044

Data values in other bases:

Hexadecimal	8	1	A	5	1	6	6	C
Binary	1000	0001	1010	0101	0001	0110	0110	1100
Decimal	0		60		176		60	
ASCII	©		‘		↑		‘	

Programming Hint: The name for this variable in code will be IP_TCP_SequenceNumber_FTP.

IP > TCP PDU > *Acknowledgement Number* for the selected FTP PDU

Field Name: *Acknowledgement Number*

Purpose and Definition:
This number is used by the receiving computer to acknowledge which packets have successfully arrived. This number will be the sequence number of the next packet the receiver is ready to receive.

Field Key: *Not applicable*

Data value: 2275627869

Data values in other bases:

Hexadecimal	8	7	A	3	5	3	5	D
Binary	1000	0111	1010	0011	0101	0011	0101	1101
Decimal	135		163		83		93	
ASCII	↑		↑		S		]	

Programming Hint: The name for this variable in code will be IP_TCP_AcknowledgementNumber_FTP.

IP > TCP PDU > *Header Length or Offset* for the selected FTP PDU

Field Name: *Header Length or Offset*

Purpose and Definition:
This is identical in concept to the header length in an IP packet, except this time it indicates the length of the TCP header.

Field Key: *Not applicable*

Data value (bytes): 32

Data values in other bases:

Hexadecimal	8	0
Binary	1000	0000
Decimal	128	
ASCII	↑	

Programming Hint: The name for this variable in code will be IP_TCP_HeaderLength_FTP.

IP > TCP PDU > *Reserved* for the selected FTP PDU

Field Name: *Reserved*

Purpose and Definition:
These 6 bits are unused and are always set to 0.

Field Key: *Not applicable*

Data value (binary): 0000 00

Data values in other bases:

Hexadecimal	0	0	0	0	0	0
Binary	0000	0000	0000	0000	0000	0000
Decimal	0		0		0	
ASCII	©		©		©	

Programming Hint: The name for this variable in code will be IP_TCP_Reserved_FTP.

IP > TCP PDU > *Control Flags* for the selected FTP PDU

Field Name: *Control Flags*

Purpose and Definition:
Every TCP packet contains this 6-bit value that indicates how many octets it can receive at once. When IP packets are received, they are placed in a temporary area of RAM known as a buffer until the receiving computer has a chance to process them; this value represents how big a buffer the receiving host has made available for this temporary storage of IP packets.

Field Key:

- Urgent (URG)
- Acknowledgement (ACK)
- Push (PSH)
- Reset (RST)
- Synchronize (SYN)
- Finish (FIN)

Data value (binary): 01 1000

Data values in other bases: *Not applicable*

Programming Hint: The name for this variable in code will be IP_TCP_ControlFlags_FTP.

IP > TCP PDU > *Window Size* for the selected FTP PDU

Field Name: *Window Size*

Purpose and Definition:
Every TCP packet contains this 16-bit value that indicates how many octets it can receive at once. When IP packets are received, they are placed in a temporary area of RAM known as a buffer until the receiving computer has a chance to process them; this value represents how big a buffer the receiving host has made available for this temporary storage of IP packets.

Field Key: *Not applicable*

Data value (decimal): 5840

Data values in other bases:

Hexadecimal	1	6	D	0
Binary	0001	0110	1110	0000
Decimal	22		224	
ASCII	©		↑	

Programming Hint: The name for this variable in code will be IP_TCP_WindowSize_FTP.

IP > TCP PDU > *Checksum* for the selected FTP PDU

Field Name: *Checksum*

Purpose and Definition:
Unlike IP, TCP is responsible for ensuring that the entire IP packet arrived intact. TCP will run a CRC on the entire IP packet (not just the header) and place the resulting checksum in this field. When the IP packet is received, TCP re-runs the CRC on the entire packet to ensure the checksum is the same.

Field Key: *Not applicable*

Data value (hexadecimal): 11 F4

Data values in other bases:

Hexadecimal	1	1	F	4
Binary	0001	0001	1111	0100
Decimal	17		244	
ASCII	©		©	

Programming Hint: The name for this variable in code will be IP_TCP_Checksum_FTP.

IP > TCP PDU > *Urgent Pointer* for the selected FTP PDU

Field Name: *Urgent Pointer*

Purpose and Definition:
If the Urgent flag is set to on, this value indicates where the urgent data is located.

Information Key: *Not applicable*

Data value: *Not applicable*

Data values in other bases: *Not applicable*

Programming Hint: The name for this variable in code will be IP_TCP_UrgentPointer_FTP.

IP > TCP PDU > *Options and Padding* for the selected FTP PDU

Field Name: *Options and Padding*

Purpose and Definition:
Like IP options, this field is optional and represents additional instructions not covered in the other TCP fields. Again, if an option does not fill up a 32-bit word, it will be filled in with padding bits.

Field Key: *Not applicable*

Data value (hexadecimal): 01 01 08 0A 1B 25 F3 A1 0B DD 73 58

Data values in other bases:

Hexadecimal	0	1	0	1	0	8	0	A	1	B
Binary	0000	0001	0000	0001	0000	1000	0000	1010	0001	1011
Decimal	1		1		8		10		27	
ASCII	©		©		©		©		©	

Hexadecimal	2	5	F	3	A	1	0	B	D	D
Binary	0010	0101	1111	0011	1010	0001	0000	1011	1101	1101
Decimal	37		243		161		11		221	
ASCII	%		↑		↑		©		↑	

Hexadecimal	7	3	5	8
Binary	0101	0011	0101	1000
Decimal	115		96	
ASCII	↑		↑	

Programming Hint: The name for this variable in code will be IP_TCP_OptionsPadding_FTP.

2.2.3 FTP PDU for the selected FTP PDU

IP >TCP > FTP Header for the FTP Packet

RFC Link: http://www.ietf.org/rfc/rfc0959.txt?number=959

PASS (Password)

The argument field is a Telnet string specifying the user's password. This command must be immediately preceded by the user name command, and, for some sites, completes the user's identification for access control.

What is Contained in the Packet

Request: PASS
Request Arg: f1a2k3user

Data Values (hexadecimal): 50 41 53 53 20 66 31 61 32 6B 33 75 73 65 72 0D 0A

Data Values in Other Bases:

ASCII	P	A	S	S	SPC	f	1	a	2
Hexadecimal	5 0	4 1	5 3	5 3	2 0	6 6	3 1	6 1	3 2
Binary	0101 0000	0100 0001	0101 0011	0101 0011	0010 0000	0110 0110	0011 0001	0110 0001	0011 0010
Decimal	80	65	83	83	32	102	49	97	59

ASCII	k	3	u	s	e	r	\r	\n
Hexadecimal	6 B	3 3	7 5	7 3	6 5	7 2	0 D	0 A
Binary	0110 1011	0011 0011	0111 0101	0111 0011	0110 0101	0111 0010	0000 1101	0000 1010
Decimal	107	51	117	115	101	114	13	10

Programming Hint: The name for this variable in code will be IP_TCP_FTP_PDU_FTP.

2.2.4 IP PDU for the selected ICMP PDU

IP PDU > *Version* for the selected ICMP PDU

Field Name: *Version*

Purpose and Definition: Version is a 4-bit field that indicates the format of the internet header.

Field Key: 4 = IPv4
6 = IPv6

Data value (decimal): 4

Data values in other bases:

Hexadecimal	4
Binary	0100
Decimal	4

Programming Hint: The name for this variable in code will be IP_Version_ICMP.

IP PDU > *Internet Header Length* for the selected ICMP PDU

Field Name: *Internet Header Length*

Purpose and Definition: The IHL field is a 4 bit field indicating the length of the internet header in 32 bit words, and thus points to the beginning of the data. The minimum value of a correct header is 5.

Field Key: *Not applicable*

Data value: The value contained in our field is 20 bytes. This is the hexadecimal and decimal value of 5 multiplied by 4 bits.

Data values in other bases:

Hexadecimal	0	5
Binary	0000	0101
Decimal	5	

Programming Hint: The name for this variable in code will be IP_Version_ICMP.

IP PDU > *Type of Service* for the selected ICMP PDU

Field Name: *Type of Service*

Purpose and Definition: Type of Service is an 8-bit field that provides and indication of the abstract parameters of the quality of service desired. These parameters guide the selection of the actual service parameters when transmitting a datagram through a particular network.

Field Key: The major choice is a three-way tradeoff between low-delay, high-reliability, and high-throughput.

0 1 2	3	4	5	6	7
Precedence	D	T	R	0	0

Bits 0-2: Precedence
Bit 3: (D) 0 = Normal Delay 1 = Low Delay
Bit 4: (T) 0 = Normal Throughput 1 = High Throughput
Bit 5: (R) 0 = Normal Reliability 1 = High Reliability

Precedence:

111 = Network Control	011 = Flash
110 = Internetwork Control	010 = Immediate
101 = CRITIC/ECP	001 = Priority
100 = Flash Overrided	000 = Routine

Data value (hexadecimal): 00

Data values in other bases:

Hexadecimal	0	0
Binary	0000	0000
Decimal	0	

Programming Hint: The name for this variable in code will be IP_TypeOfService_ICMP.

IP PDU > *Total Length of Ethernet Frame* for the selected ICMP PDU

Field Name: *Total Length of Ethernet Frame*

Purpose and Definition: Total Length is a 16-bit field that indicates the length of the frame, measured in octets, including internet header and data. The maximum size is 2^{16}-1 or 65,535 octets; however, the recommended maximum size is 576 octets.

Field Key: *Not applicable*

Data values (decimal): 84

Data values in other bases:

Hexadecimal	0	0	5	4
Binary	0000	0000	0101	0100
Decimal	0		84	
ASCII	©		T	

Programming Hint: The name for this variable in code will be IP_TotalLength_ICMP.

IP PDU > *Identification* for the selected ICMP PDU

Field Name: *Identification*

Purpose and Definition: Identification is a 16-bit field. An identifying value is assigned by the sender to aid in assembling the fragments of a datagram. The identifier is chosen based on the need to provide a way to uniquely identify the fragments and protocol for the time the datagram or any fragment could be alive in the internet

Field Key: *Not applicable*

Data value (hexadecimal): 00 00

Data values in other bases:

Hexadecimal	0	0	0	0
Binary	0000	0000	0000	0000

Programming Hint: The name for this variable in code will be IP_Identification_ICMP.

IP PDU > *Flags* for the selected ICMP PDU

Field Name: *Flags*

Purpose and Definition: Flags is a 3-bit field that indicates directions for fragmentation.

Field Key:
Bit 0: reserved, must be 0
Bit 1: (DF) 0 = May Fragment 1 = Don't Fragment
Bit 2: (MF) 0 = Last Fragment 1 = More Fragment

Data value (binary): 010

Data values in other bases: *Not applicable*

Programming Hint: The name for this variable in code will be IP_Flags_ICMP.

IP PDU > *Fragment Offset* for the selected ICMP PDU

Field Name: *Fragment Offset*

Purpose and Definition: The Fragment Offset is a 13- bit field indicating where in the Ethernet frame this fragment begins. The Fragment Offset is measured in units of 8 octets, and the first fragment has offset 0.

Field Key: *Not applicable*

Data value (decimal): 0

Data values in other bases:
Binary: 0000 0000 0000

Programming Hint: The name for this variable in code will be IP_FragmentOffset_ICMP.

IP PDU > *Time to Live* for the selected ICMP PDU

Field Name: *Time to Live*

Purpose and Definition: Time to Live is an 8-bit field that indicates the maximum time the datagram is allowed to remain in the internet. If this field contains the value 0, then the datagram must be destroyed. This field is modified in internet header processing. The time is measure in units of seconds, and is set by the sender to the maximum time the datagram is allowed to be in the internet. This field is decreased at each point that the internet header is processed. The intention is to cause undeliverable packets to be discarded, and to bind the maximum datagram lifetime.

Field Key: *Not applicable*

Data value (decimal): 64

Data values in other bases:

Hexadecimal	4	0
Binary	0100	0000
Decimal	64	

Programming Hint: The name for this variable in code will be IP_TimeToLive_ICMP.

IP PDU > *Protocol* for the selected ICMP PDU

Field Name: *Protocol*

Purpose and Definition: Protocol is an 8-bit field that indicates the next level protocol that is used in the data portion of the internet diagram.

Field Key:

Dec	Hex	Protocol	Dec	Hex	Protocol
0	00	Reserved	22	16	Multiplexing
1	01	ICMP	23	17	DCN
2	02	Unassigned	24	18	TAC Monitoring
3	03	Gateway-to-Gateway	25-76	19-4C	Unassigned
4	04	CMCC Gateway Monitoring Message	77	4D	Any local network
5	05	ST	100	64	SATNET and Backroom EXPAK
6	06	TCP	101	65	MIT Subnet Support
7	07	UCL	102-104	66-68	Unassigned
10	0A	Unassigned	105	69	SATNET Monitoring
11	0B	Secure	106	6A	Unassigned
12	0C	BBN RCC Monitoring	107	6B	Internet Packet Core Utility
13	0D	NVP	110-113	6E-71	Unassigned
14	0E	PUP	114	72	Backroom SATNET Monitoring
15	0F	Pluribus	115	73	Unassigned
16	10	Telnet	116	74	WIDEBAND Monitoring
17	11	XNET	117	75	WIDEBAND EXPAK
20	14	Chaos	120-376	78-0178	Unassigned
21	15	User Datagram	377	0179	Reserved

Data value (hexadecimal): 01

Data values in other bases:

Hexadecimal	0	6
Binary	0000	0001
Decimal	1	

RFC Link: http://www.faqs.org/rfcs/rfc790.html

Programming Hint: The name for this variable in code will be IP_Protocol_ICMP.

IP PDU > *Header Checksum* for the Selected ICMP PDU

Field Name: *Header Checksum*

Purpose and Definition: The Header Checksum is a 16-bit field. This CRC algorithm is the 16-bit one's complement sum of all the 16-bit words in the header. For purposes of computing the checksum, the value of the checksum field is initially zero. When both header checksums are the same, then the header bits are correct. If either checksums vary, then a packet will need to be resent.

This is a simple way to compute the checksum and experimental evidence indicates that it is adequate, but it is provisional and may be replaced by a CRC procedure, depending on further experience.

Field Key: *Not applicable*

Data value (hexadecimal): B8 CC

Data values in other bases:

Hexadecimal	B	8	C	C
Binary	1011	1000	1100	1100

Programming Hint: The name for this variable in code will be IP_HeaderChecksum_ICMP.

IP PDU > *Source Address* for the Selected ICMP PDU

Field Name: *Source Address*

Purpose and Definition: The Source Address is a 32-bit field that contains the IP address of the host that sent the IP Packet.

Field Key: *Not applicable*

Data value: 192.168.0.39

Data values in other bases:

Hexadecimal	C	0	A	8	0	0	2	7
Binary	1100	0000	1010	1000	0000	0000	0010	0111
Decimal	192		168		0		39	

Programming Hint: The name for this variable in code will be IP_SourceAddress_ICMP.

IP PDU > *Destination Address* for the selected ICMP PDU

Field Name: *Destination Address*

Purpose and Definition: The Destination Address is a 32-bit field that contains the address of the host that is to receive the data contained within the IP packet.

Field Key: *Not applicable*

Data value: 192.168.0.101

Data values in other bases:

Hexadecimal	C	0	A	8	0	0	6	5
Binary	1100	0000	1010	1000	0000	0000	0110	0101
Decimal	192.		168.		0.		101	

Programming Hint: The name for this variable in code will be IP_DestinationAddress_ICMP.

IP PDU > *Options and Padding* for the selected ICMP PDU

Field Name: *Options and Padding*

Purpose and Definition: The options may or may not appear in Ethernet packets. They must be implemented by all IP modules (host and gateways). What is optional is their transmission in any particular packet, not their implementation.

The option field is variable in length. There may be zero or more options. There are two cases for the format of an option.

Case 1: A single octet of option type

Case 2: An option-type octet, an option-length octet, and the actual option-data octets.

Field Key: *Not applicable*

Data values: *Not applicable*

Data values in other bases: *Not applicable*

Programming Hint: The name for this variable in code will be IP_OptionsPadding_ICMP.

2.2.5 ICMP PDU for the selected ICMP PDU

IP > ICMP Header > *Type* for the selected ICMP PDU

Field Name: *Type*

Purpose and Definition: The type is an 8-bit field that identifies what sort of message the ICMP protocol is sending.

Field Key:

Dec	Hex	Message Type	Dec	Hex	Message Type
0	00	Echo Reply	16	10	Information Reply
1	01	Unassigned	17	11	Address Mask Request
2	02	Unassigned	18	12	Address Mask Reply
3	03	Destination Unreachable	19	13	Reserved (for Security)
4	04	Source Quench	20-29	14-1D	Reserved (for Robustness Experiment)
5	05	Redirect	30	1E	Traceroute
6	06	Alternate Host Address	31	1F	Datagram Conversion Error
7	07	Unassigned	32	20	Mobile Host Redirect
8	08	Echo	33	21	IPv6 Where-Are-You
9	09	Router Advertisement	34	22	IPv6 I-Am-Here
10	0A	Router Solicitation	35	23	Mobile Registration Request
11	0B	Time Exceeded	36	24	Mobile Registration Reply
12	0C	Parameter Problem	37	25	Domain Name Request
13	0D	Timestamp	38	26	Domain Name Reply
14	0E	Timestamp Reply	39	27	SKIP
15	0F	Information Request	40	28	Photuris
			41-255	29-FF	Reserved

Data value: 8 (Echo (ping) Request)

Data values in other bases:

Hexadecimal	0	8
Binary	0000	1000
Decimal	8	

RFC Link: http://www.iana.org/assignments/icmp-parameters

Programming Hint: The name for this variable in code will be IP_ICMP_Type_ICMP.

IP > ICMP Header > *Code* for the selected ICMP PDU

Field Name: *Code*

Purpose and Definition: Code is an 8-bit field that provides further information about the associated type field.

Field Key:

Type	Name
0	Echo Reply (used by "PING")
	0 No Code
1	Unassigned
2	Unassigned
3	Destination Unreachable
	0 Net Unreachable
	1 Host Unreachable
	2 Protocol Unreachable
	3 Port Unreachable
	4 Fragmentation needed and Don't Fragment was Set
	5 Source Route Failed
	6 Destination Network Unknown
	7 Destination Host Unknown
	8 Source Host Isolated
	9 Communication with Destination Network is Administratively Prohibited
	10 Communication with Destination Host is Administratively Prohibited
	11 Destination Network Unreachable for Type of Service
	12 Destination Host Unreachable for Type of Service
4	Source Quench
	0 No Code
5	Redirect
	0 Redirect Datagram for the Network
	1 Redirect Datagram for the Host
	2 Redirect Datagram for the Type of Service and Network
	3 Redirect Datagram for the Type of Service and Host
6	Alternate Host Address
	0 Alternate Address for Host
7	Unassigned
8	Echo (used by "PING")
	0 No Code
9	Router Advertisement
	0 No Code
10	Router Selection
	0 No Code
11	Time Exceeded
	0 Time to Live exceeded in Transit
	1 Fragment Reassembly Time Exceeded
12	Parameter Problem
	0 Pointer indicates the error
	1 Missing a Required Option
	2 Bad Length
13	Timestamp
	0 No Code
14	Timestamp Reply
	0 No Code
15	Information Request
	0 No Code
16	Information Reply
	0 No Code
17	Address Mask Request
	0 No Code
18	Address Mask Reply
	0 No Code
19	Reserved (for Security)
20-29	Reserved (for Robustness Experiment)
30	Traceroute
31	Datagram Conversion Error
32	Mobile Host Redirect
33	IPv6 Where-Are-You
34	IPv6 I-Am-Here
35	Mobile Registration Request
36	Mobile Registration Reply

Data value (decimal): 0

Data values in other bases:

Hexadecimal	0	0
Binary	0000	0000
Decimal	0	
ASCII	©	

Programming Hint: The name for this variable in code will be IP_ICMP_Header_ICMP.

IP > ICMP Header > *Checksum* for the selected ICMP PDU

Field Name: *Checksum*

Purpose and Definition: The checksum is the 16-bit one's complement of the one's complement sum of the ICMP message, starting with the ICMP type. For computing the checksum, the checksum field should initially be zero.

Field Key: *Not applicable*

Data value (hexadecimal): C9 15

Data values in other bases:

Hexadecimal	C	9	1	5
Binary	1100	1001	0001	0101
Decimal	201		21	
ASCII	↑		©	

Programming Hint: The name for this variable in code will be IP_ICMP_Checksum_ICMP.

IP > ICMP Header > *Identifier* for the selected ICMP PDU

Field Name: Identifier

Purpose and Definition: The identifier is a 16-bit field that is used in matching echoes and replies for when the code field is zero.

Field Key: *Not applicable*

Data value (hexadecimal): 70 60

Data values in other bases:

Hexadecimal	7	0	6	0
Binary	0111	0000	0110	0000
Decimal	112		96	
ASCII	P		‘	

Programming Hint: The name for this variable in code will be IP_ICMP_Identifier_ICMP.

IP > ICMP Header > *Sequence* for the selected ICMP PDU

Field Name: *Sequence*

Purpose and Definition: The sequence is a 16-bit field that is used in matching echoes and replies for when the code field is zero.

Field Key: *Not applicable*

Data value (hexadecimal): 70 60

Data values in other bases:

Hexadecimal	0	0	0	0
Binary	0000	0000	0000	0000
Decimal	0		0	
ASCII	©		©	

Programming Hint: The name for this variable in code will be IP_ICMP_Sequence_ICMP

IP > ICMP Header > *Data* for the selected ICMP PDU

Field Name: *Data*

Purpose and Definition: The data is a variable-length field that contains the actual information that is sent in the ping packet.

Field Key: *Not applicable*

Data value (hexadecimal): 42 B1 89 3F 00 00 00 00 2C C6 07 00 00 00 00 00 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F 20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F 30 31 32 33 34 35 36 37

Data values in other bases:

Hexadecimal	4	2	B	1	8	9	3	F	0	0
Binary	0100	0010	1011	0001	1000	1001	0011	1111	0000	0000
Decimal	66		177		137		63		0	
ASCII	B		↑		↑		?		©	

Hexadecimal	0	0	0	0	0	0	2	C	C	6
Binary	0000	0000	0000	0000	0000	0000	0010	1100	1100	0110
Decimal	0		0		0		44		198	
ASCII	©		©		©		,		↑	

Hexadecimal	0	7	0	0	0	0	0	0	0	0
Binary	0000	0111	0000	0000	0000	0000	0000	0000	0000	0000
Decimal	7		0		0		0		0	
ASCII	©		©		©		©		©	

Hexadecimal	0	0	1	0	1	1	1	2	1	3
Binary	0000	0000	0001	0000	0001	0001	0001	0010	0001	0011
Decimal	0		16		17		18		19	
ASCII	©		©		©		©		©	

Hexadecimal	1	4	1	5	1	6	1	7	1	8
Binary	0001	0100	0001	0101	0001	0110	0001	0111	0001	1000
Decimal	20		21		22		23		24	
ASCII	©		©		©		©		©	

Hexadecimal	1	9	1	A	1	B	1	C	1	D
Binary	0001	1001	0001	1010	0001	1011	0001	1100	0001	1101
Decimal	25		26		27		28		29	
ASCII	©		©		©		©		©	

Hexadecimal	1	E	1	F	2	0	2	1	2	2
Binary	0001	1110	0001	1111	0010	0000	0010	0001	0010	0010
Decimal	30		31		32		33		34	
ASCII	©		©		SPACE		!		“	

Hexadecimal	2	3	2	4	2	5	2	6	2	7
Binary	0010	0011	0010	0100	0010	0101	0010	0110	0010	0111
Decimal	35		36		37		38		39	
ASCII	#		$		%		&		‘	

Hexadecimal	2	8	2	9	2	A	2	B	2	C
Binary	0010	1000	0010	1001	0010	1010	0010	1011	0010	1100
Decimal	40		41		42		43		44	
ASCII	(		)		*		+		,	

Hexadecimal	2	D	2	E	2	F	3	0	3	1
Binary	0010	1101	0010	1110	00010	1111	0011	0000	0011	0001
Decimal	45		46		47		48		49	
ASCII	-		.		/		0		1	

Hexadecimal	3	2	3	3	3	4	3	5	3	6
Binary	0011	0010	0011	0011	0011	0100	0011	0101	0011	0110
Decimal	50		51		52		53		54	
ASCII	2		3		4		5		6	

Hexadecimal	3	7
Binary	0011	0111
Decimal	55	
ASCII	7	

Programming Hint: The name for this variable in code will be IP_ICMP_Data_ICMP.

2.2.6 IP PDU for the selected SMTP PDU

IP PDU > *IP Version* for the selected SMTP PDU

Field Name: *IP Version*

Purpose and Definition: Version is a 4-bit field that indicates the format of the internet header.

Field Key: 4 = IPv4
6 = IPv6

Data value (decimal): 4

Data values in other bases:

Hexadecimal	4
Binary	0100
Decimal	4

Programming Hint: The name for this variable in code will be IP_IPVersion_SMTP.

IP PDU > *Internet Header Length* for the selected SMTP PDU

Field Name: *Internet Header Length*

Purpose and Definition: The IHL field is a 4-bit field indicating the length of the internet header in 32 bit words, and thus points to the beginning of the data. The minimum value of a correct header is 5.

Field Key: *Not applicable*

Data value: The value contained in our field is 20 bytes. This is the hexadecimal and decimal value of 5 multiplied by 4 bits.

Data values in other bases:

Hexadecimal	0	5
Binary	0000	0101
Decimal	5	

Programming Hint: The name for this variable in code will be IP_IHL_SMTP.

IP PDU > *Type of Service* for the selected SMTP PDU

Field Name: *Type of Service*

Purpose and Definition: Type of Service is an 8-bit field that provides and indication of the abstract parameters of the quality of service desired. These parameters guide the selection of the actual service parameters when transmitting a datagram through a particular network.

Field Key: The major choice is a three-way tradeoff between low-delay, high-reliability, and high-throughput.

0 1 2	3	4	5	6	7
Precedence	D	T	R	0	0

Bits 0-2: Precedence
Bit 3: (D) 0 = Normal Delay 1 = Low Delay
Bit 4: (T) 0 = Normal Throughput 1 = High Throughput
Bit 5: (R) 0 = Normal Reliability 1 = High Reliability

Precedence:

111 = Network Control	011 = Flash
110 = Internetwork Control	010 = Immediate
101 = CRITIC/ECP	001 = Priority
100 = Flash Overrided	000 = Routine

Data value (hexadecimal): 00

Data values in other bases:

Hexadecimal	0	0
Binary	0000	0000
Decimal	00	

Programming Hint: The name for this variable in code will be IP_TypeOfService_SMTP.

IP PDU > *Total Length of Ethernet Frame* for the selected SMTP PDU

Field Name: *Total Length of Ethernet Frame*

Purpose and Definition: Total Length is a 16-bit field that indicates the length of the frame, measured in octets, including internet header and data. The maximum size is 2^{16}-1 or 65,535 octets; however, the recommended maximum size is 576 octets.

Field Key: *Not applicable*

Data values (hexadecimal): 02 12

Data values in other bases:

Hexadecimal	0	2	1	2
Binary	0000	0010	0001	0010
Decimal	2		18	
ASCII	©		©	

Programming Hint: The name for this variable in code will be IP_TotalLength_SMTP.

IP PDU > *Identification* for the selected SMTP PDU

Field Name: *Identification*

Purpose and Definition: Identification is a 16-bit field. An identifying value is assigned by the sender to aid in assembling the fragments of a datagram. The identifier is chosen based on the need to provide a way to uniquely identify the fragments and protocol for the time the datagram or any fragment could be alive in the internet.

Field Key: *Not applicable*

Data value (hexadecimal): 6128

Data values in other bases:

Hexadecimal	6	1	2	8
Binary	0110	0001	0010	1000

Programming Hint: The name for this variable in code will be IP_Idenfification_SMTP.

IP PDU > *Flags* for the selected SMTP PDU

Field Name: *Flags*

Purpose and Definition: Flags is a 3-bit field that indicates directions for fragmentation.

Field Key:
Bit 0: reserved, must be 0
Bit 1: (DF) 0 = May Fragment 1 = Don't Fragment
Bit 2: (MF) 0 = Last Fragment 1 = More Fragment

Data value (binary): 0100

Data values in other bases: *Not applicable*

Programming Hint: The name for this variable in code will be IP_Flags_SMTP.

IP PDU > *Fragment Offset* for the selected SMTP PDU

Field Name: *Fragment Offset*

Purpose and Definition: The Fragment Offset is a 13- bit field indicating where in the Ethernet frame this fragment begins. The Fragment Offset is measured in units of 8 octets, and the first fragment has offset 0.

Field Key: *Not applicable*

Data value (decimal): 0

Data values in other bases:
Binary: 0000 0000 0000

Programming Hint: The name for this variable in code will be IP_FragmentOffset_SMTP.

IP PDU > *Time to Live* for the selected SMTP PDU

Field Name: *Time to Live*

Purpose and Definition: Time to Live is an 8-bit field that indicates the maximum time the datagram is allowed to remain in the internet. If this field contains the value 0, then the datagram must be destroyed. This field is modified in internet header processing. The time is measure in units of seconds, and is set by the sender to the maximum time the datagram is allowed to be in the internet. This field is decreased at each point that the internet header is processed. The intention is to cause undeliverable packets to be discarded, and to bind the maximum datagram lifetime.

Field Key: *Not applicable*

Data value (decimal): 64

Data values in other bases:

Hexadecimal	4	0
Binary	0100	0000
Decimal	64	

Programming Hint: The name for this variable in code will be IP_TimeToLive_SMTP.

IP PDU > *Protocol* for the selected SMTP PDU

Field Name: *Protocol*

Purpose and Definition: Protocol is an 8-bit field that indicates the next level protocol that is used in the data portion of the internet diagram.

Field Key:

Dec	Hex	Protocol	Dec	Hex	Protocol
0	00	Reserved	22	16	Multiplexing
1	01	ICMP	23	17	DCN
2	02	Unassigned	24	18	TAC Monitoring
3	03	Gateway-to-Gateway	25-76	19-4C	Unassigned
4	04	CMCC Gateway Monitoring Message	77	4D	Any local network
5	05	ST	100	64	SATNET and Backroom EXPAK
6	06	TCP	101	65	MIT Subnet Support
7	07	UCL	102-104	66-68	Unassigned
10	0A	Unassigned	105	69	SATNET Monitoring
11	0B	Secure	106	6A	Unassigned
12	0C	BBN RCC Monitoring	107	6B	Internet Packet Core Utility
13	0D	NVP	110-113	6E-71	Unassigned
14	0E	PUP	114	72	Backroom SATNET Monitoring
15	0F	Pluribus	115	73	Unassigned
16	10	Telnet	116	74	WIDEBAND Monitoring
17	11	XNET	117	75	WIDEBAND EXPAK
20	14	Chaos	120-376	78-0178	Unassigned
21	15	User Datagram	377	0179	Reserved

Data value (hexadecimal): 06

Data values in other bases:

Hexadecimal	0	6
Binary	0000	0110
Decimal	6	

RFC Link: http://www.faqs.org/rfcs/rfc790.html

Programming Hint: The name for this variable in code will be IP_Protocol_SMTP.

IP PDU > *Header Checksum* for the selected SMTP PDU

Field Name: *Header Checksum*

Purpose and Definition: The Header Checksum is a 16-bit field. The Checksum is the 16-bit one's complement sum of all 16-bit words in the header. For purposes of computing the checksum, the initial value of its field is zero. When both header checksums are equal, then the header bits are correct. If either checksums vary, then a new, correct packet will need to be sent.

This is a simple way to compute the checksum and experimental evidence indicates that it is adequate, but it is provisional and may be replaced by a CRC procedure, depending on further experience.

Field Key: *Not applicable*

Data value (hexadecimal): F1 F3

Data values in other bases:

Hexadecimal	F	1	F	3
Binary	1111	0001	1111	0011

Programming Hint: The name for this variable in code will be IP_HeaderChecksum_SMTP.

IP PDU > *Source Address* for the selected SMTP PDU

Field Name: *Source Address*

Purpose and Definition: The Source Address is a 32-bit field that contains the IP address of the host that sent the IP Packet.

Field Key: *Not applicable*

Data value (decimal): 192.168.0.101

Data values in other bases:

Hexadecimal	C	0	A	8	0	1	0	1
Binary	1100	0000	1010	1000	0000	0001	0010	0001
Decimal	192		168		101			

Programming Hint: The name for this variable in code will be IP_SourceAddress_SMTP.

IP PDU > *Destination Address* for the selected SMTP PDU

Field Name: *Destination Address*

Purpose and Definition: The Destination Address is a 32-bit field that contains the address of the host that is to receive the data contained within the IP packet.

Field Key: *Not applicable*

Data value (decimal): 192.168.100.20

Data values in other bases:

Hexadecimal	C	0	A	8	0	0	6	5
Binary	1100	0000	1010	1000	0000	0000	0110	0101
Decimal	192		168		100		20	

Programming Hint: The name for this variable in code will be IP_DestinationAddress_SMTP.

IP PDU > *Options and Padding* for the selected SMTP PDU

Field Name: *Options and Padding*

Purpose and Definition: The options may or may not appear in Ethernet packets. They must be implemented by all IP modules (host and gateways). What is optional is their transmission in any particular packet, not their implementation.

The option field is variable in length. There may be zero or more options. There are two cases for the format of an option.

Case 1: A single octet of option type

Case 2: An option-type octet, an option-length octet, and the actual option-data octets.

Field Key: *Not applicable*

Data values: *Not applicable*

Data values in other bases: *Not applicable*

Programming Hint: The name for this variable in code will be IP_OptionsPadding_SMTP.

IP PDU > *Data* for the selected SMTP PDU

Field Name: *Data*

Purpose and Definition: The Data is a variable length field which contains the actual data that is being sent from one host to another. The data field may start with a Layer 4 header, which will give additional instructions to the application that will be receiving the data; alternately, it may be an ICMP header and not contain any user data at all.

Field Key: *Not applicable*

Data values (hexadecimal) : (TCP) 0D 0A 2D 2D 2D 31 34 36 33 37 38 36 32 34 30 2D 37 32 33 37 38 33 33 32 38 2D 31 30 36 37 36 33 34 33 35 30 3D 3A 32 36 36 30 36 0D 0A 43 6F 6E 74 2D 54 79 70 65 3A 20 54 45 58 54 2F 70 6C 61 69 6E 3B 20 6E 61 6D 65 3D 22 6D 69 6d 65 74 65 73 74 2E 74 78 74 22 0D 0A 43 6F 6E 74 65 6E 74 2D 54 72 61 6E 73 66 65 72 2D 45 6E 63 6F 64 69 6E 67 3A 20 42 41 53 45 36 34 0D 0A 43 6F 6E 74 65 6E 74 2D 49 44 3A 20 3C 50 69 6E 65 2E 4C 4E 58 2E 34 2E 32 31 2E 32 31 2E 30 33 31 31 36 30 35 35 30 30 2E 32 36 36 30 36 40 63 62 31 31 38 6B 73 2E 63 73 2E 73 69 65 6E 61 2E 65 64 75 3E 0D 0A 43 6F 6E 74 65 6E 74 2D 44 65 73 63 72 69 70 74 69 6F 6E 3A 20 0D 0A 43 6F 6E 74 65 6E 74 2D 44 69 73 70 6F 73 69 74 69 6F 6E 3A 20 61 74 74 61 63 68 6D 65 6E 74 3B 20 66 69 6C 65 6E 61 6D 65 3D 22 6D 69 6D 65 74 65 73 74 2E 74 78 74 22 0D 0A 0D 0A 56 47 68 70 63 79 42 70 63 79 42 30 61 47 55 67 62 57 56 7A 63 32 46 6E 5A 53 42 30 61 47 46 30 49 48 64 70 62 47 77 67 59 57 78 73 62 33 63 67 64 58 4D 67 64 47 38 67 5A 47 6C 7A 0D 0A 63 47 78 68 65 53 42 33 61 58 52 6F 49 45 56 30 61 47 56 79 5A 57 46 73 49 41 30 4B 59 53 42 4E 53 55 31 46 49 47 46 30 64 47 46 6A 61 47 31 6C 62 6E 51 57 61 57 35 7A 61 57 52 6C 0D 0A 49 47 46 75 49 46 4E 4E 56 46 41 67 5A 6E 4A 68 62 57 55 75 44 51 6F 3D 0D 0A 2D 2D 2D 31 34 36 33 37 38 36 32 34 30 2D 37 32 33 37 38 33 33 32 38 2D 31 30 36 37 36 33 34 33 35 30 3D 3A 32 36 36 30 36 2D 2D 0D 0A 2E 0D 0A

Programming Hint: The name for this variable in code will be IP_Data_SMTP.

2.2.7 TCP PDU for the selected SMTP PDU

IP > TCP PDU > *Source Port* for the selected SMTP PDU

Field Name: *Source Port*

Purpose and Definition:
This 16-bit number represents the name of the application that sent the data in the IP packet.

Field Key: *Not applicable*

Data value (decimal): 3651

Data values in other bases:

Hexadecimal	0	D	E	9
Binary	0000	1101	1110	1001
Decimal	13		233	
ASCII	/n		↑	

Programming Hint: The name for this variable in code will be IP_TCP_SourcePort_SMTP.

IP > TCP PDU > *Destination Port* for the selected SMTP PDU

Field Name: *Destination Port*

Purpose and Definition:
This 16-bit number represents the name of the application that is to receive the data contained within the IP packet. This is one of the major differences between a Layer 3 and a Layer 4 header: the Layer 3 header contains the IP address of the computer that is to receive the IP packet; once that packet has been received, the port address in the Layer 4 header ensures that the data contained within that IP packet is passed to the correct application on that computer.

Field Key:
This key indicates assigned port number values:

Dec	**Port Numbers**
0	Reserved
1-32767	Internet registered ("well-known") protocols
32768-98303	Reserved, to allow TCPv7-TCPv4 conversion
98304 & up	Dynamic assignment

Data value (decimal): 25 (indicates SMTP)

Data values in other bases:

Hexadecimal	0	0	1	9
Binary	0000	0000	0001	1001
Decimal	0		25	
ASCII	©		©	

Source: http://www.zvon.org/tmRFC/RFC1475/Output/chapter4.html

Programming Hint: The name for this variable in code will be IP_TCP_DestinationPort_SMTP.

IP > TCP PDU > *Sequence Number* for the selected SMTP PDU

Field Name: *Sequence Number*

Purpose and Definition:
TCP is responsible for ensuring that all IP packets sent are actually received. When an application's data is packaged into IP packets, TCP will give each IP packet a sequence number. Once all the packets have arrived at the receiving computer, TCP uses the number in this 32-bit field to ensure that all of the packets actually arrived and are in the correct sequence.

Field Key: *Not applicable*

Data value (decimal): 2069207327

Data values in other bases:

Hexadecimal	7	B	5	5	9	9	1	F
Binary	0111	1011	0101	0101	1001	1001	0001	1111
Decimal	123		85		153		31	
ASCII	{		U		↑		©	

Programming Hint: The name for this variable in code will be IP_TCP_SequenceNumber_SMTP.

IP > TCP PDU > *Acknowledgement Number* for the selected SMTP PDU

Field Name: *Acknowledgement Number*

Purpose and Definition:
This number is used by the receiving computer to acknowledge which packets have successfully arrived. This number will be the sequence number of the next packet the receiver is ready to receive.

Field Key: *Not applicable*

Data value (decimal): 3827794966

Data values in other bases:

Hexadecimal	E	4	2	7	8	4	1	6
Binary	1110	0100	0010	0111	1000	0100	0001	0110
Decimal	228		39		132		22	
ASCII	↑		'		↑		©	

Programming Hint: The name for this variable in code will be IP_TCP_AcknowledgementNumber_SMTP.

IP > TCP PDU > *Header Length or Offset* for the selected SMTP PDU

Field Name: *Header Length or Offset*

Purpose and Definition:
This is identical in concept to the header length in an IP packet, except this time it indicates the length of the TCP header.

Field Key: *Not applicable*

Data value (bytes): 32

Data values in other bases:

Hexadecimal	8	0
Binary	1000	0000
Decimal	128	
ASCII	↑	

Programming Hint: The name for this variable in code will be IP_TCP_HeaderLength_SMTP.

IP > TCP PDU > *Reserved* for the selected SMTP PDU

Field Name: *Reserved*

Purpose and Definition:
These 6 bits are unused and are always set to 0.

Field Key: *Not applicable*

Data value (binary): 0000 00

Data values in other bases:

Hexadecimal	0	0	0	0	0	0
Binary	0000	0000	0000	0000	0000	0000
Decimal	0		0		0	
ASCII	©		©		©	

Programming Hint: The name for this variable in code will be IP_TCP_Reserved_SMTP

IP > TCP PDU > *Control Flags* for the selected SMTP PDU

Field Name: *Control Flags*

Purpose and Definition:
Every TCP packet contains this 6-bit value that indicates how many octets it can receive at once. When IP packets are received, they are placed in a temporary area of RAM known as a buffer until the receiving computer has a chance to process them; this value represents how big a buffer the receiving host has made available for this temporary storage of IP packets.

Field Key:
- Urgent (URG)
- Acknowledgement (ACK)
- Push (PSH)
- Reset (RST)
- Synchronize (SYN)
- Finish (FIN)

Data value (binary): 0001 1000

Data values in other bases: *Not applicable*

Programming Hint: The name for this variable in code will be IP_TCP_ControlFlags_SMTP.

IP > TCP PDU > *Window Size* for the selected SMTP PDU

Field Name: *Window Size*

Purpose and Definition:
Every TCP packet contains this 16-bit value that indicates how many octets it can receive at once. When IP packets are received, they are placed in a temporary area of RAM known as a buffer until the receiving computer has a chance to process them; this value represents how big a buffer the receiving host has made available for this temporary storage of IP packets.

Field Key: *Not applicable*

Data value (decimal): 32120

Data values in other bases:

Hexadecimal	7	D	7	8
Binary	0111	1101	0111	1000
Decimal	125		120	
ASCII	}		x	

Programming Hint: The name for this variable in code will be IP_TCP_WindowSize_SMTP.

IP > TCP PDU > *Checksum* for the selected SMTP PDU

Field Name: *Checksum*

Purpose and Definition:
Unlike IP, TCP is responsible for ensuring that the entire IP packet arrived intact. TCP will run a CRC on the entire IP packet (not just the header) and place the resulting checksum in this field. When the IP packet is received, TCP re-runs the CRC on the entire packet to ensure the checksum is the same.

Field Key: *Not applicable*

Data value (hexadecimal): 72 B5

Data values in other bases:

Hexadecimal	7	2	B	5
Binary	0111	0010	1011	0101
Decimal	114		181	
ASCII	r		↑	

Programming Hint: The name for this variable in code will be IP_TCP_Checksum_SMTP.

IP > TCP PDU > *Urgent Pointer* for the selected SMTP PDU

Field Name: *Urgent Pointer*

Purpose and Definition:
If the Urgent flag is set to on, this value indicates where the urgent data is located.

Information Key: *Not applicable*

Data value: *Not applicable*

Data values in other bases: *Not applicable*

Programming Hint: The name for this variable in code will be IP_TCP_UrgentPointer_SMTP.

IP > TCP PDU > *Options and Padding* for the selected SMTP PDU

Field Name: *Options and Padding*

Purpose and Definition:
Like IP options, this field is optional and represents additional instructions not covered in the other TCP fields. Again, if an option does not fill up a 32-bit word, it will be filled in with padding bits.

Field Key: *Not applicable*

Data value (hexadecimal): 01 01 08 0A 07 AE F6 75 00 21 66 A4

Data values in other bases:

Hexadecimal	0	1	0	1	0	8	0	A	0	7
Binary	0000	0001	0000	0001	0000	1000	0000	1010	0000	0111
Decimal	1		1		8		10		7	
ASCII	©		©		©		©		©	

Hexadecimal	A	E	F	6	7	5	0	0	2	1
Binary	1010	1110	1111	0110	0111	0101	0000	0000	0010	0001
Decimal	174		246		117		0		33	
ASCII	↑		↑		u		©		!	

Hexadecimal	6	6	A	4
Binary	0101	0011	0101	1000
Decimal	102		164	
ASCII	f		↑	

Programming Hint: The name for this variable in code will be IP_TCP_OptionsPadding_SMTP.

2.2.8 SMTP PDU for the selected SMTP PDU

IP > SMTP Header > *Command* for the selected SMTP PDU

RFC Link: http://www.ietf.org/rfc/rfc0821.txt?number=821

Field Name: *Command*

Purpose and Definition: ASCII messages sent between SMTP hosts.

Field Key:

Command	Description
DATA	Begins message composition.
EXPN <string>	Returns names on the specified mail list.
HELO <domain>	Returns identity of mail server.
HELP <command>	Returns information on the specified command.
MAIL FROM <host>	Initiates a mail session from host.
NOOP	Causes no action, except acknowledgement from server.
QUIT	Terminates the mail session.
RCPT TO <user>	Designates who receives mail.
RSET	Resets mail connection.
SAML FROM <host>	Sends mail to user terminal and mailbox.
SEND FROM <host>	Sends mail to user terminal.
SOML FROM <host>	Sends mail to user terminal or mailbox.
TURN	Switches role of receiver and sender.
VRFY <user>	Verifies the identity of a user.

Data value: Content_TEXT\Plain;name=”mimetest.txt”

Data values in other bases:

Hexadecimal	4	3	6	F	6	E	7	4
Binary	0100	0011	0110	1111	0110	1110	0111	0100
Decimal	67		111		110		116	
ASCII	C		o		n		t	

Hexadecimal	6	5	6	E	7	4	2	D
Binary	0110	0101	0110	1110	0111	0100	0010	1101
Decimal	101		110		116		45	
ASCII	e		n		t		-	

Hexadecimal	5	4	4	5	5	8	5	4
Binary	0101	0100	0100	0101	0101	1000	0101	0100
Decimal	84		69		88		84	
ASCII	T		E		X		T	

Hexadecimal	2	F	5	0	6	C	6	1
Binary	0010	1111	0101	0000	0110	1100	0110	0001
Decimal	47		80		108		97	
ASCII	/		P		l		a	

Hexadecimal	6	9	6	E	3	B	6	9
Binary	0110	1001	0110	1110	0011	1011	0110	1001
Decimal	105		110		59		110	
ASCII	i		n		;		n	

Hexadecimal	6	1	6	D	6	5	3	D
Binary	0110	0001	0110	1101	0110	0101	0011	1101
Decimal	97		109		101		61	
ASCII	a		m		e		=	

Hexadecimal	2	0	6	3	6	8	6	1
Binary	0010	0000	0110	0011	0110	1000	0110	0001
Decimal	32		99		104		97	
ASCII	“		M		I		m	

Hexadecimal	2	2	7	4	6	5	7	3
Binary	0010	0010	0111	0100	0110	0101	0111	0011
Decimal	34		116		101		115	
ASCII	e		t		e		s	

Hexadecimal	7	4	2	E	7	4	7	8
Binary	0111	0100	0010	1110	0111	0100	0111	1000
Decimal	116		46		116		120	
ASCII	t		.		t		x	

Hexadecimal	7	4	2	0
Binary	0111	0100	0010	0000
Decimal	116		32	
ASCII	t		“	

Programming Hint: The name for this variable in code will be IP_SMTP_Command_SMTP.

IP > SMTP Header > *Message* for the selected SMTP PDU

Field Name: *Message*

Purpose and Definition: Response messages consist of a response code followed by explanatory text

Field Key:

Response Code	Explanatory Text
211	(Response to system status or help request).
214	(Response to help request).
220	Mail service ready.
221	Mail service closing connection.
250	Mail transfer completed.
251	User not local, forward to <path>.
354	Start mail message, end with <CRLF><CRLF>.
421	Mail service unavailable.
450	Mailbox unavailable.
451	Local error in processing command.
452	Insufficient system storage.
500	Unknown command.
501	Bad parameter.
502	Command not implemented.
503	Bad command sequence.
504	Parameter not implemented.
550	Mailbox not found.
551	User not local, try <path>.
552	Storage allocation exceeded.
553	Mailbox name not allowed.
554	Mail transaction failed.

Data value: *Not applicable.*

Programming Hint: The name for this variable in code will be IP_SMTP_Command_SMTP.

2.2.9 IP PDU for the selected UDP PDU

IP PDU > *IP Version* for the selected UDP PDU

Field Name: *IP Version*

Purpose and Definition: Version is a 4-bit field that indicates the format of the internet header.

Field Key: 4 = IPv4
6 = IPv6

Data value (decimal): 4

Data values in other bases:

Hexadecimal	4
Binary	0100
Decimal	4

Programming Hint: The name for this variable in code will be IP_IPVersion_UDP.

IP PDU > *Internet Header Length* for the selected UDP PDU

Field Name: *Internet Header Length*

Purpose and Definition: The IHL field is a 4-bit field indicating the length of the internet header in 32 bit words, and thus points to the beginning of the data. The minimum value of a correct header is 5.

Field Key: *Not applicable*

Data value: The value contained in our field is 20 bytes. This is the hexadecimal and decimal value of 5 multiplied by 4 bits.

Data values in other bases:

Hexadecimal	0	5
Binary	0000	0101
Decimal	5	

Programming Hint: The name for this variable in code will be IP_IHL_UDP.

IP PDU > *Type of Service* for the selected UDP PDU

Field Name: *Type of Service*

Purpose and Definition: Type of Service is an 8-bit field that provides and indication of the abstract parameters of the quality of service desired. These parameters guide the selection of the actual service parameters when transmitting a datagram through a particular network.

Field Key: The major choice is a three-way tradeoff between low-delay, high-reliability, and high-throughput.

0 1 2	3	4	5	6	7
Precedence	D	T	R	0	0

Bits 0-2: Precedence
Bit 3: (D) 0 = Normal Delay 1 = Low Delay
Bit 4: (T) 0 = Normal Throughput 1 = High Throughput
Bit 5: (R) 0 = Normal Reliability 1 = High Reliability

Precedence:

111 = Network Control	011 = Flash
110 = Internetwork Control	010 = Immediate
101 = CRITIC/ECP	001 = Priority
100 = Flash Overrided	000 = Routine

Data value (hexadecimal): 10

Data values in other bases:

Hexadecimal	1	0
Binary	0001	0000
Decimal	16	

Programming Hint: The name for this variable in code will be IP_TypeOfService_UDP.

IP PDU > *Total Length of Ethernet Frame* for the selected UDP PDU

Field Name: *Total Length of Ethernet Frame*

Purpose and Definition: Total Length is a 16-bit field that indicates the length of the frame, measured in octets, including internet header and data. The maximum size is 2^{16}-1 or 65,535 octets; however, the recommended maximum size is 576 octets.

Field Key: *Not applicable*

Data values (hexadecimal): 128

Data values in other bases:

Hexadecimal	0	1	2	8
Binary	0000	0001	0010	1000
Decimal	1		40	
ASCII			↑	

Programming Hint: The name for this variable in code will be IP_TotalLength_UDP.

IP PDU > *Identification* for the selected UDP PDU

Field Name: *Identification*

Purpose and Definition: Identification is a 16-bit field. An identifying value is assigned by the sender to aid in assembling the fragments of a datagram. The identifier is chosen based on the need to provide a way to uniquely identify the fragments and protocol for the time the datagram or any fragment could be alive in the internet.

Field Key: *Not applicable*

Data value (hexadecimal): BBD7

Data values in other bases:

Hexadecimal	B	B	D	7
Binary	1011	1011	1101	0111

Programming Hint: The name for this variable in code will be IP_Idenfification_UDP.

IP PDU > *Flags* for the selected UDP PDU

Field Name: *Flags*

Purpose and Definition: Flags is a 3-bit field that indicates directions for fragmentation.

Field Key:
Bit 0: reserved, must be 0
Bit 1: (DF) 0 = May Fragment 1 = Don't Fragment
Bit 2: (MF) 0 = Last Fragment 1 = More Fragment

Data value (binary): 0000

Data values in other bases: *Not applicable*

Programming Hint: The name for this variable in code will be IP_Flags_UDP.

IP PDU > *Fragment Offset* for the selected UDP PDU

Field Name: *Fragment Offset*

Purpose and Definition: The Fragment Offset is a 13- bit field indicating where in the Ethernet frame this fragment begins. The Fragment Offset is measured in units of 8 octets, and the first fragment has offset 0.

Field Key: *Not applicable*

Data value (decimal): 0

Data values in other bases:
Binary: 0000 0000

Programming Hint: The name for this variable in code will be IP_FragmentOffset_UDP.

IP PDU > *Time to Live* for the selected UDP PDU

Field Name: *Time to Live*

Purpose and Definition: Time to Live is an 8-bit field that indicates the maximum time the datagram is allowed to remain in the internet. If this field contains the value 0, then the datagram must be destroyed. This field is modified in internet header processing. The time is measure in units of seconds, and is set by the sender to the maximum time the datagram is allowed to be in the internet. This field is decreased at each point that the internet header is processed. The intention is to cause undeliverable packets to be discarded, and to bind the maximum datagram lifetime.

Field Key: *Not applicable*

Data value (decimal): 64

Data values in other bases:

Hexadecimal	4	0
Binary	0100	0000
Decimal	64	

Programming Hint: The name for this variable in code will be IP_TimeToLive_UDP.

IP PDU > *Time to Live* for the selected UDP PDU

Field Name: *Time to Live*

Purpose and Definition: Time to Live is an 8-bit field that indicates the maximum time the datagram is allowed to remain in the internet. If this field contains the value 0, then the datagram must be destroyed. This field is modified in internet header processing. The time is measure in units of seconds, and is set by the sender to the maximum time the datagram is allowed to be in the internet. This field is decreased at each point that the internet header is processed. The intention is to cause undeliverable packets to be discarded, and to bind the maximum datagram lifetime.

Field Key: *Not applicable*

Data value (decimal): 60

Data values in other bases:

Hexadecimal	4	0
Binary	0100	0000
Decimal	60	

Programming Hint: The name for this variable in code will be IP_TimeToLive_UDP.

IP PDU > *Protocol* for the selected UDP PDU

Field Name: *Protocol*

Purpose and Definition: Protocol is an 8-bit field that indicates the next level protocol that is used in the data portion of the internet diagram.

Field Key:

Dec	Hex	Protocol	Dec	Hex	Protocol
0	00	Reserved	22	16	Multiplexing
1	01	ICMP	23	17	DCN
2	02	Unassigned	24	18	TAC Monitoring
3	03	Gateway-to-Gateway	25-76	19-4C	Unassigned
4	04	CMCC Gateway Monitoring Message	77	4D	Any local network
5	05	ST	100	64	SATNET and Backroom EXPAK
6	06	TCP	101	65	MIT Subnet Support
7	07	UCL	102-104	66-68	Unassigned
10	0A	Unassigned	105	69	SATNET Monitoring
11	0B	Secure	106	6A	Unassigned
12	0C	BBN RCC Monitoring	107	6B	Internet Packet Core Utility
13	0D	NVP	110-113	6E-71	Unassigned
14	0E	PUP	114	72	Backroom SATNET Monitoring
15	0F	Pluribus	115	73	Unassigned
16	10	Telnet	116	74	WIDEBAND Monitoring
17	11	XNET	117	75	WIDEBAND EXPAK
20	14	Chaos	120-376	78-0178	Unassigned
21	15	User Datagram	377	0179	Reserved

Data value (hexadecimal): 11

Data values in other bases:

Hexadecimal	1	1
Binary	0001	0001
Decimal	17	

RFC Link: http://www.faqs.org/rfcs/rfc790.html

Programming Hint: The name for this variable in code will be IP_Protocol_UDP

IP PDU > *Header Checksum* for the selected UDP PDU

Field Name: *Header Checksum*

Purpose and Definition: The Header Checksum is a 16-bit field. The Checksum is the 16-bit one's complement sum of all 16-bit words in the header. For purposes of computing the checksum, the initial value of its field is zero. When both header checksums are equal, then the header bits are correct. If either checksums vary, then a new, correct packet will need to be sent.

This is a simple way to compute the checksum and experimental evidence indicates that it is adequate, but it is provisional and may be replaced by a CRC procedure, depending on further experience.

Field Key: *Not applicable*

Data value (hexadecimal): 3F 47

Data values in other bases:

Hexadecimal	3	F	4	7
Binary	0011	1111	0100	0111

Programming Hint: The name for this variable in code will be IP_HeaderChecksum_UDP.

IP PDU > *Source Address* for the selected UDP PDU

Field Name: *Source Address*

Purpose and Definition: The Source Address is a 32-bit field that contains the IP address of the host that sent the IP Packet.

Field Key: *Not applicable*

Data value (decimal): 192.168.0.71

Data values in other bases:

Hexadecimal	C	0	A	8	0	0	4	7
Binary	1100	0000	1010	1000	0000	0000	0100	0111
Decimal	192		168		0		71	

Programming Hint: The name for this variable in code will be IP_SourceAddress_UDP.

IP PDU > *Destination Address* for the selected UDP PDU

Field Name: *Destination Address*

Purpose and Definition: The Destination Address is a 32-bit field that contains the address of the host that is to receive the data contained within the IP packet.

Field Key: *Not applicable*

Data value (decimal): 192.168.0.255

Data values in other bases:

Hexadecimal	C	0	A	8	0	0	F	F
Binary	1100	0000	1010	1000	0000	0000	1111	1111
Decimal	192		168		0		255	

Programming Hint: The name for this variable in code will be IP_DestinationAddress_UDP.

IP PDU > *Options and Padding* for the selected UDP PDU

Field Name: *Options and Padding*

Purpose and Definition: The options may or may not appear in Ethernet packets. They must be implemented by all IP modules (host and gateways). What is optional is their transmission in any particular packet, not their implementation.

The option field is variable in length. There may be zero or more options. There are two cases for the format of an option.

Case 1: A single octet of option type

Case 2: An option-type octet, an option-length octet, and the actual option-data octets.

Field Key: *Not applicable*

Data values: *Not applicable*

Data values in other bases: *Not applicable*

Programming Hint: The name for this variable in code will be IP_OptionsPadding_UDP.

IP PDU > *Data* for the selected UDP PDU

Field Name: *Data*

Purpose and Definition: The Data is a variable length field which contains the actual data that is being sent from one host to another. The data field may start with a Layer 4 header, which will give additional instructions to the application that will be receiving the data; alternately, it may be an ICMP header and not contain any user data at all.

Field Key: *Not applicable*

Data values (hexadecimal) : (UDP) FF FF FF FF FF FF 08 00 69 07 9A C5 08 00 45 10 01 28 BB D7 00 00 3C 11 3F 47 C0 A8 00 47 C0 A8 00 FF 02 0D 01 14 E9 DB 18 01 C2 5A 0A FF 13 D0 00 00 00 00 69 6E 64 79 00 00 00 00 7F FF 11 DB 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 7F FF 13 F0 00 00 00 00 7F FF 11 F6 00 00 00 00 00 00 00 00 00 00 00 7F FF FF FF 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 09 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 7F FF 17 E3 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 09 00 00 00 10 05 9D D8 00 00 00 00 10 00 DB A8 00 00 00 00 0F B5 B1 74 00 00 00 00 0F AB 01 C8 00 00 00 00 10 05 9D F8 00 00 00 00 0F B5 9D D8 00 00 00 00 10 05 9D FC 00 00 00 00 0F B5 B1 74 00 00 00 00 10 05 9B 18 00 00 00 00 0F AB 02 80 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Data values in other bases:
ASCII: (UDP) ↑ Extended ASCII

Programming Hint: The name for this variable in code will be IP_Data_UDP.

IP PDU > *Source Port* for the selected UDP PDU

RFC Link: http://www.ietf.org/rfc/rfc0768.txt?number=768

Field Name: *Source Port*

Purpose and Definition: Source Port is an optional field, when meaningful, it indicates the port of the sending process, and may be assumed to be the port to which a reply should be addressed in the absence of any other information. If not used, a value of zero is inserted.

Field Key: *Not applicable*

Data value (decimal): 525

Data values in other bases:

Hexadecimal	02	0D
Binary	0010	1101
Decimal	525	

Programming Hint: The name for this variable in code will be IP_SourcePort_UDP.

IP PDU > *Destination Port* for the selected UDP PDU

Field Name: *Destination Port*

Purpose and Definition: Destination Port has a meaning within the context of a particular internet destination address.

Field Key: *Not applicable*

Data value (decimal): 525

Data values in other bases:

Hexadecimal	02	0D
Binary	0010	1101
Decimal	525	

Programming Hint: The name for this variable in code will be IP_DestinationPort_UDP.

IP PDU > *Length* for the selected UDP PDU

Field Name: *Length*

Purpose and Definition: Length is the length in octets of this user datagram including this header and the data (This means the minimum value of the length is eight).

Field Key: *Not applicable*

Data value (decimal): 276

Data values in other bases:

Hexadecimal	01	14
Binary	0001	00001 0100
Decimal	276	

Programming Hint: The name for this variable in code will be IP_Length_UDP.

IP PDU > *Checksum* for the selected UDP PDU

Field Name: *Checksum*

Purpose and Definition: Checksum is the 16-bit one's complement of the one's complement sum of a pseudo header of information from the IP header, the UDP header, and the data, padded with zero octets at the end (if necessary) to make a multiple of two octets.

Field Key: *Not applicable*

Data value (decimal): E9 DB

Data values in other bases:

Hexadecimal	E	9	D	B
Binary	1110	1001	1101	1011

Programming Hint: The name for this variable in code will be IP_Checksum_UDP.

IP PDU > *Data* for the selected UDP PDU

Field Name: *Data*

Purpose and Definition:

Field Key: *Not applicable*

Data value (hexadecimal): 18 01 C2 5A 0A FF 13 D0 00 00 00 00 69 6E 64 79 00 00 00 00 7F FF 11 DB 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 7F FF 13 F0 00 00 00 00 7F FF 11 F6 00 00 00 00 00 00 00 00 00 00 00 7F FF FF FF 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 09 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 7F FF 17 E3 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 09 00 00 00 10 05 9D D8 00 00 00 00 10 00 DB A8 00 00 00 00 0F B5 B1 74 00 00 00 00 0F AB 01 C8 00 00 00 00 10 05 9D F8 00 00 00 00 0F B5 9D D8 00 00 00 00 10 05 9D FC 00 00 00 00 0F B5 B1 74 00 00 00 00 10 05 9B 18 00 00 00 00 0F AB 02 80 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Data values in other bases:
(ASCII): ↑ Extended ASCII

Programming Hint: The name for this variable in code will be IP_Data_UDP.

2.2.10 UDP PDU for the selected UDP PDU

IP > UDP Header> *Source Port* for the selected UDP PDU

RFC Link: http://www.ietf.org/rfc/rfc0768.txt?number=768

Field Name: *Source Port*

Purpose and Definition: Source Port is an optional field, when meaningful, it indicates the port of the sending process, and may be assumed to be the port to which a reply should be addressed in the absence of any other information. If not used, a value of zero is inserted.

Field Key: *Not applicable*

Data value (decimal): 525

Data values in other bases:

Hexadecimal	02	0D
Binary	0010	1101
Decimal	525	

Programming Hint: The name for this variable in code will be IP_SourcePort_UDP.

IP > UDP Header > *Destination Port* for the selected UDP PDU

Field Name: *Destination Port*

Purpose and Definition: Destination Port has a meaning within the context of a particular internet destination address.

Field Key: *Not applicable*

Data value (decimal): 525

Data values in other bases:

Hexadecimal	02	0D
Binary	0010	1101
Decimal	525	

Programming Hint: The name for this variable in code will be IP_DestinationPort_UDP.

IP > UDP Header> *Length* for the selected UDP PDU

Field Name: *Length*

Purpose and Definition: Length is the length in octets of this user datagram including this header and the data (This means the minimum value of the length is eight).

Field Key: *Not applicable*

Data value (decimal): 276

Data values in other bases:

Hexadecimal	01	14
Binary	0001	00001 0100
Decimal	276	

Programming Hint: The name for this variable in code will be IP_Length_UDP.

IP > UDP Header > *Checksum* for the selected UDP PDU

Field Name: *Checksum*

Purpose and Definition: Checksum is the 16-bit one's complement of the one's complement sum of a pseudo header of information from the IP header, the UDP header, and the data, padded with zero octets at the end (if necessary) to make a multiple of two octets.

Field Key: *Not applicable*

Data value (decimal): E9 DB

Data values in other bases:

Hexadecimal	E	9	D	B
Binary	1110	1001	1101	1011

Programming Hint: The name for this variable in code will be IP_Checksum_UDP.

IP > UDP Header > *Data* for the selected UDP PDU

Field Name: *Data*

Purpose and Definition:

Field Key: *Not applicable*

Data value (hexadecimal): 18 01 C2 5A 0A FF 13 D0 00 00 00 00 69 6E 64 79 00 00 00 00 7F FF 11 DB 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 7F FF 13 F0 00 00 00 00 7F FF 11 F6 00 00 00 00 00 00 00 00 00 00 00 7F FF FF FF 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 09 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 7F FF 17 E3 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 09 00 00 00 10 05 9D D8 00 00 00 00 10 00 DB A8 00 00 00 00 0F B5 B1 74 00 00 00 00 0F AB 01 C8 00 00 00 00 10 05 9D F8 00 00 00 00 0F B5 9D D8 00 00 00 00 10 05 9D FC 00 00 00 00 0F B5 B1 74 00 00 00 00 10 05 9B 18 00 00 00 00 0F AB 02 80 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Data values in other bases:
(ASCII): ↑ Extended ASCII

Programming Hint: The name for this variable in code will be IP_Data_UDP.

2.2.11 IP PDU for the selected SNMP PDU

IP PDU > *IP Version* for the selected SNMP PDU

Field Name: *IP Version*

Purpose and Definition: Version is a 4-bit field that indicates the format of the internet header.

Field Key: 4 = IPv4
6 = IPv6

Data value (decimal): 4

Data values in other bases:

Hexadecimal	4
Binary	0100
Decimal	4

Programming Hint: The name for this variable in code will be IP_IPVersion_SNMP.

IP PDU > *Internet Header Length* for the selected SNMP PDU

Field Name: *Internet Header Length*

Purpose and Definition: The IHL field is a 4-bit field indicating the length of the internet header in 32 bit words, and thus points to the beginning of the data. The minimum value of a correct header is 5.

Field Key: *Not applicable*

Data value: The value contained in our field is 20 bytes. This is the hexadecimal and decimal value of 5 multiplied by 4 bits.

Data values in other bases:

Hexadecimal	0	5
Binary	0000	0101
Decimal	5	

Programming Hint: The name for this variable in code will be IP_IHL_SNMP.

IP PDU > *Type of Service* for the selected SNMP PDU

Field Name: *Type of Service*

Purpose and Definition: Type of Service is an 8-bit field that provides and indication of the abstract parameters of the quality of service desired. These parameters guide the selection of the actual service parameters when transmitting a datagram through a particular network.

Field Key: The major choice is a three-way tradeoff between low-delay, high-reliability, and high-throughput.

0 1 2	3	4	5	6	7
Precedence	D	T	R	0	0

Bits 0-2: Precedence
Bit 3: (D) 0 = Normal Delay 1 = Low Delay
Bit 4: (T) 0 = Normal Throughput 1 = High Throughput
Bit 5: (R) 0 = Normal Reliability 1 = High Reliability

Precedence:

111 = Network Control	011 = Flash
110 = Internetwork Control	010 = Immediate
101 = CRITIC/ECP	001 = Priority
100 = Flash Overrided	000 = Routine

Data value (hexadecimal): 10

Data values in other bases:

Hexadecimal	1	0
Binary	0001	0000
Decimal	16	

Programming Hint: The name for this variable in code will be IP_TypeOfService_SNMP.

IP PDU > *Total Length of Ethernet Frame* for the selected SNMP PDU

Field Name: *Total Length of Ethernet Frame*

Purpose and Definition: Total Length is a 16-bit field that indicates the length of the frame, measured in octets, including internet header and data. The maximum size is 2^{16}-1 or 65,535 octets; however, the recommended maximum size is 576 octets.

Field Key: *Not applicable*

Data values (decimal): 109

Data values in other bases:

Hexadecimal	6	D
Binary	110	1101
Decimal	109	

Programming Hint: The name for this variable in code will be IP_TotalLength_SNMP.

IP PDU > *Identification* for the selected SNMP PDU

Field Name: *Identification*

Purpose and Definition: Identification is a 16-bit field. An identifying value is assigned by the sender to aid in assembling the fragments of a datagram. The identifier is chosen based on the need to provide a way to uniquely identify the fragments and protocol for the time the datagram or any fragment could be alive in the internet.

Field Key: *Not applicable*

Data value (hexadecimal): D5 1A

Data values in other bases:

Hexadecimal	D	5	1	A
Binary	1101	0101	0001	1010

Programming Hint: The name for this variable in code will be IP_Idenfification_SNMP.

IP PDU > *Flags* for the selected SNMP PDU

Field Name: *Flags*

Purpose and Definition: Flags is a 3-bit field that indicates directions for fragmentation.

Field Key:
Bit 0: reserved, must be 0
Bit 1: (DF) 0 = May Fragment 1 = Don't Fragment
Bit 2: (MF) 0 = Last Fragment 1 = More Fragment

Data value (binary): 010

Data values in other bases: *Not applicable*

Programming Hint: The name for this variable in code will be IP_Flags_SNMP.

IP PDU > *Fragment Offset* for the selected SNMP PDU

Field Name: *Fragment Offset*

Purpose and Definition: The Fragment Offset is a 13- bit field indicating where in the Ethernet frame this fragment begins. The Fragment Offset is measured in units of 8 octets, and the first fragment has offset 0.

Field Key: *Not applicable*

Data value (decimal): 0

Data values in other bases:
Binary: 0000 0000 0000

Programming Hint: The name for this variable in code will be IP_FragmentOffset_SNMP.

IP PDU > *Time to Live* for the selected SNMP PDU

Field Name: *Time to Live*

Purpose and Definition: Time to Live is an 8-bit field that indicates the maximum time the datagram is allowed to remain in the internet. If this field contains the value 0, then the datagram must be destroyed. This field is modified in internet header processing. The time is measure in units of seconds, and is set by the sender to the maximum time the datagram is allowed to be in the internet. This field is decreased at each point that the internet header is processed. The intention is to cause undeliverable packets to be discarded, and to bind the maximum datagram lifetime.

Field Key: *Not applicable*

Data value (decimal): 64

Data values in other bases:

Hexadecimal	4	0
Binary	0100	0000
Decimal	64	

Programming Hint: The name for this variable in code will be IP_TimeToLive_SNMP.

IP PDU > *Protocol* for the selected SNMP PDU

Field Name: *Protocol*

Purpose and Definition: Protocol is an 8-bit field that indicates the next level protocol that is used in the data portion of the internet diagram.

Field Key:

Dec	Hex	Protocol	Dec	Hex	Protocol
0	00	Reserved	22	16	Multiplexing
1	01	ICMP	23	17	DCN
2	02	Unassigned	24	18	TAC Monitoring
3	03	Gateway-to-Gateway	25-76	19-4C	Unassigned
4	04	CMCC Gateway Monitoring Message	77	4D	Any local network
5	05	ST	100	64	SATNET and Backroom EXPAK
6	06	TCP	101	65	MIT Subnet Support
7	07	UCL	102-104	66-68	Unassigned
10	0A	Unassigned	105	69	SATNET Monitoring
11	0B	Secure	106	6A	Unassigned
12	0C	BBN RCC Monitoring	107	6B	Internet Packet Core Utility
13	0D	NVP	110-113	6E-71	Unassigned
14	0E	PUP	114	72	Backroom SATNET Monitoring
15	0F	Pluribus	115	73	Unassigned
16	10	Telnet	116	74	WIDEBAND Monitoring
17	11	XNET	117	75	WIDEBAND EXPAK
20	14	Chaos	120-376	78-0178	Unassigned
21	15	User Datagram	377	0179	Reserved

Data value (hexadecimal): 06

Data values in other bases:

Hexadecimal	1	1
Binary	1	0001
Decimal	17	

RFC Link: http://www.faqs.org/rfcs/rfc790.html

Programming Hint: The name for this variable in code will be IP_Protocol_SNMP.

IP PDU > *Header Checksum* for the selected SNMP PDU

Field Name: *Header Checksum*

Purpose and Definition: The Header Checksum is a 16-bit field. The Checksum is the 16-bit one's complement sum of all 16-bit words in the header. For purposes of computing the checksum, the initial value of its field is zero. When both header checksums are equal, then the header bits are correct. If either checksums vary, then a new, correct packet will need to be sent.

This is a simple way to compute the checksum and experimental evidence indicates that it is adequate, but it is provisional and may be replaced by a CRC procedure, depending on further experience.

Field Key: *Not applicable*

Data value (hexadecimal): 22 F0

Data values in other bases:

Hexadecimal	2	2	F	0
Binary	10	0010	1111	0000

Programming Hint: The name for this variable in code will be IP_HeaderChecksum_SNMP

IP PDU > *Source Address* for the selected SNMP PDU

Field Name: *Source Address*

Purpose and Definition: The Source Address is a 32-bit field that contains the IP address of the host that sent the IP Packet.

Field Key: *Not applicable*

Data value (decimal): 192.168.0.39

Data values in other bases:

Hexadecimal	C	0	A	8	0	0	2	7
Binary	1100	0000	1010	1000	0000	0000	0010	0111
Decimal	192		168		0		39	

Programming Hint: The name for this variable in code will be IP_SourceAddress_SNMP

.

IP PDU > *Destination Address* for the selected SNMP PDU

Field Name: *Destination Address*

Purpose and Definition: The Destination Address is a 32-bit field that contains the address of the host that is to receive the data contained within the IP packet.

Field Key: *Not applicable*

Data value (decimal): 192.168.0.143

Data values in other bases:

Hexadecimal	C	0	A	8	0	0	8	f
Binary	1100	0000	1010	1000	0000	0000	1000	1111
Decimal	192		168		0		143	

Programming Hint: The name for this variable in code will be IP_DestinationAddress_SNMP

2.2.12 UDP PDU for the selected SNMP PDU

IP > UDP PDU > *Source Port* for the selected SNMP PDU

RFC Link: http://www.ietf.org/rfc/rfc0768.txt?number=768

Field Name: *Source Port*

Purpose and Definition: Source Port is an optional field, when meaningful, it indicates the port of the sending process, and may be assumed to be the port to which a reply should be addressed in the absence of any other information. If not used, a value of zero is inserted.

Field Key: *Not applicable*

Data value (decimal): 161

Data values in other bases:

Hexadecimal	A	1
Binary	1010	0001
Decimal	161	

Programming Hint: The name for this variable in code will be IP_SourcePort_UDP_SMNP.

IP > UDP PDU> D*estination Port* for the selected SNMP PDU

Field Name: *Destination Port*

Purpose and Definition: Destination Port has a meaning within the context of a particular internet destination address.

Field Key: *Not applicable*

Data value (decimal): 1034

Data values in other bases:

Hexadecimal	40	0A
Binary	0010	1101
Decimal	1034	

Programming Hint: The name for this variable in code will be IP_DestinationPort_UDP_SMNP.

IP > UDP *Length* for the selected SNMP PDU

Field Name: *Length*

Purpose and Definition: Length is the length in octets of this user datagram including this header and the data (This means the minimum value of the length is eight).

Field Key: *Not applicable*

Data value (decimal): 89

Data values in other bases:

Hexadecimal	5	9
Binary	0101	1001
Decimal	89	

Programming Hint: The name for this variable in code will be IP_Length_UDP_SNMP.

IP > UDP PDU > *Checksum* for the selected SNMP PDU

Field Name: *Checksum*

Purpose and Definition: Checksum is the 16-bit one's complement of the one's complement sum of a pseudo header of information from the IP header, the UDP header, and the data, padded with zero octets at the end (if necessary) to make a multiple of two octets.

Field Key: *Not applicable*

Data value (hexadecimal): 9A25

Data values in other bases:

Hexadecimal	9	A	2	5
Binary	1001	0000	0010	0101
Decimal	154		37	

Programming Hint: The name for this variable in code will be IP_Checksum_UDP_SNMP.

IP > UDP PDU > *Data* for the selected SNMP PDU

Field Name: *Data*

Purpose and Definition:

Field Key: *Not applicable*

Data value (hexadecimal): see SNMP

Data values in other bases:
(ASCII): ↑ Extended ASCII

Programming Hint: The name for this variable in code will be IP_Data_UDP_SNMP.

2.2.13 SNMP PDU for the selected SNMP PDU

IP > UDP > SNMP Header > *Version* for the selected SNMP PDU

Field Name: *Version*

Purpose and Definition: Version is a 6-bit field that indicates the format of the protocol

Field Key: *Not applicable*

Data value (hexadecimal): 02 01 00

Data values in other bases:

Hexadecimal	0	2	0	1	0	0
Binary	0000	0010	0000	0001	0000	0000
Decimal	2		1		0	

Programming Hint: The name for this variable in code will be IP_UDP_SNMP_Version.

IP > UDP > SNMP Header > *Community* for the selected SNMP PDU

Field Name: *Community*

Purpose and Definition:

Field Key: Public: all users
Private: Selected users

Data value: The value contained in our field determines who view the information

Data values in other bases:

Hexadecimal	0	6	7	0	7	5	6	2	6	C
Binary	0000	0110	0111	0000	0111	0101	0110	0010	0110	1100
Decimal	1		1		8		10		7	
ASCII	©		©		©		©		©	

Hexadecimal	6	9	6	3
Binary	0110	1001	0110	0011
Decimal	105		99	
ASCII	i		c	

Programming Hint: The name for this variable in code will be IP_UDP_SNMP_Community.

IP > UDP > SNMP Header > *PDU Type* for the selected SNMP PDU

Field Name: *PDU Type*

Purpose and Definition: The type of data.

Field Key: *Not applicable*

Data value (hexadecimal): A2 42

Values in other bases:

Hexadecimal	A	2	4	2
Binary	1010	0010	0100	0010
Decimal	162		66	
ASCII	↑		B	

Programming Hint: The name for this variable in code will be IP_UDP_SNMP_PDUType

IP > UDP > SNMP Header > *Request ID* for the selected SNMP PDU

Field Name: *Request ID*

Purpose and Definition: ID of the requester.

Field Key: *Not applicable*

Data value (hexadecimal): 51 EB

Data values in other bases:

Hexadecimal	5	1	E	B
Binary	0101	0001	1110	1011
Decimal	81		235	
ASCII	Q		↑	

Programming Hint: The name for this variable in code will be IP_UDP_SNMP_Request.

IP > UDP > SNMP Header > *Error Status* for the selected SNMP PDU

Field Name: *Error Status*

Purpose and Definition: If there is an error, it will show here

Field Key: *Not applicable.*

Data value: No error

Data values in other bases: *Not applicable*

Programming Hint: The name for this variable in code will be IP_UDP_SNMP_ErrorStatus.

IP > UDP > SNMP Header > *Error ID* for the selected SNMP PDU

Field Name: *Error Index*

Purpose and Definition: How the error is defined.

Field Key: *Not applicable.*

Data value (hexadecimal): 02 01 00

Data values in other bases:

Hexadecimal	0	2	0	1	0	0
Binary	0000	0010	0000	0001	0000	0000
Decimal	2		1		0	

Programming Hint: The name for this variable in code will be IP_UDP_SNMP_ErrorIndex.

IP > UDP > SNMP Header > *Object ID* for the selected SNMP PDU

Field Name: *Object ID*

Purpose and Definition: How the packet is identified.

Field Key: *Not applicable*

Data value: 1.3.6.1.2.1.25.3.5.1.1.1

Data values in other bases:

Hexadecimal	1	3	6	1	2	1	2	5	3	5
Binary	0000	0011	0110	0001	0010	0001	0010	0101	0011	0101
Decimal	1	3	6	1	2	1	37		3	5
ASCII	©	©	©	©	©	©	%		©	©

Hexadecimal	1	1	1
Binary	0001	0001	0001
Decimal	1	1	1
ASCII	©	©	©

Programming Hint: The name for this variable in code will be IP_UDP_SNMP_ObjectID

IP > UDP > SNMP Header > *Value Integer* for the selected SNMP PDU

Field Name: *Value Integer*

Purpose and Definition: The size of the integer

Field Key: *Not applicable.*

Data value (hexadecimal): 3

Data values in other bases:

Hexadecimal	3
Binary	11
Decimal	3

Programming Hint: The name for this variable in code will be IP_UDP_SNMP_Value_Integer

IP > UDP > SNMP Header > *Object ID* for the selected SNMP PDU

Field Name: *Object ID*

Purpose and Definition: How the packet is identified.

Field Key: *Not applicable*

Data value: 1.3.6.1.2.1.25.3.5.1.1.1

Data values in other bases:

Hexadecimal	1 3 6 1 2 1 25 3 5 1 1 1	
Binary	0001 0011 0110 0001 0010 0001 11001 0011 0101 0001 0001 0001	
Decimal	n/a	

Programming Hint: The name for this variable in code will be SNMP_Object Id

2.2.14 IP PDU for the TELNET PDU

IP PDU > *IP Version* for the selected TELNET PDU

Field Name: *IP Version*

Purpose and Definition: Version is a 4-bit field that indicates the format of the internet header.

Field Key: 4 = IPv4
6 = IPv6

Data value (decimal): 4

Data values in other bases:

Hexadecimal	4
Binary	0100
Decimal	4

Programming Hint: The name for this variable in code will be IP_IPVersion_TELNET.

IP PDU > *Internet Header Length* for the selected TELNET PDU

Field Name: *Internet Header Length*

Purpose and Definition: The IHL field is a 4-bit field indicating the length of the internet header in 32 bit words, and thus points to the beginning of the data. The minimum value of a correct header is 5.

Field Key: *Not applicable*

Data value: The value contained in our field is 20 bytes. This is the hexadecimal and decimal value of 5 multiplied by 4 bits.

Data values in other bases:

Hexadecimal	5
Binary	0101
Decimal	5

Programming Hint: The name for this variable in code will be IP_IHL_TELNET.

IP PDU > *Type of Service* for the selected TELNET PDU

Field Name: *Type of Service*

Purpose and Definition: Type of Service is an 8-bit field that provides and indication of the abstract parameters of the quality of service desired. These parameters guide the selection of the actual service parameters when transmitting a datagram through a particular network.

Field Key: The major choice is a three-way tradeoff between low-delay, high-reliability, and high-throughput.

0 1 2	3	4	5	6	7
Precedence	D	T	R	0	0

Bits 0-2: Precedence
Bit 3: (D) 0 = Normal Delay 1 = Low Delay
Bit 4: (T) 0 = Normal Throughput 1 = High Throughput
Bit 5: (R) 0 = Normal Reliability 1 = High Reliability

Precedence:

111 = Network Control	011 = Flash
110 = Internetwork Control	010 = Immediate
101 = CRITIC/ECP	001 = Priority
100 = Flash Overrided	000 = Routine

Data value (hexadecimal): 10

Data values in other bases:

Hexadecimal	1	0
Binary	0001	0000
Decimal	16	

Programming Hint: The name for this variable in code will be IP_TypeOfService_TELNET.

IP PDU > *Total Length of Ethernet Frame* for the selected TELNET PDU

Field Name: *Total Length of Ethernet Frame*

Purpose and Definition: Total Length is a 16-bit field that indicates the length of the frame, measured in octets, including internet header and data. The maximum size is 2^{16}-1 or 65,535 octets; however, the recommended maximum size is 576 octets.

Field Key: *Not applicable*

Data values (decimal): 128

Data values in other bases:

Hexadecimal	8	0
Binary	1000	0000
Decimal	128	

Programming Hint: The name for this variable in code will be IP_TotalLength_TELNET.

IP PDU > *Identification* for the selected TELNET PDU

Field Name: *Identification*

Purpose and Definition: Identification is a 16-bit field. An identifying value is assigned by the sender to aid in assembling the fragments of a datagram. The identifier is chosen based on the need to provide a way to uniquely identify the fragments and protocol for the time the datagram or any fragment could be alive in the internet.

Field Key: *Not applicable*

Data value (hexadecimal): C7 57

Data values in other bases:

Hexadecimal	C	7	5	7
Binary	1100	0111	0101	0111
Decimal	199		87	

Programming Hint: The name for this variable in code will be IP_Idenfification_TELNET.

IP PDU > *Flags* for the selected TELNET PDU

Field Name: *Flags*

Purpose and Definition: Flags is a 3-bit field that indicates directions for fragmentation.

Field Key:
Bit 0: reserved, must be 0
Bit 1: (DF) 0 = May Fragment 1 = Don't Fragment
Bit 2: (MF) 0 = Last Fragment 1 = More Fragment

Data value (binary): 001

Data values in other bases: *Not applicable*

Programming Hint: The name for this variable in code will be IP_Flags_TELNET.

IP PDU > *Fragment Offset* for the selected TELNET PDU

Field Name: *Fragment Offset*

Purpose and Definition: The Fragment Offset is a 13- bit field indicating where in the Ethernet frame this fragment begins. The Fragment Offset is measured in units of 8 octets, and the first fragment has offset 0.

Field Key: *Not applicable*

Data value (decimal): 0

Data values in other bases:
Binary: 0000 0000 0000

Programming Hint: The name for this variable in code will be IP_FragmentOffset_TELNET.

IP PDU > *Time to Live* for the selected TELNET PDU

Field Name: *Time to Live*

Purpose and Definition: Time to Live is an 8-bit field that indicates the maximum time the datagram is allowed to remain in the internet. If this field contains the value 0, then the datagram must be destroyed. This field is modified in internet header processing. The time is measure in units of seconds, and is set by the sender to the maximum time the datagram is allowed to be in the internet. This field is decreased at each point that the internet header is processed. The intention is to cause undeliverable packets to be discarded, and to bind the maximum datagram lifetime.

Field Key: *Not applicable*

Data value (decimal): 64

Data values in other bases:

Hexadecimal	4	0
Binary	0100	0000
Decimal	64	

Programming Hint: The name for this variable in code will be IP_TimeToLive_TELNET.

IP PDU > *Protocol* for the selected TELNET PDU

Field Name: *Protocol*

Purpose and Definition: Protocol is an 8-bit field that indicates the next level protocol that is used in the data portion of the internet diagram.

Field Key:

Dec	Hex	Protocol	Dec	Hex	Protocol
0	00	Reserved	22	16	Multiplexing
1	01	ICMP	23	17	DCN
2	02	Unassigned	24	18	TAC Monitoring
3	03	Gateway-to-Gateway	25-76	19-4C	Unassigned
4	04	CMCC Gateway Monitoring Message	77	4D	Any local network
5	05	ST	100	64	SATNET and Backroom EXPAK
6	06	TCP	101	65	MIT Subnet Support
7	07	UCL	102-104	66-68	Unassigned
10	0A	Unassigned	105	69	SATNET Monitoring
11	0B	Secure	106	6A	Unassigned
12	0C	BBN RCC Monitoring	107	6B	Internet Packet Core Utility
13	0D	NVP	110-113	6E-71	Unassigned
14	0E	PUP	114	72	Backroom SATNET Monitoring
15	0F	Pluribus	115	73	Unassigned
16	10	Telnet	116	74	WIDEBAND Monitoring
17	11	XNET	117	75	WIDEBAND EXPAK
20	14	Chaos	120-376	78-0178	Unassigned
21	15	User Datagram	377	0179	Reserved

Data value (hexadecimal): 06

Data values in other bases:

Hexadecimal	0	6
Binary	0000	0110
Decimal	6	

RFC Link: http://www.faqs.org/rfcs/rfc790.html

Programming Hint: The name for this variable in code will be IP_Protocol_TELNET

IP PDU > *Header Checksum* for the selected TELNET PDU

Field Name: *Header Checksum*

Purpose and Definition: The Header Checksum is a 16-bit field. The Checksum is the 16-bit one's complement sum of all 16-bit words in the header. For purposes of computing the checksum, the initial value of its field is zero. When both header checksums are equal, then the header bits are correct. If either checksums vary, then a new, correct packet will need to be sent.

This is a simple way to compute the checksum and experimental evidence indicates that it is adequate, but it is provisional and may be replaced by a CRC procedure, depending on further experience.

Field Key: *Not applicable*

Data value (hexadecimal): F1 85

Data values in other bases:

Hexadecimal	F	1	8	5
Binary	1111	0001	0100	0101

Programming Hint: The name for this variable in code will be IP_HeaderChecksum_TELNET.

IP PDU > *Source Address* for the selected TELNET PDU

Field Name: *Source Address*

Purpose and Definition: The Source Address is a 32-bit field that contains the IP address of the host that sent the IP Packet.

Field Key: *Not applicable*

Data value (decimal): 192.168.0.101

Data values in other bases:

Hexadecimal	C	0	A	8	0	0	6	5
Binary	1100	0000	1010	1000	0000	0000	0110	0101
Decimal	192		168		0		101	

Programming Hint: The name for this variable in code will be IP_SourceAddress_TELNET.

IP PDU > *Destination Address* for the selected TELNET PDU

Field Name: *Destination Address*

Purpose and Definition: The Destination Address is a 32-bit field that contains the address of the host that is to receive the data contained within the IP packet.

Field Key: *Not applicable*

Data value (decimal): 192.168.0.39

Data values in other bases:

Hexadecimal	C	0	A	8	0	0	2	7
Binary	1100	0000	1010	1000	0000	0000	0010	0111
Decimal	192		168		0		39	

Programming Hint: The name for this variable in code will be IP_DestinationAddress_TELNET.

2.2.15 TCP PDU for TELNET PDU

IP > TCP PDU > *Source Port* for the selected TELNET PDU

Field Name: *Source Port*

Purpose and Definition:
This 16-bit number represents the name of the application that sent the data in the IP packet.

Field Key: *Not applicable*

Data value: TELNET (23)

Data values in other bases:

Hexadecimal	0	0	1	7
Binary	0000	0000	0001	0111
Decimal	0		23	
ASCII	©		©	

Programming Hint: The name for this variable in code will be IP_TCP_SourcePort_TELNET.

IP > TCP PDU > *Destination Port* for the selected TELNET PDU

Field Name: *Destination Port*

Purpose and Definition:
This 16-bit number represents the name of the application that is to receive the data contained within the IP packet. This is one of the major differences between a Layer 3 and a Layer 4 header: the Layer 3 header contains the IP address of the computer that is to receive the IP packet; once that packet has been received, the port address in the Layer 4 header ensures that the data contained within that IP packet is passed to the correct application on that computer.

Field Key:
This key indicates assigned port number values:

Dec	**Port Numbers**
0	Reserved
1-32767	Internet registered ("well-known") protocols
32768-98303	Reserved, to allow TCPv7-TCPv4 conversion
98304 & up	Dynamic assignment

Data value (hexadecimal): 8025

Data values in other bases:

Hexadecimal	8	0	2	5
Binary	1000	0000	0010	0101
Decimal	128		37	
ASCII	↑		%	

Source: http://www.zvon.org/tmRFC/RFC1475/Output/chapter4.html

Programming Hint: The name for this variable in code will be IP_TCP_DestinationPort_TELNET.

IP > TCP PDU > *Sequence Number* for the selected TELNET PDU

Field Name: *Sequence Number*

Purpose and Definition:
TCP is responsible for ensuring that all IP packets sent are actually received. When an application's data is packaged into IP packets, TCP will give each IP packet a sequence number. Once all the packets have arrived at the receiving computer, TCP uses the number in this 32-bit field to ensure that all of the packets actually arrived and are in the correct sequence.

Field Key: *Not applicable*

Data value (decimal): 2635302920

Data values in other bases:

Hexadecimal	9	D	1	3	8	8	0	8
Binary	1001	1101	0001	0011	1000	1000	0000	1000
Decimal	157		19		136		8	
ASCII	↑		©		↑		©	

Programming Hint: The name for this variable in code will be IP_TCP_SequenceNumber_TELNET.

IP > TCP PDU > *Acknowledgement Number* for the selected TELNET PDU

Field Name: *Acknowledgement Number*

Purpose and Definition:
This number is used by the receiving computer to acknowledge which packets have successfully arrived. This number will be the sequence number of the next packet the receiver is ready to receive.

Field Key: *Not applicable*

Data value: 2526101253

Data values in other bases:

Hexadecimal	9	6	9	1	3	F	0	5
Binary	1001	0110	1001	0001	0011	1111	0000	0101
Decimal	150		145		63		5	
ASCII	↑		↑		?		©	

Programming Hint: The name for this variable in code will be IP_TCP_AcknowledgementNumber_TELNET.

IP > TCP PDU > *Header Length or Offset* for the selected TELNET PDU

Field Name: *Header Length or Offset*

Purpose and Definition:
This is identical in concept to the header length in an IP packet, except this time it indicates the length of the TCP header.

Field Key: *Not applicable*

Data value (bytes): 32

Data values in other bases:

Hexadecimal	8	0
Binary	1000	0000
Decimal	128	
ASCII	↑	

Programming Hint: The name for this variable in code will be IP_TCP_HeaderLength_TELNET.

IP > TCP PDU > *Control Flags* for the selected TELNET PDU

Field Name: *Control Flags*

Purpose and Definition:
Every TCP packet contains this 6-bit value that indicates how many octets it can receive at once. When IP packets are received, they are placed in a temporary area of RAM known as a buffer until the receiving computer has a chance to process them; this value represents how big a buffer the receiving host has made available for this temporary storage of IP packets.

Field Key:
- Urgent (URG)
- Acknowledgement (ACK)
- Push (PSH)
- Reset (RST)
- Synchronize (SYN)
- Finish (FIN)

Data value (binary): 0001 1000

Data values in other bases:

Hexadecimal	1	8
Binary	0001	1000
Decimal	24	
ASCII	©	

Programming Hint: The name for this variable in code will be IP_TCP_ControlFlags_TELNET.

IP > TCP PDU > *Window Size* for the selected TELNET PDU

Field Name: *Window Size*

Purpose and Definition:
Every TCP packet contains this 16-bit value that indicates how many octets it can receive at once. When IP packets are received, they are placed in a temporary area of RAM known as a buffer until the receiving computer has a chance to process them; this value represents how big a buffer the receiving host has made available for this temporary storage of IP packets.

Field Key: *Not applicable*

Data value (decimal): 32120

Data values in other bases:

Hexadecimal	7	D	7	8
Binary	0111	1101	0111	1000
Decimal	125		120	
ASCII	}		x	

Programming Hint: The name for this variable in code will be IP_TCP_WindowSize_TELNET.

IP > TCP PDU > *Checksum* for the selected TELNET PDU

Field Name: *Checksum*

Purpose and Definition:
Unlike IP, TCP is responsible for ensuring that the entire IP packet arrived intact. TCP will run a CRC on the entire IP packet (not just the header) and place the resulting checksum in this field. When the IP packet is received, TCP re-runs the CRC on the entire packet to ensure the checksum is the same.

Field Key: *Not applicable*

Data value (hexadecimal): 59 89

Data values in other bases:

Hexadecimal	5	9	8	9
Binary	0101	1001	1000	1001
Decimal	89		137	
ASCII	Y		↑	

Programming Hint: The name for this variable in code will be IP_TCP_Checksum_TELNET.

IP > TCP PDU > *No Operation* for the selected TELNET PDU

Field Name: *No Operation*

Purpose and Definition:
This option may be used between options, for example, to align the beginning of a subsequent option on a 32 bit boundary. May be copied, introduced, or deleted on fragmentation, or for any other reason. This option may be created by one or many octet fields.

Field Key: *Not applicable*

Data value (hexadecimal): 01 01

Data values in other bases:

Hexadecimal	0	1	0	1
Binary	0000	0001	0000	0001
Decimal	©		©	

Programming Hint: The name for this variable in code will be IP_TCP_NoOperation_TELNET.

IP > TCP PDU > *Timestamp* for the selected TELNET PDU

Field Name: *Timestamp*

Purpose and Definition:
Unlike IP, TCP is responsible for ensuring that the entire IP packet arrived intact. TCP will run a CRC on the entire IP packet (not just the header) and place the resulting checksum in this field. When the IP packet is received, TCP re-runs the CRC on the entire packet to ensure the checksum is the same.

Field Key:
This key indicates assigned flag options:

Dec	Port Numbers
0	time stamps only, stored in consecutive 32-bit words,
1	each timestamp is preceded with internet address of the registering entity,
3	the internet address fields are pre-specified. An IP module only registers its timestamp if it matches its own address with the next specified internet address.

Data value (hexadecimal): 08 0A 0B D1 8D EC 1A AC 06 AB

Data values in other bases:

Hexadecimal	0	8	0	A	0	B	D	1	8	D
Binary	0000	1000	0000	1010	0000	1011	1101	0001	1000	1101
Decimal	8		10		11		209		141	
ASCII	©		©		©		↑		↑	

Hexadecimal	E	C	1	A	A	C	0	6	A	B
Binary	1110	1100	0001	1010	1010	1100	0000	0110	1010	1011
Decimal	236		26		172		6		171	
ASCII	↑		©		↑		©		↑	

Programming Hint: The name for this variable in code will be IP_TCP_Timestamp_TELNET.

IP > TCP PDU > *Options and Padding* for the selected TELNET PDU

Field Name: *Options and Padding*

Purpose and Definition:
Like IP options, this field is optional and represents additional instructions not covered in the other TCP fields. Again, if an option does not fill up a 32-bit word, it will be filled in with padding bits.

Field Key: *Not applicable*

Data value (hexadecimal): 01 01 08 0A 0B D1 8D EC 1A AC 06 AB

Data values in other bases:

Hexadecimal	0	1	0	1	0	8	0	A	0	B
Binary	0000	0001	0000	0001	0000	1000	0000	1010	0000	1011
Decimal	1		1		8		10		11	
ASCII	©		©		©		©		©	

Hexadecimal	D	1	8	D	E	C	1	A	A	C
Binary	1101	0001	1000	1101	1110	1100	0001	1010	1010	1100
Decimal	209		141		236		26		172	
ASCII	↑		↑		↑		©		↑	

Hexadecimal	0	6	A	B
Binary	0000	0110	1010	1011
Decimal	6		171	
ASCII	©		↑	

Programming Hint: The name for this variable in code will be IP_TCP_OptionsPadding_TELNET.

2.2.16 TELNET PDU for the TELNET PDU

IP >TCP > TELNET PDU for the TELNET Packet

RFC Link: http://www.ietf.org/rfc/rfc0959.txt?number=959

PASS (Password)

The argument field is a Telnet string specifying the user's password. This command must be immediately preceded by the user name command, and, for some sites, completes the user's identification for access control.

What is Contained in the Packet
Request: PASS

Data Values (hexadecimal): 50 61 73 73 77 6F 72 64 3A 20

Data Values in Other Bases

Hexadecimal	5	0	6	1	7	3	7	7	6	F
Binary	0101	0000	0110	0001	0111	0011	0111	0111	0110	1111
Decimal	80		97		115		119		111	
ASCII	P		a		s		w		o	

Hexadecimal	7	2	6	4	3	A	2	0
Binary	0111	0010	0110	0100	0011	1010	0010	0000
Decimal	114		100		58		32	
ASCII	r		d		:		©	

Programming Hint: The name for this variable in code will be IP_TCP_TELNET_PDU_TELNET.

2.2.17 IP PDU for the selected SSH PDU

IP PDU > *IP Version* for the selected SSH PDU

Field Name: *IP Version*

Purpose and Definition: Version is a 4-bit field that indicates the format of the internet header.

Field Key: 4 = IPv4
6 = IPv6

Data value (decimal): 4

Data values in other bases:

Hexadecimal	4
Binary	0100
Decimal	4

Programming Hint: The name for this variable in code will be IP_Version_SSH.

IP PDU > *Internet Header Length* for the selected SSH PDU

Field Name: *Internet Header Length*

Purpose and Definition: The IHL field is a 4-bit field indicating the length of the internet header in 32 bit words, and thus points to the beginning of the data. The minimum value of a correct header is 5.

Field Key: *Not applicable*

Data value: The value contained in our field is 20 bytes. This is the hexadecimal and decimal value of 5 multiplied by 4 bits.

Data values in other bases:

Hexadecimal	5
Binary	0101
Decimal	5

Programming Hint: The name for this variable in code will be IP_IHL_SSH.

IP PDU > *Type of Service* for the selected SSH PDU

Field Name: *Type of Service*

Purpose and Definition: Type of Service is an 8-bit field that provides and indication of the abstract parameters of the quality of service desired. These parameters guide the selection of the actual service parameters when transmitting a datagram through a particular network.

Field Key: The major choice is a three-way tradeoff between low-delay, high-reliability, and high-throughput.

0 1 2	3	4	5	6	7
Precedence	D	T	R	0	0

Bits 0-2: Precedence
Bit 3: (D) 0 = Normal Delay 1 = Low Delay
Bit 4: (T) 0 = Normal Throughput 1 = High Throughput
Bit 5: (R) 0 = Normal Reliability 1 = High Reliability

Precedence:

111 = Network Control	011 = Flash
110 = Internetwork Control	010 = Immediate
101 = CRITIC/ECP	001 = Priority
100 = Flash Overrided	000 = Routine

Data value (hexadecimal): 00

Data values in other bases:

Hexadecimal	0	0
Binary	0000	0000
Decimal	0	

Programming Hint: The name for this variable in code will be IP_TypeOfService_SSH.

IP PDU > *Total Length of Ethernet Frame* for the selected SSH PDU

Field Name: *Total Length of Ethernet Frame*

Purpose and Definition: Total Length is a 16-bit field that indicates the length of the frame, measured in octets, including internet header and data. The maximum size is 2^{16}-1 or 65,535 octets; however, the recommended maximum size is 576 octets.

Field Key: *Not applicable*

Data values (hexadecimal): 00 64

Data values in other bases:

Hexadecimal	0	0	6	4
Binary	0000	0000	0110	0100
Decimal	0		100	
ASCII	.		d	

Programming Hint: The name for this variable in code will be IP_TotalLength_SSH.

IP PDU > *Identification* for the selected SSH PDU

Field Name: *Identification*

Purpose and Definition: Identification is a 16-bit field. An identifying value is assigned by the sender to aid in assembling the fragments of a datagram. The identifier is chosen based on the need to provide a way to uniquely identify the fragments and protocol for the time the datagram or any fragment could be alive in the internet.

Field Key: *Not applicable*

Data value (hexadecimal): 30 CA

Data values in other bases:

Hexadecimal	3	0	C	A
Binary	0011	0000	1100	1010

Programming Hint: The name for this variable in code will be IP_Idenfification_SSH.

IP PDU > *Flags* for the selected SSH PDU

Field Name: *Flags*

Purpose and Definition: Flags is a 3-bit field that indicates directions for fragmentation.

Field Key:
Bit 0: reserved, must be 0
Bit 1: (DF) 0 = May Fragment 1 = Don't Fragment
Bit 2: (MF) 0 = Last Fragment 1 = More Fragment

Data value (binary): 001

Data values in other bases: *Not applicable*

Programming Hint: The name for this variable in code will be IP_Flags_SSH.

IP PDU > *Fragment Offset* for the selected SSH PDU

Field Name: *Fragment Offset*

Purpose and Definition: The Fragment Offset is a 13- bit field indicating where in the Ethernet frame this fragment begins. The Fragment Offset is measured in units of 8 octets, and the first fragment has offset 0.

Field Key: *Not applicable*

Data value (decimal): 0

Data values in other bases:
Binary: 0 0000 0000 0000

Programming Hint: The name for this variable in code will be IP_FragmentOffset_SSH.

IP PDU > *Time to Live* for the selected SSH PDU

Field Name: *Time to Live*

Purpose and Definition: Time to Live is an 8-bit field that indicates the maximum time the datagram is allowed to remain in the internet. If this field contains the value 0, then the datagram must be destroyed. This field is modified in internet header processing. The time is measure in units of seconds, and is set by the sender to the maximum time the datagram is allowed to be in the internet. This field is decreased at each point that the internet header is processed. The intention is to cause undeliverable packets to be discarded, and to bind the maximum datagram lifetime.

Field Key: *Not applicable*

Data value (decimal): 64

Data values in other bases:

Hexadecimal	4	0
Binary	0100	0000
Decimal	64	

Programming Hint: The name for this variable in code will be IP_TimeToLive_SSH.

IP PDU > *Protocol* for the selected SSH PDU

Field Name: *Protocol*

Purpose and Definition: Protocol is an 8-bit field that indicates the next level protocol that is used in the data portion of the internet diagram.

Field Key:

Dec	Hex	Protocol	Dec	Hex	Protocol
0	00	Reserved	22	16	Multiplexing
1	01	ICMP	23	17	DCN
2	02	Unassigned	24	18	TAC Monitoring
3	03	Gateway-to-Gateway	25-76	19-4C	Unassigned
4	04	CMCC Gateway Monitoring Message	77	4D	Any local network
5	05	ST	100	64	SATNET and Backroom EXPAK
6	06	TCP	101	65	MIT Subnet Support
7	07	UCL	102-104	66-68	Unassigned
10	0A	Unassigned	105	69	SATNET Monitoring
11	0B	Secure	106	6A	Unassigned
12	0C	BBN RCC Monitoring	107	6B	Internet Packet Core Utility
13	0D	NVP	110-113	6E-71	Unassigned
14	0E	PUP	114	72	Backroom SATNET Monitoring
15	0F	Pluribus	115	73	Unassigned
16	10	SSH	116	74	WIDEBAND Monitoring
17	11	XNET	117	75	WIDEBAND EXPAK
20	14	Chaos	120-376	78-0178	Unassigned
21	15	User Datagram	377	0179	Reserved

Data value (hexadecimal): 11

Data values in other bases:

Hexadecimal	0	6
Binary	0000	0110
Decimal	6	

RFC Link: http://www.faqs.org/rfcs/rfc790.html

Programming Hint: The name for this variable in code will be IP_Protocol_SSH

IP PDU > *Header Checksum* for the selected SSH PDU

Field Name: *Header Checksum*

Purpose and Definition: The Header Checksum is a 16-bit field. The Checksum is the 16-bit one's complement sum of all 16-bit words in the header. For purposes of computing the checksum, the initial value of its field is zero. When both header checksums are equal, then the header bits are correct. If either checksums vary, then a new, correct packet will need to be sent.

This is a simple way to compute the checksum and experimental evidence indicates that it is adequate, but it is provisional and may be replaced by a CRC procedure, depending on further experience.

Field Key: *Not applicable*

Data value (hexadecimal): 87 AE

Data values in other bases:

Hexadecimal	8	7	A	E
Binary	1000	0111	1010	1110

Programming Hint: The name for this variable in code will be IP_HeaderChecksum_SSH.

IP PDU > *Source Address* for the selected SSH PDU

Field Name: *Source Address*

Purpose and Definition: The Source Address is a 32-bit field that contains the IP address of the host that sent the IP Packet.

Field Key: *Not applicable*

Data value (decimal): 192.168.0.101

Data values in other bases:

Hexadecimal	C	0	A	8	0	0	6	5
Binary	1100	0000	1010	1000	0000	0000	0110	0101
Decimal	192		168		0		101	

Programming Hint: The name for this variable in code will be IP_SourceAddress_SSH.

IP PDU > *Destination Address* for the selected SSH PDU

Field Name: *Destination Address*

Purpose and Definition: The Destination Address is a 32-bit field that contains the address of the host that is to receive the data contained within the IP packet.

Field Key: *Not applicable*

Data value (decimal): 192.168.0.39

Data values in other bases:

Hexadecimal	C	0	A	8	0	0	2	7
Binary	1100	0000	1010	1000	0000	0000	0010	0111
Decimal	192		168		0		39	

Programming Hint: The name for this variable in code will be IP_DestinationAddress_SSH.

2.2.18 TCP PDU for the selected SSH PDU

IP > TCP PDU > *Source Port* for the selected SSH PDU

Field Name: *Source Port*

Purpose and Definition:
This 16-bit number represents the name of the application that sent the data in the IP packet.

Field Key: *Not applicable*

Data value: 1243

Data values in other bases:

Hexadecimal	0	4	D	B
Binary	0000	0100	1101	1011
Decimal	4		219	
ASCII	©		↑	

Programming Hint: The name for this variable in code will be IP_TCP_SourcePort_SSH.

IP > TCP PDU > *Destination Port* for the selected SSH PDU

Field Name: *Destination Port*

Purpose and Definition:
This 16-bit number represents the name of the application that is to receive the data contained within the IP packet. This is one of the major differences between a Layer 3 and a Layer 4 header: the Layer 3 header contains the IP address of the computer that is to receive the IP packet; once that packet has been received, the port address in the Layer 4 header ensures that the data contained within that IP packet is passed to the correct application on that computer.

Field Key:
This key indicates assigned port number values:

Dec	**Port Numbers**
0	Reserved
1-32767	Internet registered ("well-known") protocols
32768-98303	Reserved, to allow TCPv7-TCPv4 conversion
98304 & up	Dynamic assignment

Data value (decimal): 1243

Data values in other bases:

Hexadecimal	0	4	D	B
Binary	0000	0100	1101	1011
Decimal	4		219	
ASCII	©		↑	

Source: http://www.zvon.org/tmRFC/RFC1475/Output/chapter4.html

Programming Hint: The name for this variable in code will be IP_TCP_DestinationPort_SSH.

IP > TCP PDU > *Sequence Number* for the selected SSH PDU

Field Name: *Sequence Number*

Purpose and Definition:
TCP is responsible for ensuring that all IP packets sent are actually received. When an application's data is packaged into IP packets, TCP will give each IP packet a sequence number. Once all the packets have arrived at the receiving computer, TCP uses the number in this 32-bit field to ensure that all of the packets actually arrived and are in the correct sequence.

Field Key: *Not applicable*

Data value (decimal): 4008673261

Data values in other bases:

Hexadecimal	E	E	E	F	7	F	E	D
Binary	1110	1110	1110	1111	0111	1111	1110	1101
Decimal	238		239		127		237	
ASCII	↑		↑		□		↑	

Programming Hint: The name for this variable in code will be IP_TCP_SequenceNumber_SSH.

IP > TCP PDU > *Acknowledgement Number* for the selected SSH PDU

Field Name: *Acknowledgement Number*

Purpose and Definition:
This number is used by the receiving computer to acknowledge which packets have successfully arrived. This number will be the sequence number of the next packet the receiver is ready to receive.

Field Key: *Not applicable*

Data value: 3798775616

Data values in other bases:

Hexadecimal	E	2	6	C	B	7	4	0
Binary	1110	0010	0110	1100	1011	0111	0100	0000
Decimal	226		108		183		64	
ASCII	↑		l		↑		@	

Programming Hint: The name for this variable in code will be IP_TCP_AcknowledgementNumber_SSH.

IP > TCP PDU > *Header Length or Offset* for the selected SSH PDU

Field Name: *Header Length or Offset*

Purpose and Definition:
This is identical in concept to the header length in an IP packet, except this time it indicates the length of the TCP header.

Field Key: *Not applicable*

Data value (bytes): 32

Data values in other bases:

Hexadecimal	8	0
Binary	1000	0000
Decimal	128	
ASCII	↑	

Programming Hint: The name for this variable in code will be IP_TCP_HeaderLength_SSH.

IP > TCP PDU > *Control Flags* for the selected SSH PDU

Field Name: *Control Flags*

Purpose and Definition:
Every TCP packet contains this 6-bit value that indicates how many octets it can receive at once. When IP packets are received, they are placed in a temporary area of RAM known as a buffer until the receiving computer has a chance to process them; this value represents how big a buffer the receiving host has made available for this temporary storage of IP packets.

Field Key:

- Urgent (URG)
- Acknowledgement (ACK)
- Push (PSH)
- Reset (RST)
- Synchronize (SYN)
- Finish (FIN)

Data value (binary): 0001 1000

Data values in other bases:

Hexadecimal	1	8
Binary	0001	1000
Decimal	24	
ASCII	©	

Programming Hint: The name for this variable in code will be IP_TCP_ControlFlags_SSH.

IP > TCP PDU > *Window Size* for the selected SSH PDU

Field Name: *Window Size*

Purpose and Definition:
Every TCP packet contains this 16-bit value that indicates how many octets it can receive at once. When IP packets are received, they are placed in a temporary area of RAM known as a buffer until the receiving computer has a chance to process them; this value represents how big a buffer the receiving host has made available for this temporary storage of IP packets.

Field Key: *Not applicable*

Data value (decimal): 32120

Data values in other bases:

Hexadecimal	7	D	7	8
Binary	0111	1101	0111	1000
Decimal	125		120	
ASCII	}		x	

Programming Hint: The name for this variable in code will be IP_TCP_WindowSize_SSH.

IP > TCP PDU > *Checksum* for the selected SSH PDU

Field Name: *Checksum*

Purpose and Definition:
Unlike IP, TCP is responsible for ensuring that the entire IP packet arrived intact. TCP will run a CRC on the entire IP packet (not just the header) and place the resulting checksum in this field. When the IP packet is received, TCP re-runs the CRC on the entire packet to ensure the checksum is the same.

Field Key: *Not applicable*

Data value (hexadecimal): 8B CA

Data values in other bases:

Hexadecimal	8	B	C	A
Binary	1000	1011	1100	1010
Decimal	139		202	
ASCII	↑		↑	

Programming Hint: The name for this variable in code will be IP_TCP_Checksum_SSH.

IP > TCP PDU > *Options and Padding* for the selected SSH PDU

Field Name: *Options and Padding*

Purpose and Definition:
Like IP options, this field is optional and represents additional instructions not covered in the other TCP fields. Again, if an option does not fill up a 32-bit word, it will be filled in with padding bits.

Field Key: *Not applicable*

Data value (hexadecimal): 01 01 08 0A 0B D1 8D EC 1A AC 06 AB

Data values in other bases:

Hexadecimal	0	1	0	1
Binary	0000	0001	0000	0001
Decimal	©		©	

Programming Hint: The name for this variable in code will be IP_TCP_OptionsPadding_SSH.

IP > TCP PDU > *Timestamp* for the selected SSH PDU

Field Name: *Timestamp*

Purpose and Definition:
Unlike IP, TCP is responsible for ensuring that the entire IP packet arrived intact. TCP will run a CRC on the entire IP packet (not just the header) and place the resulting checksum in this field. When the IP packet is received, TCP re-runs the CRC on the entire packet to ensure the checksum is the same.

Field Key:
This key indicates assigned flag options:

Dec	Port Numbers
0	time stamps only, stored in consecutive 32-bit words,
1	each timestamp is preceded with internet address of the registering entity,
3	the internet address fields are pre-specified. An IP module only registers its timestamp if it matches its own address with the next specified internet address.

Data value (hexadecimal): 08 0A 14 42 6F 30 27 47 32 1F

Data values in other bases:

Hexadecimal	0	8	0	A	1	4	4	2	6	F
Binary	0000	1000	0000	1010	0001	0100	0100	0010	0110	1111
Decimal	8		10		20		66		111	
ASCII	©		©		©		B		o	

Hexadecimal	3	0	2	7	4	7	3	2	1	F
Binary	0011	0000	0010	0111	0100	0111	0011	0010	0001	1111
Decimal	48		39		71		50		31	
ASCII	0		‘		G		2		©	

Programming Hint: The name for this variable in code will be IP_TCP_Timestamp_SSH.

2.2.18 SSH PDU for the selected SSH PDU

IP >TCP > SSH PDU for the SSH Packet

RFC Link: http://www.ietf.org/rfc/rfc0959.txt?number=959

PASS (Password)

The argument field is a SSH string specifying the user's password. This command must be immediately preceded by the user name command, and, for some sites, completes the user's identification for access control. All information below is encrypted.

What is Contained in the Packet
Request: PASS

Data Values (hexadecimal): 03 B6 51 11 6A 46 12 36 4F 46 C9 63 B1 A4 B5 48 A2 BA 68 1C 42 17 AB D2 CE 8E 6D 3F 49 7E EB 36 A0 1B 16 62 E4 0F D7 55 DD 5F EB 52 64 B9 A7 62

Data Values in Other Bases

Hexadecimal	0	3	B	6	5	1	1	1	6	A
Binary	0000	0011	1011	0110	0101	0001	0001	0001	0110	1010
Decimal	3		182		81		17		106	
ASCII	©		↑		Q		©		j	

Hexadecimal	4	6	1	2	3	6	4	F	4	6
Binary	0100	0110	0001	0010	0011	0110	0100	1111	0100	0110
Decimal	70		18		54		79		70	
ASCII	F		©		6		O		F	

Hexadecimal	C	9	6	3	B	1	A	4	B	5
Binary	1100	1001	0110	0011	1011	0001	1010	0100	1011	0101
Decimal	201		99		177		164		181	
ASCII	↑		c		↑		↑		↑	

Hexadecimal	4	8	A	2	B	A	6	8	1	C
Binary	0100	1000	1010	0010	1011	1010	0110	1000	0001	1000
Decimal	72		162		178		104		28	
ASCII	H		↑		↑		H		©	

Hexadecimal	4	2	1	7	A	B	D	2	C	E
Binary	0100	0010	0001	0111	1010	1011	1101	0010	1100	1110
Decimal	66		23		171		210		206	
ASCII	B		©		↑		↑		↑	

Hexadecimal	8	E	6	D	3	F	4	9	7	E
Binary	1000	1110	0110	1101	0011	1111	0100	1001	0111	1110
Decimal	142		109		63		73		126	
ASCII	↑		m		?		I		~	

Hexadecimal	E	B	3	6	A	0	1	B	1	6
Binary	1110	1011	0011	0110	1010	0000	0001	1011	0001	0110
Decimal	235		54		160		27		22	
ASCII	↑		6		↑		©		©	

Hexadecimal	6	2	E	4	0	F	D	7	5	5
Binary	0110	0010	1110	0100	0000	1111	1101	0111	0101	0101
Decimal	98		228		15		215		85	
ASCII	b		↑		©		↑		U	

Hexadecimal	D	D	5	F	E	B	5	2	6	4
Binary	1101	1101	0101	1111	1110	1011	0101	0010	0110	0100
Decimal	221		95		235		82		100	
ASCII	↑		_		↑		R		d	

Hexadecimal	B	9	A	7	6	2
Binary	1011	1001	1010	0111	0110	0010
Decimal	185		167		98	
ASCII	↑		↑		b	

Programming Hint: The name for this variable in code will be IP_TCP_SSH_PDU_SSH.

2.2.20 ARP PDU for the selected ARP PDU

ARP PDU> *Hardware Address Type* > for the selected ARP PDU

Field Name: *Hardware Address Type*

Purpose and Definition: The physical media that communicates on the network.

Field Key: 1 for Ethernet
2 for IEEE 802 LAN

Data value (hexadecimal): 00 01

Data values in other bases:

Hexadecimal	0	0	0	1
Binary	0000	0000	0000	0001
Decimal	0		1	

Programming Hint: The name for this variable in code will be ARP_HardwareAddressType

ARP PDU > *Protocol Address Type* > for the selected ARP PDU

Field Name: *Protocol Address Type*

Purpose and Definition: Defines the protocol that the terminals are using to connect with each other.

Field Key: 2048 IPv4 (0x0800)

Data value (hexadecimal): 08 00

Data values in other bases:

Hexadecimal	0	8	0	0
Binary	0000	1000	0000	0000
Decimal	8		0	

Programming Hint: The name for this variable in code will be ARP_ProtocolAddressType.

ARP PDU > *Hardware Address Length* > for the selected ARP PDU

Field Name: *Hardware Address Length*

Purpose and Definition: This field determines the type of hardware used.

Field Key: 6 Ethernet / IEE 802

Data value (hexadecimal): 00 06

Data values in other bases:

Hexadecimal	0	0	0	6
Binary	0000	1000	0000	0110
Decimal	0		6	

Programming Hint: The name for this variable in code will be ARP_HardwareAddressLength.

ARP PDU >*Protocol Address Length*>for the selected ARP PDU

Field Name: *Protocol Address Length*

Purpose and Definition: Determines the protocol used in the request or response**.** .

Field Key: 4 for IPv4

Data value (hexadecimal): 02 01 00

Data values in other bases:

Hexadecimal	0	2	0	1	0	0
Binary	0000	0010	0000	0001	0000	0000
Decimal	2		1		0	

Programming Hint: The name for this variable in code will be ARP_ProtocolAddressLength.

ARP PDU > *Operation* > for the selected ARP PDU

Field Name: *Operation*

Purpose and Definition: Determines weather a request or a response is being called upon.

Field Key: 1 for request
2 for reply

Data value (hexadecimal): 00 01

Data values in other bases:

Hexadecimal	0	0	0	1
Binary	0000	0000	0000	0001
Decimal	0		1	

Programming Hint: The name for this variable in code will be ARP_Operation

ARP PDU > *Sender Hardware Address* > for the selected ARP PDU

Field Name: *Sender Hardware Address*

Purpose and Definition: The Physical address or MAC address of the network adapter of the sender's terminal.

Field Key:

00000C Cisco
00000E Fujitsu
00000F NeXT
000010 Sytek
00001D Cabletron
000020 DIAB (Data Intdustrier AB)
000022 Visual Technology
00002A TRW
000032 GPT Limited (reassigned from GEC Computers Ltd)
00005A S & Koch
00005E IANA
000065 Network General
00006B MIPS
000077 MIPS
00007A Ardent
000089 Cayman Systems Gatorbox
000093 Proteon
00009F Ameristar Technology
0000A2 Wellfleet
0000A3 Network Application Technology
0000A6 Network General (internal assignment, not for products)
0000A7 NCD X-terminals
0000A9 Network Systems
0000AA Xerox Xerox machines
0000B3 CIMLinc
0000B7 Dove Fastnet
0000BC Allen-Bradley
0000C0 Western Digital
0000C5 Farallon phone net card
0000C6 HP Intelligent Networks Operation (formerly Eon Systems)
0000C8 Altos
0000C9 Emulex Terminal Servers
0000D7 Dartmouth College (NED Router)
0000D8 3Com? Novell? PS/2
0000DD Gould
0000DE Unigraph
0000E2 Acer Counterpoint
0000EF Alantec
0000FD High Level Hardvare (Orion, UK)
000102 BBN BBN internal usage (not registered)
0020AF 3COM ???
001700 Kabel
008064 Wyse Technology / Link Technologies
00802B IMAC ???
00802D Xylogics, Inc. Annex terminal servers

00808C Frontier Software Development
0080C2 IEEE 802.1 Committee
0080D3 Shiva
00AA00 Intel
00DD00 Ungermann-Bass
00DD01 Ungermann-Bass
020701 Racal InterLan
020406 BBN BBN internal usage (not registered)
026086 Satelcom MegaPac (UK)
02608C 3Com IBM PC; Imagen; Valid; Cisco
02CF1F CMC Masscomp; Silicon Graphics; Prime EXL
080002 3Com (Formerly Bridge)
080003 ACC (Advanced Computer Communications)
080005 Symbolics Symbolics LISP machines
080008 BBN
080009 Hewlett-Packard
08000A Nestar Systems
08000B Unisys
080011 Tektronix, Inc.
080014 Excelan BBN Butterfly, Masscomp, Silicon Graphics
080017 NSC
08001A Data General
08001B Data General
08001E Apollo
080020 Sun Sun machines
080022 NBI
080025 CDC
080026 Norsk Data (Nord)
080027 PCS Computer Systems GmbH
080028 TI Explorer
08002B DEC
08002E Metaphor
08002F Prime Computer Prime 50-Series LHC300
080036 Intergraph CAE stations
080037 Fujitsu-Xerox
080038 Bull
080039 Spider Systems
080041 DCA Digital Comm. Assoc.
080045 ???? (maybe Xylogics, but they claim not to know this number)
080046 Sony
080047 Sequent
080049 Univation
08004C Encore
08004E BICC
080056 Stanford University
080058 ??? DECsystem-20
08005A IBM
080067 Comdesign
080068 Ridge
080069 Silicon Graphics
08006E Concurrent Masscomp
080075 DDE (Danish Data Elektronik A/S)
08007C Vitalink TransLAN III
080080 XIOS
080086 Imagen/QMS
080087 Xyplex terminal servers

080089 Kinetics AppleTalk-Ethernet interface
08008B Pyramid
08008D XyVision XyVision machines
080090 Retix Inc Bridges
484453 HDS ???
800010 AT&T
AA0000 DEC obsolete
AA0001 DEC obsolete
AA0002 DEC obsolete
AA0003 DEC Global physical address for some DEC machines
AA0004 DEC Local logical address for systems running DECNET

Data value (hexadecimal): 00:00:E6: 34:ED:A3

Data values in other bases:

Hexadecimal	0	0	0	0	E	6	3	4
Binary	0000	0000	0000	0000	1110	0110	0011	0100
Decimal	0		0		230		52	

Hexadecimal	E	D	A	3
Binary	1110	1101	1010	0011
Decimal	237		163	

Programming Hint: The name for this variable in code will be ARP_SenderAddress

ARP PDU > *Sender Protocol Address* > for the selected ARP PDU

Field Name: *Sender Protocol Address*

Purpose and Definition: The protocol of the sender computer. This is used to identify the senders Protocol.

Field Key: N/A

Data value (decimal): 192.168.0.101

Hexadecimal	C	0	A	8	0	0	6	5
Binary	1100	0000	1010	1000	0000	0000	0110	0101
Decimal	192		168		0		101	

Programming Hint: The name for this variable in code will be ARP_SenderProtocolAddress

ARP PDU > *Target Hardware Address* > for the selected ARP PDU

Field Name: *Target Hardware Address*

Purpose and Definition: The Physical address or MAC address of the network adapter of the target terminal.

Field Key:

00000C Cisco
00000E Fujitsu
00000F NeXT
000010 Sytek
00001D Cabletron
000020 DIAB (Data Intdustrier AB)
000022 Visual Technology
00002A TRW
000032 GPT Limited (reassigned from GEC Computers Ltd)
00005A S & Koch
00005E IANA
000065 Network General
00006B MIPS
000077 MIPS
00007A Ardent
000089 Cayman Systems Gatorbox
000093 Proteon
00009F Ameristar Technology
0000A2 Wellfleet
0000A3 Network Application Technology
0000A6 Network General (internal assignment, not for products)
0000A7 NCD X-terminals
0000A9 Network Systems
0000AA Xerox Xerox machines
0000B3 CIMLinc

0000B7 Dove Fastnet
0000BC Allen-Bradley
0000C0 Western Digital
0000C5 Farallon phone net card
0000C6 HP Intelligent Networks Operation (formerly Eon Systems)
0000C8 Altos
0000C9 Emulex Terminal Servers
0000D7 Dartmouth College (NED Router)
0000D8 3Com? Novell? PS/2
0000DD Gould
0000DE Unigraph
0000E2 Acer Counterpoint
0000EF Alantec
0000FD High Level Hardvare (Orion, UK)
000102 BBN BBN internal usage (not registered)
0020AF 3COM ???
001700 Kabel
008064 Wyse Technology / Link Technologies
00802B IMAC ???
00802D Xylogics, Inc. Annex terminal servers
00808C Frontier Software Development
0080C2 IEEE 802.1 Committee
0080D3 Shiva
00AA00 Intel
00DD00 Ungermann-Bass
00DD01 Ungermann-Bass
020701 Racal InterLan
020406 BBN BBN internal usage (not registered)
026086 Satelcom MegaPac (UK)
02608C 3Com IBM PC; Imagen; Valid; Cisco
02CF1F CMC Masscomp; Silicon Graphics; Prime EXL
080002 3Com (Formerly Bridge)
080003 ACC (Advanced Computer Communications)
080005 Symbolics Symbolics LISP machines
080008 BBN
080009 Hewlett-Packard
08000A Nestar Systems
08000B Unisys
080011 Tektronix, Inc.
080014 Excelan BBN Butterfly, Masscomp, Silicon Graphics
080017 NSC
08001A Data General
08001B Data General
08001E Apollo
080020 Sun Sun machines
080022 NBI
080025 CDC
080026 Norsk Data (Nord)
080027 PCS Computer Systems GmbH
080028 TI Explorer
08002B DEC

08002E Metaphor
08002F Prime Computer Prime 50-Series LHC300
080036 Intergraph CAE stations
080037 Fujitsu-Xerox

080038 Bull
080039 Spider Systems
080041 DCA Digital Comm. Assoc.
080045 ???? (maybe Xylogics, but they claim not to know this number)
080046 Sony
080047 Sequent
080049 Univation
08004C Encore
08004E BICC
080056 Stanford University
080058 ??? DECsystem-20
08005A IBM
080067 Comdesign
080068 Ridge
080069 Silicon Graphics
08006E Concurrent Masscomp
080075 DDE (Danish Data Elektronik A/S)
08007C Vitalink TransLAN III
080080 XIOS
080086 Imagen/QMS
080087 Xyplex terminal servers
080089 Kinetics AppleTalk-Ethernet interface
08008B Pyramid
08008D XyVision XyVision machines
080090 Retix Inc Bridges
484453 HDS ???
800010 AT&T
AA0000 DEC obsolete
AA0001 DEC obsolete
AA0002 DEC obsolete
AA0003 DEC Global physical address for some DEC machines
AA0004 DEC Local logical address for systems running DECNET

Data value (hexadecimal): 00:00:00:00:00:00

Data values in other bases: *Not applicable*

Programming hint: ARP_TargetHardwareAddress.

ARP PDU>*Target Protocol Address*> for the selected ARP PDU

Field Name: *Target Protocol Address*

Purpose and Definition: The protocol of the sender computer. This is used to identify the targets Protocol.

Field Key: 4 for IPv4

Data value (decimal): 192.168.0.145

Hexadecimal	C	0	A	8	0	0	9	1
Binary	1100	0000	1010	1000	0000	0000	1001	0001
Decimal	192		168		0		145	

Programming hint: ARP_TargetProtocolAddress

2.2.21 IP PDU for the selected PING PDU

IP PDU > *Differentiated Services Field* for the selected PING

RFC Link: http://www.ietf.org/rfc/rfc0768.txt?number=768

Field Name: *Differentiated Services Field*

Purpose and Definition: Type of Service is an 8-bit field that provides and indication of the abstract parameters of the quality of service desired. These parameters guide the selection of the actual service parameters when transmitting a datagram through a particular network.

Field Key: The major choice is a three-way tradeoff between low-delay, high-reliability, and high-throughput.

0 1 2	3	4	5	6	7
Precedence	D	T	R	0	0

Bits 0-2: Precedence
Bit 3: (D) 0 = Normal Delay 1 = Low Delay
Bit 4: (T) 0 = Normal Throughput 1 = High Throughput
Bit 5: (R) 0 = Normal Reliability 1 = High Reliability

Precedence:

111 = Network Control	011 = Flash
110 = Internetwork Control	010 = Immediate
101 = CRITIC/ECP	001 = Priority
100 = Flash Overrided	000 = Routine

Data value (decimal): 00

Data values in other bases:

Hexadecimal	0	0
Binary	0000	0000
Decimal	0	

Programming Hint: The name for this variable in code will be IP_DSF_PING.

IP PDU> *Flags* for the selected PING

Field Name: ***Flags***

Purpose and Definition: Flags is a 3-bit field that indicates directions for fragmentation.

Field Key: Bit 0: reserved, must be 0
Bit 1: (DF) 0 = May Fragment 1 = Don't Fragment
Bit 2: (MF) 0 = Last Fragment 1 = More Fragment

Data value (hexadecimal): 04

Data values in other bases:

Hexadecimal	0	4
Binary	0000	0100
Decimal	4	

Programming Hint: The name for this variable in code will be IP_Flags_PING.

IP PDU> *Fragment offset* for the selected PING

Field Name: *Fragment offset*

Purpose and Definition: The Fragment Offset is a 13- bit field indicating where in the Ethernet frame this fragment begins. The Fragment Offset is measured in units of 8 octets, and the first fragment has offset 0.

Field Key: *Not applicable*

Data value (decimal): 0

Data values in other bases:
Binary: 0000

Programming Hint: The name for this variable in code will be IP_Fragment Offset_PING.

IP PDU > *Time to Live* for the selected PING

Field Name: *Time to Live*

Purpose and Definition: Time to Live is an 8-bit field that indicates the maximum time the datagram is allowed to remain in the internet. If this field contains the value 0, then the datagram must be destroyed. This field is modified in internet header processing. The time is measure in units of seconds, and is set by the sender to the maximum time the datagram is allowed to be in the internet. This field is decreased at each point that the internet header is processed. The intention is to cause undeliverable packets to be discarded, and to bind the maximum datagram lifetime.

Field Key: *Not applicable*

Data value (decimal): 40

Data values in other bases:

Hexadecimal	2	8
Binary	0010	1000
Decimal	40	

Programming Hint: The name for this variable in code will be IP_TimetoLive_PING.

IP PDU> *Protocol* for the selected PING

Field Name: *Protocol*

Purpose and Definition: Protocol is an 8-bit field that indicates the next level protocol that is used in the data portion of the internet diagram.

Field Key:

Dec	Hex	Protocol	Dec	Hex	Protocol
0	00	Reserved	22	16	Multiplexing
1	01	ICMP	23	17	DCN
2	02	Unassigned	24	18	TAC Monitoring
3	03	Gateway-to-Gateway	25-76	19-4C	Unassigned
4	04	CMCC Gateway Monitoring Message	77	4D	Any local network
5	05	ST	100	64	SATNET and Backroom EXPAK
6	06	TCP	101	65	MIT Subnet Support
7	07	UCL	102-104	66-68	Unassigned
10	0A	Unassigned	105	69	SATNET Monitoring
11	0B	Secure	106	6A	Unassigned
12	0C	BBN RCC Monitoring	107	6B	Internet Packet Core Utility
13	0D	NVP	110-113	6E-71	Unassigned
14	0E	PUP	114	72	Backroom SATNET Monitoring
15	0F	Pluribus	115	73	Unassigned
16	10	Telnet	116	74	WIDEBAND Monitoring
17	11	XNET	117	75	WIDEBAND EXPAK
20	14	Chaos	120-376	78-0178	Unassigned
21	15	User Datagram	377	0179	Reserved

Data value (decimal): 01

Data values in other bases:

Hexadecimal	0	1
Binary	0000	0001
Decimal	1	

Programming Hint: The name for this variable in code will be IP_Protocol_PING.

IP PDU> *Header Checksum* for the selected PING

Field Name: *Header Checksum*

Purpose and Definition: The Header Checksum is a 16-bit field. The Checksum is the 16-bit one's complement sum of all 16-bit words in the header. For purposes of computing the checksum, the initial value of its field is zero. When both header checksums are equal, then the header bits are correct. If either checksums vary, then a new, correct packet will need to be sent.

Field Key: *Not applicable*

Data value (decimal): B8 CC

Data values in other bases:

Hexadecimal	B	8	C	C
Binary	1011	1000	1100	1100

Programming Hint: The name for this variable in code will be IP_HeaderChecksum_PING.

IP PDU> *Source* for the selected PING

Field Name: *Source*

Purpose and Definition: The Source Address is a 32-bit field that contains the IP address of the host that sent the IP Packet.

Field Key: *Not applicable*

Data value (decimal): 192.168.0.39

Data values in other bases:

Hexadecimal	C	0	A	8	0	0	2	7
Binary	1100	0000	1010	1000	0000	0000	0010	0111
Decimal	192		168		0		39	

Programming Hint: The name for this variable in code will be IP_Source_PING.

IP PDU > *Destination* for the selected PING

Field Name: *Destination*

Purpose and Definition: The Destination Address is a 32-bit field that contains the address of the host that is to receive the data contained within the IP packet.

Field Key: *Not applicable*

Data value (decimal): 192.168.0.101

Data values in other bases:

Hexadecimal	C	0	A	8	0	0	6	5
Binary	1100	0000	1010	1000	0000	0000	0110	0101
Decimal	192		168		0		101	

Programming Hint: The name for this variable in code will be Ethernet_Destination_PING.

2.2.22 ICMP PDU for the selected PING PDU

IP > ICMP Header > *Type* for the selected PING PDU

Field Name: *Type*

Purpose and Definition: The type is an 8-bit field that identifies what sort of message the ICMP protocol is sending.

Field Key:

Dec	Hex	Message Type	Dec	Hex	Message Type
0	00	Echo Reply	16	10	Information Reply
1	01	Unassigned	17	11	Address Mask Request
2	02	Unassigned	18	12	Address Mask Reply
3	03	Destination Unreachable	19	13	Reserved (for Security)
4	04	Source Quench	20-29	14-1D	Reserved (for Robustness Experiment)
5	05	Redirect	30	1E	Traceroute
6	06	Alternate Host Address	31	1F	Datagram Conversion Error
7	07	Unassigned	32	20	Mobile Host Redirect
8	08	Echo	33	21	IPv6 Where-Are-You
9	09	Router Advertisement	34	22	IPv6 I-Am-Here
10	0A	Router Solicitation	35	23	Mobile Registration Request
11	0B	Time Exceeded	36	24	Mobile Registration Reply
12	0C	Parameter Problem	37	25	Domain Name Request
13	0D	Timestamp	38	26	Domain Name Reply
14	0E	Timestamp Reply	39	27	SKIP
15	0F	Information Request	40	28	Photuris
			41-255	29-FF	Reserved

Data value: 8 (Echo (ping) Request)

Data values in other bases:

Hexadecimal	0	8
Binary	0000	1000
Decimal	8	

RFC Link: http://www.iana.org/assignments/icmp-parameters

Programming Hint: The name for this variable in code will be IP_ICMP_Type_PING.

IP > ICMP Header > *Code* for the selected PING PDU

Field Name: *Code*

Purpose and Definition: Code is an 8-bit field that provides further information about the associated type field.

Field Key:

Type	Name
0	Echo Reply (used by "PING")
	0 No Code
1	Unassigned
2	Unassigned
3	Destination Unreachable
	0 Net Unreachable
	1 Host Unreachable
	2 Protocol Unreachable
	3 Port Unreachable
	5 Fragmentation needed and Don't Fragment was Set
	5 Source Route Failed
	6 Destination Network Unknown
	7 Destination Host Unknown
	8 Source Host Isolated
	13 Communication with Destination Network is Administratively Prohibited
	14 Communication with Destination Host is Administratively Prohibited
	15 Destination Network Unreachable for Type of Service
	16 Destination Host Unreachable for Type of Service
4	Source Quench
	0 No Code
5	Redirect
	0 Redirect Datagram for the Network
	1 Redirect Datagram for the Host
	2 Redirect Datagram for the Type of Service and Network
	4 Redirect Datagram for the Type of Service and Host
6	Alternate Host Address
	0 Alternate Address for Host
7	Unassigned
8	Echo (used by "PING")
	0 No Code
9	Router Advertisement
	0 No Code
10	Router Selection
	0 No Code
11	Time Exceeded
	0 Time to Live exceeded in Transit
	1 Fragment Reassembly Time Exceeded
12	Parameter Problem
	0 Pointer indicates the error
	1 Missing a Required Option
	2 Bad Length
13	Timestamp
	0 No Code
14	Timestamp Reply
	0 No Code
15	Information Request
	0 No Code
16	Information Reply
	0 No Code
17	Address Mask Request
	0 No Code
18	Address Mask Reply
	0 No Code
19	Reserved (for Security)
20-29	Reserved (for Robustness Experiment)
30	Traceroute
31	Datagram Conversion Error
32	Mobile Host Redirect
33	IPv6 Where-Are-You
34	IPv6 I-Am-Here
35	Mobile Registration Request
36	Mobile Registration Reply

Data value (decimal): 0

Data values in other bases:

Hexadecimal	0	0
Binary	0000	0000
Decimal	0	
ASCII	©	

Programming Hint: The name for this variable in code will be IP_ICMP_Header_PING.

IP > ICMP Header > *Checksum* for the selected PING PDU

Field Name: *Checksum*

Purpose and Definition: The checksum is the 16-bit one's complement of the one's complement sum of the ICMP message, starting with the ICMP type. For computing the checksum, the checksum field should initially be zero.

Field Key: *Not applicable*

Data value (hexadecimal): C9 15

Data values in other bases:

Hexadecimal	C	9	1	5
Binary	1100	1001	0001	0101
Decimal	201		21	
ASCII	↑		©	

Programming Hint: The name for this variable in code will be IP_ICMP_Checksum_PING.

IP > ICMP Header > *Identifier* for the selected PING PDU

Field Name: *Identifier*

Purpose and Definition: The identifier is a 16-bit field that is used in matching echoes and replies for when the code field is zero.

Field Key: *Not applicable*

Data value (hexadecimal): 70 60

Data values in other bases:

Hexadecimal	7	0	6	0
Binary	0111	0000	0110	0000
Decimal	112		96	
ASCII	P		‘	

Programming Hint: The name for this variable in code will be IP_ICMP_Identifier_PING.

IP > ICMP Header > *Sequence* for the selected PING PDU

Field Name: *Sequence*

Purpose and Definition: The sequence is a 16-bit field that is used in matching echoes and replies for when the code field is zero.

Field Key: *Not applicable*

Data value (hexadecimal): 70 60

Data values in other bases:

Hexadecimal	0	0	0	0
Binary	0000	0000	0000	0000
Decimal	0		0	
ASCII	©		©	

Programming Hint: The name for this variable in code will be IP_ICMP_Sequence_PING.

IP > ICMP Header > *Data* for the selected PING PDU

Field Name: *Data*

Purpose and Definition: The data is a variable-length field that contains the actual information that is sent in the ping packet.

Field Key: *Not applicable*

Data value (hexadecimal): 00 01 03 1E E2 24 00 00 F8 1F 00 85 08 00 45 00 00 54 00 00 40 00 40 01 B8 CC C0 A8 00 27 C0 A8 00 65 08 00 C9 15 70 60 00 00 42 B1 89 3F 00 00 00 00 2C C6 07 00 00 00 00 00 10 11 12 13 14 15 26 27 28 29 2A 2B 2C 2D 2E 2F 30 31 32 33 34 35 36 37

Data values in other bases:

Hexadecimal	0	0	0	1	0	3	1	E	E	2
Binary	0000	0000	0000	0000	000	0011	0001	1110	1110	0010
Decimal	0		1		3		30		226	
ASCII	©		©		©		©		↑	

Hexadecimal	2	4	0	0	0	0	F	8	1	F
Binary	0010	0100	0000	0000	0000	0000	1111	1000	0001	1111
Decimal	36		0		0		242		31	
ASCII	$		©		©		↑		©	

Hexadecimal	0	0	8	5	0	8	0	0	4	5
Binary	0000	0000	1000	0101	0000	1000	0000	0000	0100	0101
Decimal	0		133		8		0		69	
ASCII	©		↑		©		©		E	

Hexadecimal	0	0	0	0	5	4	0	0	0	0
Binary	0000	0000	0000	0000	0101	0100	0000	0000	0000	0000
Decimal	0		0		84		0		0	
ASCII	©		©		T		©		©	

Hexadecimal	4	0	0	0	4	0	0	1	B	8
Binary	0100	0000	0000	0000	0100	0000	0000	0001	1011	1000
Decimal	64		0		64		1		184	
ASCII	@		©		@		©		↑	

Hexadecimal	C	C	C	0	A	8	0	0	2	7
Binary	1100	1100	1100	0000	1010	1000	0000	0000	0010	0111
Decimal	204		192		168		0		39	
ASCII	↑		↑		↑		©		‘	

Hexadecimal	C	0	A	8	0	0	6	5	0	8
Binary	1100	0000	1010	1000	0000	0000	0110	0101	0000	1000
Decimal	192		168		0		101		8	
ASCII	↑		↑		©		e		©	

Hexadecimal	0	0	C	9	1	5	7	0	6	0
Binary	0000	0000	1100	1001	0001	0101	0111	0000	0110	0000
Decimal	0		201		21		112		96	
ASCII	©		↑		©		p		‘’	

Hexadecimal	0	0	0	0	4	2	b	1	8	9
Binary	0000	0000	0000	0000	0100	0010	1011	0001	1000	1001
Decimal	0		0		66		177		137	
ASCII	©		©		B		↑		↑	

Hexadecimal	3	F	0	0	0	0	0	0	0	0
Binary	0011	1111	0000	0000	0000	0000	0000	0000	0000	0000
Decimal	63		0		0		0		0	
ASCII	?		©		©		©		©	

Hexadecimal	2	C	c	6	0	7	0	0	0	0
Binary	0010	1100	1100	0110	0000	0111	0000	0000	0000	0000
Decimal	44		198		7		0		0	
ASCII	,		↑		©		©		©	

Hexadecimal	0	0	0	0	0	0	1	0	1	1
Binary	0000	0000	0000	0000	0000	0000	0001	0000	0001	0001
Decimal	0		0		0		16		17	
ASCII	©		©		©		©		©	

Hexadecimal	1	2	1	3	1	4	1	5	2	6
Binary	0001	0010	0001	0011	0001	0100	0001	0101	0010	0110
Decimal	18		19		20		21		38	
ASCII	©		©		©		©		&	

Hexadecimal	2	7	2	8	2	9	2	A	2	B
Binary	0010	0111	0001	1000	0010	1001	0010	1010	0010	1011
Decimal	39		40		41		42		43	
ASCII	‘		(		)		*		+	

Hexadecimal	2	C	2	F	3	0	3	1	3	2
Binary	0010	1100	0010	111	0011	0000	0011	0001	0011	0010
Decimal	44		47		48		49		50	
ASCII	,		/		0		1		2	

Hexadecimal	3	3	3	4	3	5	3	6	3	7
Binary	0011	0011	0011	0100	0000	0101	0011	0110	0011	0111
Decimal	51		52		53		54		55	
ASCII	3		4		5		6		7	

Programming Hint: The name for this variable in code will be IP_ICMP_Data_PING.

2.2.23 IP PDU for the selected HTTP PDU

IP PDU > *IP Version* for the selected HTTP PDU

Field Name: *IP Version*

Purpose and Definition: Version is a 4-bit field that indicates the format of the internet header.

Field Key: 4 = IPv4
6 = IPv6

Data value (decimal): 4

Data values in other bases:

Hexadecimal	4
Binary	0100
Decimal	4

Programming hint: The name for this variable in code will be IP_Version_HTTP.

IP PDU> *Internet Header Length* for the selected HTTP PDU

Field Name: *Internet Header Length*

Purpose and Definition: The IHL field is a 4-bit field indicating the length of the internet header in 32 bit words, and thus points to the beginning of the data. The minimum value of a correct header is 5.

Field Key: *Not applicable*

Data value: The value contained in our field is 20 bytes. This is the hexadecimal and decimal value of 5 multiplied by 4 bits.

Data values in other bases:

Hexadecimal	0	5
Binary	0000	0101
Decimal	5	

Programming hint: The name for this variable in code will be IP_IHL_HTTP.

IP PDU > *Type of Service* for the selected HTTP PDU

Field Name: *Type of Service*

Purpose and Definition: Type of Service is an 8-bit field that provides and indication of the abstract parameters of the quality of service desired. These parameters guide the selection of the actual service parameters when transmitting a datagram through a particular network.

Field Key: The major choice is a three-way tradeoff between low-delay, high-reliability, and high-throughput.

0 1 2	3	4	5	6	7
Precedence	D	T	R	0	0

Bits 0-2: Precedence
Bit 3: (D) 0 = Normal Delay 1 = Low Delay
Bit 4: (T) 0 = Normal Throughput 1 = High Throughput
Bit 5: (R) 0 = Normal Reliability 1 = High Reliability

Precedence:

111 = Network Control	011 = Flash
110 = Internetwork Control	010 = Immediate
101 = CRITIC/ECP	001 = Priority
100 = Flash Overrided	000 = Routine

Data value (hexadecimal): 00

Data values in other bases:

Hexadecimal	0	0
Binary	0000	0000
Decimal	00	

Programming hint: The name for this variable in code will be IP_TypeOfService_HTTP.

IP PDU > *Total Length of Ethernet Frame* for the selected HTTP PDU

Field Name: *Total Length of Ethernet Frame*

Purpose and Definition: Total Length is a 16-bit field that indicates the length of the frame, measured in octets, including internet header and data. The maximum size is 2^{16}-1 or 65,535 octets; however, the recommended maximum size is 576 octets.

Field Key: *Not applicable*

Data values (hexadecimal): 570

Data values in other bases:

Hexadecimal	0	2	3	A
Binary	0000	0010	0011	1010
Decimal	2		58	
ASCII	©		:	

Programming hint: The name for this variable in code will be IP_TotalLength_HTTP.

IP PDU > *Identification* for the selected HTTP PDU

Field Name: *Identification*

Purpose and Definition: Identification is a 16-bit field. An identifying value is assigned by the sender to aid in assembling the fragments of a datagram. The identifier is chosen based on the need to provide a way to uniquely identify the fragments and protocol for the time the datagram or any fragment could be alive in the internet.

Field Key: *Not applicable*

Data value (hexadecimal): 3C 05

Data values in other bases:

Hexadecimal	3	C	0	5
Binary	0011	1100	0000	0101

Programming hint: The name for this variable in code will be IP_Identification_HTTP,

IP PDU > *Flags* for the selected HTTP PDU

Field Name: *Flags*

Purpose and Definition: Flags is a 3-bit field that indicates directions for fragmentation.

Field Key:
Bit 0: reserved, must be 0
Bit 1: (DF) 0 = May Fragment 1 = Don't Fragment
Bit 2: (MF) 0 = Last Fragment 1 = More Fragment

Data value (binary): 010

Data values in other bases: *Not applicable*

Programming hint: The name for this variable in code will be IP_Flags_HTTP.

IP PDU > *Fragment Offset* for the selected HTTP PDU

Field Name: *Fragment Offset*

Purpose and Definition: The Fragment Offset is a 13- bit field indicating where in the Ethernet frame this fragment begins. The Fragment Offset is measured in units of 8 octets, and the first fragment has offset 0.

Field Key: *Not applicable*

Data value (decimal): 0

Data values in other bases:
Binary: 0 0000 0000 0000

Programming hint: The name for this variable in code will be IP_FragmentOffset_HTTP.

IP PDU > *Time to Live* for the selected HTTP PDU

Field Name: *Time to Live*

Purpose and Definition: Time to Live is an 8-bit field that indicates the maximum time the datagram is allowed to remain in the internet. If this field contains the value 0, then the datagram must be destroyed. This field is modified in internet header processing. The time is measure in units of seconds, and is set by the sender to the maximum time the datagram is allowed to be in the internet. This field is decreased at each point that the internet header is processed. The intention is to cause undeliverable packets to be discarded, and to bound the maximum datagram lifetime.

Field Key: *Not applicable*

Data value (decimal): 64

Data values in other bases:

Hexadecimal	4	0
Binary	0100	0000
Decimal	64	

Programming hint: The name for this variable in code will be IP_TimeToLive_HTTP.

IP PDU > *Protocol* for the selected HTTP PDU

Field Name: *Protocol*

Purpose and Definition: Protocol is an 8-bit field that indicates the next level protocol that is used in the data portion of the internet diagram.

Field Key:

Dec	Hex	Protocol	Dec	Hex	Protocol
0	0	Reserved	22	16	Multiplexing
1	1	ICMP	23	17	DCN
2	2	Unassigned	24	18	TAC Monitoring
3	3	Gateway-to-Gateway	25-76	19-4C	Unassigned
4	4	CMCC Gateway Monitoring Message	77	4D	Any local network
5	5	ST	100	64	SATNET and Backroom EXPAK
6	6	TCP	101	65	MIT Subnet Support
7	7	UCL	102-104	66-68	Unassigned
10	A	Unassigned	105	69	SATNET Monitoring
11	B	Secure	106	6A	Unassigned
12	C	BBN RCC Monitoring	107	6B	Internet Packet Core Utility
13	D	NVP	110-113	6E-71	Unassigned
14	E	PUP	114	72	Backroom SATNET Monitoring
15	F	Pluribus	115	73	Unassigned
16	10	Telnet	116	74	WIDEBAND Monitoring
17	11	XNET	117	75	WIDEBAND EXPAK
20	14	Chaos	120-376	78-178	Unassigned
21	15	User Datagram	377	179	Reserved

Data value (hexadecimal): 06

Data values in other bases:

Hexadecimal	0	6
Binary	0000	0110
Decimal	6	

RFC Link: http://www.faqs.org/rfcs/rfc790.html

Programming hint: The name for this variable in code will be IP_Protocol_HTTP.

IP PDU > *Header Checksum* for the selected HTTP PDU

Field Name: *Header Checksum*

Purpose and Definition: The Header Checksum is a 16-bit field. The Checksum is the 16-bit one's complement sum of all 16-bit words in the header. For purposes of computing the checksum, the initial value of its field is zero. When both header checksums are equal, then the header bits are correct. If either checksums vary, then a new, correct packet will need to be sent.

This is a simple way to compute the checksum and experimental evidence indicates that it is adequate, but it is provisional and may be replaced by a CRC procedure, depending on further experience.

Field Key: *Not applicable*

Data value (hexadecimal): 7A F7

Data values in other bases:

Hexadecimal	7	A	5	7
Binary	0111	1010	0101	0111

Programming hint: The name for this variable in code will be IP_HeaderChecksum_HTTP.

IP PDU > *Source Address* for the selected HTTP PDU

Field Name: *Source Address*

Purpose and Definition: The Source Address is a 32-bit field that contains the IP address of the host that sent the IP Packet.

Field Key: *Not applicable*

Data value (decimal): 192.168.0.12

Data values in other bases:

Hexadecimal	C	0	A	8	0	0	0	C
Binary	1100	0000	1010	1000	0000	0000	0000	1100
Decimal	192		168		0		12	

Programming hint: The name for this variable in code will be IP_SourceAddress_HTTP.

IP PDU > *Destination Address* for the selected HTTP PDU

Field Name: *Destination Address*

Purpose and Definition: The Destination Address is a 32-bit field that contains the address of the host that is to receive the data contained within the IP packet.

Field Key: *Not applicable*

Data value (decimal): 192.168.0.101

Data values in other bases:

Hexadecimal	C	0	A	8	0	0	6	5
Binary	1100	0000	1010	1000	0000	0000	0110	0101
Decimal	192		168		0		101	

Programming hint: The name for this variable in code will be IP_DestinationAddress_HTTP.

IP PDU > *Options and Padding* for the selected HTTP PDU

Field Name: *Options and Padding*

Purpose and Definition: The options may or may not appear in Ethernet packets. They must be implemented by all IP modules (host and gateways). What is optional is their transmission in any particular packet, not their implementation.

The option field is variable in length. There may be zero or more options. There are two cases for the format of an option.

Case 1: A single octet of option type

Case 2: An option-type octet, an option-length octet, and the actual option-data octets.

Field Key: *Not applicable*

Data values: *Not applicable*

Data values in other bases: *Not applicable*

Programming hint: The name for this variable in code will be IP_OptionsPadding_HTTP.

IP PDU > *Data* for the selected HTTP PDU

Field Name: *Data*

Purpose and Definition: The Data is a variable length field which contains the actual data that is being sent from one host to another. The data field may start with a Layer 4 header, which will give additional instructions to the application that will be receiving the data; alternately, it may be an ICMP header and not contain any user data at all.

Field Key: *Not applicable*

Data values (hexadecimal) : (TCP) 80 30 00 15 81 A5 16 6C 87 A3 53 5D 80 18 16 D0 11 F4 00 00 01 01 08 0A 1B 25 F3 A1 0b DD 73 58
(FTP) 50 41 53 53 20 66 31 61 32 6B 33 75 73 65 72 0D 0A

Data values in other bases:
Hexadecimal: (TCP) 0 x 80 30 00 15 81 A5 16 6C 87 A3 53 5D 80 18 16 D0 11 F4 00 00 01 01 08 0A 1B 25 F3 A1 0B DD 73 58
(FTP) 50 41 53 53 20 66 31 61 32 6B 33 75 73 65 72 0D 0A

ASCII: (TCP) ↑ 0 © © ↑ ↑ © ↑ ↑ S] ↑ © © ↑ © ↑ © © © © © © © % ↑ ↑ © ↑ s X
(FTP) P A S S S © f 1 a 2 k 3 u s e r © ©

Programming hint: The name for this variable in code will be IP_Data_HTTP.

2.2.24 TCP PDU for the selected HTTP PDU

IP > TCP PDU > *Source Port* for the selected HTTP PDU

Field Name: *Source Port*

Purpose and Definition:
This 16-bit number represents the name of the application that sent the data in the IP packet.

Field Key: *Not applicable*

Data value: www (80)

Data values in other bases:

Hexadecimal	0	0	5	0
Binary	0000	0000	0101	0000
Decimal	0		80	
ASCII	©		P	

Programming hint: The name for this variable in code will be IP_TCP_SourcePort_HTTP.

IP > TCP PDU > *Destination Port* for the selected HTTP PDU

Field Name: *Destination Port*

Purpose and Definition:
This 16-bit number represents the name of the application that is to receive the data contained within the IP packet. This is one of the major differences between a Layer 3 and a Layer 4 header: the Layer 3 header contains the IP address of the computer that is to receive the IP packet; once that packet has been received, the port address in the Layer 4 header ensures that the data contained within that IP packet is passed to the correct application on that computer.

Field Key:
This key indicates assigned port number values:

Dec	**Port Numbers**
0	Reserved
1-32767	Internet registered ("well-known") protocols
32768-98303	Reserved, to allow TCPv7-TCPv4 conversion
98304 & up	Dynamic assignment

Data value (decimal): 4255

Data values in other bases:

Hexadecimal	1	0	9	F
Binary	0001	0000	1001	1111
Decimal	16		159	
ASCII	©		↑	

Source: http://www.zvon.org/tmRFC/RFC1475/Output/chapter4.html

Programming hint: The name for this variable in code will be IP_TCP_DestinationPort_HTTP.

IP > TCP PDU > *Sequence Number* for the selected HTTP PDU

Field Name: *Sequence Number*

Purpose and Definition:
TCP is responsible for ensuring that all IP packets sent are actually received. When an application's data is packaged into IP packets, TCP will give each IP packet a sequence number. Once all the packets have arrived at the receiving computer, TCP uses the number in this 32-bit field to ensure that all of the packets actually arrived and are in the correct sequence.

Field Key: *Not applicable*

Data value (decimal): 988014608

Data values in other bases:

Hexadecimal	3	A	E	3	E	8	1	0
Binary	0011	1010	1110	0011	1110	1000	0001	0000
Decimal	58		227		232		16	
ASCII	:		↑		↑		©	

Programming hint: The name for this variable in code will be IP_TCP_SequenceNumber_HTTP.

IP > TCP PDU > *Acknowledgement Number* for the selected HTTP PDU

Field Name: *Acknowledgement Number*

Purpose and Definition:
This number is used by the receiving computer to acknowledge which packets have successfully arrived. This number will be the sequence number of the next packet the receiver is ready to receive.

Field Key: *Not applicable*

Data value: 1398299764

Data values in other bases:

Hexadecimal	5	3	5	8	5	C	7	4
Binary	0101	0011	0101	0111	0101	1010	0111	0100
Decimal	83		88		92		116	
ASCII	S		X		\		t	

Programming hint: The name for this variable in code will be IP_TCP_AcknowledgmentNumber_HTTP.

IP > TCP PDU > *Header Length or Offset* for the selected HTTP PDU

Field Name: *Header Length or Offset*

Purpose and Definition:
This is identical in concept to the header length in an IP packet, except this time it indicates the length of the TCP header.

Field Key: *Not applicable*

Data value (bytes): 32

Data values in other bases:

Hexadecimal	8	0
Binary	1000	0000
Decimal	128	
ASCII	↑	

Programming hint: The name for this variable in code will be IP_TCP_IHL_HTTP.

IP > TCP PDU > *Control Flags* for the selected HTTP PDU

Field Name: *Control Flags*

Purpose and Definition:
Every TCP packet contains this 6-bit value that indicates how many octets it can receive at once. When IP packets are received, they are placed in a temporary area of RAM known as a buffer until the receiving computer has a chance to process them; this value represents how big a buffer the receiving host has made available for this temporary storage of IP packets.

Field Key:

- Urgent (URG)
- Acknowledgement (ACK)
- Push (PSH)
- Reset (RST)
- Synchronize (SYN)
- Finish (FIN)

Data value (binary): 01 1000

Data values in other bases: *Not applicable*

Programming hint: The name for this variable in code will be IP_TCP_Flags_HTTP.

IP > TCP PDU > *Window Size* for the selected HTTP PDU

Field Name: *Window Size*

Purpose and Definition:
Every TCP packet contains this 16-bit value that indicates how many octets it can receive at once. When IP packets are received, they are placed in a temporary area of RAM known as a buffer until the receiving computer has a chance to process them; this value represents how big a buffer the receiving host has made available for this temporary storage of IP packets.

Field Key: *Not applicable*

Data value (decimal): 7504

Data values in other bases:

Hexadecimal	1	D	5	0
Binary	0001	1101	0101	0000
Decimal	29		80	
ASCII	©		P	

Programming hint: The name for this variable in code will be IP_TCP_WindowSize_HTTP.

IP > TCP PDU > *Urgent Pointer* for the selected HTTP PDU

Field Name: *Urgent Pointer*

Purpose and Definition:
If the Urgent flag is set to on, this value indicates where the urgent data is located.

Information Key: *Not applicable*

Data value: *Not applicable*

Data values in other bases: *Not applicable*

Programming hint: The name for this variable in code will be IP_TCP_UrgentPointer_HTTP.

IP > TCP PDU > *Checksum* for the selected HTTP PDU

Field Name: *Checksum*

Purpose and Definition:
Unlike IP, TCP is responsible for ensuring that the entire IP packet arrived intact. TCP will run a CRC on the entire IP packet (not just the header) and place the resulting checksum in this field. When the IP packet is received, TCP re-runs the CRC on the entire packet to ensure the checksum is the same.

Field Key: *Not applicable*

Data value (hexadecimal): F0 F6

Data values in other bases:

Hexadecimal	F	0	F	6
Binary	1111	0000	1111	0110
Decimal	240		246	
ASCII	↑		↑	

Programming hint: The name for this variable in code will be IP_TCP_Checksum_HTTP.

IP > TCP PDU > *Options and Padding* for the selected HTTP PDU

Field Name: *Options and Padding*

Purpose and Definition:
Like IP options, this field is optional and represents additional instructions not covered in the other TCP fields. Again, if an option does not fill up a 32-bit word, it will be filled in with padding bits.

Field Key: *Not applicable*

Data value (hexadecimal): 08 0A 39 22 DB 5B 06 2F 44 96

Data values in other bases:

Hexadecimal	0	8	0	A	3	9	2	2	D	B
Binary	0000	1000	0000	1010	0011	1001	0010	0010	1101	1011
Decimal	8		10		57		34		219	
ASCII	©		©		9		“		↑	

Hexadecimal	5	B	0	6	2	F	4	4	9	6
Binary	0101	1011	0000	0110	0010	1111	0100	0100	1001	0110
Decimal	91		6		47		68		150	
ASCII	[		©		/		F		↑	

Programming hint: The name for this variable in code will be IP_TCP_Options_HTTP.

2.2.25 HTTP PDU for the selected HTTP PDU

IP > TCP > HTTP PDU > *Content Type* for the selected HTTP PDU

Field Name: *Content Type*

Purpose and Definition: The Content-Type entity-header field indicates the media type of the Entity-Body sent to the recipient.

Field Key: *Not applicable*

Data value (ASCII): text/html; charset=iso – 8859-1\r\n

Data values in other bases:

Hexadecimal	4	3	6	F	6	E	7	4
Binary	0100	0011	0110	1111	0110	1110	0111	0100
Decimal	67		111		110		116	
ASCII	C		o		n		t	

Hexadecimal	6	5	6	E	7	4	2	D
Binary	0110	0101	0110	1110	0111	0100	0010	1101
Decimal	101		110		116		45	
ASCII	e		n		t		-	

Hexadecimal	5	4	7	9	7	0	6	5
Binary	0101	0100	0111	1001	0111	0000	0110	0101
Decimal	84		121		112		101	
ASCII	T		y		p		e	

Hexadecimal	3	A	2	0	7	4	6	5
Binary	0110	1010	0010	0000	0111	0100	0110	0101
Decimal	58		32		116		101	
ASCII	:				t		e	

Hexadecimal	7	8	7	4	2	F	6	8
Binary	0111	1000	0111	0100	0010	1111	0110	1000
Decimal	120		116		47		104	
ASCII	x		t		/		h	

Hexadecimal	7	4	6	D	6	C	3	B
Binary	0111	0100	0110	1101	0110	1100	0011	1011
Decimal	116		109		108		59	
ASCII	t		m		l		;	

Hexadecimal	2	0	6	3	6	8	6	1
Binary	0010	0000	0110	0011	0110	1000	0110	0001
Decimal	32		99		104		97	
ASCII			c		h		a	

Hexadecimal	7	2	7	3	6	5	7	4
Binary	0111	0010	0111	0011	0110	0101	0111	0100
Decimal	114		115		101		116	
ASCII	r		s		e		t	

Hexadecimal	3	D	6	9	7	3	6	F
Binary	0011	1101	0110	1001	0111	0011	0110	1111
Decimal	61		105		115		111	
ASCII	=		i		s		o	

Hexadecimal	2	D	3	8	3	8	3	5
Binary	0010	1101	0011	1000	0011	1000	0011	0101
Decimal	45		56		56		53	
ASCII	-		8		8		5	

Hexadecimal	3	9	2	D	3	1	0	D
Binary	0011	1001	0010	1101	0011	0001	0000	1101
Decimal	57		45		49		13	
ASCII	9		-		1		\r	

Hexadecimal	0	A
Binary	0000	1010
Decimal	10	
ASCII	\n	

Programmer's Hint: The name for this variable in code will be IP_TCP_HTTP_Content-Type_HTTP.

IP > TCP > HTTP PDU > *Date* for the selected HTTP PDU

Field Name: *Date*

Purpose and Definition: This field contains the date and time on which the web page was accessed.

Field Key: *Not applicable*

Data value (ASCII): Date: Tue, 03 Feb 2004 23:08:10 GMT\r\n

Data values in other bases:

Hexadecimal	4	6	6	1	7	4	6	5
Binary	0110	0110	0110	0001	0111	0100	0110	0101
Decimal	70		97		116		101	
ASCII	D		a		t		e	

Hexadecimal	3	A	2	0	5	4	7	5
Binary	0010	1010	0010	0000	0101	0100	0111	0101
Decimal	58		32		84		117	
ASCII	:				T		u	

Hexadecimal	6	5	2	C	2	0	3	0
Binary	0110	0101	0010	1100	0010	0000	0011	0000
Decimal	101		44		32		48	
ASCII	e		,				0	

Hexadecimal	3	3	2	0	4	6	6	5
Binary	0011	0011	0010	0000	0100	0110	0110	0101
Decimal								
ASCII	3				F		e	

Hexadecimal	6	2	2	0	3	2	3	0
Binary	0110	0010	0010	0000	0011	0010	0011	0000
Decimal	98		32		50		48	
ASCII	b				2		0	

Hexadecimal	3	0	3	4	3	2	3	3
Binary	0011	0000	0011	0100	0011	0010	0011	0011
Decimal	48		52		50		51	
ASCII	0		4		2		3	

Hexadecimal	3	A	3	0	3	8	3	A
Binary	0011	1010	0011	0000	0011	1000	0011	1010
Decimal	58		48		56		58	
ASCII	:		0		8		:	

Hexadecimal	3	1	3	0	2	0	4	7
Binary	0011	0001	0011	0000	0010	0000	0100	0111
Decimal	49		48		32		71	
ASCII	1		0				G	

Hexadecimal	4	D	5	4	0	D	0	A
Binary	0100	1101	0101	0100	0000	1101	0000	1010
Decimal	77		84		13		10	
ASCII	M		T		\r		\n	

Programmer's Hint: The name for this variable in code will be IP_TCP_HTTP_Date_HTTP.

IP > TCP > HTTP PDU > *HTTP* for the selected HTTP PDU

Field Name: *HTTP*

Purpose and Definition: This field displays the category of the page that is being displayed.

Field Key: *Not applicable*

Data value (ASCII): HTTP/1.1 404 Not Found\r\n

Data values in other bases:

Hexadecimal	4	8	5	4	5	4	5	0
Binary	0100	1000	0101	0100	0101	0100	0101	0000
Decimal	72		84		84		80	
ASCII	H		T		T		P	

Hexadecimal	2	F	3	1	2	E	3	1
Binary	0010	1111	0011	0001	0010	1110	0011	0001
Decimal	47		49		46		49	
ASCII	/		1		.		1	

Hexadecimal	2	0	3	4	3	0	3	4
Binary	0010	0000	0011	0100	0011	0000	0011	0100
Decimal	32		52		48		52	
ASCII			4		0		4	

Hexadecimal	2	0	4	E	6	F	7	4
Binary	0010	0000	0100	1110	0110	1111	0111	0100
Decimal	32		78		111		116	
ASCII			N		o		t	

Hexadecimal	2	0	4	6	6	F	7	5
Binary	0010	0000	0100	0110	0110	1111	0111	0101
Decimal	32		70		111		117	
ASCII			F		o		u	

Hexadecimal	6	E	6	4	0	D	0	A
Binary	0110	1110	0110	0100	0000	1101	0000	1010
Decimal	110		100		13		10	
ASCII	n		d		\r		\n	

Programmer's Hint: The name for this variable in code will be IP_TCP_HTTP_HTTP_HTTP.

IP > TCP > HTTP PDU > *Server* for the selected HTTP PDU

Field Name: *Server*

Purpose and Definition: The Server response-header field contains information about the software used by the origin server to handle the request.

Field Key: *Not applicable*

Data value (ASCII): Server: Apache/1.3.24 (Unix) PHP/4.2.1\r\n

Data values in other bases:

Hexadecimal	5	3	6	5	7	2	7	6
Binary	0101	0011	0110	0101	0111	0010	0111	0110
Decimal	83		101		114		118	
ASCII	S		e		r		v	

Hexadecimal	6	5	7	2	3	A	2	0
Binary	0110	0101	0111	0010	0011	1010	0010	0000
Decimal	101		114		58		32	
ASCII	e		r		:			

Hexadecimal	4	1	7	0	6	1	6	3
Binary	0110	0001	0111	0000	0110	0001	0110	0011
Decimal	65		112		97		99	
ASCII	A		p		a		c	

Hexadecimal	6	8	6	5	2	F	3	1
Binary	0110	1000	0110	0101	0010	1111	0011	0001
Decimal	104		101		47		49	
ASCII	h		e		/		1	

Hexadecimal	2	E	3	3	2	E	3	2
Binary	0010	1110	0011	0011	0010	1110	0011	0010
Decimal	46		51		46		50	
ASCII	.		3		.		2	

Hexadecimal	3	4	2	0	2	8	5	5
Binary	0011	0100	0010	0000	0010	1000	0101	0101
Decimal	52		32		40		85	
ASCII	4				(		U	

Hexadecimal	6	E	6	9	7	8	2	9
Binary	0110	1110	0110	1001	0111	1000	0010	1001
Decimal	110		105		120		41	
ASCII	n		i		x		)	

Hexadecimal	2	0	5	0	4	8	5	0
Binary	0010	0000	0101	0000	0100	1000	0101	0000
Decimal	32		80		72		80	
ASCII			P		H		P	

Hexadecimal	2	5	3	4	2	E	3	2
Binary	0010	0101	0011	0100	0010	1110	0011	0010
Decimal	37		52		46		50	
ASCII	/		4		.		2	

Hexadecimal	2	E	3	1	0	D	0	A
Binary	0010	1110	0011	0001	0000	1101	0000	1010
Decimal	46		49		13		10	
ASCII	.		1		\r		\n	

Programmer's Hint: The name for this variable in code will be IP_TCP_HTTP_Server_HTTP.

IP > TCP > HTTP PDU > *Data* for the selected HTTP PDU

Field Name: *Data*

Purpose and Definition: This field stores the information that is actually contained in the HTTP Protocol.

Field Key: *Not applicable*

Data value (ASCII): <!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">\n
<HTML><HEAD>\n
<TITLE>404 Not Found</TITLE>\n
</HEAD><BODY>\n
<H1>Not Found</H1>\n
The requested URL /~csis410/2003/bluetech/Requirements Speicfication Document Final-files/image002.gif was not found on this server.<p>\n
<HR>\n
<ADDRESS>Apache/1.3.24 Server at ares.cs.siena.edu Port 80</ADDRESS>\n
</BODY></HTML>\n

Data values in other bases:

Hexadecimal	3	C	2	1	4	4	4	5
Binary	0011	1100	0010	0001	0100	0100	0100	0101
Decimal	60		33		68		69	
ASCII	<		!		D		O	

Hexadecimal	4	3	5	4	5	9	5	0
Binary	0100	0011	0101	0100	0101	1001	0101	0000
Decimal	67		84		89		80	
ASCII	C		T		Y		P	

Hexadecimal	4	5	2	0	4	8	5	4
Binary	0100	0101	0010	0000	0100	1000	0101	0100
Decimal	69		32		72		84	
ASCII	E				H		T	

Hexadecimal	4	D	4	C	2	0	5	0
Binary	0100	1101	0110	1100	0010	0000	0101	0000
Decimal	77		76		32		80	
ASCII	M		L				P	

Hexadecimal	5	5	4	2	4	C	4	9
Binary	0101	0101	0100	0010	0100	1100	0100	1001
Decimal	85		66		76		73	
ASCII	U		B		L		I	

Hexadecimal	4	3	2	0	2	2	2	D
Binary	0100	0011	0010	0000	0010	0010	0010	1101
Decimal	67		32		34		45	
ASCII	C				“		-	

Hexadecimal	2	F	2	F	4	9	4	5
Binary	0010	1111	0010	1111	0100	1001	0100	0101
Decimal	47		47		73		69	
ASCII	/		/		I		E	

Hexadecimal	5	4	4	6	2	F	2	F
Binary	0101	0100	0100	0110	0010	1111	0010	1111
Decimal	84		70		47		47	
ASCII	T		F		/		/	

Hexadecimal	4	4	5	4	4	4	2	0
Binary	0100	0100	0101	0100	0100	0100	0010	0000
Decimal	68		84		68		32	
ASCII	D		T		D			

Hexadecimal	4	8	5	4	4	D	4	C
Binary	0100	1000	0101	0100	0100	1101	0100	1100
Decimal	72		84		77		76	
ASCII	H		T		M		L	

Hexadecimal	2	0	3	2	2	E	3	0
Binary	0010	0000	0011	0010	0010	1110	0011	0000
Decimal	32		50		46		48	
ASCII			2		.		0	

Hexadecimal	2	F	2	F	4	5	4	E
Binary	0010	1111	0010	1111	0100	0101	0100	1110
Decimal	47		47		69		78	
ASCII	/		/		E		N	

Hexadecimal	2	2	3	E	0	A	3	C
Binary	0010	0010	0011	1110	0000	1010	0011	1100
Decimal	34		62		10		60	
ASCII	“		>		\n		<	

Hexadecimal	4	8	5	4	4	D	4	C
Binary	0100	1000	0101	0100	0100	1101	0100	1100
Decimal	72		84		77		76	
ASCII	H		T		M		L	

Hexadecimal	3	E	3	C	4	8	4	5
Binary	0011	1110	0011	1100	0100	1000	0100	0101
Decimal	62		60		72		69	
ASCII	>		<		H		E	

Hexadecimal	4	1	4	4	3	E	0	A
Binary	0100	0001	0100	0100	0011	1110	0000	1010
Decimal	65		68		62		10	
ASCII	A		D		>		\n	

Hexadecimal	3	C	5	4	4	9	5	4
Binary	0011	1100	0101	0100	0100	1001	0101	0100
Decimal	60		84		73		84	
ASCII	<		T		I		T	

Hexadecimal	4	C	4	5	3	E	3	4
Binary	0100	1100	0100	0101	0011	1110	0011	0100
Decimal	76		69		62		52	
ASCII	L		E		>		4	

Hexadecimal	3	0	3	4	2	0	4	E
Binary	0011	0000	0011	0100	0010	0000	0100	1110
Decimal	48		52		32		78	
ASCII	0		4				N	

Hexadecimal	6	F	7	4	2	0	4	6
Binary	0110	1111	0111	0100	0010	0000	0100	0110
Decimal	111		116		32		70	
ASCII	o		t				F	

Hexadecimal	6	F	7	5	6	E	6	4
Binary	0110	1111	0111	0101	0110	1110	0110	0100
Decimal	111		117		110		100	
ASCII	o		u		n		d	

Hexadecimal	3	C	2	F	5	4	4	9
Binary	0011	1100	0010	1111	0101	0100	0100	1001
Decimal	60		47		84		73	
ASCII	<		/		T		I	

Hexadecimal	5	4	4	C	4	5	3	E
Binary	0101	0100	0100	1100	0100	0101	0011	1110
Decimal	84		76		69		62	
ASCII	T		L		E		>	

Hexadecimal	0	A	3	C	2	F	4	8
Binary	0000	1010	0011	1100	0010	1111	0100	1000
Decimal	10		60		47		72	
ASCII	\n		<		/		H	

Hexadecimal	4	5	4	1	4	4	3	E
Binary	0100	0101	0100	0001	0100	0100	0011	1110
Decimal	69		65		68		62	
ASCII	E		A		D		>	

Hexadecimal	3	C	4	2	4	F	4	4
Binary	0011	1100	0100	0010	0100	1111	0100	0100
Decimal	60		66		79		68	
ASCII	<		B		O		D	

Hexadecimal	5	9	3	E	0	A	3	C
Binary	0101	1001	0011	1110	0000	1010	0011	1100
Decimal	89		62		10		60	
ASCII	Y		>		\n		<	

Hexadecimal	4	8	3	1	3	E	4	E
Binary	0100	1000	0011	0001	0011	1110	0100	1110
Decimal	72		49		62		78	
ASCII	H		1		>		N	

Hexadecimal	6	F	7	4	2	0	4	6
Binary	0110	1111	0111	0100	0010	0000	0100	0110
Decimal	111		116		32		70	
ASCII	o		t				F	

Hexadecimal	6	F	7	5	6	E	6	4
Binary	0100	1111	0111	0101	0110	1110	0110	0100
Decimal	111		117		110		100	
ASCII	o		u		n		d	

Hexadecimal	3	C	2	F	4	8	3	1
Binary	0011	1100	0010	1111	0100	1000	0011	0001
Decimal	60		47		72		49	
ASCII	<		/		H		1	

Hexadecimal	3	E	0	A	5	4	6	8
Binary	0011	1110	0000	1010	0101	0100	0110	1000
Decimal	62		10		84		104	
ASCII	>		\n		T		h	

Hexadecimal	6	5	2	0	7	2	6	5
Binary	0110	0101	0010	0000	0111	0010	0110	0101
Decimal	101		32		114		101	
ASCII	e				r		e	

Hexadecimal	7	1	7	5	6	5	7	3
Binary	0111	0001	0111	0101	0110	0101	0111	0011
Decimal	113		117		101		115	
ASCII	q		u		e		s	

Hexadecimal	7	4	6	5	6	4	2	0
Binary	0111	0100	0110	0101	0110	0100	0010	0000
Decimal	116		101		100		32	
ASCII	t		e		d			

Hexadecimal	5	5	5	2	4	C	2	0
Binary	0101	0101	0101	0010	0100	1100	0010	0000
Decimal	85		82		76		32	
ASCII	U		R		L			

Hexadecimal	2	F	7	E	6	3	7	3
Binary	0010	1111	0111	1110	0110	0011	0111	0011
Decimal	47		126		99		115	
ASCII	/		~		c		s	

Hexadecimal	6	9	7	3	3	4	3	1
Binary	0110	1001	0111	0011	0011	0100	0011	0001
Decimal	105		115		52		49	
ASCII	i		s		4		1	

Hexadecimal	3	0	2	F	3	2	3	0
Binary	0011	0000	0010	1111	0011	0010	0011	0000
Decimal	48		47		50		48	
ASCII	0		/		2		0	

Hexadecimal	3	0	3	3	2	F	6	2
Binary	0011	0000	0011	0011	0010	1111	0110	0010
Decimal	48		51		47		98	
ASCII	0		3		/		b	

Hexadecimal	6	C	7	5	6	5	7	4
Binary	0110	1100	0111	0101	0110	0101	0111	0100
Decimal	108		117		101		116	
ASCII	l		u		e		t	

Hexadecimal	6	5	6	3	6	8	2	F
Binary	0110	0101	0110	0011	0110	1000	0010	1111
Decimal	101		99		104		47	
ASCII	e		c		h		/	

Hexadecimal	5	2	6	5	7	1	7	5
Binary	0101	0010	0110	0101	0111	0001	0111	0101
Decimal	82		101		113		117	
ASCII	R		e		q		u	

Hexadecimal	6	9	7	2	6	5	6	D
Binary	0110	1001	0111	0010	0110	0101	0110	1101
Decimal	105		114		101		109	
ASCII	i		r		e		m	

Hexadecimal	6	5	6	E	7	4	7	3
Binary	0110	0101	0110	1110	0111	0100	0111	0011
Decimal	101		110		116		115	
ASCII	e		n		t		s	

Hexadecimal	2	0	5	3	7	0	6	5
Binary	0010	0000	0101	0011	0111	0000	0110	0101
Decimal	32		83		112		101	
ASCII			S		p		e	

Hexadecimal	6	3	6	9	6	6	6	9
Binary	0110	0011	0110	1001	0110	0110	0110	1001
Decimal	99		105		102		105	
ASCII	c		i		f		i	

Hexadecimal	6	3	6	1	7	4	6	9
Binary	0110	0011	0110	0001	0111	0100	0110	1001
Decimal	99		97		116		105	
ASCII	c		a		t		i	

Hexadecimal	6	F	6	E	2	0	4	4
Binary	0110	1111	0110	1110	0010	0000	0100	0100
Decimal	111		110		32		68	
ASCII	o		n				D	

Hexadecimal	6	F	6	3	7	5	6	D
Binary	0110	1111	0110	0011	0111	0101	0110	1101
Decimal	111		99		117		109	
ASCII	o		c		u		m	

Hexadecimal	6	5	6	E	7	4	2	0
Binary	0110	0101	0110	1110	0111	0100	0010	0000
Decimal	101		110		116		32	
ASCII	e		n		t			

Hexadecimal	4	6	6	9	6	E	6	1
Binary	0100	0110	0110	1001	0110	1110	0110	0001
Decimal	70		105		110		97	
ASCII	F		i		n		a	

Hexadecimal	6	C	5	F	6	6	6	9
Binary	0110	1100	0101	1111	0110	0110	0110	1001
Decimal	108		95		102		105	
ASCII	l		-		f		i	

Hexadecimal	6	C	6	5	7	3	2	F
Binary	0110	1100	0110	0101	0111	0011	0010	1111
Decimal	63		101		115		47	
ASCII	l		e		s		/	

Hexadecimal	6	9	6	D	6	1	6	7
Binary	0110	1001	0110	1101	0110	0001	0110	0111
Decimal	105		109		97		103	
ASCII	i		m		a		g	

Hexadecimal	6	5	3	0	3	0	3	2
Binary	0110	0101	0011	0000	0011	0000	0011	0010
Decimal	101		48		48		50	
ASCII	e		0		0		2	

Hexadecimal	2	E	6	7	6	9	6	6
Binary	0010	1110	0110	0111	0110	1001	0110	0110
Decimal	46		103		105		102	
ASCII	.		g		i		f	

Hexadecimal	2	0	7	7	6	1	7	3
Binary	0001	0000	0111	0111	0110	0001	0111	0011
Decimal	32		119		97		115	
ASCII			w		a		s	

Hexadecimal	2	0	6	E	6	F	7	4
Binary	0010	0000	0110	1110	0110	1111	0111	0100
Decimal	32		110		111		116	
ASCII			n		o		t	

Hexadecimal	2	0	6	6	6	F	7	5
Binary	0010	0000	0110	0110	0110	1111	0111	0101
Decimal	32		102		111		117	
ASCII			f		o		u	

Hexadecimal	6	E	6	4	2	0	6	F
Binary	0110	1110	0110	0100	0010	0000	0110	1111
Decimal	110		100		32		111	
ASCII	n		d				o	

Hexadecimal	6	E	2	0	7	4	6	8
Binary	0110	1110	0010	0000	0111	0100	0110	1000
Decimal	110		32		116		104	
ASCII	n				t		h	

Hexadecimal	6	9	7	3	2	0	7	3
Binary	0110	1001	0111	0011	0010	0000	0111	0011
Decimal	105		115		32		115	
ASCII	i		s				s	

Hexadecimal	6	5	7	2	7	6	6	5
Binary	0110	0101	0111	0010	0111	0110	0110	0101
Decimal	101		114		118		101	
ASCII	e		r		v		e	

Hexadecimal	7	2	2	E	3	C	5	0
Binary	0111	0010	0010	1110	0011	1100	0101	0000
Decimal	114		46		60		80	
ASCII	r		.		<		p	

Hexadecimal	3	E	0	A	3	C	4	8
Binary	0011	1110	0000	1010	0011	1100	0100	1000
Decimal								
ASCII	>		\n		<		H	

Hexadecimal	5	2	3	E	0	4	3	C
Binary	0101	0010	0011	1110	0000	0100	0011	1100
Decimal	82		62		10		60	
ASCII	R		>		\n		<	

Hexadecimal	4	1	4	4	4	4	5	2
Binary	0100	0001	0100	0100	0100	0100	0101	0010
Decimal	65		68		68		82	
ASCII	A		D		D		R	

Hexadecimal	4	5	5	3	5	3	3	E
Binary	0100	0101	0101	0011	0101	0011	0011	1110
Decimal	69		83		83		62	
ASCII	E		S		S		>	

Hexadecimal	4	1	7	0	6	1	6	3
Binary	0100	0001	0111	0000	0110	0001	0110	0011
Decimal	65		112		97		99	
ASCII	A		p		a		c	

Hexadecimal	6	8	6	5	2	F	3	1
Binary	0110	1000	0110	0101	0010	1111	0011	0001
Decimal	104		101		47		49	
ASCII	h		e		/		1	

Hexadecimal	2	E	3	3	2	E	3	2
Binary	0010	1110	0011	0011	0010	1110	0011	0010
Decimal	46		51		46		50	
ASCII	.		3		.		2	

Hexadecimal	3	4	2	0	5	3	6	5
Binary	0011	0100	0010	0000	0101	0011	0110	0101
Decimal	52		32		83		101	
ASCII	4				S		e	

Hexadecimal	7	2	7	6	6	5	7	2
Binary	0111	0010	0111	0110	0110	0101	0111	0010
Decimal								
ASCII	r		v		e		r	

Hexadecimal	2	0	6	1	7	4	2	0
Binary	0010	0000	0110	0001	0111	0100	0010	0000
Decimal	32		97		116		323	
ASCII			a		t			

Hexadecimal	3	1	7	2	6	5	7	3
Binary	0011	0001	0111	0010	0110	0101	0111	0011
Decimal	97		114		101		115	
ASCII	a		r		e		s	

Hexadecimal	2	E	6	3	7	3	2	E
Binary	0010	1110	0110	0011	0111	0011	0010	1110
Decimal	46		99		115		46	
ASCII	.		c		s		.	

Hexadecimal	7	3	6	9	6	5	6	E
Binary	0111	0011	0110	1001	0110	0101	0110	1110
Decimal	115		105		101		110	
ASCII	s		i		e		n	

Hexadecimal	6	1	2	E	6	5	6	4
Binary	0110	0001	0010	1110	0110	0101	0110	0100
Decimal	97		46		101		100	
ASCII	a		.		e		d	

Hexadecimal	7	5	2	0	5	0	6	F
Binary	0111	0101	0010	0000	0101	0000	0110	1111
Decimal	117		32		80		111	
ASCII	u				P		o	

Hexadecimal	7	2	7	4	2	0	3	8
Binary	0111	0010	0111	0100	0010	0000	0011	1000
Decimal	114		116		32		56	
ASCII	r		t				8	

Hexadecimal	3	0	3	C	2	F	4	1
Binary	0011	0000	0010	1100	0010	1111	0100	0001
Decimal	48		60		47		65	
ASCII	0		<		/		A	

Hexadecimal	4	4	4	4	5	2	4	5
Binary	0100	0100	0100	0100	0101	0010	0100	0101
Decimal	68		68		82		69	
ASCII	D		D		R		E	

Hexadecimal	5	3	5	3	3	E	0	A
Binary	0101	0011	0101	0011	0011	1110	0000	1010
Decimal	83		83		62		10	
ASCII	S		S		>		\n	

Hexadecimal	3	C	2	F	4	2	4	F
Binary	0011	1100	0010	1111	0100	0010	0100	1111
Decimal	60		47		66		79	
ASCII	<		/		B		O	

Hexadecimal	4	4	5	9	3	E	3	C
Binary	0100	0100	0101	1001	0011	1110	0011	1100
Decimal	68		89		62		60	
ASCII	D		Y		>		<	

Hexadecimal	2	F	4	8	5	4	4	D
Binary	0010	1111	0100	1000	0101	0100	0100	1101
Decimal	47		72		84		77	
ASCII	/		H		T		M	

Hexadecimal	4	C	3	E	0	A
Binary	0100	1100	0011	1110	0000	1010
Decimal	76		62		10	
ASCII	L		>		\n	

Programmer's Hint: The name for this variable in code will be IP_TCP_HTTP_Data_HTTP.

3.0 Testing Requirements

3.1 Testing Overview

We will be assigning someone from outside of our company to test our software. This will eliminate biases and create a fair environment to ensure that all requirements are met.

We will be conducting these tests for the detailed design portion of development, which is the basis for the final development of the software.

We will be implementing gray box testing in the detailed design portion of our development. Gray box testing is a testing procedure done with some knowledge of how the internals work.

Attributes Tested:
The result from a right/left click of the mouse on the:

- Individual field of the packets
- Hierarchical tree
- Options

3.2 Test Cases

- Did the computer connect to the web-based client?
  - Does each screen load up promptly when navigating through the client?
  - Is scrolling to a minimum?
- At the main menu:
  - Do the buttons bring you to correct/next logical screen/PDU?
  - Does changing the radians show affect on all PDU's?
  - Is the hierarchical tree dynamic?
  - Is the "Choose a Protocol" hierarchical tree displayed when the user clicks "Choose a Protocol?"
  - Does a message box appear when user selects "IPv6" stating that it is not currently available?
- Graphical User Interface (GUI):
  - Is the GUI clearly visible on 1024x768 projectors?
    - Is it visible from the farthest corners in the room?
  - Are all colors easily distinguishable?
  - Are information boxes placed so that the current PDU is not covered?
  - When a field is clicked, is the information box the same color as the field?
  - Does each protocol have a link to its RFC?

3.3 Testing Sheets

3.3.1 Functional Requirements

Functional Requirements

Date: _____

Tester: _____

Screen: **Pass** **Fail**

Requirement	Actual Result	Comments	Pass	Fail
Contains information for various protocols.				

Requirement	Actual Result	Comments	Pass	Fail
Produces GUI that colorfully and clearly displays contents of the specified protocol.				

Requirement	Actual Result	Comments	Pass	Fail
Displays clearly on a 1024x768 pixel screen.				

Requirement	Actual Result	Comments	Pass	Fail
Menus on top of screen should be visible on every page to allow user to change protocol, or switch between layers.				

Requirement	Actual Result	Comments	Pass	Fail
Ability to view either an independent field or an entire protocol in a different radix				

Functional Requirements

Requirement	Actual Result	Comments	Pass	Fail
Produces information box for each field of a protocol which displays Purpose of Field, Options for Pattern, Bit Pattern Form, and Minimum/Maximum Length				

Requirement	Actual Result	Comments	Pass	Fail
Displays RFC link for entire protocol or specific field when available.				

3.3.2 Ethernet Testing Sheet

Screen: Ethernet

Date: _____

Tester: _____

Screen: **Pass** **Fail**

Field Name: ***"Choose a Protocol" Button***

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	Tree with all possible protocols is displayed			

Field Name*: IPv4 Button*

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	By default, IPv4 will be displayed			

Field Name*: IPv6 Button*

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	Message box will be displayed saying that IPv6 is currently not available			

Field Name*: Preamble Sof Sync Data Field*

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	Preamble information box is displayed in a magenta box			

Field Name*: Datalink Header*

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	Datalink information field is displayed in a green box			

Field Name: ***Frame Check Sequence (FCS)***

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	FCS information field is displayed in a yellow box			

Screen: Ethernet

Field Name*: IP PDU*

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	IP PDU information field is displayed in a cyan box			

Field Name*: "Different Protocol is Selected"*

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	Dynamic Road Map will change to show hierarchy of available protocols			

3.3.3 IP Testing Sheet

Screen: IP PDU

Date: _____

Tester: _____

Screen: **Pass** **Fail**

Field Name: ***Version***

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	Version information field pops up in a pink box			

Field Name*: Internet Header Length*

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	Internet Header Length information field pops up in a cyan box			

Field Name*: Type of Service*

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	Type of Service information field pops up in a yellow box			

Field Name: ***Total Length***

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	Total Length information field pops up in a green box			

Field Name*: Identification*

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	Identification information field pops up in an orange box			

Screen: IP PDU

Field Name*: Flags*

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	Flags information field pops up in a cyan box			

Field Name*: Fragment Offset*

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	Fragment Offset information field pops up in a magenta box			

Field Name*: Time to Live*

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	Time to Live information field pops up in a cyan box			

Field Name*: Protocol*

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	Protocol information field pops up in a pink box			

Field Name*: Header Checksum*

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	Header Checksum information field pops up in a yellow box			

Field Name*: Source IP Address*

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	Source IP Address information field pops up in a green box			

Screen: IP PDU

Field Name*: Destination IP Address*

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	Destination IP Address information field pops up in an orange box			

Field Name*: Options*

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	Options information field pops up in a cyan box			

Field Name*: Data*

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	Data information field pops up in a magenta box			

3.3.4 TCP Testing Sheet

Screen: TCP PDU

Date: _____

Tester: _____

Screen: **Pass** **Fail**

Field Name: ***Source Port***

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	Source Port information field pops up in a pink box			

Field Name*: Destination Port*

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	Destination Port information field pops up in a cyan box			

Field Name*: Sequence Number*

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	Sequence Number information field pops up in a yellow box			

Field Name*: Acknowledgement Number*

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	Acknowledgment Number information field pops up in a green box			

Field Name: ***Length***

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	Length information field pops up in a orange box			

Field Name: ***Reserved***

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	Reserved information field pops up in a magenta box			

Screen: TCP PDU

Field Name: *URG*

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	URG information field pops up in a orange box			

Field Name: *ACK*

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	ACK information field pops up in a pink box			

Field Name: *PSH*

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	PSH information field pops up in a cyan box			

Field Name: *RST*

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	RST information field pops up in a magenta box			

Field Name: *SYN*

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	SYN information field pops up in a green box			

Field Name: *FIN*

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	FIN information field pops up in a yellow box			

Field Name: *Window Size*

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	Window Size information field pops up in pink box			

Screen: TCP PDU

Field Name: *Checksum*

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	Checksum information field pops up in a green box			

Field Name: *Urgent Pointer*

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	Urgent Pointer information field pops up in a yellow box			

Field Name: *Options*

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	Options information field pops up in a magenta box			

Field Name: *Data*

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	FTP PDU pops up			

3.3.5 FTP Testing Sheet

Screen: FTP PDU

Date: _____

Tester: _____

Screen: **Pass** **Fail**

Field Name: ***Destination Address***

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	Destination Address information field pops up in the appropriate colored box			

Field Name***: Source Address***

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	Source Address information field pops up in the appropriate colored box			

***Field Name*: Pass**

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	Pass information field pops up in the appropriate colored box			

3.3.6 ICMP Testing Sheet

Screen: ICMP PDU

Date: _____

Tester: _____

Screen: **Pass** **Fail**

Field Name: ***Type***

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	Type information field pops up in an appropriately colored box			

Field Name*: Code*

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	Code information field pops up in an appropriately colored box			

Field Name*: Checksum*

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	Checksum information field pops up in an appropriately colored box			

Field Name*: Identifier*

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	Identifier information field pops up in an appropriately colored box			

***Field Name*: Sequence**

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	Sequence information field pops up in an appropriately colored box			

Screen: ICMP PDU

Date: _____

Tester: _____

Screen: **Pass** **Fail**

Field Name*: Data*

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	Data information field pops up in an appropriately colored box			

3.3.7 SMTP Testing Sheet

Screen: SMTP PDU

Date: _____

Tester: _____

Screen: **Pass** **Fail**

Field Name: ***Command***

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	Command information field pops up in an appropriately colored box			

Field Name***: Message***

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	Message information field pops up in an appropriately colored box			

3.3.8 UDP Testing Sheet

Screen: UDP PDU

Date: _____

Tester: _____

Screen: **Pass** **Fail**

Field Name: ***Source Port***

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	Source Port information field pops up in the appropriately colored box			

Field Name*: Destination Port*

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	Destination Port information field pops up in the appropriately colored box			

Field Name*: Length*

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	Length information field pops up in the appropriately colored box			

Field Name*: Checksum*

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	Checksum information field pops up in the appropriately colored box			

Field Name: ***Data***

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	Data information field pops up in the appropriately colored box			

3.3.9 SNMP Testing Sheet

Screen: SNMP PDU

Date: _____

Tester: _____

Screen: **Pass** **Fail**

Field Name: ***Version***

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	Version information field pops up in the appropriate colored box			

Field Name***: Community***

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	Community information field pops up in the appropriate colored box			

Field Name***: PDU Type***

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	PDU information field pops up in the appropriate colored box			

Field Name***: Request ID***

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	Request ID information field pops up in the appropriate colored box			

Field Name: ***Error Status***

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	Error status information field pops up in the appropriate colored box			

Screen: SNMP PDU

Field Name*: Object Id*

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	Object information field pops up in the appropriate colored box			

Field Name*: Value Integer*

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	Value information field pops up in the appropriate colored box			

Field Name*: Value ID*

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	Value ID information field pops up in the appropriate colored box			

***Field Name*: Object ID**

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	Object information field pops up in the appropriate colored box			

3.3.10 TELNET Testing Sheet

Screen: TELNET PDU

Date: _____

Tester: _____

Screen: **Pass** **Fail**

Field Name: ***Data***

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	Data information field pops up in the appropriately colored box			

3.3.11 SSH Testing Sheet

Screen: SSH PDU

Date: _____

Tester: _____

Screen: **Pass** **Fail**

Field Name: ***Data***

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	Data information field pops up in the appropriately colored box			

3.3.12 ARP Testing Sheet

Screen: ARP PDU

Date: _____

Tester: _____

Screen: **Pass** **Fail**

Field Name: ***Hardware Address Type***

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	Hardware Address Type information field pops up in the appropriately colored box			

Field Name*: Protocol Address Type*

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	Protocol Address Type information field pops up in the appropriately colored box			

Field Name: ***Hardware Address Length***

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	Hardware Address Length information field pops up in the appropriately colored box			

Field Name: ***Protocol Address Length***

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	Protocol Address Length information field pops up in the appropriately colored box			

Field Name: ***Operation***

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	Operation information field pops up in the appropriately colored box			

Screen: ARP PDU

Field Name: ***Sender Hardware Address***

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	Sender Hardware Address information field pops up in the appropriately colored box			

Field Name*: Protocol Address Type*

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	Protocol Address Type information field pops up in the appropriately colored box			

Field Name: ***Target Hardware Address***

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	Target Hardware Address information field pops up in the appropriately colored box			

Field Name: ***Target Protocol Address***

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	Target Protocol Address information field pops up in the appropriately colored box			

3.3.13 PING Testing Sheet

Screen: PING

Date: _____

Tester: _____

Screen: **Pass** **Fail**

Field Name: ***Destination***

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	Destination information field pops up in the appropriately colored box			

Field Name***: Source***

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	Source information field pops up in the appropriately colored box			

Field Name***: Fragment***

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	Fragment offset information field pops up in the appropriately colored box			

Field Name***: Time to Live***

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	Time to Live information field pops up in the appropriately colored box			

Field Name: ***Protocol***

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	Protocol information field pops up in the appropriately colored box			

Screen: PING

Field Name*: Header Checksum*

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	Header Checksum information field pops up in the appropriately colored box			

Field Name: *Source*

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	Source information field pops up in the appropriately colored box			

Field Name: *Destination*

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	Destination information field pops up in the appropriately colored box			

Field Name*: Checksum*

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	Checksum information field pops up in the appropriately colored box			

Field Name: *Identifier*

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	Identifier information field pops up in the appropriately colored box			

Screen: PING

Field Name: ***Sequence Number***

Attempted	**Expected Result**	**Comments**	**Pass**	**Fail**
1. Right Click	Nothing pops up			
2. Left Click	Sequence number information field pops up in the appropriately colored box			

3.3.14 HTTP Testing Sheet

Screen: HTTP

Date: _____

Tester: _____

Screen: **Pass** **Fail**

Field Name: ***Fragment offset***

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	Fragment offset information field pops up in the appropriate colored box			

Field Name: **Time to Live**

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	Time to Live field pops up in the appropriate colored box			

Field Name: **Protocol**

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	Protocol information field pops up in the appropriate colored box			

Field Name: **Header Checksum**

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	Header Checksum information field pops up in the appropriate colored box			

Field Name: ***Source***

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	Source field pops up in the appropriate colored box			

Screen: HTTP

Field Name: **Destination**

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	Destination information field pops up in the appropriate colored box			

Field Name: ***Header Length***

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	Header Length field pops up in the appropriate colored box			

Field Name: ***Window Size***

Attempted	Expected Result	Comments	Pass	Fail
1. Right Click	Nothing pops up			
2. Left Click	Window Size information field pops up in the appropriate colored box			

4.0 Detailed Design Specification

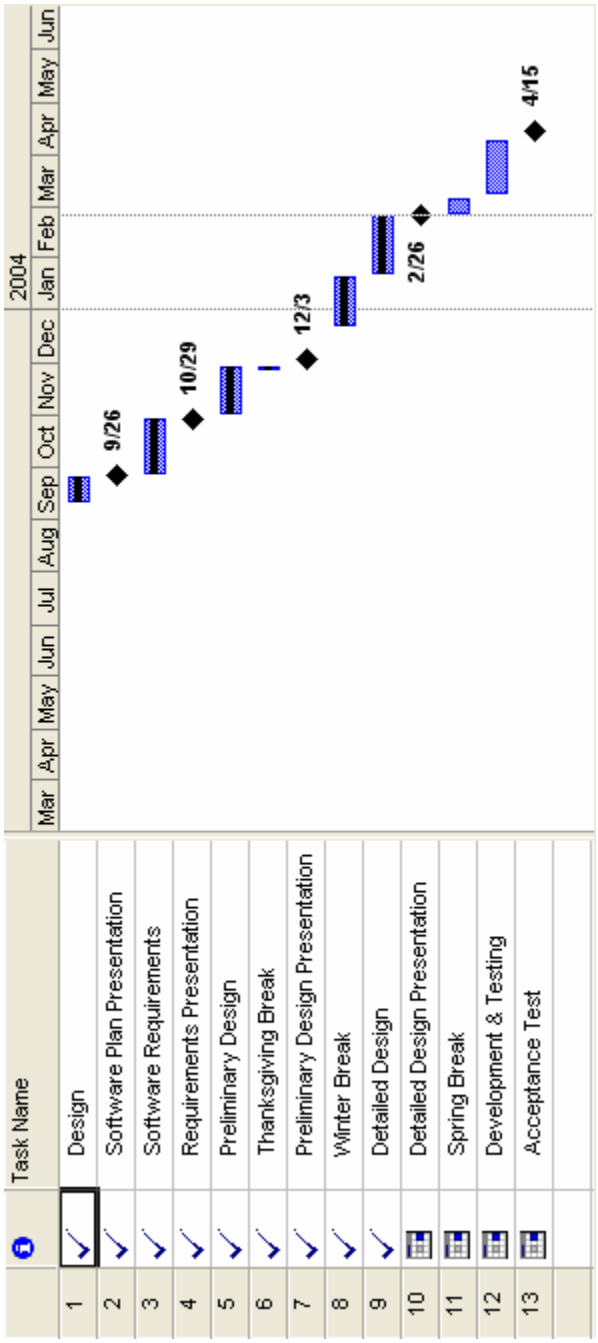
4.1 Packaging and Deployment Specifications

Mirage Incorporated will install its Packet Descriptor application and all necessary files on the Siena Computer Science network, Oraserv, in a password protected area, enabling only Siena students to access the program. An installation executable will be created and will be able to be run by the end user. The user will then be able to load all the necessary files into a user-configurable directory on the local hard drive and create a shortcut on the start menu for the application.

A CD-ROM, which will include the application, all documents, and all presentations, will be given to our client, Mr. Swarner, for back-up.

5.0 Appendix

5.1 Gantt Chart (Year)



5.2 Glossary

ASCII:

American Standard Code for Information Interchange: a code for representing English characters as numbers, with each letter assigned a number from 0 to 127.

Attribute:

A named value or relationship that exists for some or all instances of some entity and is directly associated with that instance.

Binary:

Pertaining to a number system that has just two unique digits, 0 and 1. Computers operate on a binary number system.

Code:

The symbolic arrangement of data or instructions in a computer program or the set of such instructions.

Data Flow Diagram:

A graphical notation used to describe how data flows between processes in a system. They are a representation of the functional decomposition of a system.

Decimal:

Refers to numbers in base 10—the numbers we use in everyday life.

Dynamic Combo Menu:

Menu showing all actions possible at the current moment.

Frame:

A feature that divides a browser's window into separate segments that can be scrolled independently of each other; a single step in a sequence of programmed instructions

GUI:

Graphical User Interface: A user interface based on graphics (icons, pictures, and menus) instead of text; uses a mouse as well as a keyboard as an input device.

Gantt Chart:

A chart that depicts progress in relation to time, often used in planning and tracking a project

Gray Box Testing: Testing procedure done with some knowledge of the internals.

HTML:

Hypertext Transfer Markup Language: A markup language used to structure text and multimedia documents and to set up hypertext links between documents, used extensively on the World Wide Web.

Hexadecimal:

Refers to the base-16 number system which consists of 16 unique symbols: the numbers 0 to 9 and the letters A to F.

Hypertext:

A computer-based text retrieval system that enables a user to access particular locations in web pages or other electronic documents by clicking on links within specific web pages or documents.

Internet:

An interconnected system of networks that connects computers around the world via the TCP/IP protocol.

Linear Sequential Model:

Sometimes called the *classic life cycle* or the *waterfall model*, this model suggests a systematic, sequential approach to software development that begins at the system level and progresses through analysis, design, coding, testing, and support.

Linux:

A trademark for an open-source version of the UNIX operating system.

Network:

A group of two or more computer systems linked together.

Open-Source:

A method and philosophy for software licensing and distribution designed to encourage use and improvement of software written by volunteers by ensuring that anyone can copy the source code.

PHP:

PHP Hypertext Preprocessor (server-side scripting language)

Packet:

A short block of data transmitted in a packet switching network.

PDU:

Protocol Data Unit: A packet of data passed across a network.

Protocol:

A set of formal rules describing how to transmit data, especially across a network.

Prototype:

An original type, form, or instance serving as a basis or standard for later stages.

RFC:

Request for Comments: One of a long-establish series of numbered Internet informational documents and standards widely followed by commercial software and freeware in the Internet and Unix communities.

Software:

The code executed by a computer, as opposed to the physical device which they run on.

TCP/IP:

Transmission Control Protocol/Internet Protocol: A suite of protocols for communication between computers, used as a standard for transmitting data over networks and as the basis for standard Internet protocols.

UNIX:

A powerful operating system developed at the ATT Bell Laboratories.

Use Case:

The specification of sequences of actions that a system, subsystem, or class can perform by interacting with outside actors.

Visible Analyst:

Project management software used in Computer-Aided Software Engineering (CASE) to create such illustrations as the data flow diagrams.