Capítulo 5 & 6

A camada de Rede

(fim do capítulo 5 e parte do capitulo 6)

Provinha 5 11.03.2010

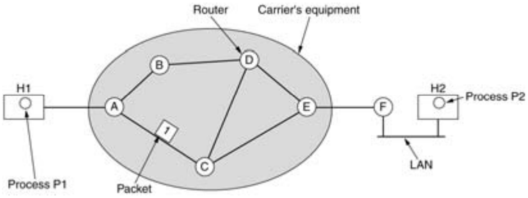
A rede da USP ocupa toda a faixa provida pelos endereços 143.107.X.X Imagine um esquema de divisão de endereços em que a distribuição é feita por unidades (FEA, POLI, ICMC, Direito). Cada unidade, então divide sua porção entre os seus departamentos (SCC, SSC, SMA, SME).

- mostre os esquemas de netmasking e dê alguns exemplos
- como seria o processo do roteamento de pacote enviado do Computer Lab (Cambridge) para o SSC (ICMC) nos seguintes pontos:
  - um roteador intermediário em Miami;
  - o roteador de borda da USP (CCE - são paulo)
  - o roteador de borda do ICMC
- indique as possíveis vantagens e desvantagens do esquema de endereçamento da Internet.

Network Layer Design Isues

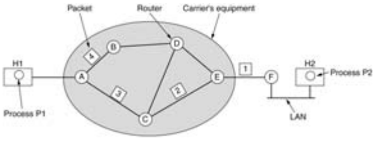
- Store-and-Forward Packet Switching
- Services Provided to the Transport Layer
- Implementation of Connectionless Service
- Implementation of Connection-Oriented Service
- Comparison of Virtual-Circuit and Datagram Subnets

Store-and-Forward Packet Switching



The environment of the network layer protocols.

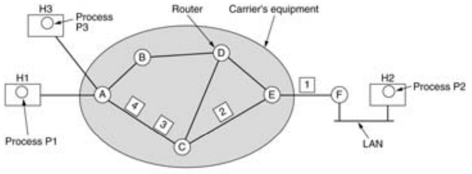
Implementation of Connectionless Service



A's table initially		A's table later		C's table		E's table	
A	–	A	–	A	A	A	C
B	B	B	B	B	A	B	D
C	C	C	C	C	–	C	C
D	B	D	B	D	D	D	D
E	C	E	B	E	E	E	–
F	C	F	B	F	E	F	F
Dest.	Line						

Routing within a diagram subnet.

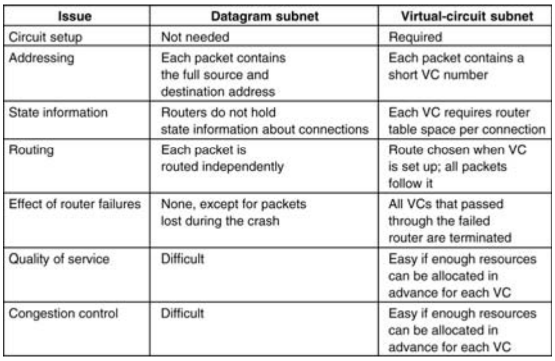
Implementation of Connection-Oriented Service



A's table In		A's table Out		C's table				E's table			
H1	1	C	1	A	1	E	1	C	1	F	1
H3	1	C	2	A	2	E	2	C	2	F	2

Routing within a virtual-circuit subnet.

Comparison of Virtual-Circuit and Datagram Subnets



Issue	Datagram subnet	Virtual-circuit subnet
Circuit setup	Not needed	Required
Addressing	Each packet contains the full source and destination address	Each packet contains a short VC number
State information	Routers do not hold state information about connections	Each VC requires router table space per connection
Routing	Each packet is routed independently	Route chosen when VC is set up; all packets follow it
Effect of router failures	None, except for packets lost during the crash	All VCs that passed through the failed router are terminated
Quality of service	Difficult	Easy if enough resources can be allocated in advance for each VC
Congestion control	Difficult	Easy if enough resources can be allocated in advance for each VC

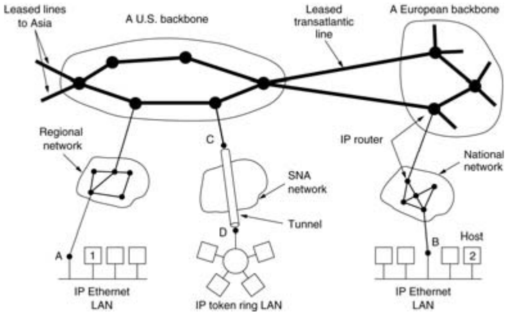
The Network Layer in the Internet

- The IP Protocol
- IP Addresses
- Internet Control Protocols
- OSPF – The Interior Gateway Routing Protocol
- BGP – The Exterior Gateway Routing Protocol
- Internet Multicasting
- Mobile IP
- IPv6

Design Principles for Internet

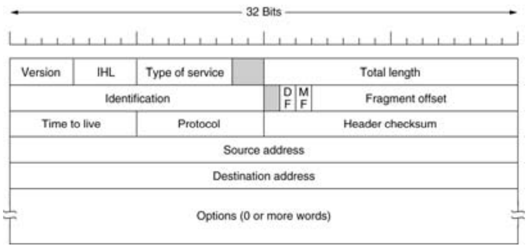
1. Make sure it works.
2. Keep it simple.
3. Make clear choices.
4. Exploit modularity.
5. Expect heterogeneity.
6. Avoid static options and parameters.
7. Look for a good design; it need not be perfect.
8. Be strict when sending and tolerant when receiving.
9. Think about scalability.
10. Consider performance and cost.

Collection of Subnetworks



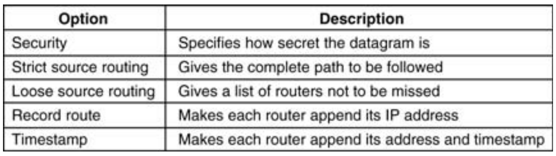
The Internet is an interconnected collection of many networks.

The IP Protocol



The IPv4 (Internet Protocol) header.

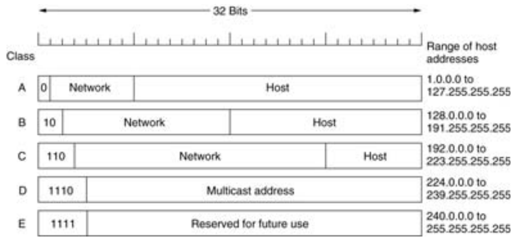
The IP Protocol (2)



Option	Description
Security	Specifies how secret the datagram is
Strict source routing	Gives the complete path to be followed
Loose source routing	Gives a list of routers not to be missed
Record route	Makes each router append its IP address
Timestamp	Makes each router append its address and timestamp

Some of the IP options.

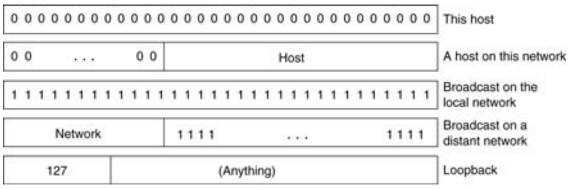
IP Addresses


32 Bits

Class				Range of host addresses
A	0	Network	Host	1.0.0.0 to 127.255.255.255
B	10	Network	Host	128.0.0.0 to 191.255.255.255
C	110	Network	Host	192.0.0.0 to 223.255.255.255
D	1110	Multicast address		224.0.0.0 to 239.255.255.255
E	1111	Reserved for future use		240.0.0.0 to 255.255.255.255

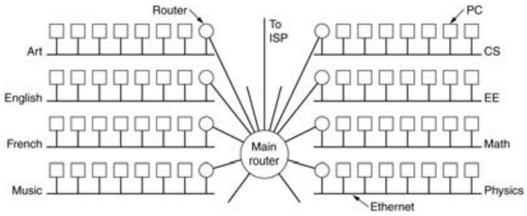
IP address formats.

IP Addresses (2)


0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	This host
0 0 . . . 0 0 | Host	A host on this network
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	Broadcast on the local network
Network | 1 1 1 1 . . . 1 1 1 1	Broadcast on a distant network
127 | (Anything)	Loopback

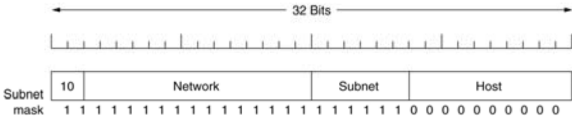
Special IP addresses.

Subnets



A campus network consisting of LANs for various departments.

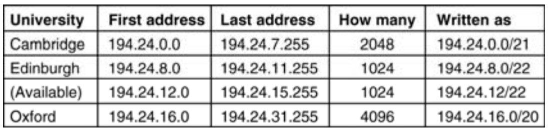
Subnets (2)


32 Bits

Subnet mask	10	Network	Subnet	Host
	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0			

A class B network subnetted into 64 subnets.

CDR – Classless InterDomain Routing



University	First address	Last address	How many	Written as
Cambridge	194.24.0.0	194.24.7.255	2048	194.24.0.0/21
Edinburgh	194.24.8.0	194.24.11.255	1024	194.24.8.0/22
(Available)	194.24.12.0	194.24.15.255	1024	194.24.12/22
Oxford	194.24.16.0	194.24.31.255	4096	194.24.16.0/20

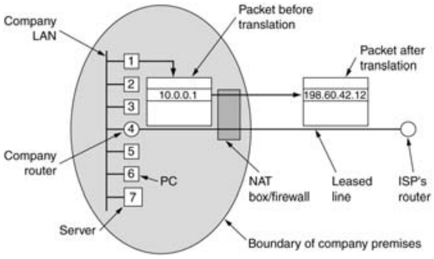
A set of IP address assignments.

Provinha 5 11.03.2010

A rede da USP ocupa toda a faixa provida pelos endereços 143.107.X.X Imagine um esquema de divisão de endereços em que a distribuição é feita por unidades (FEA, POLI, ICMC, Direito). Cada unidade, então divide sua porção entre os seus departamentos (SCC, SSC, SMA, SME).

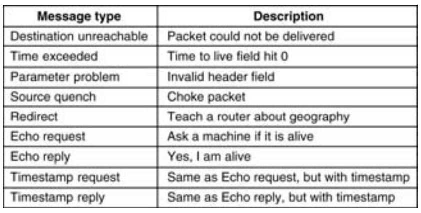
- mostre os esquemas de netmasking e dê alguns exemplos
- como seria o processo do roteamento de pacote enviado do Computer Lab (Cambridge) para o SSC (ICMC) nos seguintes pontos:
  - um roteador intermediário em Miami;
  - o roteador de borda da USP (CCE - são paulo)
  - o roteador de borda do ICMC
- indique as possíveis vantagens e desvantagens do esquema de endereçamento da Internet.

NAT – Network Address Translation



Placement and operation of a NAT box.

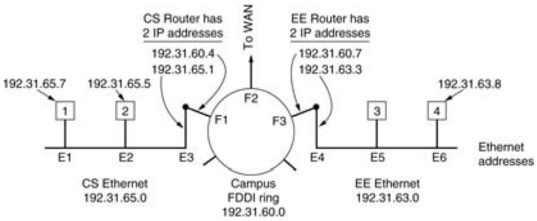
Internet Control Message Protocol



Message type	Description
Destination unreachable	Packet could not be delivered
Time exceeded	Time to live field hit 0
Parameter problem	Invalid header field
Source quench	Choke packet
Redirect	Teach a router about geography
Echo request	Ask a machine if it is alive
Echo reply	Yes, I am alive
Timestamp request	Same as Echo request, but with timestamp
Timestamp reply	Same as Echo reply, but with timestamp

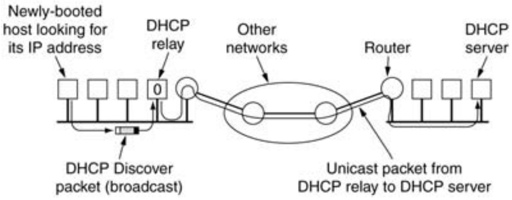
The principal ICMP message types.

ARP– The Address Resolution Protocol



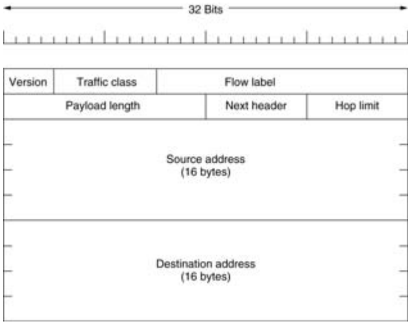
Three interconnected /24 networks: two Ethernets and an FDDI ring.

Dynamic Host Configuration Protocol



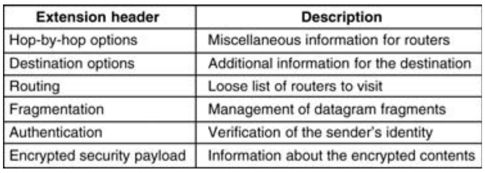
Operation of DHCP.

The Main IPv6 Header



The IPv6 fixed header (required).

Extension Headers



Extension header	Description
Hop-by-hop options	Miscellaneous information for routers
Destination options	Additional information for the destination
Routing	Loose list of routers to visit
Fragmentation	Management of datagram fragments
Authentication	Verification of the sender's identity
Encrypted security payload	Information about the encrypted contents

IPv6 extension headers.

Extension Headers (2)

Next header	0	194	4
Jumbo payload length			

The hop-by-hop extension header for large datagrams (jumbograms).

Extension Headers (3)

Next header	Header extension length	Routing type	Segments left
Type-specific data			

The extension header for routing.

Chapter 6

The Transport Layer

Provinha 23.03.2010

a) Porque são utilizados números de sequência aleatórios, para segmentos TCP?

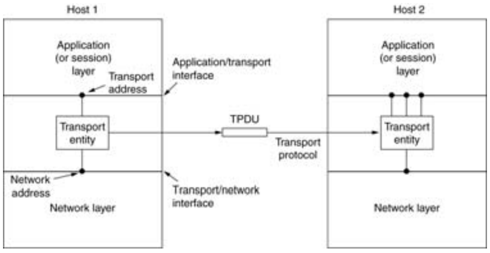
b) Como é feito o controle de fluxo, em conexões TCP? Porque controle de fluxo é importante?

c) Como funciona o processo de estabelecimento de conexões no TCP?

The Transport Service

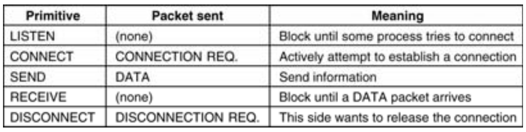
- Services Provided to the Upper Layers
- Transport Service Primitives
- Berkeley Sockets
- An Example of Socket Programming:
  - An Internet File Server

Services Provided to the Upper Layers



The network, transport, and application layers.

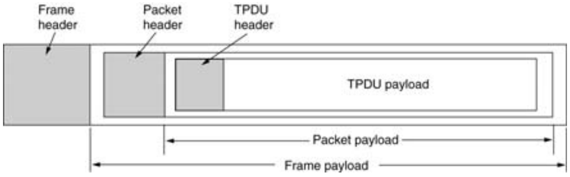
Transport Service Primitives



Primitive	Packet sent	Meaning
LISTEN	(none)	Block until some process tries to connect
CONNECT	CONNECTION REQ.	Actively attempt to establish a connection
SEND	DATA	Send information
RECEIVE	(none)	Block until a DATA packet arrives
DISCONNECT	DISCONNECTION REQ.	This side wants to release the connection

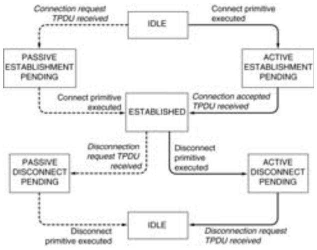
The primitives for a simple transport service.

Transport Service Primitives (2)



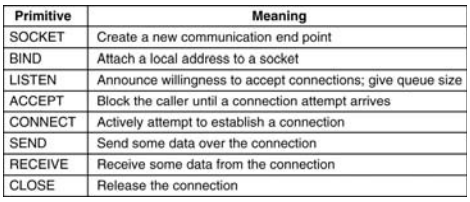
The nesting of TPDUs, packets, and frames.

Transport Service Primitives (3)


Connection request TPDU received
IDLE
Connect primitive executed
PASSIVE ESTABLISHMENT PENDING
ACTIVE ESTABLISHMENT PENDING
Connect primitive executed
ESTABLISHED
Connection accepted TPDU received
Disconnection request TPDU received
Disconnect primitive executed
PASSIVE DISCONNECT PENDING
ACTIVE DISCONNECT PENDING
IDLE
Disconnect primitive executed
Disconnection request TPDU received

A state diagram for a simple connection management scheme. Transitions labeled in italics are caused by packet arrivals. The solid lines show the client's state sequence. The dashed lines show the server's state sequence.

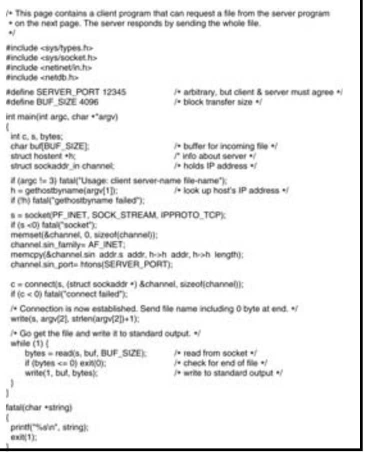
Berkeley Sockets



Primitive	Meaning
SOCKET	Create a new communication end point
BIND	Attach a local address to a socket
LISTEN	Announce willingness to accept connections; give queue size
ACCEPT	Block the caller until a connection attempt arrives
CONNECT	Actively attempt to establish a connection
SEND	Send some data over the connection
RECEIVE	Receive some data from the connection
CLOSE	Release the connection

The socket primitives for TCP.

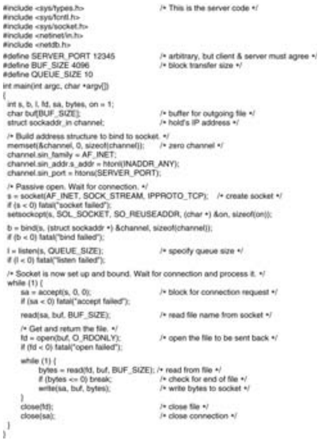
Socket Programming Example: Internet File Server



```
/* This page contains a client program that can request a file from the server program
 * on the next page. The server responds by sending the whole file.
 */

#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>

#define SERVER_PORT 12345                  /* arbitrary, but client & server must agree */
#define BUF_SIZE 4096                      /* block transfer size */

int main(int argc, char **argv)
{
  int c, s, bytes;
  char buf[BUF_SIZE];                      /* buffer for incoming file */
  struct hostent *h;                       /* info about server */
  struct sockaddr_in channel;              /* holds IP address */

  if (argc != 3) fatal("Usage: client server-name file-name");
  h = gethostbyname(argv[1]);              /* look up host's IP address */
  if (!h) fatal("gethostbyname failed");

  s = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP);
  if (s <0) fatal("socket");
  memset(&channel, 0, sizeof(channel));
  channel.sin_family= AF_INET;
  memcpy(&channel.sin_addr.s_addr, h->h_addr, h->h_length);
  channel.sin_port= htons(SERVER_PORT);

  c = connect(s, (struct sockaddr *) &channel, sizeof(channel));
  if (c < 0) fatal("connect failed");

  /* Connection is now established. Send file name including 0 byte at end. */
  write(s, argv[2], strlen(argv[2])+1);

  /* Go get the file and write it to standard output. */
  while (1) {
       bytes = read(s, buf, BUF_SIZE);     /* read from socket */
       if (bytes <= 0) exit(0);            /* check for end of file */
       write(1, buf, bytes);               /* write to standard output */
  }
}

fatal(char *string)
{
  printf("%s\n", string);
  exit(1);
}
```

Client code using sockets.

Socket Programming Example: Internet File Server (2)



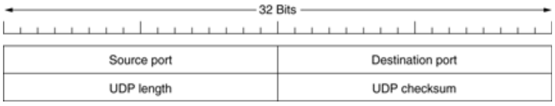
```
#include <sys/types.h>                     /* This is the server code */
#include <sys/fcntl.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>

#define SERVER_PORT 12345                  /* arbitrary, but client & server must agree */
#define BUF_SIZE 4096                      /* block transfer size */
#define QUEUE_SIZE 10

int main(int argc, char *argv[])
{
  int s, b, l, fd, sa, bytes, on = 1;
  char buf[BUF_SIZE];                      /* buffer for outgoing file */
  struct sockaddr_in channel;              /* hold's IP address */

  /* Build address structure to bind to socket. */
  memset(&channel, 0, sizeof(channel));    /* zero channel */
  channel.sin_family = AF_INET;
  channel.sin_addr.s_addr = htonl(INADDR_ANY);
  channel.sin_port = htons(SERVER_PORT);

  /* Passive open. Wait for connection. */
  s = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);    /* create socket */
  if (s < 0) fatal("socket failed");
  setsockopt(s, SOL_SOCKET, SO_REUSEADDR, (char *) &on, sizeof(on));

  b = bind(s, (struct sockaddr *) &channel, sizeof(channel));
  if (b < 0) fatal("bind failed");

  l = listen(s, QUEUE_SIZE);               /* specify queue size */
  if (l < 0) fatal("listen failed");

  /* Socket is now set up and bound. Wait for connection and process it. */
  while (1) {
       sa = accept(s, 0, 0);               /* block for connection request */
       if (sa < 0) fatal("accept failed");

       read(sa, buf, BUF_SIZE);            /* read file name from socket */

       /* Get and return the file. */
       fd = open(buf, O_RDONLY);           /* open the file to be sent back */
       if (fd < 0) fatal("open failed");

       while (1) {
            bytes = read(fd, buf, BUF_SIZE); /* read from file */
            if (bytes <= 0) break;         /* check for end of file */
            write(sa, buf, bytes);         /* write bytes to socket */
       }
       close(fd);                          /* close file */
       close(sa);                          /* close connection */
  }
}
```

Client code using sockets.

The Internet Transport Protocols: UDP

- Introduction to UDP
- Remote Procedure Call
- The Real-Time Transport Protocol

Introduction to UDP



The UDP header.

The Internet Transport Protocols: TCP

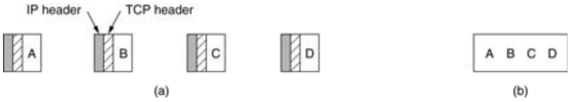
- Introduction to TCP
- The TCP Service Model
- The TCP Protocol
- The TCP Segment Header
- TCP Connection Establishment
- TCP Connection Release
- TCP Connection Management Modeling
- TCP Transmission Policy
- TCP Congestion Control
- TCP Timer Management
- Wireless TCP and UDP
- Transactional TCP

The TCP Service Model

Port	Protocol	Use
21	FTP	File transfer
23	Telnet	Remote login
25	SMTP	E-mail
69	TFTP	Trivial File Transfer Protocol
79	Finger	Lookup info about a user
80	HTTP	World Wide Web
110	POP-3	Remote e-mail access
119	NNTP	USENET news

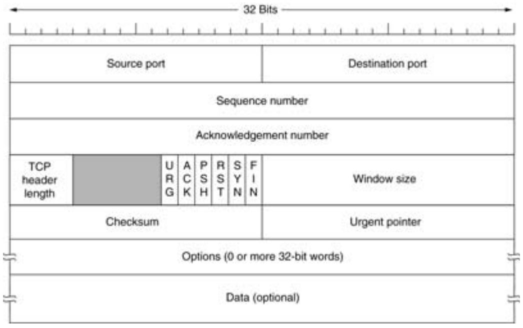
Some assigned ports.

The TCP Service Model (2)



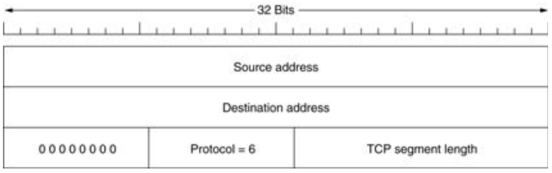
(a) Four 512-byte segments sent as separate IP datagrams.
(b) The 2048 bytes of data delivered to the application in a single READ CALL.

The TCP Segment Header



TCP Header.

The TCP Segment Header (2)



32 Bits

Source address		
Destination address		
0 0 0 0 0 0 0 0	Protocol = 6	TCP segment length

The pseudoheader included in the TCP checksum.

Provinha 23.03.2010

a) Porque são utilizados números de sequência aleatórios, para segmentos TCP?

b) Como é feito o controle de fluxo, em conexões TCP? Porque controle de fluxo é importante?

c) Como funciona o processo de estabelecimento de conexões no TCP?