

# Essential IPAM

OSI, Addressing, Routing, and More

VOLUME 1

---

## TCP/IP Networking

This paper examines TCP/IP networking with an emphasis on protocol layers and the interactions of these layers. It is meant to be an introductory level paper.

# Table of Contents

<i>Section 1 – The Protocol Wars of the 90’s</i> .....	3
<i>Section 2 – Networking Models</i> .....	4
<i>Section 3 – IP Addressing</i> .....	8
<i>Section 4 – IP Packets and IP Routing</i> .....	11
<i>Section 5 – The Transport Layer, UDP and TCP</i> .....	13
<i>Section 6 – Review</i> .....	15
<i>SolarWinds IPAM</i> .....	16
<i>About SolarWinds</i> .....	17
<i>About the Author</i> .....	17

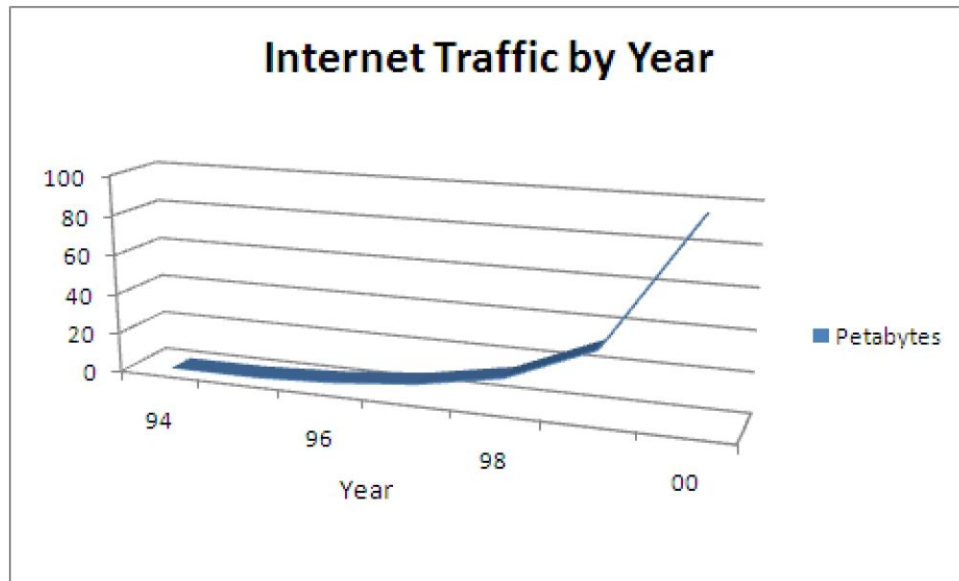
Copyright© 1995–2013 SolarWinds. All rights reserved worldwide. No part of this document may be reproduced by any means nor modified, decompiled, disassembled, published or distributed, in whole or in part, or translated to any electronic medium or other means without the written consent of SolarWinds. All right, title and interest in and to the software and documentation are and shall remain the exclusive property of SolarWinds and its licensors. SolarWinds Orion™, SolarWinds Cirrus™, and SolarWinds Toolset™ are trademarks of SolarWinds and SolarWinds.net® and the SolarWinds logo are registered trademarks of SolarWinds All other trademarks contained in this document and in the Software are the property of their respective owners.

SOLARWINDS DISCLAIMS ALL WARRANTIES, CONDITIONS OR OTHER TERMS, EXPRESS OR IMPLIED, STATUTORY OR OTHERWISE, ON SOFTWARE AND DOCUMENTATION FURNISHED HEREUNDER INCLUDING WITHOUT LIMITATION THE WARRANTIES OF DESIGN, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL SOLARWINDS, ITS SUPPLIERS OR ITS LICENSORS BE LIABLE FOR ANY DAMAGES, WHETHER ARISING IN TORT, CONTRACT OR ANY OTHER LEGAL THEORY EVEN IF SOLARWINDS HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

## SECTION 1

### The Protocol Wars of the 90's

The early 1990's were a difficult time for network engineers. Not only did engineers need to be fluent in TCP/IP and Ethernet but there was an equal need to be knowledgeable in Apple Talk, NetBIOS/NetBEUI, IPX/SPX (Novell), DECnet, ARCNET, Token ring, FDDI, and a plethora of other standards. It was common to see networks with mixed Token ring, Ethernet and FDDI moving traffic for DECnet, AppleTalk, IPX/SPX, and TCP/IP. Had it not been for the emergence of the "World Wide Web", the "Information Super Highway", and Fast Ethernet, we might be working today in a very different networking environment. The World Wide Web (WWW) and the Information Super Highway are really only different parts of what we call the Internet today. TCP/IP has been used from the inception of the Internet as a Department of Defense (DoD) and academic information exchange tool. In the mid-1990's use of the Internet began to expand into the general public as user-friendly tools like Internet browsers and GUI email became available. The subsequent popularity of Internet usage resulted to the first real increase in Internet traffic since inception. The chart below shows Internet usage in this timeframe as measured by Cisco Systems.



As you can see, the trend in traffic growth explodes starting about 1996. The growth rate from 2010 forward is expected to flatten out adding about 10 Petabytes a year. The increase in public use of the Internet was followed closely by the adoption of corporate Internet usage for low cost VPN tunneling, Web-based commerce and Web-based application providers. A great deal of the traffic that was once handled by private WAN links and snail mail/phone is now being sent over the Internet.

Because the Internet was built around TCP/IP, there was a lot of pressure to bring the IP stack to the desktop in order to minimize protocol translations and simplify network implementations and network trouble shooting. If the end user can send Internet data using TCP/IP, the data never needs to be translated to another protocol stack.

Novell added the TCP/IP stack to NetWare in 1996 and switched completely from their proprietary IPX/SPX to native TCP/IP in 1998. Microsoft entered the server market with a server that only supported NetBEUI. NetBEUI is a non-routable protocol, severely inhibiting adoption of the new server. Microsoft then released NT server 3.1 with TCP/IP. Other vendors followed suit and by about 1999 TCP/IP had won the protocol wars and was the native protocol stack for most equipment from then forward.

## SECTION 2

# Networking Models

The OSI model and the TCP/IP model are the prevalent methods to describe the interdependency of networking protocols. Both of these are conceptual models only and simply describe, not prescribe, how networking can be broken up into categories.

### The OSI Model

The International Organization for Standardization (IOS) is an internationally represented standards body founded in the late 1940's. The IOS created the Open Systems Interconnection (OSI) model to describe the very complex task of moving computer data from something you understand on a monitor, to bits on a wire, and back to readable data on a remote system. Although the OSI model is the most well-known ISO standard, the IOS is very active in other areas. Ever mounted an IOS image as a CD? This is an ISO standard too. The 7 layer OSI model is described below.

**Layer 1, The Physical Layer** describes the hardware used to move data from one machine to another. This includes cables, connectors, pin functions, voltage, and higher level functions such as bit level flow control and signaling (modulation/demodulation). Data in layer 1 is *bit* level data.

**Layer 2, The Data Link Layer** describes two primary functions, or sublayers. The lower sublayer, which communicates directly with the physical layer is the **Media Access Control (MAC) sublayer**. The MAC is responsible for supplying orderly access to the network media for the higher layers. This includes sending and receiving data to/from the physical layer, detecting errors from the physical layer and using MAC addressing at an operational level. The **Logical Link Control (LLC) sublayer** is responsible for MAC address control and forming the layer 2 frame by perpending the upper level protocol packet with the layer 2 frame header. Data in layer 2 is *frame* level data.

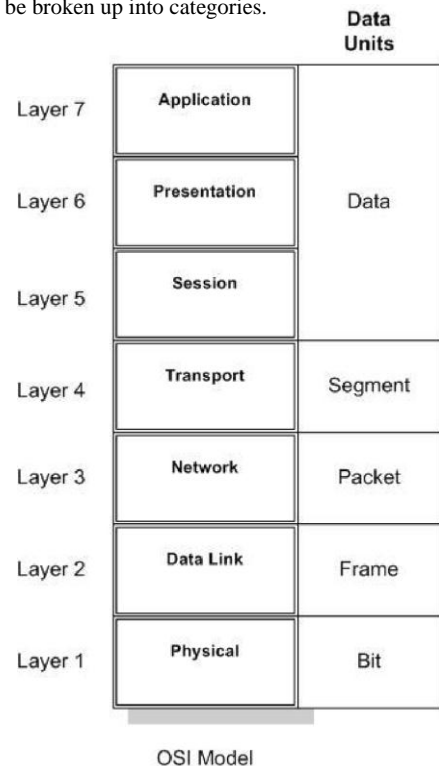
**Layer 3** is known as **The Network Layer**. The main responsibility of the network layer is to route traffic from one network to another using network level addressing. In TCP/IP networking, this is the job of IP. Routing traffic at level 3 includes maintaining and sharing logical routing tables and applying the rules of the appropriate routing algorithms. The network layer also enacts the QoS requested by the transport layer and fragments packets when they're too large to be transmitted by lower level protocols. Even though the network layer is responsible for upper layer requests, layer 3 communications are connectionless and therefore, unreliable. The OSI model allows for several NON-TCP/IP protocols at the level including IPX, AppleTalk, and DECnet. Data at layer 3 is *packet* level data.

**Layer 4, The Transport Layer**, allows for connection-oriented communications to other devices at layer 4 and reliable services to upper layer protocols. This layer can also take advantage of using connectionless transport when speed is preferred over reliability. Layer 4 offers both native and tunneling services to carry non TCP/IP data over a TCP/IP network. Layer 4 data units are called *segments*.

**Layer 5** is called **The Session Layer**. This layer watches over the communications between two devices and sets the rules for communication with other devices at layer 5. Layer 5 has the ability create, reset and terminate layer 5 sessions with other computers.

**Layer 6 is The Presentation Layer**. Data formatting is the main function of layer 6. This layer receives data from the session layer in various formats and reformats the data to a form that can be used by the application layer. When receiving data from the application layer, the presentation layer formats the data for use by lower layer protocols.

**The Application Layer** interacts directly with the user application, if that application implements a communication component, such as wireless networking or other LAN services. This involves synchronizing information exchange with the application and making the data available to the presentation layer.



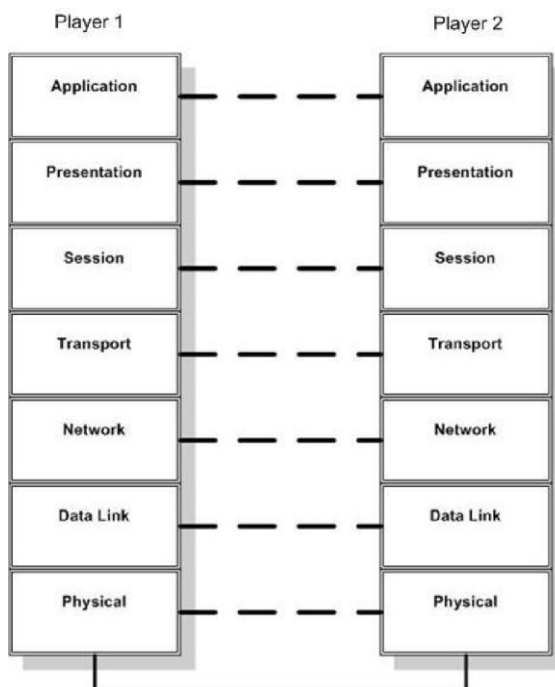
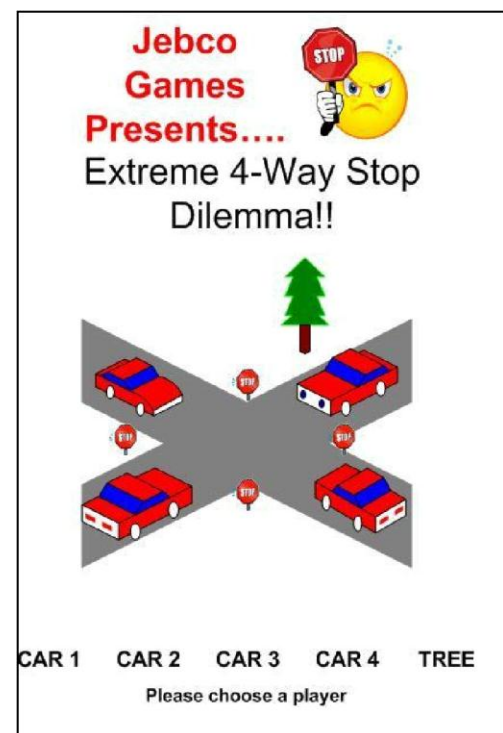
Each layer has two jobs to do:

1. Communicate to the peer system at the same level. In other words, computer 1 will send transport layer information to the transport layer of computer 2.
2. Pass information up and down the stack to the adjacent layers as required. Because there is no direct connection between machines at level 4, to get the level 4 message to the other machine, level 4 must hand the information down the stack and rely on lower layers to make sure it's delivered.

Since the only real connection between these devices is at the physical layer, the transport layer on computer 1 requests network level services to help move the transport segment to computer 2. Layer 3 on computer 1 accepts this segment, encapsulates the layer 4 segments in a layer 3 packet by adding the layer 3 header with IP addresses, and then passes it on to layer 2. Layer 2 accepts the layer 3 packet and appends it with the layer 2 header giving it a MAC address that can be used at the physical layer. The packet is then handed down to layer one for transmission on the media. On computer 2, the bits are received from the media and the information is sent up to layer 2. Each layer now makes the information ready for the layer above by stripping off the header for that layer and passing it up.

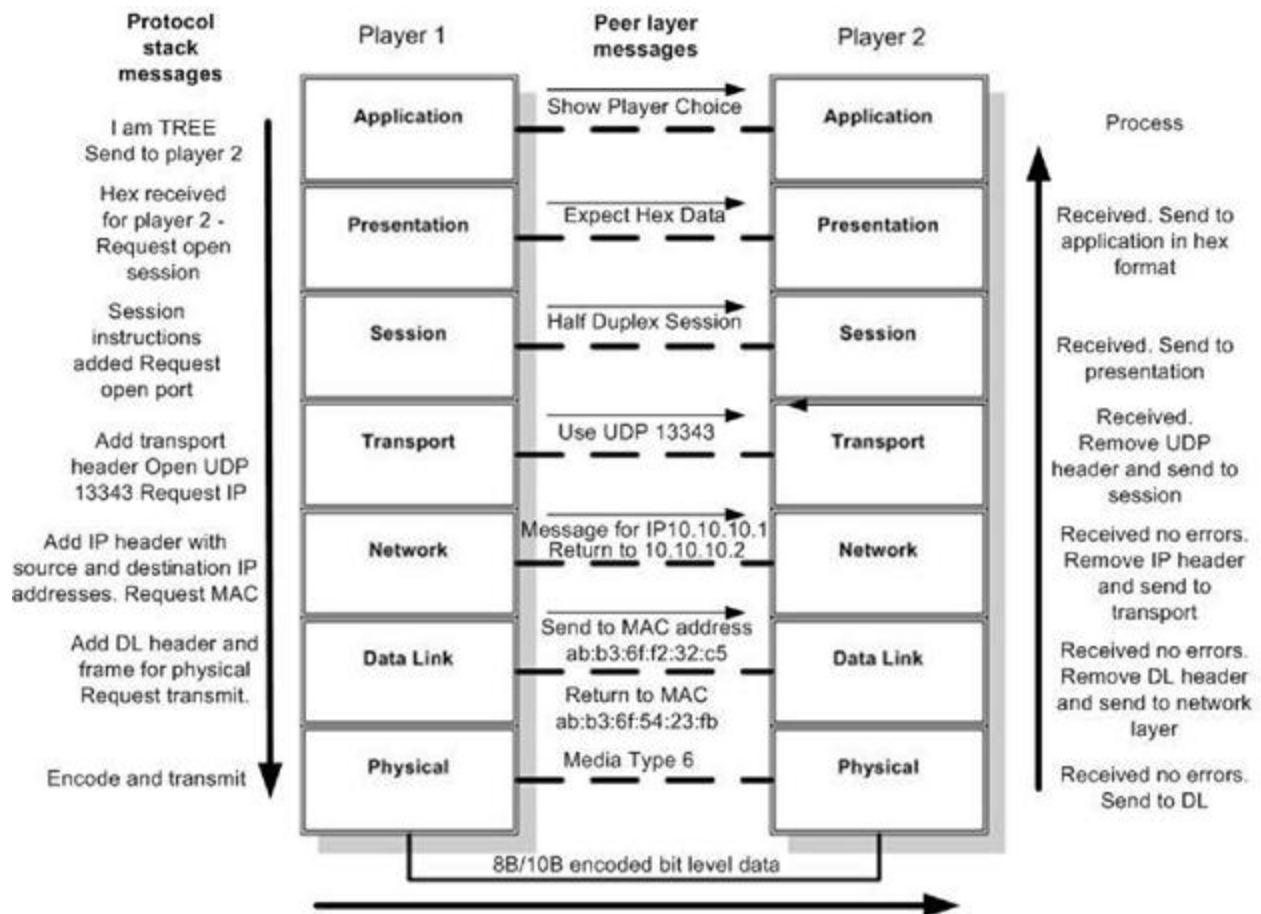
Below is a hypothetical example of this interaction for two connected gaming computers.

Imagine we have two folks playing the very popular video game *Extreme 4-Way Stop Dilemma* over the Internet. When player 1 chooses which character they want to be, the application needs to show player 2 that a car, or tree, has been chosen. As there is no direct connection to the remote player at the application layer, the application must rely on the protocols in the lower stacks to get the data to the physical layer where it can be transmitted. At the other end, the physical layer receives the bits and passes them up the stack until they are readable by the application. The application then informs player 2 that the worthy opponent has chosen "TREE" and the application marks the TREE choice as taken. Player 2 then chooses a car and the application shows player 1 that CAR 1 has been selected using the same methodology as above.



To the left is how the 7 layer OSI model might enable this communication.

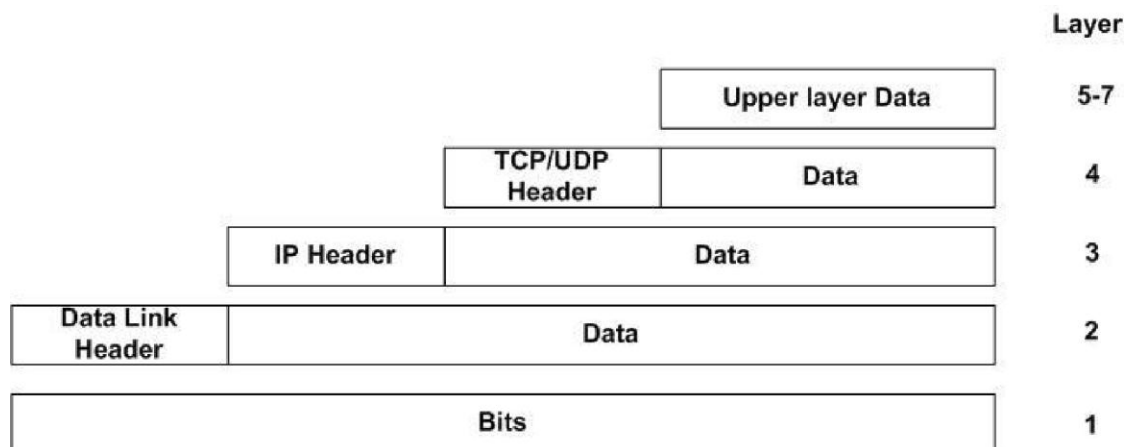
The dotted lines between same-level layers show that there are messages within each layer that are meant for the matching peer layer on the other end of the conversation. The only way to get those messages over to the peer layer is down the local stack, transmit, then up the remote stack. Here is what that might look like:



The selection of player 1 as TREE starts in the upper left. The application has the “show player 1 choice made” message now for the application layer on player 2’s machine. This avoids the unhappy chance that both players choose to be TREE, which severely limits the game’s action potential.

Again, this is a hypothetical example of how inter-layer communications might take place using the OSI model.

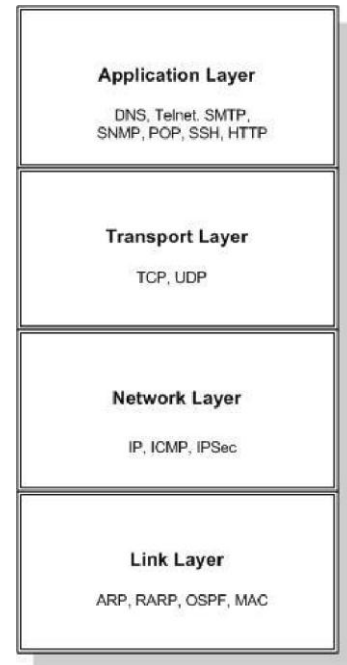
The way that lower layers accomplish providing services for upper layers is by adding protocol headers to all data from upper (higher) layers. This header addition is seen at the transport layer and below, so that is where I’ll focus. I’ll also assume TCP/IP will be used.



The data from upper layers relies upon each lower layer to accomplish its task in moving the data to where it can be transmitted as bits. As data moves down to lower layers, you can see that each layer considers all of the information, including upper layer headers as data only. Once all the data reaches layer 1, it is all bits. The opposite happens on the receiving side with each layer reading its header, taking the actions the header requests, stripping off the header, and passing to the next layer.

## The TCP/IP Model

This model was formed many years before the OSI model. As a result, the upper layer protocols were not well defined and all tasks above the transport layer were grouped together in the application layer. The TCP model does not address physical layer tasks typical to the OSI model, making it an entirely abstract model. Here we see a depiction of the TCP/IP model with some of the more popular protocols shown in their proper layer. Although the OSI model came along after the TCP/IP model to address some oversights and changes in technologies, the TCP model is used throughout networking and network troubleshooting tools. Why? From the transport layer down, there are actions a network engineer can take to remedy faulty communications. For example, if the engineer sees that there is a port conflict in TCP for a particular application, the engineer may be able to alter the transport port and restart the application on a non-conflicting port. On the other hand, if there is an application level error, there is not much the engineer can do other than work with the application vendor. Most network engineering does not involve OSI levels 5-7.



Now if we consider the case of *Extreme 4-Way Stop Dilemma* set to TPC/IP, everything would look identical except all interactions above the transport layer would simply be application layer. It should be noted that the TCP/IP model can be thought of as a supporting model for the more complex OSI model because the TCP/IP model only defines TCP/IP applications such as DNS and POP. These applications are normally used to support higher level applications such as email or HTTP browsing. Other than that, the layers of the TCP/IP model are analogous in function to the same name layers of the OSI model.

## SECTION 3

### IP Addressing

I have always thought of IP addressing as the meat and potatoes of TCP/IP. Without a good understanding of IP addressing, many aspects of TCP/IP networking will seem much more complicated than they need be. As the next volume in this series is entirely dedicated to IP addressing, the explanations in this section will cover the basics you need to know and will leave the details to the next paper.

#### IP Version 4 (IPv4) Addressing

IP addressing uniquely identifies a machine within a TCP/IP internet. (Notice the use of the lower case i to specify a generic TCP/IP network, not necessarily part of the network we all share worldwide, the "uppercase I Internet". IPv4 addresses are 32-bit designations, usually expressed in dotted decimal format such as 10.14.125.1. Although uncommon, the address can be displayed in hex or binary formats too. These are shown below for this sample address. Later, we'll see examples on why you would want to view IP address in different formats.

Dotted Decimal	Hex	Binary
10.14.125.1	0A:0E:7D:01	00001010.0001110.01111101.00000001

IPv4 addresses have two main fields, the network address field and the host address field IP addresses. IPv4 addresses fall into one of five classes, although only three classes are available for unique public use. For the most part, these addresses are ones that can be uniquely assigned to a machine on the Internet. These three classes are shown below.

Class	Applicable Networks	Number of Networks	Number of Address per Network
A	1.0.0.0 - 126.0.0.0	126	16,777,214+
B	128.0.0.0 - 191.255.0.0	16,384+	65,534
C	192.0.1.0 - 223.255.255.0	2,097,151	254

Each class has a range of usable address which are unique and registered with a registration authority. This eliminates the possibility of duplicate address between interconnected private entities.

The reasoning behind the segregation of addresses into classes is so that address blocks can be given out in a manner that is most economical for the requirement. Say for instance that AcmeCo requested a class A IP address block for its 3 offices and 65 staff members. Seeing that AcmeCo is such a small company, most all of the 16+ million addresses allocated to AcmeCo would go wasted. It would make more sense to give AcmeCo a single class C address space allowing them to have up to 254 devices directly on the Internet. So, AcmeCo might receive the sole usage of the registered network 199.12.21.0, which allows for connected devices on 191.12.2.1 to 191.12.21.254. You may ask, "What about the 199.12.21.0 and 199.12.21.255 addresses?" There are two rules governing the top most and lower most addresses in any network.

1. The top most address is reserved for broadcast on that network, an address machines can use to send a single message to all other machines on that network.
2. The lowest most address is reserved as the network address and is used for routing purposes.

So on the Acme network, the network address is 191.12.21.0 and the broadcast address is 191.12.21.255.



## Private IP Address Space

IPv4 addressing has special address ranges that are either allowed for special circumstances, or not allowed for use by the public. Here is a listing of those ranges:

Address Space	Reserved For	Allowed Use
10.0.0.0 – 10.255.255.255	Private Class A networks	Private
172.13.0.0 – 172.31.255.255	Private Class B networks	Private
192.168.0.0 – 192.168.255.255	Private Class C networks	Private
224.0.0.0 – 239.255.255.255	Multicast	Public
240.0.0.0 – 247.255.255.255	Reserved	None

The top three are called the *private IP address spaces*. These addresses may be used by anyone, as long as they're not used to connect directly to registered address spaces.

When IP addressing was first released, the 3.7 billion available unique addresses were assumed to be plenty for the future on the Internet. The problem today is that there are over 200 million registered addresses assigned each year, bringing the total used registered addresses to almost 2.7 billion. Without some help, we would run out of address space in the very near future. This is where the use on the above private addresses and something called Network Address Translation (NAT) comes in. Another method of expanding the usable IP address space was the creation of IPv6.

### Classless Internet Domain Routing (CIDR) Addresses and Subnetting

As just a quick note, addresses may be classful, as seen above, or be *classless*. Classless addresses do not strictly follow the rules of being a class A, B or C address, but rather a variable subnet mask is assigned to maximize IP address efficiency by creating subnetworks. This is done in a bit-wise fashion borrowing bits from the host address space and assigning them to the network address space. As an example, let's take a look at the network 10.10.21.0. This network could be implemented with a subnet mask of 255.255.255.0 (/24) and will allow one network with 254 usable addresses. Let's say we actually have four smaller networks that we want to use this address space for and we need to be able to route between the networks. Here is what this network address looks like in bit format with the subnet mask.

Network Address	00001010.00001010.00010101.00000000	10.10.21.0
Subnet Mask	11111111.11111111.11111111.00000000	255.255.255.0

By borrowing the 2 most significant (left most) bits from the host portion of the network address, we create more networks with fewer potential hosts.

New Subnet Mask	11111111.11111111.11111111.11000000	255.255.255.192
New Network Addresses	00001010.00001010.00010101.00000000	10.10.21.0
	00001010.00001010.00010101.01000000	10.10.21.64
	00001010.00001010.00010101.10000000	10.10.21.128
	00001010.00001010.00010101.11000000	10.10.21.192

Borrowing the 2 host bits gave us three additional usable networks (subnets) from the single /24 original network. But how many hosts can each subnet have? Seeing that we have 6 bits left over in each subnet, this leaves  $2^6$  or 64 hosts in each subnet. But there are 2 reserved addresses in all subnets:

- The lowest possible address for the subnet, also called the **zero subnet address** or **subnet address**. This address is used to identify the subnet for routing.
- The highest possible address in any subnet. This is the subnet **broadcast address**.

So, the subnets we have created above have the following addresses:

Subnet Address	Broadcast Address	Host Range
10.10.21.0	10.10.21.63	10.10.21.1 to 10.10.21.62
10.10.21.64	10.10.21.127	10.10.21.65 to 10.10.21.126
10.10.21.128	10.10.21.191	10.10.21.129 to 10.10.21.190
10.10.21.192	10.10.21.255	10.10.21.192 to 10.10.21.254

## Network Address Translation (NAT)

While subnetting is helpful in avoiding the waste of IP address space, NAT offers a unique solution to preserving IP address, the controlled reuse of IP addresses. The idea behind NAT is this: If we allow users to reuse a limited number of controlled IP address in completely distinct IP domains, we can have routers translate those addresses to unique, registered addresses when access to registered IP space is needed. Below is an example of how NAT can be applied.

- AcmeCo uses a 10.10.1.0 private network address. Because this address is part of the private IP range, it cannot be used on the public Internet. The router to the Internet has been set up with a pool of usable, registered public addresses it can use to communicate over the Internet when a privately addressed machine requests Internet services.
- The router sees the request to <http://cisco.com> from 10.10.1.55
- Using an available registered address of 211.12.13.14 the router marks that public address as being mapped to 10.10.1.5 on the inside and sends the request to [cisco.com](http://cisco.com) from the public address 211.12.13.14.
- When [cisco.com](http://cisco.com) responds to 211.12.13.14, the router remaps the packet to the internal address of 10.10.1.5 and sends the packet to the workstation.

In this way, if there are 500 employees at AcmeCo but only 6 need Internet access at a time, AcmeCo only needs to use 6 registered IP addresses. This is only an example of one type of NAT and there are many. With the use of NAT, the pressing need for more registered IP addresses has been somewhat relieved. This subject will be covered in further detail in Essential IPAM Vol. 2.

## IPv6

Along with some features not addressed in IPv4, IPv6 conquers the issue of diminishing IP address space by expanding the usable IP address range from 32-bits to 128-bits. The result is  $3.4 \times 10^{28}$  usable IP addresses. This is an extremely large number. Not surprising, IPv6 numbers look very large. Hex is the common format for showing numbers in the least amount of space, but even in hex numbers, IPv6 addresses are large. For example, here is a hypothetical IPv6 number in hex.

**2001:24c8:85d3:08de:3145:c82e:0371:1237**

There is a feature to shorten IPv6 numbers. Often the numbers will contain a series of zeros. These can be replaced by a double colon (::). For example:

**2001:24c8:0000:0000:3145:c82e:0000:1237**

Can be expressed as:

**2001:24c8::3145:c82e::1237**

As volume 2 of Essential IPAM is dedicated to IP Addressing, we will not further examine IP addressing here.

## Address Resolution Protocol (ARP) – Connecting layers 2 and 3

One of the protocols that tie layers together is ARP. ARP functions to tie logical IP addresses on local segments to the MAC address on that segment. When an IP packet reaches the final network segment and is ready for delivery from the gateway router to the end system, the gateway needs some way to know what MAC address to deliver it to. Since IP addresses don't contain any MAC address information, the gateway must map IP addresses to MAC addresses. This is accomplished by the gateway sending local broadcasts asking who has an IP address the gateway needs to deliver. The local system with that IP address tells the gateway it has the IP address of interest and also gives the gateway the local system's MAC address. The gateway can now deliver the data to the layer 2 MAC address. The gateway also stores this mapping of IP and MAC addresses in its ARP cache so that the gateway does not have to make new ARP requests for every packet.

## SECTION 4

### IP Packets and IP Routing

In volume 3 of Essential IPAM, we'll do a thorough review of IP Routing. For this reason, I will keep this section to a high-level explanation of routing. Again, the primary function of IP is the orderly routing of data to the proper network or subnetwork. Routers are required to move IP datagrams from one network to another if possible or drop the packet if routing is not possible. In order to route, the router requires a physical connection to two or more networks and a method of understanding where the datagrams needs to be sent to move towards to its final destination. This understanding comes in the form of routing tables. Routing tables are formed according to the routing protocol enabled on the routers. There are very simple routing protocols, used primarily for small networks and complex routing protocols for very large networks.

An IP workstation is configured with IP address and an IP gateway. The gateway is where the workstation is instructed to send all IP communications. Once the gateway receives the IP datagram, the gateway is responsible for sending it on its way to the destination IP address if possible, or dropping the packet. If the packet destination address is on the same segment as the sending device, the gateway sends the packet to the MAC address of the destination. If the IP destination is on an IP network in the router's routing table, the packet is sent out according to the rules of the routing protocol. If the packet is for an unknown destination, it's dropped. Once an IP datagram reaches its destination network, the router is responsible for delivering it to the proper local MAC address. This is accomplished by referring to the routers Address Resolution Protocol (ARP) table, which maps all local machines IP address to MAC addresses.

### The IP Datagram

The IP packet structure is fairly simple as it is designed for one main task—to move data from one network to another. IP is not responsible for establishing a connection with the remote system, reliability, acknowledging messages, reporting dropped packets, and other connection features. This is why IP is called connectionless. Below is the IP packet header format.

	0	4	8	16	19	31
0	Version	IHL	Type of Service	Length		
32	Identification			Flags	Fragment Offset	
64	Time to Live		Protocol	Header Checksum		
96	Source IP Address					
128	Destination IP Address					
160	Options				Padding	

Each field on the IP datagram has specific functions as described below:

- **Version** – 4-bits. Indicates the version of the IP header. Normally this will be set to v4.
- **IHL** – 4-bits. The Inter Header Length identifies where the IP header ends and the payload data begins.
- **Type of Service** – 8-bits, 6 used, 2 unused. This is used to be the 3 bit ToS which was expanded to allow for the 6 bit DSCP values requesting special or prioritized delivery.
- **Length** – 16-bits. The length of the entire IP datagram, including payload.
- **Identification** – 16-bits. Used to identify a datagram. Set by the sender to assist the receiver in reordering fragmented

datagrams.

- **Flags** – 3-bits. Used to control fragmentation. Defines 4 states
  - Fragmentation allowed
  - Fragmentation not allowed
  - More fragments to come
  - Last fragment
- **Fragment Offset** – 13-bits. Maps where this fragment fits in a series of fragments.
- **Time to Live** – 8-bits. Sets a maximum time the datagram can exist. Each system routing an IP datagram decrements this field. Once the value is zero, the packet is dropped. This keeps orphaned packets from endlessly circling a network.
- **Protocol** – 8-bits. Specifies the transport layer protocol used.
- **Header Checksum** – 16-bits. Contains the results of a checksum on the header only. If the receiving device does not obtain the same calculated value, the packet is dropped.
- **Source and Destination IP Addresses** – 32-bits each
- **Options** – Variable length. Provides several options beyond the scope of this document.
- **Padding** - Variable length. Used to pad the datagram to ensure the header ends on a 32-bit boundary

IP uses the fields above to send data in a connectionless fashion. IP does not care if the destination machine is available, it just sends data.

## IP Routing Algorithms

As an example of a simple routing protocol, we'll take a look at Routing Information Protocol (RIP). RIP uses what is called a **distance-vector routing** algorithm. RIP keeps track of the reachability of networks by advertising the networks each router knows of every thirty seconds. Router A on network 1 has network 1 in its RIP table, as network 1 is a directly connected network. Every thirty seconds, router A sends a broadcast on every directly connected network saying, "I have network 1 directly connected (hop count =0)." All routers connected to networks router A receive this message, update their routing tables and the broadcast every thirty seconds: "I know where network 1 is and it is 1 hop from me". This route update is sending the direction to get to network 1 as well as how far it is, thus distance-vector. A distance vector protocol only knows where to send a packet next, not the complexities of the whole network.

Other routing protocols keep what is essentially a complete map of the routing domain by exchanging network information between routers throughout the network and creating a map at each router. These are called **link-state routing** protocols. While distance-vector protocols can only determine the fewest hops to reach a destination, link state protocols can use other information such as fastest path or most reliable path. Open Shortest Path First (OSPF) is an example of a popular link-state protocol.

For very large networks like the Internet, the above two routing methods are too limited to make the complex decisions needed. Not only that, but when network changes occur, the routing systems have to converge, or agree upon a new set of routing data. When an IP network is converging, routing is severely hampered or stopped altogether. **Path-vector routing** such as Border Gateway Protocol (BGP) were created to solve this issue. Because BGP is able to work with autonomous systems, or autonomous routing sub environments, the issues that affect one autonomous system are isolated from impacting the whole internet. Without this, the Internet would be constantly reconverging and so impossible to use.

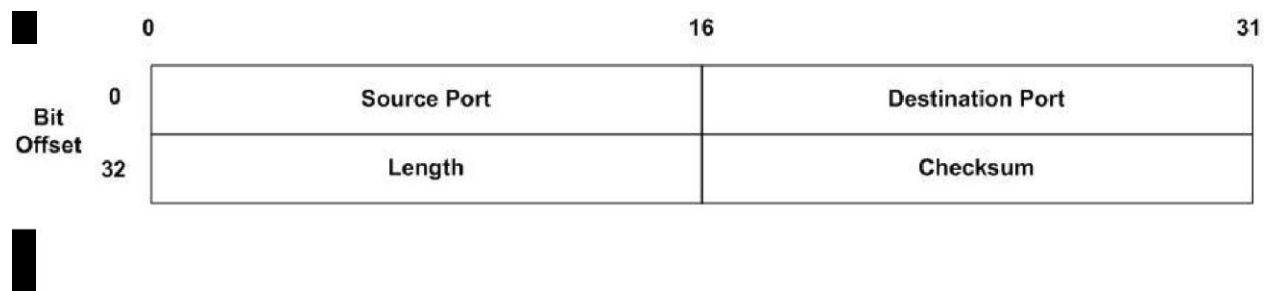
## SECTION 5

# The Transport Layer, UDP and TCP

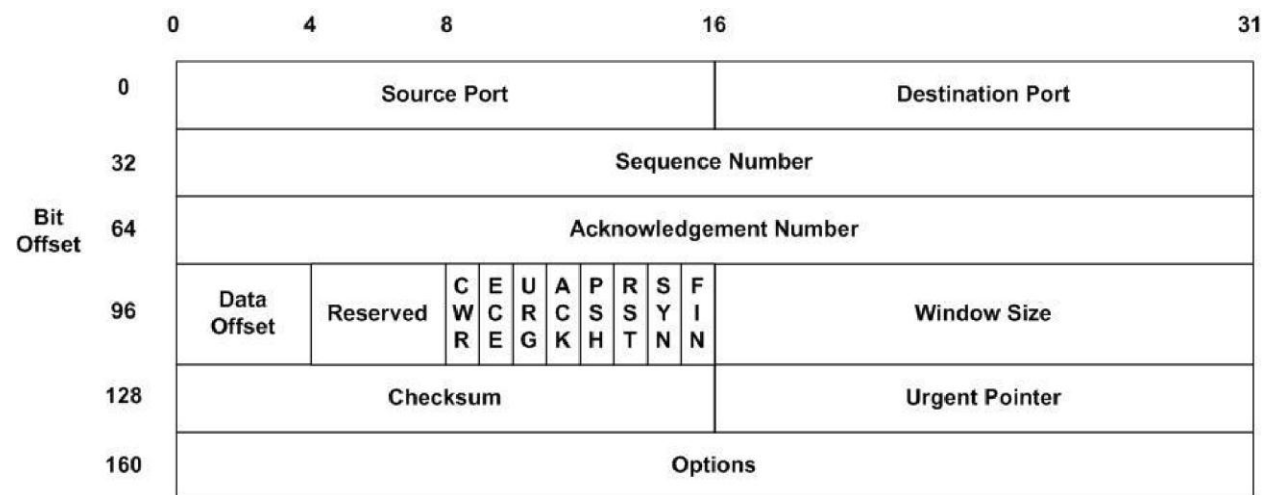
**UDP** is a transport layer protocol offering connectionless and unreliable transport of IP packets from one machine to another. Because UDP is not burdened with connection establishment or acknowledging delivery, it supplies faster communications compared to TCP. UDP operates by accepting data from the session layer requesting UDP services, assigning a UDP port to the data, and handing it down to IP for network layer service. UDP can detect errors but does nothing to correct them. UDP is very similar to IP as they are both connectionless and unreliable.

What the simple UDP header adds to IP is a UDP port number. This allows UDP/IP to multiplex conversations between hosts by separating the individual conversations with UDP ports. The 64-bit **UDP header** has only four fields as shown below.

- **Source and destination UDP ports** – 16 bits each. These indicate the UDP ports used.
- **Message Length** – 16-bits. Indicates the length of the entire datagram.
- **Checksum** – 16-bits. Computes a checksum on the entire datagram, including the payload.



Below is the **TCP header format**.



While UDP was designed for simplicity and speed, TCP was designed for reliability. This reliability all starts with a procedure known as the 3-way handshake. The handshake ensures two items critical to a connection oriented, reliable data exchange:

- Verify the two systems can reach each other
- Verify that the two systems can interpret data from each other

Here is how this is accomplished:

1. The system initiating the connection (A) sends a TCP datagram to the intended target machine

(B). A sets the SYN bit to 1 and inserts a random number in the sequence field.

2. B receives the connection request and accepts the connection by setting the ACK bit to 1, takes the sequence number and adds 1 to it and places this number in the acknowledgement field. B then inserts a random number into the sequence field and sends this all back to A.
3. A send acknowledges B's response and demonstrates that it can read data from B by adding 1 to the sequence number from B, placing that number in the acknowledgement field and placing a new random number in the sequence field.

Here is a description of the other TCP fields:

- **Source and Destination Ports** – 16-bits each. Assigned TCP ports mapped to known applications. Random high number ports may also be used when port/application identification mapping is not required.
- **Sequence Number** – 32-bits. Used to establish connections as described above when the SYN bit is 1. When the SYN bit is 0, this is the sequence number of the current data.
- **Acknowledgement** -Used in the connection establishment as described above.
- **Data Offset** – 4-bits. Indicates where the TCP header ends and payload data begins.
- **Reserved** – 4-bits. Always set to all zeros.
- **Control Bits** – 8 single bit fields.
  - **CWR** – Congestion Window Reduced. Used when both systems have the ability use explicit congestion control.
  - **ECE** – ECN echo. Also used in congestion control.
  - **URG** – When set to 1 this indicates that the urgent field is active.
  - **ACK** - When set to 1 this indicates that the acknowledgement field is active.
  - **PSH** – Push. When set to 1 this requests that data be delivered to the application immediately rather than buffered.
  - **RST** – An unrecoverable error has occurred in this connection. When set to 1, the connection is terminated and a new handshake procedure occurs.
  - **SYN** - Used in the connection establishment as described above.
  - **FIN** – End of data. Close connection.
- **Window Size** – 16-bit. Allow systems to communicate the maximum size of data it can handle per datagram.
- **Checksum** – 16-bit. Checks the integrity of the entire datagram.
- **Urgent Pointer** – 16-bit. Indicates the bit offset to urgent data. Valid only when the URG control bit is set.

TCP incorporates a variety of mechanisms that ensure reliable connections including:

**Flow control** – When a system no longer has buffer space it can ask the remote system to back off until the receiving system can process the buffered data. This is handled by the window field described above.

**Retransmission of lost data** - Because TCP tracks the delivery of all data and maintains an active connection, TCP can correct for lost data by retransmitting.

**Congestion Control** – TCP can implement several mechanisms to help avoid network congestion or at least to minimize the effect of congestion on TCP delivered data.

TCP achieves this level of connection quality at the cost of processing speed. For applications which require real-time data transfer like VoIP, UDP is a better choice.

I hope this has been helpful and I'll see you on [thwack](#) !

## SECTION 6

### Review

- Several protocol groups exist in the 1990's with TCP/IP winning out, probably due to the popularity of the Internet.
- Two network models are used today to describe network communications
  - OSI 7 layer model
  - TCP/IP 4 layer model
  - Protocols are responsible for communication with the peer system protocols at the same layer, using lower layers to pass the messages.
- IPv4 addresses are 32 bit in length, allowing for about 3.7 billion unique addresses
  - 2.7 billion IPv4 address are already in use
  - NAT and IP v6 have been put in place to alleviate the v4 address space limitations
  - Subnetting helps conserve address space to a limited extent
- IP routing is generally done by distance-vector, link-state, or algorithms
- Transport layer protocols most often used are
  - TCP – connection oriented and reliable
  - UDP – connectionless and fast

## SolarWinds IP Address Manager (IPAM)

Are you still using spreadsheets to manage your IP space? SolarWinds IP Address Manager (IPAM) enables you and your team to ditch your spreadsheets and switch to easy-to-use, centralized IP address management software. Now it's easier than ever to manage and monitor Microsoft® DHCP and DNS, as well as Cisco® DHCP servers, all from a single, intuitive Web console.

With SolarWinds IPAM, you can:

- Centrally manage, monitor, alert, & report on entire IP infrastructure
- Maintain Microsoft® DHCP/DNS & Cisco® DHCP services from a single Web interface
- Optimize IP space utilization & avoid IP conflicts via automatic scans & preventative alerts
- Deliver role-based access control along with detailed event recording & activity logs
- Gain critical insight into IP address space through real-time views & historical tracking

[Get more information on IP Address Manager here.](#)

[Evaluate IP Address Manager in a live demo environment here.](#)

[Download the fully functioning, 30-day evaluation here.](#)



## About SolarWinds

Founded in 1999, SolarWinds delivers powerful and affordable IT management and monitoring software to over 100,000 customers worldwide – from Global 1000 enterprises to small businesses. Named by Forbes as one of the top 10 fastest growing technology companies, SolarWinds is improving the way IT management software is developed, priced, purchased, delivered, and used.

Our IT monitoring and management software is built for SysAdmins and network engineers who need powerful, affordable, and easy-to-use solutions that save time and simply get the job done. Evaluating, buying, deploying, and using enterprise software shouldn't be complex. IT management solutions from SolarWinds are easy to try, buy, deploy and use. That's unexpected simplicity.

SolarWinds is headquartered in Austin, Texas, with sales and product development offices around the world. Join our online community of experts at [thwack.com](http://thwack.com)!

## About the Author

Andy McBride is a Technical Specialist for SolarWinds focusing on making knowledge of networking and network management accessible to customers and prospects of all levels. The “Essential IPAM” series is specifically written for an audience with limited prior exposure to these technologies. His technical background includes seven years at International Network Services (INS) as a Network Engineer and Managing Consultant, three years as a Novell Certified Instructor and five years as a Network Performance Products Manager with BT-Infonet. Prior to entering technology, Andy worked in aerospace on projects such as the SR-71, F-117, F-22, L-1011, F-18, and the space shuttle main engine and has a degree in Chemistry from California State University Northridge. Go Matadors!