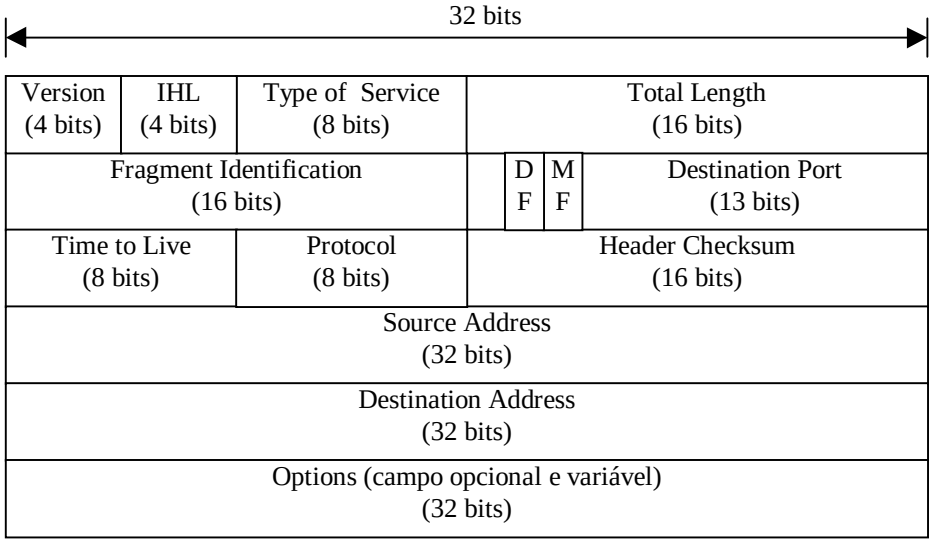
1. PRINCIPAIS PROTOCOLOS TCP/IP

1.1 IP - Internet Protocol – RFC 791

Esse protocolo foi introduzido na ARPANET no início dos anos 80, e tem sido utilizado juntamente com o TCP desde então. A principal característica desse protocolo é que a transmissão é efetuada sem a necessidade de uma conexão entre máquina fonte e máquina destino, sendo baseada no envio de datagramas que podem passar por muitas redes intermediárias até chegarem ao destino.

Um datagrama IP consiste de um cabeçalho e uma área de dados. O cabeçalho ocupa uma área fixa de 20 bytes e uma área de tamanho variável (correspondente ao campo *options*). A seguir é visto o formato desse cabeçalho.



- **Version**: 4 bits: contém a versão do protocolo IP que o datagrama pertence. Dessa forma, há a possibilidade de modificar o protocolo com a rede em funcionamento;
- **IHL**: 4 bits: indica o número de palavras de 32 bits que existem no cabeçalho. O valor mínimo é 5;
- **Type of service**: indica a qualidade de serviço que o datagrama deve ser enviado. Várias combinações de velocidade e segurança são possíveis. Para voz digitalizada, a velocidade é muito mais importante que segurança. Já para transferência de arquivos, uma transmissão com segurança é muito mais importante que a velocidade. Na prática esse campo não é usado (TAN 96 p. 472);
- **Total length**: indica o número total de bytes (octetos) existentes no datagrama (cabeçalho + dados). O tamanho máximo é 65.535 bytes;
- **Fragment ID**: todos os fragmentos de um datagrama contém o mesmo valor de identificação, que é utilizado para a máquina destino identificar o datagrama que pertence cada fragmento;

- **DF**: Don't Fragment - aviso para os roteadores não fragmentarem o datagrama, pois o destino não é capaz de remontá-los novamente.
- **MF**: More Fragments - todos os fragmentos, com exceção do último, devem possuir este bit ligado. Ele é utilizado juntamente com o campo total length para garantir que nenhum fragmento esteja faltando. Quando essa flag for 0, quer dizer que se trata do último fragmento (ou que o pacote não foi fragmentado).
- **Fragment offset**: indica a posição à qual pertence o fragmento atual. Todos fragmentos com exceção do último devem ser múltiplos de 8 bytes (unidade básica do fragmento). Como são utilizados 13 bits, há um máximo de 8192 fragmentos por datagrama, dando um tamanho máximo de 65536 bytes, coerentemente com o campo total length.
- **Time to live**: é um contador utilizado para limitar a vida dos pacotes. Quando ele chega a zero, o pacote é destruído. A unidade utilizada é segundo, e a máxima vida de um pacote é 255 segundos.
- **Protocol**: este campo identifica o protocolo da camada de transporte ao qual esse datagrama está associado, sendo que o mais comum é o TCP (0x06) ou UDP (0x11);
- **Header checksum**: faz o controle de erros do cabeçalho. Soma de todos bytes deve dar zero. É modificado a cada hop, pois pelo menos TTL se altera;
- **Source address**: indica o número da rede e número do host origem da mensagem;
- **Destination address**: indica o número da rede e número do host destino da mensagem;
- **Options**: é utilizado para controle de segurança, roteamento, relatório de erros, depuração, informação de hora, etc. Basicamente, foi proposto para permitir a evolução do protocolo através deste campo variável. Detalhes em TAN 96 p. 473

/**/ Mostrar fragmentação com o aplicativo PRAV Multicast

1.2 O protocolo TCP – RFC793

Entidades TCP trocam dados na forma de segmentos, que consiste num cabeçalho de 20 bytes (fixo) mais uma parte opcional, seguido de zero ou mais bytes de dados.

O software TCP decide qual deve ser o tamanho dos segmentos, limitado de duas formas:

- Tamanho máximo do segmento, de 65535 bytes, menos os cabeçalhos TCP e IP. (p. 597e 599).
- Tamanho da MTU (Maximum Transfer Unit) da rede. Se o segmento passar por uma rede com MTU menor, deve ser fragmentado, recebendo um novo cabeçalho TCP e IP (40 bytes a mais). P. 598 TAN. /**/

Originalmente, a ARPANET assumia que o nível de subnet oferecia um serviço totalmente confiável de transmissão de dados. Dessa forma, o primeiro protocolo de nível de transporte era bastante simples, e ficou conhecido como NCP (*Network Control Protocol*), sendo satisfatório para o ambiente inicial da ARPANET.

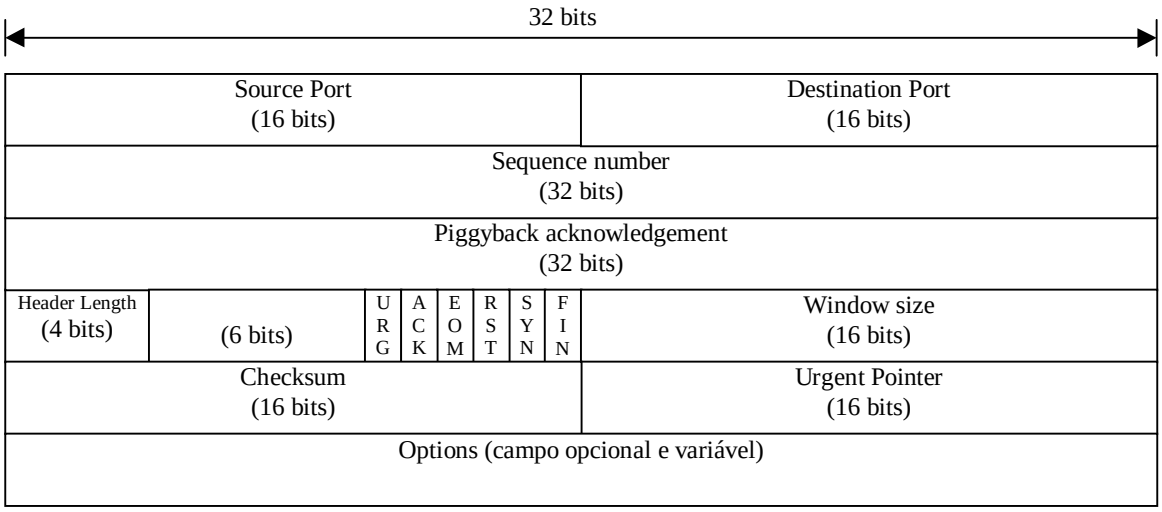
À medida que o tempo foi passando, a ARPANET cresceu e começou a incluir na sua estrutura muitas redes locais, algumas subredes baseadas em rádio, e também muitos canais via satélite, fazendo com que a confiabilidade entre a máquina fonte e máquina destino fosse diminuída. Isso forçou a criação de um novo protocolo de nível de transporte, que ficou conhecido como TCP (*Transfer Control Protocol*), e foi especificamente desenvolvido para tolerar uma subrede não confiável.

TCP é um protocolo de nível de transporte orientado à conexão, portanto, oferece uma comunicação confiável entre máquina fonte e destino, além de controle de fluxo e recuperação de erros, permitindo uma transmissão de dados *full-duplex*.

O processo de usuário pode enviar mensagens de qualquer tamanho, e o TCP divide essa mensagem em blocos iguais ou menores que 64 Kbytes, enviando cada peça separadamente. O nível de rede não oferece garantias que os pedaços vão ser entregues na mesma seqüência que foram transmitidos, portanto, o TCP deve prever uma forma de remontar os pedaços em ordem. Caso uma mensagem seja perdida, o TCP deve prever mecanismos de *time-out* para retransmitir o pedaço perdido.

1.2.1 Cabeçalho TCP

A figura a seguir mostra o formato do cabeçalho do TCP.



A primeira observação é que o cabeçalho mínimo do TCP é de 20 bytes. A seguir é feita uma descrição de cada campo existente no cabeçalho.

- **Source Port e Destination Port**: portas TCP (16 bits cada) através das quais será feita a conexão entre as entidades de transporte. A RFC 1700 especifica algumas portas padrão. Até 256 é reservado.

- **Sequence Number**: Todo byte tem um número de seqüência, portanto, pode ter um acknowledge relativo a ele. Permite a remontagem dos dados no destino.
- **Acknowledge number**: especifica o próximo byte aguardado pelo destino, informando também que até o byte anterior foi recebido corretamente. Só é válido se o bit de ACK do campo *flags* estiver setado.
- **TCP Header Length**: número de palavras de 32 bits do cabeçalho TCP. Essa informação é necessária pois o campo *options* é variável.
- Seis bits não utilizados
- **Flags**:

1. URG: indica que o campo Urgent Pointer é válido
2. ACK: indica que o campo Acknowledge number é válido
3. PSH: PUSH – o receptor é solicitado a entregar os dados à aplicação imediatamente, sem esperar que o buffer fique completo.
4. RST: Usado para reiniciar uma conexão que ficou confusa por uma falha no host ou por qualquer outra razão. Também usado para rejeitar uma tentativa de conexão ou rejeitar um segmento inválido.
5. SYN: usado para estabelecer a conexão. Primeiro segmento deve ter SYN=1 e ACK=0 (solicitação de conexão e informação do número de seqüência – CONNECTION REQUEST). A resposta contém uma confirmação (ver three way handshake), tendo SYN=1 e ACK=1 (CONNECTION ACCEPTED).
6. FIN: usado para encerrar a conexão, dizendo que o transmissor não tem mais dados a enviar.
7. **Window Size**: Efetua controle de fluxo através de uma janela deslizante de tamanho variável. Esse campo indica quantos bytes podem ser enviados a partir do byte confirmado (Acknowledge number). O valor de 0 é válido, e serve para indicar que os bytes até Acknowledge number –1 foram recebidos bem, mas o receptor quer um descanso (o transmissor deve parar de enviar dados). Quando o receptor estiver mais folgado, envia outro segmento com o mesmo campo acknowledge number e um byte de window diferente de zero.

- **Checksum:** soma de todos bytes mais checksum deve dar zero.
- ***Urgent Pointer:*** indica um *offset* do número de bytes a partir do número de seqüência atual no qual existem dados urgentes;
- ***Options:*** possui diversas funções, como por exemplo comunicar tamanho de buffer durante o período de *setup*. Maiores detalhes em TAN 96 p. 601.

1.2.2 Three Way Handshake (RFC 793)

1) A --> B SYN my sequence number is X
2) A <-- B ACK your sequence number is X
3) A <-- B SYN my sequence number is Y
4) A --> B ACK your sequence number is Y

Como os passos 2 e 3 podem ser feitos juntos, o reconhecimento é feito em três etapas. Ele é necessário pois cada entidade pode ter um número de seqüência diferente, e os dois devem saber qual o número do outro.

1.2.3 Timeout no TCP

É adaptativo, ou seja, o host tenta estabelecer a conexão (SYN no IP e porta destino). Se não obtiver resposta (SYN ACK) em 3s, tenta 2ª vez, aumentando timeout para 6s. Se não vier, dobra timeout (12s), tentando 3ª vez. Se não vier resposta, tenta novamente. Se não vier resposta, dá erro dizendo que conexão falhou.

Durante a conexão é similar, se adaptando ao tempo que a resposta está demorando para chegar.

/**/ telnet para ip inválido e análise no NetXRay.

1.3 UDP - User Datagram Protocol – RFC 768

O UDP é um protocolo de nível de transporte orientado à transmissão de mensagens sem o estabelecimento de uma conexão entre máquina fonte e destino, fornecendo uma comunicação menos confiável que o TCP. Ele envia as mensagens (sem conexão) e não oferece nenhuma garantia de entrega ou seqüência. O formato do cabeçalho do UDP é mostrado na figura a seguir.

Source Port (16 bits)	Destination Port (16 bits)
Length (16 bits)	Checksum (16 bits)

O campo *length* dá o tamanho do cabeçalho mais o campo de dados. O campo de *checksum* também é para o cabeçalho mais dados.

1.3.1 Exemplos de usos do UDP

1. Protocolo TFTP (Trivial FTP) - /* link para servidor de TFTP */
2. Transmissão de voz e vídeo pela rede
3. Transmissão em multicast (normalmente é não confiável, apesar de existirem protocolos multicast confiáveis.

1.4 Ipv6 – RFCs 1883 e 1884

As principais alterações em relação ao Ipv4 são:

- Ampliação de 32 bits para 128 bits
- Término das classes de endereços
- Fim da fragmentação no cabeçalho

1.4.1 Formas de representação dos endereços Ipv6:

1. x:x:x:x:x:x:x:x, onde os "x" são números hexadecimais, ou seja, o endereço é dividido em oito partes de 16 bits, com por exemplo: 1080:0:0:0:8:800:200C:417A
2. Utilizando a notação :: para substituir uma seqüência de zeros (uma única vez no endereço). Por exemplo:
   Unicast: 1080:0:0:0:8:800:200C:417A. - > 1080::8:800:200C:417^{A}
   Multicast: FF01:0:0:0:0:0:0:43 -> FF01::43
   Loopback: 0:0:0:0:0:0:0:1 -> ::1
   Unspecified: 0:0:0:0:0:0:0:0 -> ::
3. Para usar na transição de IPv4 e IPv6: x:x:x:x:x:x:d:d:d:d, onde os "x" são números hexadecimais (16 bits) e os "d" são valores decimais de 8 bits, referentes à representação padrão do IPv4. Por exemplo:
   0:0:0:0:0:0:192.168.20.30, ou, na forma abreviada: ::192.168.20.30

1.4.2 Número de endereços Ipv6

www.rnp.br/newsgen (Frank Ned)

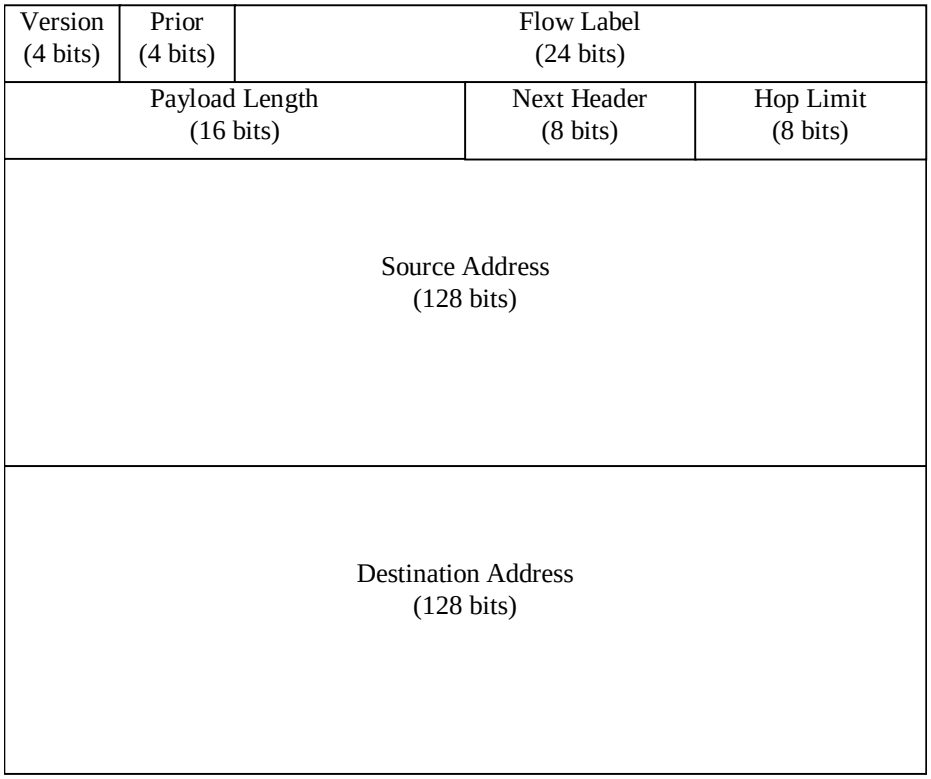
O IPng suporta endereços com número de bits 4 vezes maior que o endereço IPv4 (128 - [16 bytes] vs. 32 - [4 bytes]). Isto significa que o IPv6 é 4 bilhões (2^{96}) de vezes maior que o IPv4 (2^{32}), ou seja, a quantidade de endereços é de:

340.282.366.920.938.463.463.374.607.431.768.211.456

Esta é uma faixa de endereçamento extremamente grande. Teoricamente, isto representa aproximadamente 665.570.793.348.866.943.898.599 endereços por metro quadrado da superfície do nosso planeta (assumindo que a superfície da Terra seja de 511.263.971.197.990 m^2).

Em termos mais práticos, a tarefa de distribuição e roteamento de endereços requer a criação de hierarquias eficientes para o uso dos endereços. Christian Huitema fez uma análise na qual avaliou a eficiência de outras arquiteturas de endereçamento (inclusive o sistema telefônico francês, o sistema telefônico dos E.U.A, a Internet atual que usa IPv4 e nós IEEE 802). Ele concluiu que o enderçamanto de 128 bits pode acomodar entre $8x10^{17}$ a $2x10^{33}$ nós, assumindo a eficiência de blocos semelhantes em outras arquiteturas de endereçamento. Até mesmo a sua estimativa mais pessimista assume que seria possível ter 1.564 endereços para cada metro quadrado da superfície do planeta Terra. A estimativa otimista permitiria 3.911.873.538.269.506.102 de endereços para cada metro quadrado.

1.4.3 Formato do cabeçalho



Version - 4 bits com a versão do IP utilizado (6).
Prior - Diz o nível de prioridade (4 bits). Permite que uma origem especifique a prioridade de entrega para determinados pacotes em relação a outros pacotes da mesma origem.
Flow Label - Campo de 24 bits, relacionado com a qualidade de serviço do pacote.
Payload Length - Inteiro sem sinal (16 bits). Tamanho do *payload*, isto é, o resto do pacote que segue o cabeçalho IPng em octetos.
Next Header - Campo de 8 bits. Identifica o tipo de cabeçalho que segue o cabeçalho IPng. Usar o mesmo valor do protocolo IPv4.
Hop Limit - Inteiro sem sinal (8 bits). Decrementado de 1 a cada *node* que passa o pacote. O pacote é descartado caso o *hop limit* seja igual a zero. Semelhante ao TTL no Ipv4.
Source Address e **Destination Address** - Campos de 128 bits com os endereços fonte e destino do pacote.