BCC361 – Redes de Computadores
Universidade Federal de Ouro Preto
Departamento de Ciência da Computação

Prof. Reinaldo Silva Fortes
www.decom.ufop.br/reinaldo
2012/01

Camada	Nome
5	Aplicação
4	Transporte
3	Rede
2	Enlace
1	Física

Camada de Transporte

Agenda

- O serviço de transporte;
- Elementos dos protocolos de transporte;
- Protocolo UDP;
- Protocolo TCP;
- Implementação de sockets.

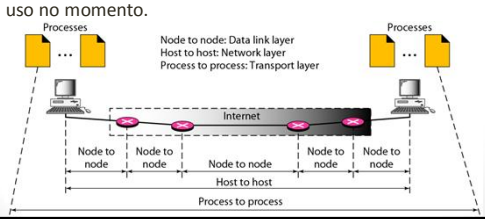
O SERVIÇO DE TRANSPORTE

Tópicos

- Serviços oferecidos às camadas superiores;
- Primitivas de serviços de transporte.

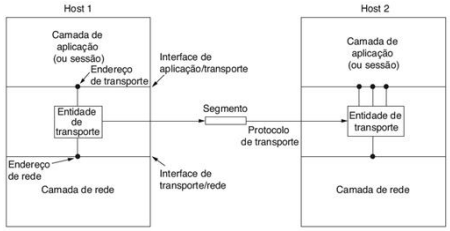
Serviços oferecidos...

- A camada de transporte se baseia na camada de rede para oferecer transporte de dados de um ***processo*** *em uma máquina de origem* para um **processo** em uma *máquina de destino*;
- Este transporte de dados deve ser feito com um nível de confiabilidade desejado, independente das redes físicas em uso no momento.



Serviços oferecidos...

- Relacionamento lógico entre as camadas de rede, transporte e aplicação:



- **Entidade de transporte**: hardware e software que executa o "trabalho";
- **Segmento**: mensagens enviadas entre duas entidades de transporte;
  - Também denominado de **TPDU** (***Transport Protocol Data Unit***).

Serviços oferecidos...

- Existem dois tipos de serviços oferecidos:
  - Orientado a conexões;
  - Não orientado a conexões;
- Ambos são semelhantes aos serviços oferecidos pela camada de redes;
- Diante disto, pergunta-se:
  - Por que há duas camadas distintas?
  - Uma única camada não seria suficiente?

Serviços oferecidos...

- O código de transporte funciona inteiramente nas máquinas dos usuários;
  - A camada de rede funciona principalmente nos roteadores, que na maioria dos casos está sob a responsabilidade das concessionárias de comunicação;
- A camada de transporte imuniza as camadas superiores da tecnologia, projeto e imperfeições de rede;
  - Primitivas de serviços podem ser implementadas como chamadas de procedimentos em bibliotecas, tornando-as independentes da rede (primitivas-padrão);
  - Rede (não confiável) *v.s.* Transporte (confiável).

Primitivas de serviços...

- Focaremos neste momento no serviço orientado a conexão;
- Função da camada de transporte: oferecer um serviço confiável sobre uma rede não confiável;
- Muitas das aplicações (seus programadores) farão uso da camada de transporte para comunicação, por isso, o serviço de transporte deve ser adequado e fácil de usar;
- A seguir veremos um conjunto simples de primitivas e um diagrama de estados para ilustrar o funcionamento do serviço da camada de transporte.

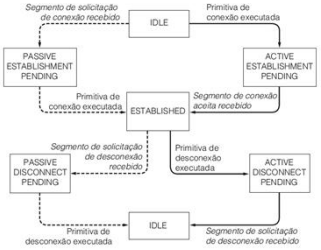
Primitivas de serviços...

- Primitivas para um serviço de transporte simples:

Primitiva	Pacote enviado	Significado
LISTEN	(nenhum)	Bloqueia até algum processo tentar se conectar.
CONNECT	CONNECTION REQ.	Tenta ativamente estabelecer uma conexão.
SEND	DATA	Envia informação.
RECEIVE	(nenhum)	Bloqueia até que um pacote de dados chegue.
DISCONECT	DISCONNECT REQ.	Solicita uma liberação da conexão.

Primitivas de serviços...

- Diagrama de estados:



- As transições marcadas em itálico são causadas pelos pacotes de chegada;
- As linhas sólidas mostram a sequência de estados do cliente;
- As linhas tracejadas mostram a sequência de estados do servidor.

O serviço de transporte;
Elementos dos protocolos de transporte;
Protocolo UDP;
Protocolo TCP;
Implementação de sockets.

ELEMENTOS DOS PROTOCOLOS DE TRANSPORTE

Tópicos

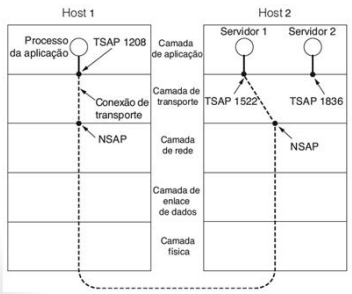
- Endereçamento;
- Estabelecimento de conexões;
- Encerramento de conexões;
- Controle de erro e fluxo;
- Multiplexação;
- Recuperação de falhas.

Endereçamento

- Para estabelecer uma conexão ou enviar uma mensagem é necessário que um processo da aplicação do cliente saiba como especificar a aplicação remota;
- Na camada de transporte isso é feito a partir de **portas**, cujo termo genérico é **TSAP** (***Transport Service Access Point***);
  - Na camada de rede é usado o termo **NSAP** (***Network Service Access Point***);

Endereçamento

- Possível cenário para uma conexão de transporte:



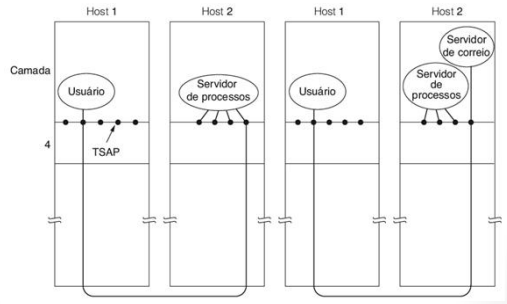
1. Um processo servidor de correio no host 2 se associa ao TSAP 1522;
2. Um processo de aplicação no *host* 1 transmite uma solicitação CONNECT especificando como origem o TSAP 1208 e como destino o TSAP 1522 do *host* 2;
3. O processo da aplicação envia a mensagem de correio;
4. O servidor de correio responde que entregará a mensagem;
5. A conexão é encerrada.

Endereçamento

- Como descobrir o endereço TSAP de um servidor remoto?
  - Endereçamento fixo;
  - ***Portmapper***: um processo especial que gerencia o mapeamento de serviços (nome) a portas (número);
  - Ambos possuem endereços estáticos;
- Ocupar portas permanentemente para serviços que normalmente são pouco utilizados é um desperdício;
- Solução: utilizar o **protocolo de conexão inicial**.

Endereçamento

- Funcionamento do **protocolo de conexão inicial**:

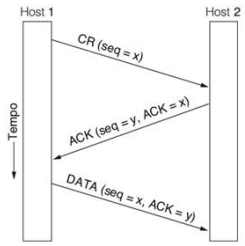


Estabelecimento de conexões

- Estabelecer uma conexão pode parecer simples, mas não é;
- Pacotes podem ser perdidos, atrasados, corrompidos e duplicados;
- Uma solução foi proposta por *Tomlinson* em 1975: o ***handshake* de três vias**:
  - Cada segmento é numerado;
  - Esta numeração não se repete por um tempo *T*;
  - Esquema a seguir.

Estabelecimento de conexões

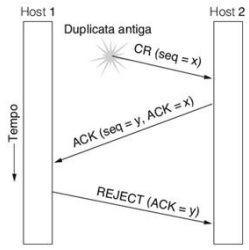
- ***Handshake* de três vias** (1/3):
  - Situação normal:



CR = Connection Request

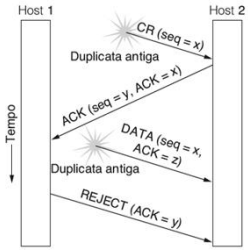
Estabelecimento de conexões

- ***Handshake* de três vias** (2/3):
  - Duplicata antiga de CONNECTION REQUEST que surge repentinamente:



Estabelecimento de conexões

- ***Handshake* de três vias** (3/3):
  - CONNECTION REQUEST e ACK duplicadas:

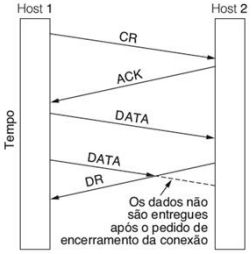


Encerramento de conexões

- Pode acontecer de duas formas:
  - **Assimétrico**:
    - Trata a conexão de maneira semelhante ao sistema telefônico;
    - Quando um dos interlocutores termina a conexão é interrompida;
  - **Simétrico**:
    - Trata a conexão como duas conexões unidimensionais isoladas;
    - Exige que cada conexão seja encerrada separadamente.

Encerramento de conexões

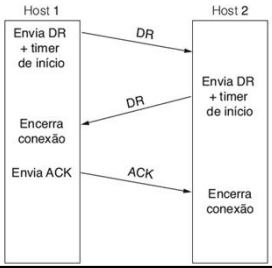
- **Encerramento Assimétrico**:
  - Dados podem ser perdidos:



DR = Disconnection Request

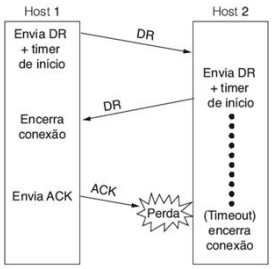
Encerramento de conexões

- **Encerramento Simétrico** (1/5):
  - Indicado quando uma quantidade fixa de dados será transmitida e é possível saber quando a transmissão termina;
  - **Situação 1**: Caso normal de *handshake* de três vias:



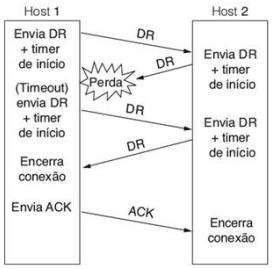
Encerramento de conexões

- **Encerramento Simétrico** (2/5):
  - **Situação 2**: ACK final perdido:



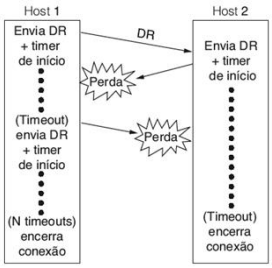
Encerramento de conexões

- **Encerramento Simétrico** (3/5):
  - **Situação 3**: Resposta perdida:



Encerramento de conexões

- **Encerramento Simétrico** (4/5):
  - **Situação 4**: Resposta perdida e DRs subsequentes perdidos:



Encerramento de conexões

- **Encerramento Simétrico** (5/5):
  - Ainda assim existe a possibilidade de falha;
  - Imagine que a primeira DR e todas as demais retransmissões se perdessem;
  - Um *host* encerraria a conexão e o outro não, ocasionando em uma conexão "semiaberta";
  - Uma alternativa é que a conexão seja encerrada após um determinado tempo de inatividade.

Controle de erro e fluxo

- Estabelecido como as conexões podem ser iniciadas e finalizadas é hora de verificar algumas questões relacionadas ao gerenciamento das conexões enquanto em uso;
- As principais questões são:
  - **Controle de erro**: garantir que os dados sejam entregues em um nível de confiabilidade desejado, normalmente que todos os dados sejam entregues sem nenhum erro;
  - **Controle de fluxo**: impedir que um transmissor rápido sobrecarregue um receptor lento;
- Mas isso já não foi feito na camada de enlace?

Controle de erro e fluxo

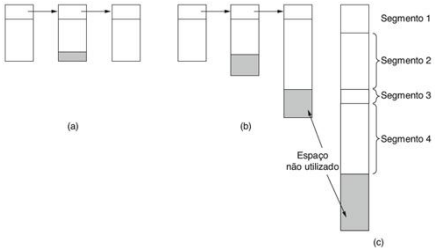
- Sim, e na camada de transporte utiliza-se dos mesmos mecanismos estudados na camada de enlace;
- Embora utilize os mesmos mecanismos, existem diferenças em função e grau;
- **Controle de erro**:
  - O *checksum* da camada de enlace protege um quadro quando ele atravessa um único enlace;
  - O *checksum* da camada de transporte protege um segmento enquanto ele atravessa um caminho de rede inteiro;
  - Erros de quadros podem passar desapercebidos e serem identificados apenas na camada de transporte;
    - Ex.: erro de processamento em um roteador;

Controle de erro e fluxo

- **Controle de fluxo** (1/4):
  - Feito através da gerência de *buffer* e do tamanho da janela deslizante;
  - Reserva de espaço em *buffer* de um receptor limita o fluxo máximo de transmissão de um transmissor:
    - Durante a conexão são feitas negociações de reserva de *buffer* em um receptor para uma conexão específica;
    - O transmissor deve possuir *buffer* para armazenar segmentos que ainda não foram confirmados;
    - Como definir o tamanho dos *buffers*?

Controle de erro e fluxo

- **Controle de fluxo** (2/4):
  - Tamanho dos *buffers*:



(a) Encadeamento de *buffers* de tamanho fixo.
(b) Encadeamento com tamanho variável. (c) Um grande *buffer* circular por conexão.

Controle de erro e fluxo

- **Controle de fluxo** (3/4):
  - Alocação dinâmica de *buffer*:

	A	Mensagem	B	Comentários
1	→	<request 8 buffers>	→	A deseja 8 buffers
2	←	<ack = 15, buf = 4>	←	B concede mensagens 0-3 apenas
3	→	<seq = 0, data = m0>	→	A tem 3 buffers restantes agora
4	→	<seq = 1, data = m1>	→	A tem 2 buffers restantes agora
5	→	<seq = 2, data = m2>	•••	Mens. perdida, mas A acha que tem 1 restante
6	←	<ack = 1, buf = 3>	←	B confirma 0 e 1, permite 2-4
7	→	<seq = 3, data = m3>	→	A tem 1 buffer restante
8	→	<seq = 4, data = m4>	→	A tem 0 buffers restantes e deve parar
9	→	<seq = 2, data = m2>	→	A atinge o timeout e retransmite
10	←	<ack = 4, buf = 0>	←	Tudo confirmado, mas A ainda bloqueado
11	←	<ack = 4, buf = 1>	←	A pode agora enviar 5
12	←	<ack = 4, buf = 2>	←	B encontrou novo buffer em algum lugar
13	→	<seq = 5, data = m5>	→	A tem 1 buffer restante
14	→	<seq = 6, data = m6>	→	A agora está bloqueado novamente
15	←	<ack = 6, data = m6>	←	A ainda está bloqueado
16	•••	<ack = 6, buf = 4>	←	Impasse em potencial

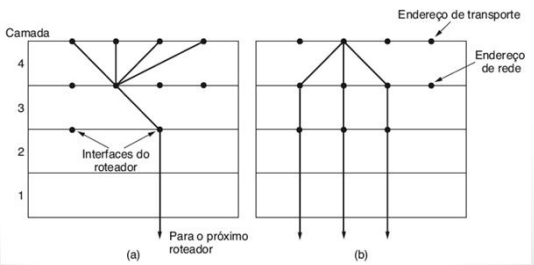
As setas mostram a direção de transmissão. (...) indica perda segmentos.

Controle de erro e fluxo

- **Controle de fluxo** (4/4):
  - Quando o espaço em buffer deixar de limitar o fluxo máximo, surgirá outro gargalo: a capacidade de transporte da rede;
  - Então, é necessário definir um mecanismo que limite as transmissões com base nesta capacidade;
  - Uma solução proposta por *Belsnes* é o ajuste dinâmico do tamanho da janela deslizante:
    - Se a rede puder transmitir ***c*** segmentos/s e o tempo de ciclo for de ***r*** (considerando todos os fatores de transmissão e confirmação), então o tamanho da janela deverá ser de ***c*r***;
    - Como a capacidade da rede varia com o tempo, o tamanho da janela deverá ser ajustado com frequência;
  - Esta estratégia permite fazer o controle de fluxo e de congestionamento através do tamanho da janela deslizante;

Multiplexação

- Pode ser de diversas formas:
  a) Caso exista apenas um endereço de rede;
  b) Um usuário que necessitar de mais largura de banda pode utilizar vários caminhos de rede (multiplexação inversa).



Recuperação de falhas

- Poderão ocorrer falhas nos *hosts* ou nos roteadores;
- Com a entidade de transporte totalmente no *host*, uma falha em roteador é facilmente tratada:
  - As entidades de transporte esperam segmentos perdidos o tempo todo, e sabem como lidar com eles usando retransmissões;
- O problema maior é quando um *host* falha:
  - Em particular, pode-se desejar que o cliente continue funcionando quando um servidor falhar e retornar instantes depois;
  - Sempre haverá situações em que o protocolo não recuperará o funcionamento de modo apropriado.

Recuperação de falhas

- Diferentes combinações de estratégias de cliente e servidor:

	Estratégia usada pelo host receptor					
	Primeiro ACK, depois gravar			Primeiro gravar, depois ACK		
Estratégia usada pelo host transmissor	AC(W)	AWC	C(AW)	C(WA)	WAC	WC(A)
Sempre retransmitir	OK	DUP	OK	OK	DUP	DUP
Nunca retransmitir	LOST	OK	LOST	LOST	OK	OK
Retransmitir em S0	OK	DUP	LOST	LOST	DUP	OK
Retransmitir em S1	LOST	OK	OK	OK	OK	DUP

OK = Protocolo funciona corretamente
DUP = Protocolo gera uma mensagem duplicada
LOST = Protocolo perde uma mensagem

S0: nenhum seguimento pendente; **S1**: um seguimento pendente;
A: enviar confirmação (ACK); **W**: gravar no processo de saída; **C**: sofrer uma pane;

- **Conclusão**: Recuperação de falha na camada **N** só pode ser realizada na camada **N+1**.

O serviço de transporte;
Elementos dos protocolos de transporte;
Protocolo UDP;
Protocolo TCP;
Implementação de sockets.

PROTOCOLO UDP

Tópicos

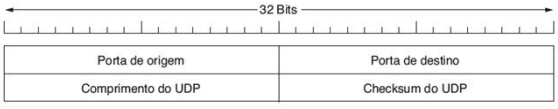
- Introdução;
- Chamada de Procedimentos Remotos (RPC);
- Protocolos de transporte em tempo real;
  - RTP;
  - RTCP.

Introdução

- Protocolo de transporte não orientado a conexões;
- Significa *Protocolo de Datagrama do Usuário* ou:
  - **UDP** (***User Datagram Protocol***);
- Descrito na RFC 768;
- Utilizado quando o custo da conexão é mais alto do que a transferência, exemplos:
  - Cenário *cliente/servidor* com *requisição/resposta* pequenas;
  - Programas de multimídia em tempo real;
  - Protocolo de aplicação **DNS** (***Domain Name System***).

Introdução

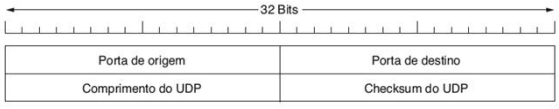
- O UDP transmite segmentos que consistem de um cabeçalho de 8 bytes, seguido pela carga útil;
- O cabeçalho UDP (1):



  - **Portas**:
    - Usadas para identificar os pontos extremos da origem e o destino;

Introdução

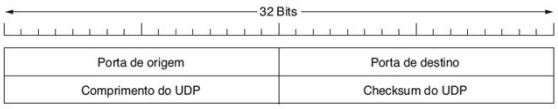
- O UDP transmite segmentos que consistem de um cabeçalho de 8 bytes, seguido pela carga útil;
- O cabeçalho UDP (2):



  - **Comprimento UDP**:
    - Inclui o tamanho do cabeçalho e dos dados;
    - Máximo de 65.515 bits.

Introdução

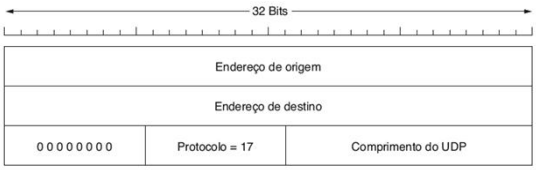
- O UDP transmite segmentos que consistem de um cabeçalho de 8 bytes, seguido pela carga útil;
- O cabeçalho UDP (3):



  - ***Checksum* UDP**:
    - Campo opcional para gerar confiabilidade extra;
    - Faz o *checksum* do cabeçalho, dados e um pseudocabeçalho conceitual IP.

Introdução

- Pseudocabeçalho IP:
  - Não é transmitido no cabeçalho do datagrama UDP;
  - É acrescentado ao *checksum* do UDP para ajudar a detectar se o segmento chegou ao destino correto;
  - Para o IPv4:

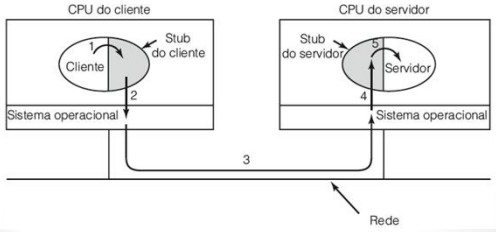


RPC

- **RPC** = ***Remote Procedure Call***, ou, *Chamada Remota de Procedimento*;
- De certa forma, enviar uma mensagem a um servidor remoto e aguardar uma resposta se assemelha muito a uma chamada de função em uma linguagem de programação;
- A ideia do RPC é tornar uma chamada de procedimento remoto o mais semelhante possível de uma chamada local;
  - Nenhuma troca de mensagens é visível pelo programador.

RPC

- O procedimento que faz a chamada é denominado **cliente** e o que a recebe é denominado **servidor**;
- A chamada é feita através de procedimentos que ocultam a característica de chamada remota denominados **stubs**;
- Etapas de uma chamada RPC:

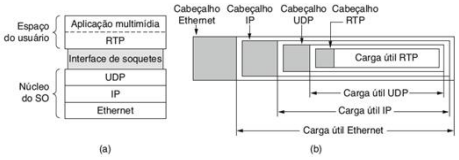


Protocolos de tempo real

- Além do RPC, o UDP é amplamente utilizado para outra área: aplicações multimídia em tempo real;
- À medida que as aplicações multimídia foram ganhando espaço na Internet, rádios online, telefonia sobre Internet, música por demanda, videoconferência, etc., cada aplicação foi criando seus recursos próprios;
- Com muita semelhança entre estes recursos, foi criado o **RTP** (***Real-time Transport Protocol***), para criar um padrão para as várias aplicações.

Protocolos de tempo real

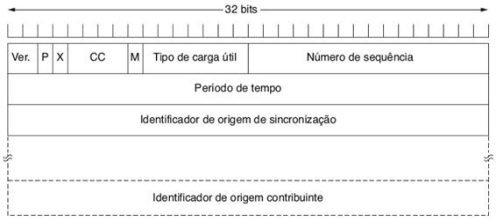
- O RTP normalmente opera no espaço do usuário sobre o UDP;
- Ele pode ser visto como um protocolo de transporte implementado na camada de aplicação, por isso está sendo visto na camada de transporte;
- Como se encaixa na pilha de protocolos:



(a) A posição do RTP na pilha de protocolos.
(b) Pacotes aninhados.

Protocolos de tempo real

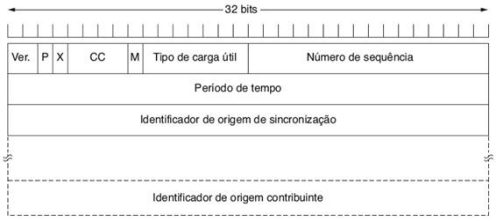
- Cabeçalho RTP (1):



- **Versão**:
  - Indicação da versão do protocolo utilizado;

Protocolos de tempo real

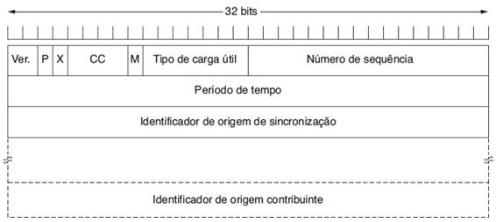
- Cabeçalho RTP (2):



- **P**:
  - Indica que o segmento foi completado para atingir um tamanho múltiplo de 4 bytes (32 bits);

Protocolos de tempo real

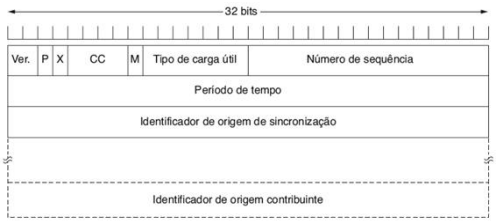
- Cabeçalho RTP (3):



- **X**:
  - Indica que o cabeçalho de extensão está presente;

Protocolos de tempo real

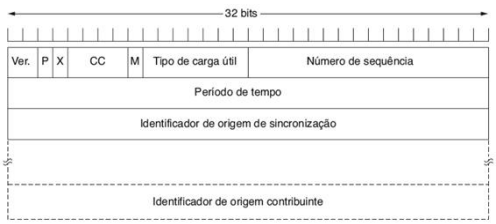
- Cabeçalho RTP (4):



- **CC**:
  - Informa quantas *origens de contribuição* estão presentes;

Protocolos de tempo real

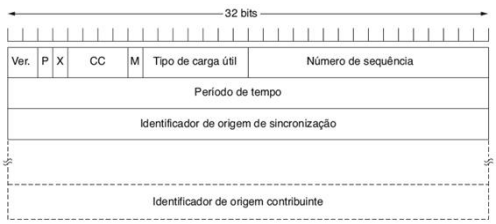
- Cabeçalho RTP (5):



- **M**:
  - Bit marcador reservado para aplicação;
  - A própria aplicação define seu uso;

Protocolos de tempo real

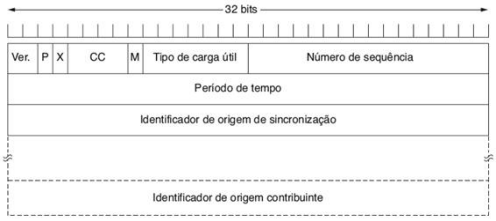
- Cabeçalho RTP (6):



- **Tipo de carga útil**:
  - Informa o tipo de codificação utilizado (MP3, etc...);

Protocolos de tempo real

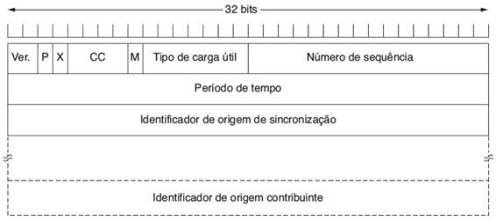
- Cabeçalho RTP (7):



- **Número de sequência**:
  - Timer incrementado a cada pacote;
  - Usado para identificar perdas;

Protocolos de tempo real

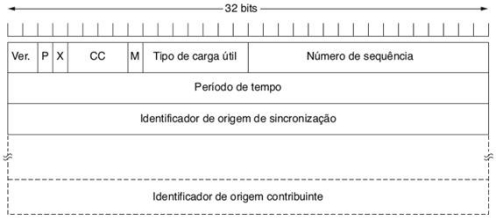
- Cabeçalho RTP (8):



- **Período de tempo**:
  - Utilizado para anotar o tempo em relação à primeira amostra;
  - Ajuda a reduzir a flutuação de sincronização no receptor, desacoplando a reprodução da chegada do segmento;

Protocolos de tempo real

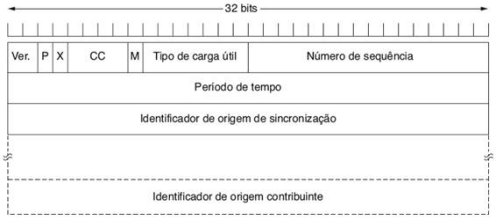
- Cabeçalho RTP (9):



- **Identificador de origem de sincronização**:
  - Determina o fluxo ao qual o segmento pertence;
  - Utilizado para multiplexar e demultiplexar vários fluxos de dados em um único fluxo de segmentos UDP;

Protocolos de tempo real

- Cabeçalho RTP (10):



- **Identificador de origem de contribuinte**:
  - Usado quando houver *mixers* de fluxos. Se em algum ponto, fluxos diferentes forem misturados (duas fontes de áudio por exemplo), os identificadores originais podem ser colocados neste campo;

Protocolos de tempo real

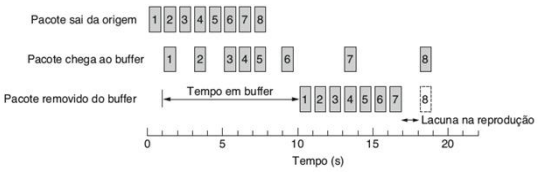
- O protocolo RTP é acompanhado de um protocolo de controle, o **RTCP** (***Real-Time Transport Control Protocol***);
- Cada participante de uma sessão RTP envia periodicamente pacotes RTCP para todos os outros participante;
- O RTCP cuida do *feedback*, da sincronização e da interface do usuário, mas não transporta nenhuma amostra de mídia.

Protocolos de tempo real

- O RTCP possui quatro funções:
  1. Prover *feedback* da qualidade da distribuição dos dados;
  2. Agrupar fluxos diferentes, por exemplo, áudio e vídeo, através de um identificador;
  3. Possibilitar que cada sessão possa observar o número de participantes;
  4. Opcional, refere-se à distribuição de informação sobre um participante:
     - Esta informação poderia ser usada numa interface de usuário, por exemplo.

Protocolos de tempo real

- Suavizando o fluxo de saída através de *buffer*:



O serviço de transporte;
Elementos dos protocolos de transporte;
Protocolo UDP;
Protocolo TCP;
Implementação de sockets.

PROTOCOLO TCP

Tópicos

- Introdução;
- O Modelo de serviço;
- O protocolo;
- Cabeçalho do segmento;
- Estabelecimento de conexões;
- Uso de TCP *v.s.* UDP.

Introdução

- Protocolo de Controle de Transmissão: **TCP** (***Transmission Control Protocol***);
- Projetado originalmente para fornecer fluxo de bytes **fim a fim confiável** em uma rede interligada não confiável;
- Definido formalmente na RFP 793 (em 1981), mas evoluiu muito com o tempo:
  - RFC 1122: esclarecimentos e soluções de bugs;
  - RFC 1323: extensões para alto desempenho;
  - RFC 2873: alterações de cabeçalho para QoS;
  - RFC 2988: melhorias na sincronização de retransmissões;
  - RFC 3168: notificação explícita de congestionamento;
  - ...
  - **RFC 4614**: guia para as RFCs relacionadas ao TCP.

O Modelo de serviço

- O serviço é obtido através de **soquetes** (***sockets***);
- Cada soquete tem um número (endereço) que consiste do endereço IP do *host* e um número de **porta** (TSAP):
  - O **inetd** (***Internet daemon***) funciona como um Proxy (protocolo de conexão inicial) para atribuição de portas;
- Para que o serviço funcione é necessário o estabelecimento de uma conexão entre um soquete transmissor e um receptor;
- Um soquete pode ser utilizado para várias conexões ao mesmo tempo;

O Modelo de serviço

- Portas de números abaixo de 1024 (**portas conhecidas**) são reservadas para usuários privilegiados, exemplos:

Porta	Protocolo	Uso
20, 21	FTP	Transferência de arquivos.
22	SSH	Login remoto (substituto do Telnet).
25	SMTP	Correio eletrônico.
80	HTTP	*World Wide Web*.
110	POP-3	Acesso remoto a correio eletrônico.
143	IMAP	Acesso remoto a correio eletrônico.
443	HTTPS	Web segura (HTTP sobre SSL/TLS).
543	RTPS	Controle de *player* de música.
631	IPP	Compartilhamento de impressora.

- Lista completa: www.iana.org.

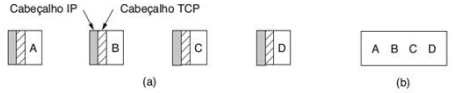
O Modelo de serviço

- Primitivas de soquetes para o TCP:

Primitiva	Significado
SOCKET	Criar um novo ponto final de comunicação.
BIND	Anexar um endereço local a um soquete.
LISTEN	Anunciar a disposição para aceitar conexões; Mostrar o tamanho da fila.
ACCEPT	Bloquear o responsável pela chamada até uma tentativa de conexão ser recebida.
CONNECT	Tentar estabelecer uma conexão ativamente.
SEND	Enviar alguns dados através da conexão.
RECEIVE	Receber alguns dados da conexão.
CLOSE	Encerrar a conexão.

O Modelo de serviço

- Todas as conexões são ponto-a-ponto e *full-duplex*;
- Não admite processos de *multicast* ou *broadcast*;
- Uma conexão TCP é um fluxo de bytes, não de mensagens, as fronteiras de mensagens não são preservadas de uma extremidade a outra:

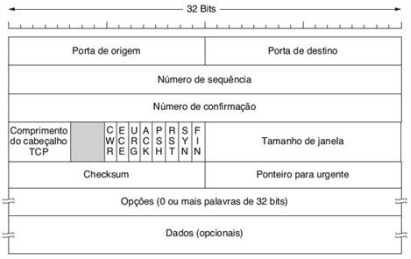


(a) 4 segmentos de 512 bytes enviados em datagramas IP separados.
(b) Os 2.048 bytes de dados entregues à aplicação em uma única chamada READ.

O protocolo

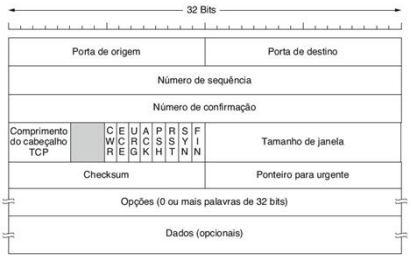
- As entidades transmissoras e receptoras trocam dados através de **segmentos**:
  - Cabeçalho de 20 bytes fixos mais uma parte opcional;
  - Zero ou mais bytes de dados;
  - Um segmento não pode ultrapassar 65.515 bytes ou a MTU do caminho (para caber em uma carga útil IP);
  - Por isso, utiliza a **descoberta de MTU do caminho**;
- Utiliza o protocolo de janela deslizante para controle;
- Cada byte possui seu próprio número de sequência.

Cabeçalho do segmento



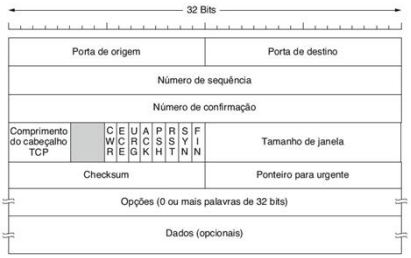
- **Portas**:
  - Identificam os pontos terminais da conexão;

Cabeçalho do segmento



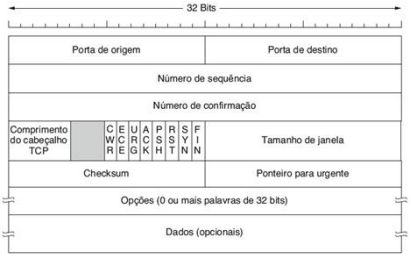
- **Números de sequência e confirmação**:
  - Identifica os bytes enviados e sua confirmação;
  - O número de confirmação corresponde ao próximo byte esperado, e não ao último recebido;
  - Cada byte é numerado em um único fluxo TCP;

Cabeçalho do segmento



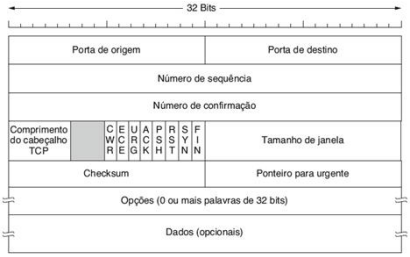
- **Comprimento do cabeçalho**:
  - Informa quantas palavras de 32 bits compõem o cabeçalho;

Cabeçalho do segmento



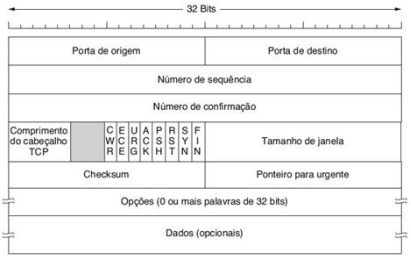
- **Oito flags de 1 bit** (1):
  - **CWR** e **ECE**: controle de congestionamento;
  - **URG**: indica entrega urgente;
  - **ACK**: indica se há uma confirmação no segmento;

Cabeçalho do segmento



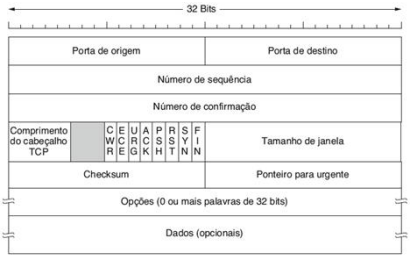
- **Oito flags de 1 bit** (2):
  - **PSH**: *push*, significa enviar o dado imediatamente (não espera o *buffer* ser preenchido);
  - **RST**: indica algum problema, a conexão deve ser reiniciada;

Cabeçalho do segmento



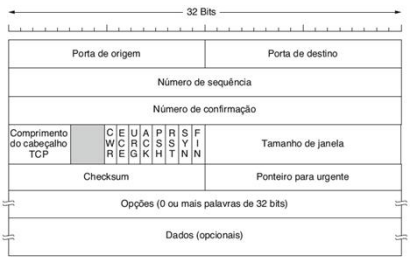
- **Oito flags de 1 bit** (3):
  - **SYN**: usado para estabelecer conexão;
  - **FIN**: usado para finalizar conexão;

Cabeçalho do segmento



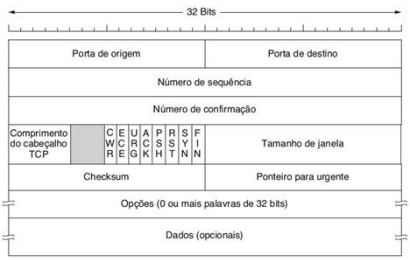
- **Tamanho da janela**:
  - O TCP funciona com janela deslizante de tamanho variável;
  - Utilizado para definição do tamanho da janela;

Cabeçalho do segmento



- **Ponteiro para urgente**:
  - Usado apenas quando a *flag* URG está setada para 1;
  - Representa o deslocamento positivo a ser aplicado ao campo de número de sequência para identificar o último byte de dados urgentes;

Cabeçalho do segmento



- **Opções**:
  - Projetado para permitir a oferta de recursos extra, ou seja, recursos não previstos pelo cabeçalho comum;
  - Devem preencher um múltiplo de 32 bits e pode ser no máximo de 40 bytes (o limite de tamanho do cabeçalho é 60 bytes).

Estabelecimento de conexões

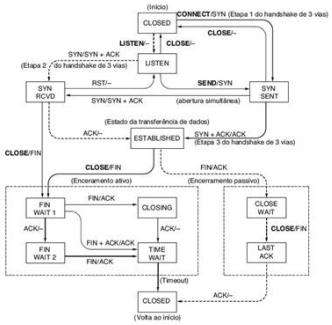
- Utiliza o ***handshake* de três vias**;
- Utiliza a *flag* de sincronização (SYN) para o início de uma conexão;
- Por outro lado, utiliza a *flag* de finalização (FIN) para o encerramento da conexão.

Estabelecimento de conexões

- As etapas para o estabelecimento e encerramento de conexões pode ser expresso em uma máquina de estados finitos com 11 estados:

Estado	Descrição
CLOSED	Nenhuma conexão ativa ou pendente.
LISTEN	O servidor está esperando a chegada de uma chamada.
SYN RCVD	Uma solicitação de conexão chegou; espera por ACK.
SYN SENT	A aplicação começou a abrir uma conexão.
ESTABLISHED	O estado normal para a transferência de dados.
FIN WAIT 1	A aplicação informou que terminou de transmitir.
FIN WAIT 2	O outro lado concordou em encerrar.
TIME WAIT	Aguarda a entrega de todos os pacotes.
CLOSING	Ambos os lados tentaram encerrar a transmissão simultaneamente.
CLOSE WAIT	O outro lado deu início a um encerramento.
LAST ACK	Aguarda a entrega de todos os pacotes.

Estabelecimento de conexões



- Gerenciamento de conexão TCP em uma máquina de estados finitos:
  - As linhas sólidas grossas são os percursos comuns ao cliente. As linhas tracejadas são os percursos comuns ao servidor. As linhas suaves são eventos incomuns. Cada transição é rotulada pelo evento que a produziu e ação resultante separada por barra.

Uso de TCP *v.s.* UDP

- Protocolos de aplicação *v.s.* Transporte:

Serviço	Protoc. de Aplicação	Protoc. de Transporte
Transf. de arquivos	FTP	TCP
Terminal remoto	Telnet	TCP
Correio eletrônico	SMTP	TCP
Serviço Web	HTTP	TCP
Trivial FTP	TFTP	UDP
Endereçamento dinâmico	DHCP	UDP
Gerência remota	SNMP	UDP
Serviço de nomes	DNS	UDP / TCP
Serviço de arquivos remotos	NFS	UDP / TCP

O serviço de transporte;
Elementos dos protocolos de transporte;
Protocolo UDP;
Protocolo TCP;
Implementação de sockets.

IMPLEMENTAÇÃO DE SOCKETS

Tópicos

- Trabalhos práticos extraclasse.

Fim!

REFERÊNCIAS:

- **A.S. TANENBAUM, *Redes de Computadores*, Prentice Hall, 5a. edição, 2011;**
- Materiais didáticos dos professores:
  - **Rande A. Moreira**, UFOP / 2011-01
    Disponível em: http://randearievilo.com.br/redes/ (acesso em 17/08/2011);