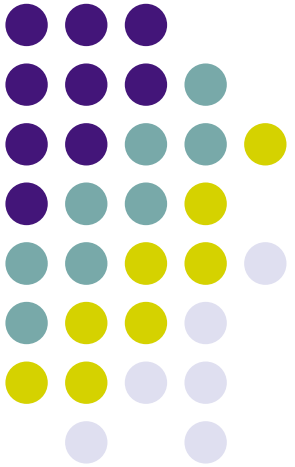
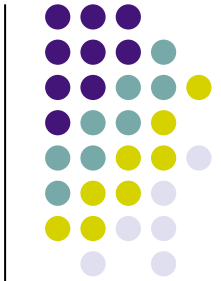
Camada de Transporte

Kalinka Castelo Branco



Camada de Transporte



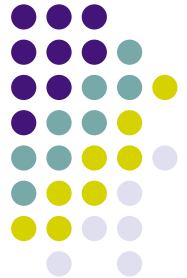
Objetivos do Capítulo:

- entender os princípios por trás dos serviços da camada de transporte:
  - multiplexação/demultiplexação
  - transferência de dados confiável
  - controle de fluxo
  - controle de congestionamento
- instanciação e implementação na Internet

Resumo do Capítulo:

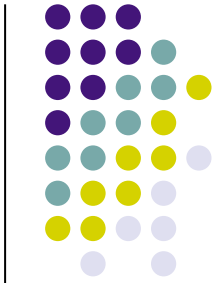
- serviços da camada de transporte
- multiplexação/demultiplexação
- transporte sem conexão: UDP
- princípios de transferência confiável de dados
- transporte orientado à conexão: TCP
  - transferência confiável
  - controle de fluxo
  - gerenciamento de conexão
- princípios de controle de congestionamento
- controle de congestionamento do TCP

Camada de Transporte - Funções



- A função básica da camada de transporte é aceitar dados da camada de aplicação, dividí-los em unidades menores em caso de necessidade, passá-los para a camada de rede e garantir que todas essas unidades cheguem corretamente à outra extremidade.
- Tudo deve ser feito com eficiência de forma que as camadas superiores fiquem isoladas das mudanças na tecnologia de hardware.

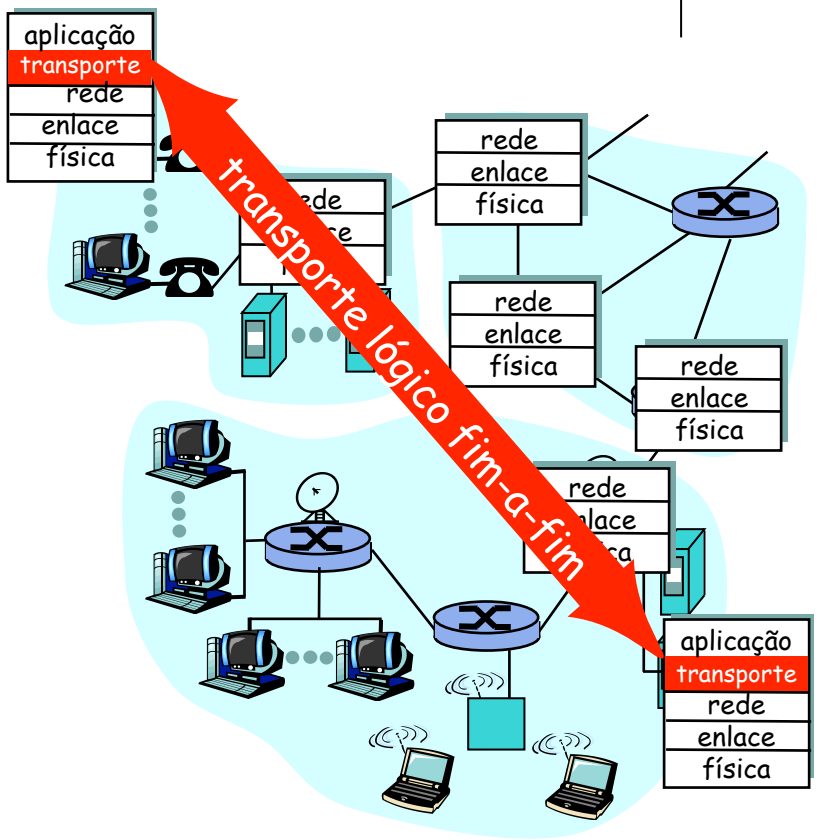
Camada de Transporte - Funções



- A camada de transporte é uma camada fim a fim, que liga a origem ao destino.
- Um programa da máquina de origem mantém uma conversa com um programa semelhante instalado na máquina de destino, utilizando cabeçalhos de mensagem e mensagens de controle.
  - Nessas camadas de transporte de diferentes hosts são trocadas TPDUs (*Transport Protocol Data Units*) chamados de segmentos. Um segmento é composto pelo cabeçalho da camada de transporte e os dados da camada de aplicação.

Protocolos e Serviços de Transporte

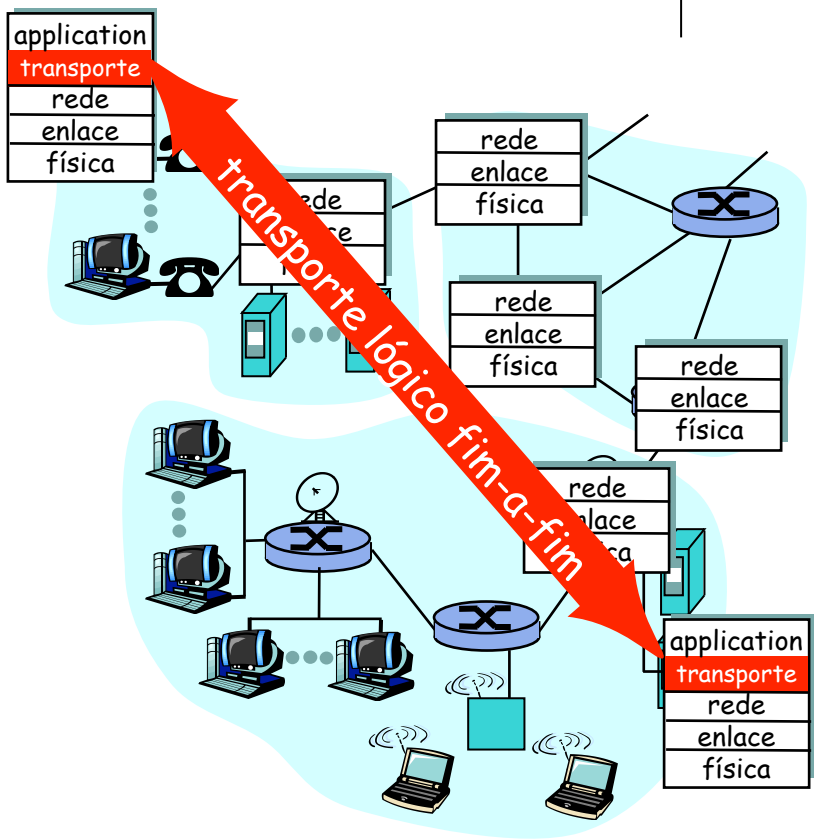
- Fornecem comunicação lógicas entre processos de aplicação em diferentes hosts
- Os protocolos de transporte são executados nos sistemas finais da rede
- serviço de transporte vs serviços de rede :
- *camada de rede:* transferência de dados entre computadores (*end systems*)
- *camada de transporte:* transferência de dados entre processos
  - utiliza e aprimora os serviços oferecidos pela camada de rede



Protocolos da Camada de Transporte

Serviços de Transporte da Internet:

- Confiável (TCP)
  - congestão
  - controle de fluxo
  - orientado à conexão
- não confiável ("best-effort") UDP
- serviços não disponíveis:
  - tempo-real
  - garantia de banda

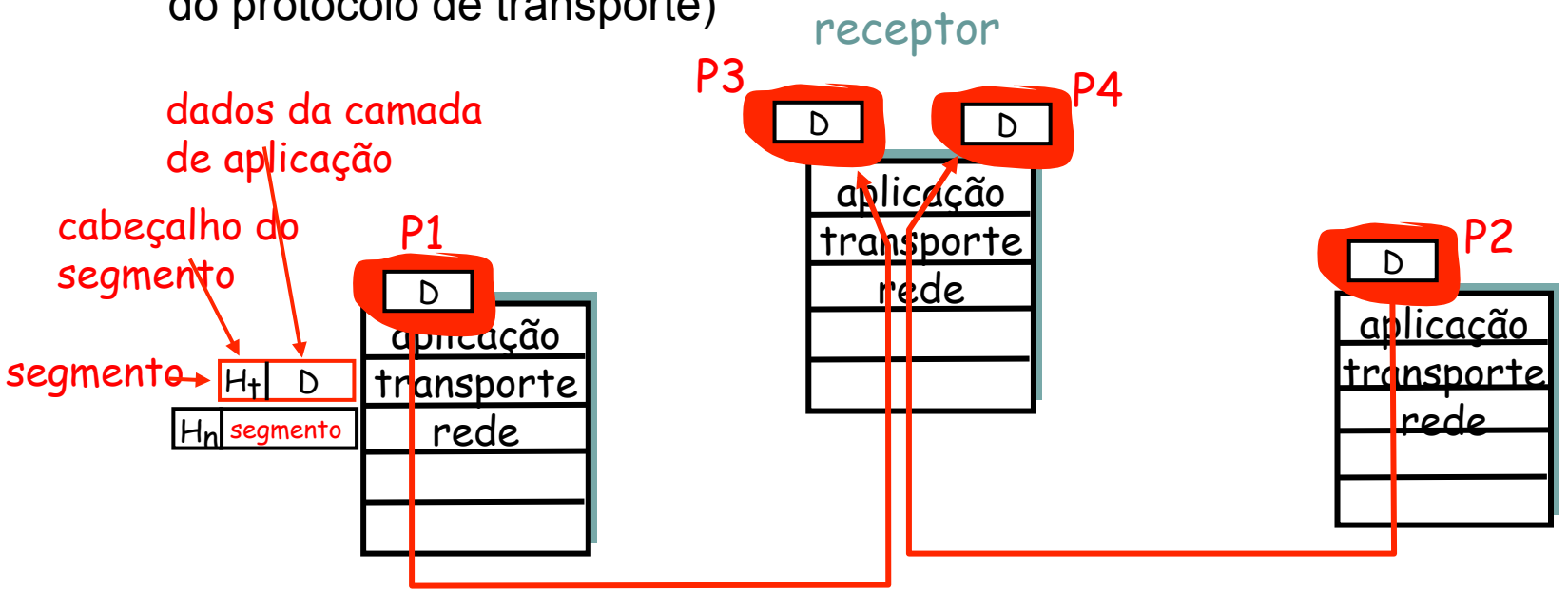


Multiplexação de Aplicações

Segmento - unidade de dados trocada entre entidades da camada de transporte

- TPDU: *transport protocol data unit* (unidade de dados do protocolo de transporte)

Demultiplexação: entrega de segmentos recebidos aos processos de aplicação corretos

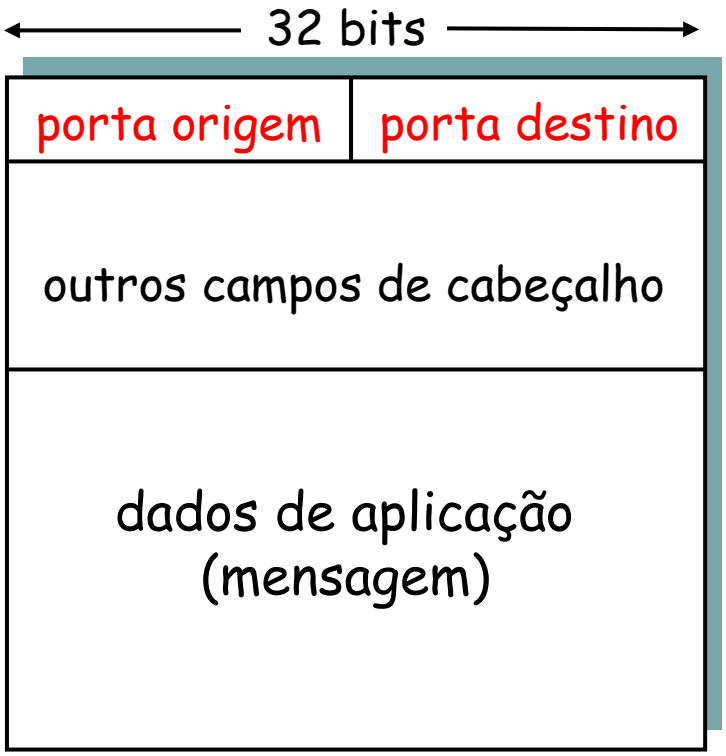


Multiplexação de Aplicações

Multiplexação:
reunir dados de múltiplos processo de aplicação, juntar cabeçalhos com informações para demultiplexação

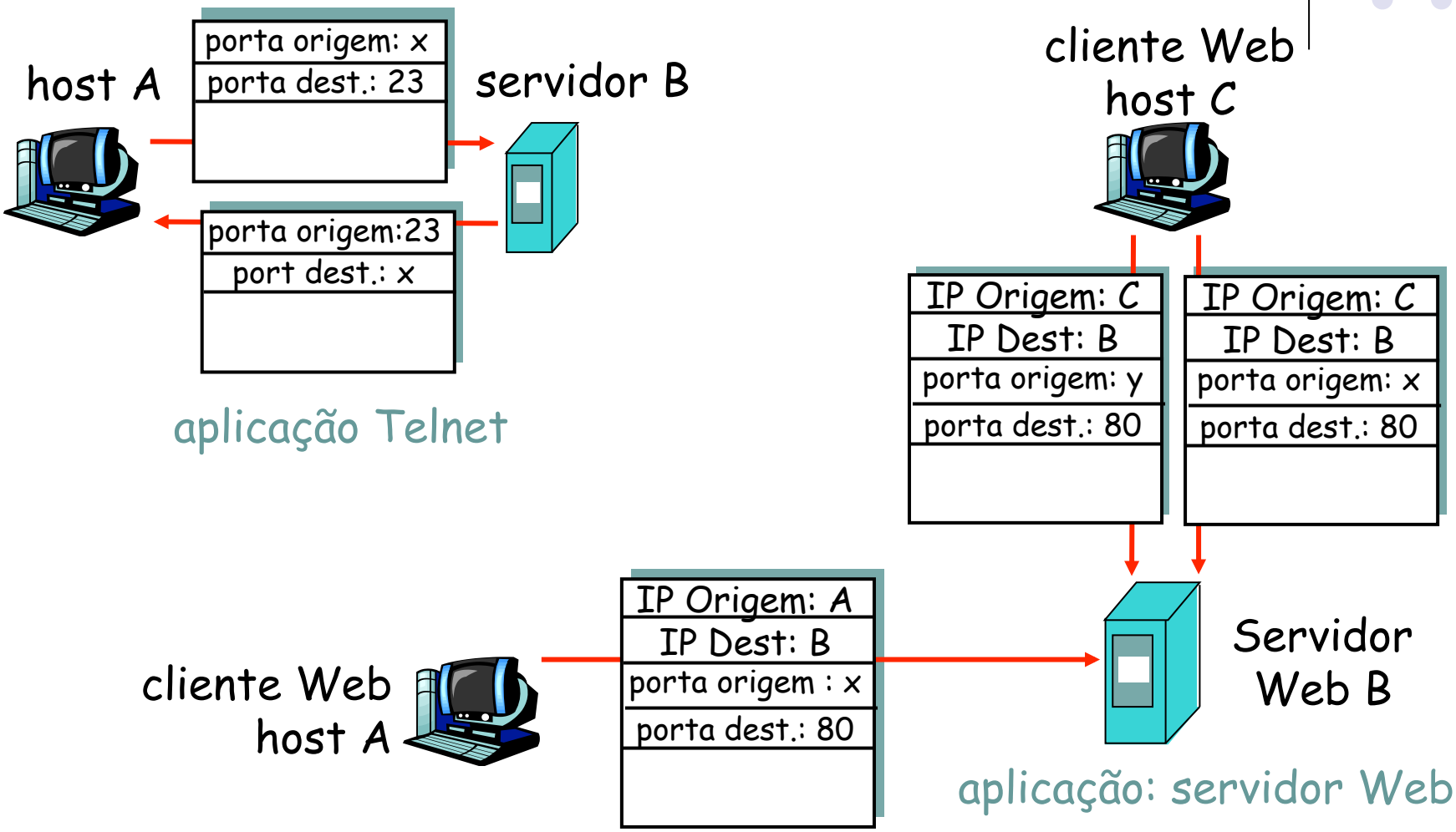
multiplexação/demultiplexação:

- baseada nos número de porta do transmissor, número de porta do receptor e endereços IP
  - números de porta origem e destino em cada segmento
  - lembre: portas com números bem-conhecidos são usadas para aplicações específicas

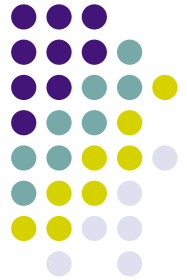


formato do segmento TCP/UDP

Multiplexação: exemplos

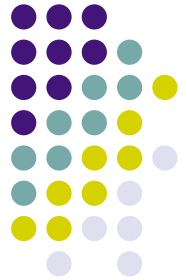


Multiplexação



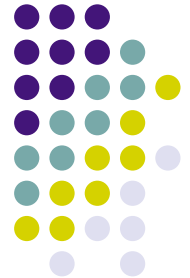
- Muitos hosts são multiprogramados;
  - isso significa que muitas conexões estarão entrando e saindo de cada host.
- É responsabilidade da camada de transporte multiplexar todas as comunicações em um único canal determinando a qual conexão uma mensagem pertence.
- É responsabilidade desta camada estabelecer conexões, encerrá-las e controlá-las de forma que um host muito rápido não possa sobrecarregar um host muito lento (controle de fluxo). Em redes IP são utilizados dois protocolos para a implementação destas funções: o TCP e o UDP.

Endereçamento da Camada de Transporte



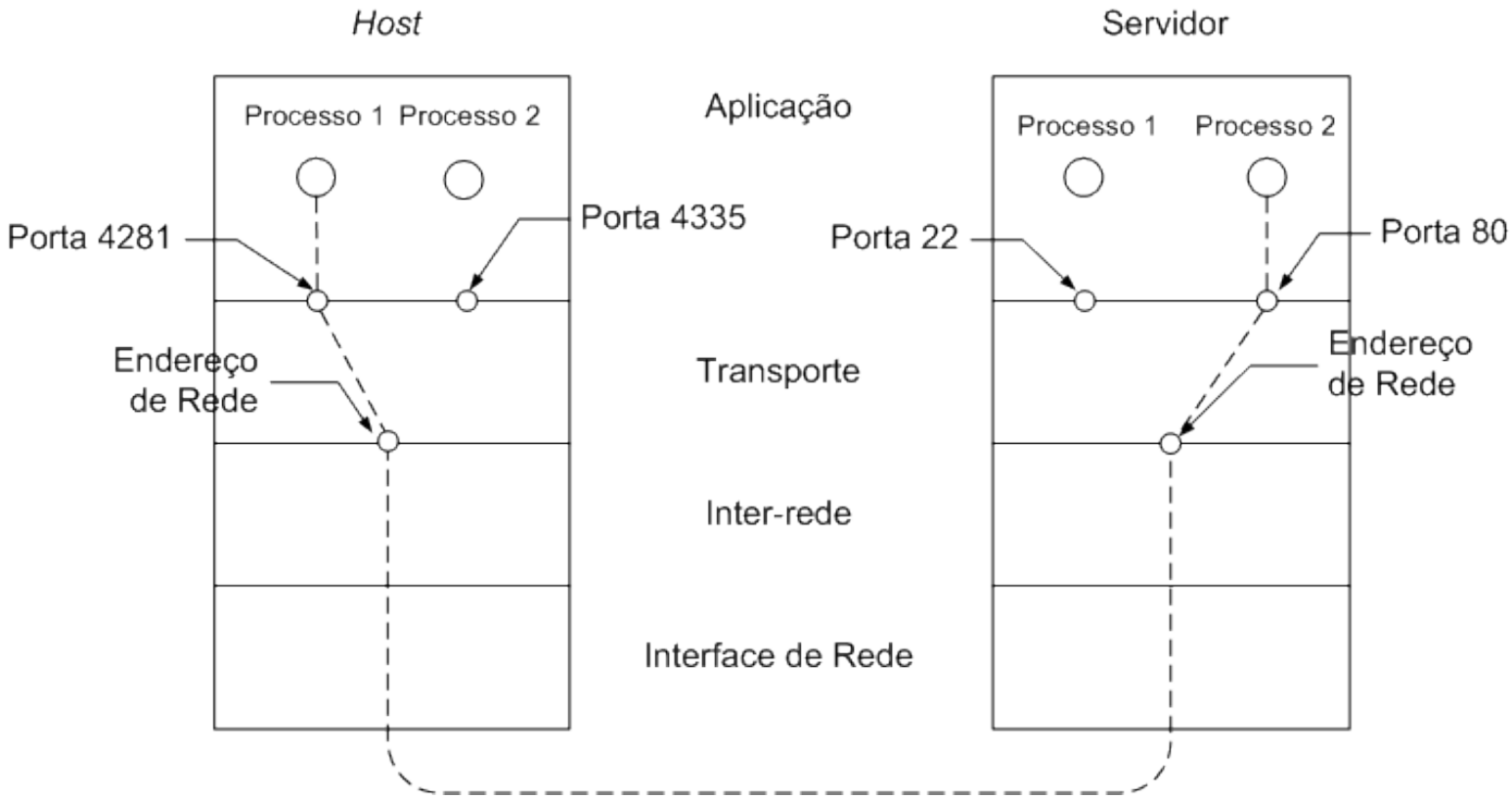
- Da mesma forma que em outras camadas, a camada de transporte também possui um endereçamento.
- Quando um processo de aplicação deseja estabelecer uma conexão com um processo de aplicação remoto, é necessário especificar a aplicação com a qual ele irá se conectar.
  - O método utilizado é definir os endereços de transporte que os processos podem ouvir para receber solicitações de conexão.

Endereçamento da Camada de Transporte

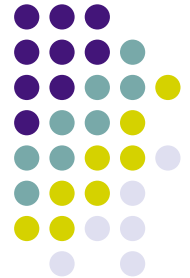


- Os processos utilizam os TSAP (*Transport Service Access Point* – Ponto de Acesso de Serviços de Transporte) para se intercomunicarem.
- Em redes IP, o TSAP é um número de 16 bits chamado de porta. O endereço da camada de transporte é um número de 48 bits, composto pela agregação do endereço IP do host e o número da porta.
- Os serviços da camada de transporte são obtidos por meio da comunicação entre os *sockets* do transmissor e do receptor.

Endereçamento da Camada de Transporte

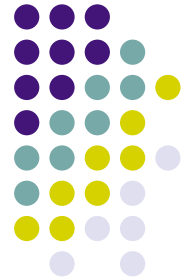


Endereçamento da Camada de Transporte



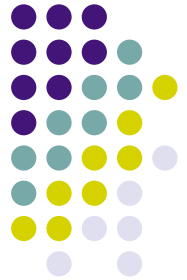
- Para uma melhor organização de serviços, algumas portas foram definidas pela IANA como “portas bem conhecidas” (*well-known ports*). Estas são as portas abaixo de 1024, para aplicações não padronizadas são utilizadas portas acima deste valor.

Endereçamento da Camada de Transporte



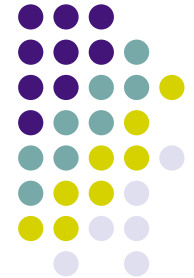
- Os *sockets* são diferentes para cada protocolo de transporte, desta forma mesmo que um *socket* TCP possua o mesmo número que um *socket* UDP, ambos são responsáveis por aplicações diferentes.
- Os *sockets* de origem e destino são responsáveis pela identificação única da comunicação. Desta forma é possível a implementação da função conhecida como multiplexação.
- A multiplexação possibilita que haja várias conexões partindo de um único host ou terminando em um mesmo servidor.

Endereçamento da Camada de Transporte



- A formação do *socket* de origem e destino se dá da seguinte forma:
  - Ao iniciar uma comunicação é especificado para a aplicação o endereço IP de destino e a porta de destino;
  - A porta de origem é atribuída dinamicamente pela camada de transporte. Ele geralmente é um número seqüencial randômico acima de 1024;
  - O endereço IP de origem é atribuído pela camada 3.

UDP: User Datagram Protocol [RFC 768]



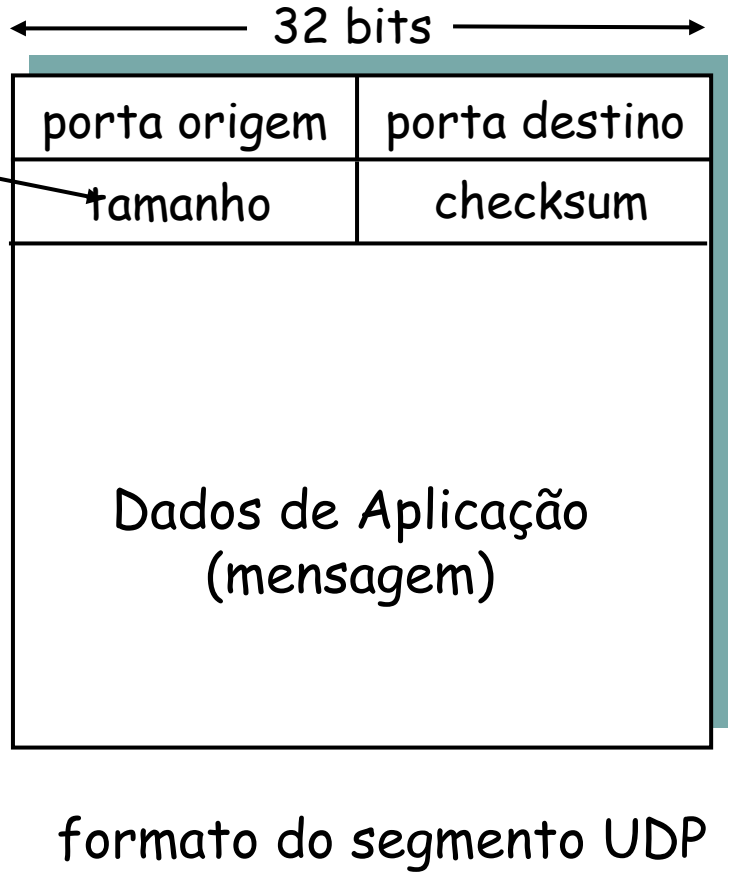
- protocolo de transporte da Internet “sem gorduras”
- serviço “best effort” , segmentos UDP podem ser:
  - perdidos
  - entregues fora de ordem para a aplicação
- *sem conexão:*
  - não há apresentação entre o UDP transmissor e o receptor
  - cada segmento UDP é tratado de forma independente dos outros

Porque existe um UDP?

- não há estabelecimento de conexão (que pode redundar em atrasos)
- simples: não há estado de conexão nem no transmissor, nem no receptor
- cabeçalho de segmento reduzido
- não há controle de congestionamento: UDP pode enviar segmentos tão rápido quanto desejado (e possível)

Mais sobre UDP

- muito usado por aplicações de mutimídia contínua (Voz e vídeo)
  - tolerantes à perda
  - sensíveis à taxa
- outros usos do UDP
  - DNS
  - SNMP
- transferência confiável sobre UDP: acrescentar confiabilidade na camada de aplicação
  - recuperação de erro específica de cada aplicação
- Porta de Origem e Porta de Destino – Indicam os pares de porta que estão executando a comunicação;
- Comprimento – Indica o comprimento de todo o datagrama isto é, cabeçalho e dados;
- *Checksum* – Verificação de integridade do datagrama;



formato do segmento UDP

UDP checksum

Objetivo: detectar "erros" (ex.,bits trocados) no segmento transmitido

Transmissor:

- trata o conteúdo do segmento como seqüência de inteiros de 16 bits
- *checksum*: soma (complemento de 1 da soma) do conteúdo do segmento
- transmissor coloca o valor do *checksum* no campo de *checksum* do UDP

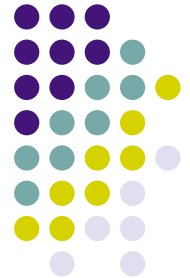
Receptor:

- computa o *checksum* do segmento recebido
- verifica se o *checksum* calculado é igual ao valor do campo *checksum*:
  - NÃO - erro detectado
  - SIM - não há erros

TCP – Transport Control Protocol

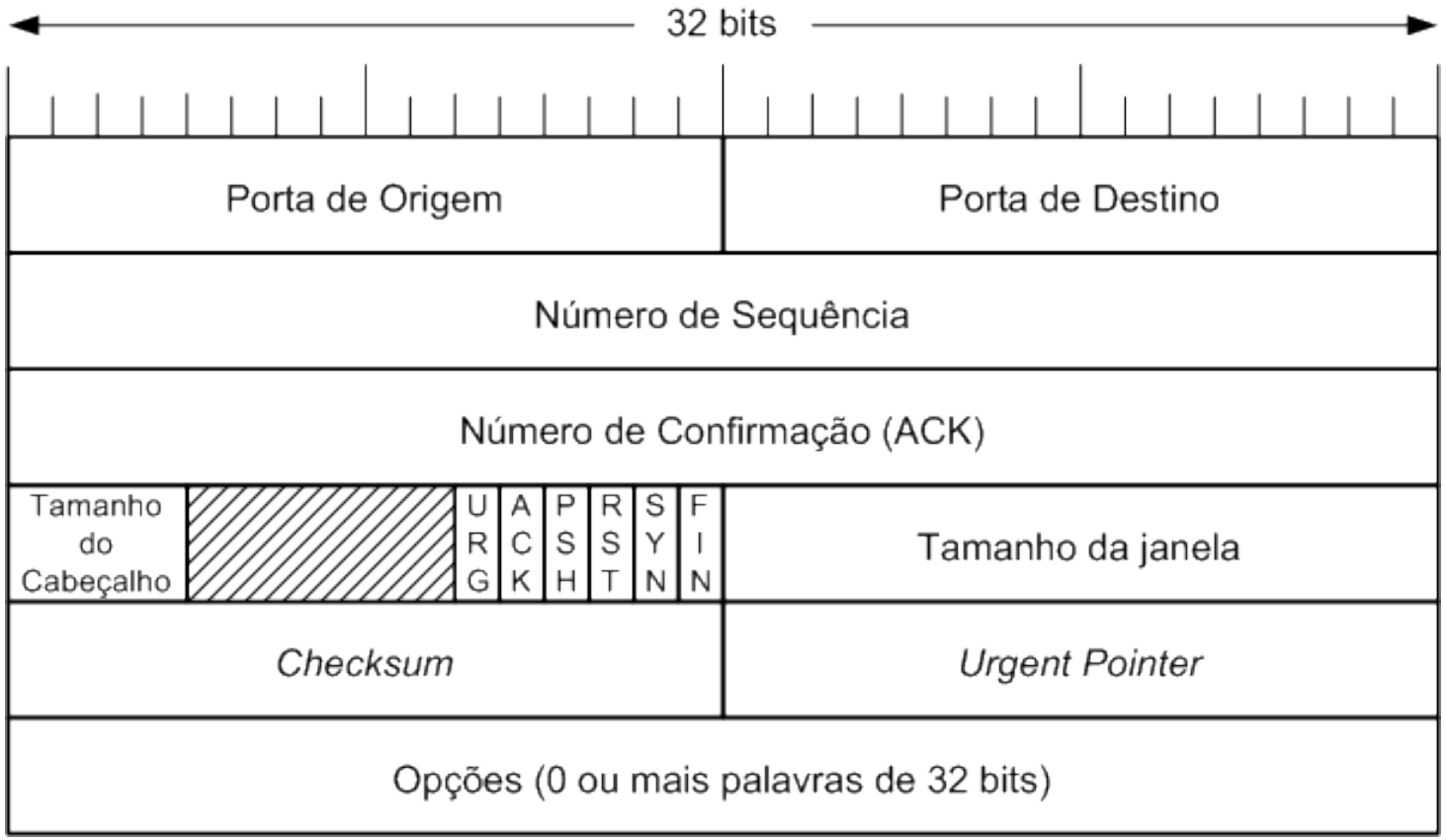
- O TCP (Protocolo de Controle de Transmissão) foi projetado para oferecer um fluxo de bytes fim a fim confiável em uma inter-rede não confiável.
- Ele é um protocolo orientado a conexão que permite a entrega sem erros de um fluxo de bytes originado de uma determinada máquina para qualquer computados da rede.
- Fragmenta o fluxo de entrada em mensagens e passa cada uma delas para a camada de redes. No destino, o processo TCP remonta as mensagens recebidas gerando o fluxo de saída. O TCP foi projetado para se adaptar dinamicamente às propriedades da camada de rede e ser robusto diante dos muitos tipos de falhas que podem ocorrer

TCP – Transport Control Protocol

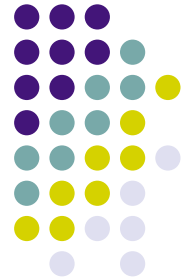


- O TCP foi formalmente definido na RFC 793, posteriormente alguns erros foram corrigidos e o TCP foi definido na RFC 1122.
- O TCP define um cabeçalho para suas mensagens composto dos seguintes campos:

TCP – Transport Control Protocol

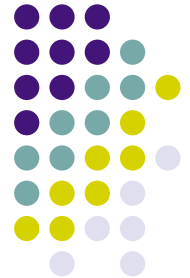


TCP – Transport Control Protocol



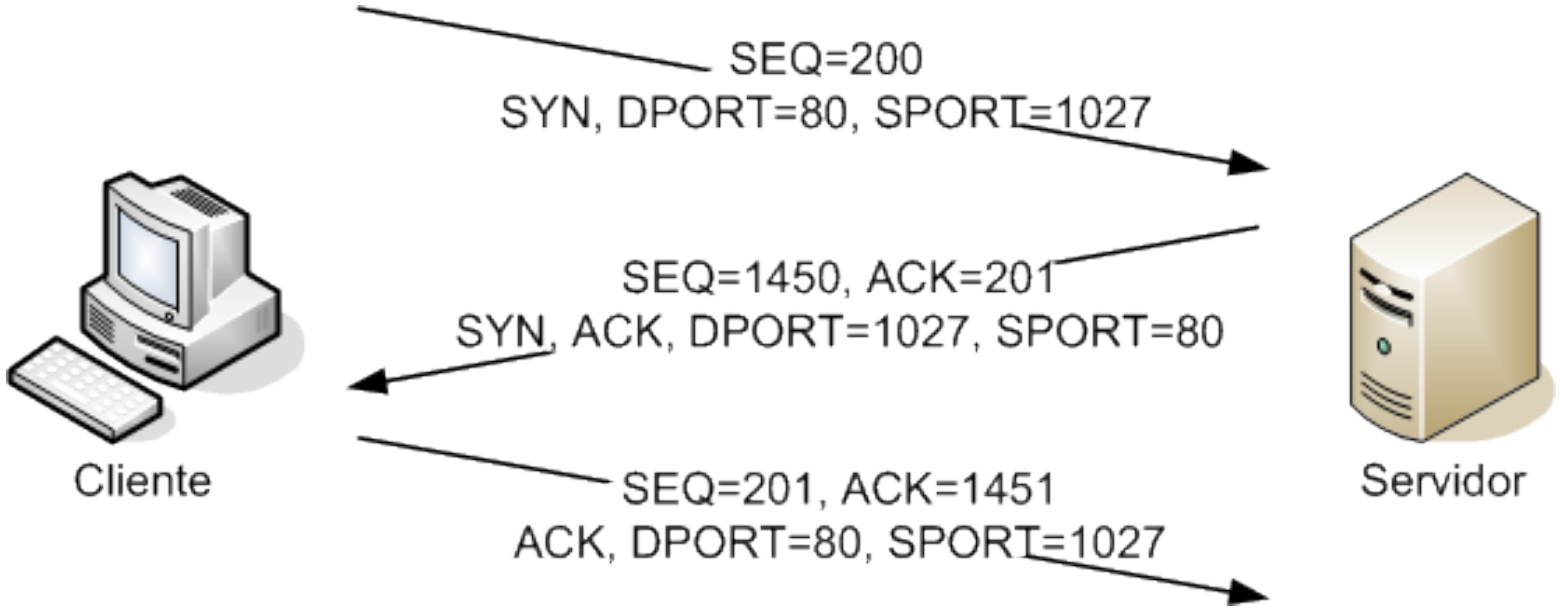
- **Porta de Origem e Porta de Destino** – identifica os pontos terminais locais da conexão;
- **Número de Seqüência –** Identifica o fragmento dentro de todo o fluxo gerado;
- **Numero de Confirmação –** Indica qual o próximo byte esperado;
- **Tamanho do Cabeçalho** – Informa quantas palavras de 32 bits compõem o cabeçalho TCP;
- **URG** – Indica a utilização do *urgent pointer*;
- **ACK** – É utilizado para indicar que este segmento é um ACK e que o campo Número de Confirmação deve ser interpretado;
- **PSH** – Indica que este segmento não deve ser enfileirado como todos os outros, mas sim posto à frente na fila;
- **RST** – É utilizado para reiniciar uma conexão que tenha ficado confusa devido a falhas no *host* ou por qualquer outra razão;
- **SYN** – Este bit é utilizado para indicar um pedido de conexão e a confirmação da conexão;
- **FIN –** Utilizado para indicar que o emissor não possui mais dados para enviar e deseja finalizar a conexão;
- **Tamanho da Janela** – Indica quantos bytes podem ser enviados a partir do byte confirmado. Este campo é utilizado no controle de fluxo do TCP;
- ***Checksum*** – Indicador de integridade do segmento;
- ***Urgent Pointer*** – Indica um deslocamento de bytes a partir do número de seqüência atual em que os dados urgentes devem ser encontrados;
- **Opções** – Projetado para que o TCP possa oferecer recursos extras que não foram previstos em seu protocolo;

Estabelecimento de Conexões

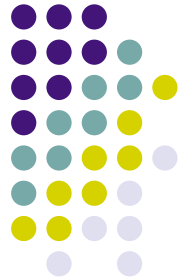


- O estabelecimento de uma conexão TCP ocorre antes que qualquer outro recurso TCP possa começar seu trabalho.
- O estabelecimento da conexão se fundamenta no processo de inicialização dos campos referentes à seqüência, aos ACKs e na troca dos números de *sockets* usados.
- As conexões são estabelecidas no TCP por meio do *three way handshake* (*handshake* de três vias).

Estabelecimento de Conexões

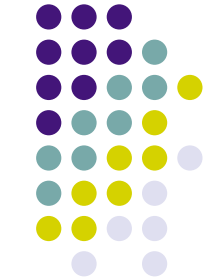


Estabelecimento de Conexões



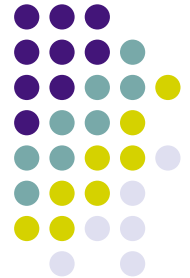
- O estabelecimento da conexão é feito usando dois bits na cabeçalho TCP: SYN e ACK.
- Um segmento que possua a *flag* SYN ativa sinaliza uma requisição de sincronia do número de seqüência. Essa sincronização é necessária em ambos os sentidos, pois origem e destino utilizam números de seqüência distintos.
- Cada pedido de conexão é seguido de uma confirmação utilizando o bit ACK.
- O segundo segmento do *three way handshake* exerce as duas funções ao mesmo tempo: Confirma a sincronização do servidor com o cliente e requisita a sincronização do cliente com o servidor.

Estabelecimento de Conexões



- Basicamente o *Three way handshake* 'simula' um acordo.
  - O Cliente pergunta pro servidor: "Você está ai?" e;
  - servidor responde: "Sim estou...".
  - Depois o servidor pergunta: "Você está ai?" e;
  - o cliente responde: "Sim estou...".
  - Mas espere! Como tem 4 sentenças em apenas 3 trocas de mensagens?? Simples, a segunda mensagem contém uma resposta e uma pergunta.

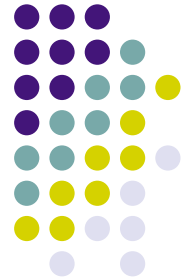
Estabelecimento de Conexões



- **Cliente:** Servidor, você está ai?? (SYN)
  Servidor: Sim estou... (ACK) E você, está ai? (SYN)
  Cliente: Sim, estou... (ACK)

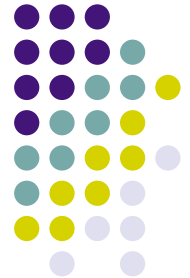
- O servidor precisa perguntar se o cliente está lá pela simples necessidade de sincronização do número de seqüência. O número de seqüência é utilizado para garantir a entrega de todas as mensagens.

Estabelecimento de Conexões



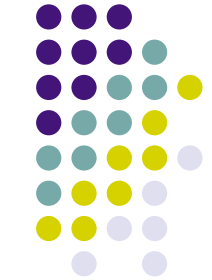
- **Cliente:** Cambio servidor, mensagem 2000 (Número de seqüência do cliente), o senhor está disponível (SYN)?
  Servidor: Positivo cliente! Mensagem 1450 (Numero de seqüência do servidor) Prossiga com a mensagem 2001 (ACK=2001), cambio.
  Cliente: Positivo servidor! Mensagem 2001 (numero de sequência), confirmando número da próxima mensagem: 1451, cambio!

Estabelecimento de Conexões



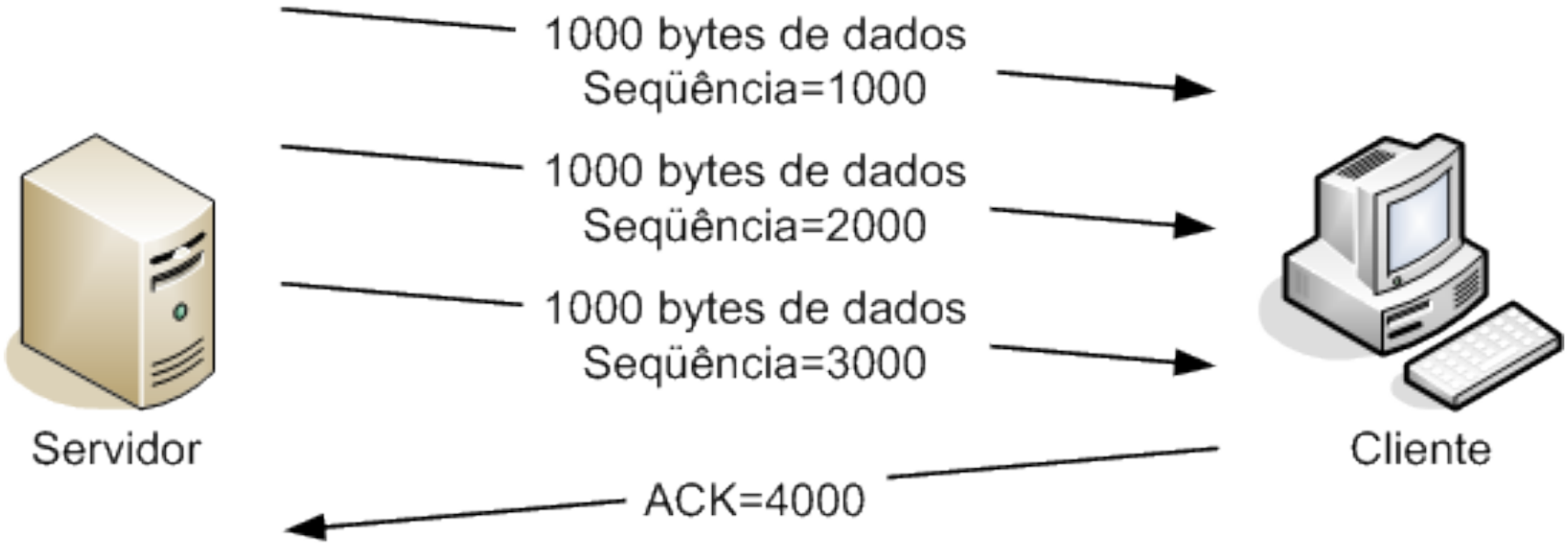
- Dessa forma eles trocam o número de seqüência, que tem como função “enumerar” as mensagens de cada um.
- Por exemplo, se a última mensagem foi a 2001 e a mensagem que chegou pro servidor foi a 2003, ele tem completa certeza que uma mensagem (2002) se perdeu no caminho! Então basta somente solicitar uma retransmissão.
- O número de seqüência nem sempre é incrementado por 1, ele pode ser incrementado com base no número de bytes enviados pela origem.
  O ACK tem como objetivo solicitar a continuidade das mensagens. Pode-se interpretar um ACK=210 como sendo: “Pronto, recebi até a 209, pode mandar a 210”.

Recuperação de Erros

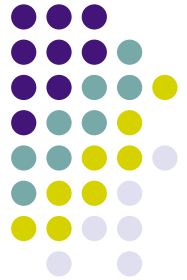


- O TCP proporciona uma transferência confiável de dados, o que também é chamado de confiabilidade ou recuperação de erros.
- Para conseguir a confiabilidade o TCP enumera os bytes de dados usando os campos referentes à seqüência e aos ACKs no cabeçalho TCP.
- O TCP alcança a confiabilidade em ambas as direções, usando um campo referente ao número de seqüência de uma direção, combinado com o campo referente ao ACK na direção oposta.

Recuperação de Erros

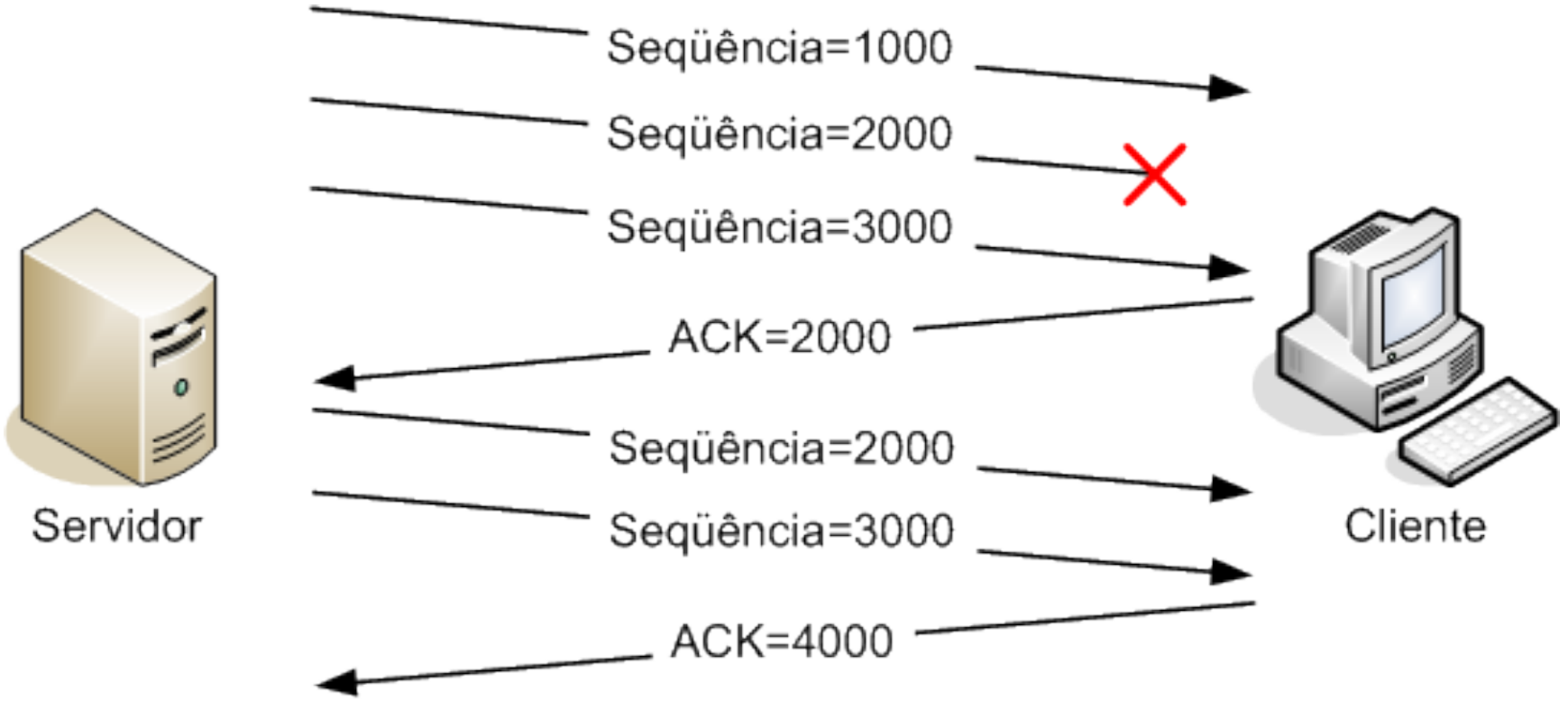


Recuperação de Erros



- Como dito anteriormente o campo ACK implica o próximo byte a ser recebido. O número de seqüência indica o número do primeiro byte do segmento correspondente à sua posição no fluxo de dados

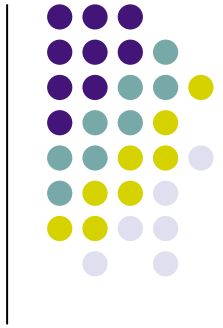
Recuperação de Erros



Portas de Serviços

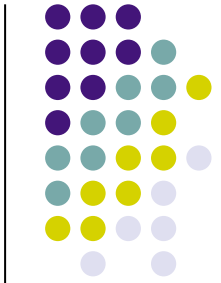
- A numeração de portas utiliza 16 bits. Logo (2^16) existem ao todo 65536 portas a serem utilizadas.
- Isso para cada protocolo da camadas de transporte. Logo 65536 portas para o TCP, e 65536 portas para o UDP, 65536 portas.
- Conhecer essas portas é fundamental para operar um *Firewall* de forma satisfatória.

Portas de Seviço



- Com tanta porta como saber todas elas??".
- Não precisa conhecer todas, uma vez que a maior parte delas não são especificadas.
- Apenas as primeiras 1024 são especificadas.
- Para um administrador de rede é imprescindível saber pelo menos as portas dos serviços básicos de Rede: telnet, SSh, FTP, SMTP, POP, HTTP, HTTPS... Não são muitas, mas antes de ver isso, vamos entender que controla essas portas.
- O uso das portas de 1 a 1024 é padronizada pela IANA (***I****nternet* ***A****ssigned* ***N****umbers* ***A****uthority*).
  - Essa entidade é responsável por alocar portas para determinados serviços. Essas portas são chamadas de ***well-known ports***.

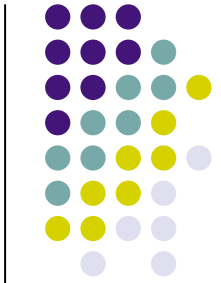
Portas de Serviço



- Lista de algumas principais portas TCP:

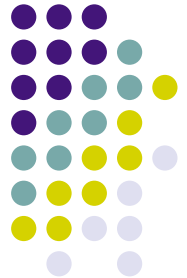
serviço	porta	protocolo
daytime	13	tcp e udp
ftp-data	20	tcp
ftp	21	tcp
ssh	22	tcp
telnet	23	tcp
smtp	25	tcp e udp
name	42	tcp e udp
nameserver	42	tcp e udp
tftp	69	tcp e udp
www	80	tcp
pop3	110	tcp e udp
netbios-ns	137	tcp e udp
netbios-dgm	138	tcp e udp
netbios-ssn	139	tcp e udp

Camada de Transporte



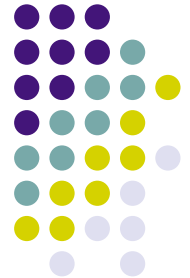
- Camada de transporte – é o coração de toda a hierarquia de protocolos.
  - Função – “promover uma transferência de dados confiável, eficiente e econômica entre a máquina de origem e a máquina de destino , INDEPENDENTEMENETE da(s) rede(s) física(s)”
- Utiliza vários serviços oferecidos pela camada de redes.

Camada de Transporte

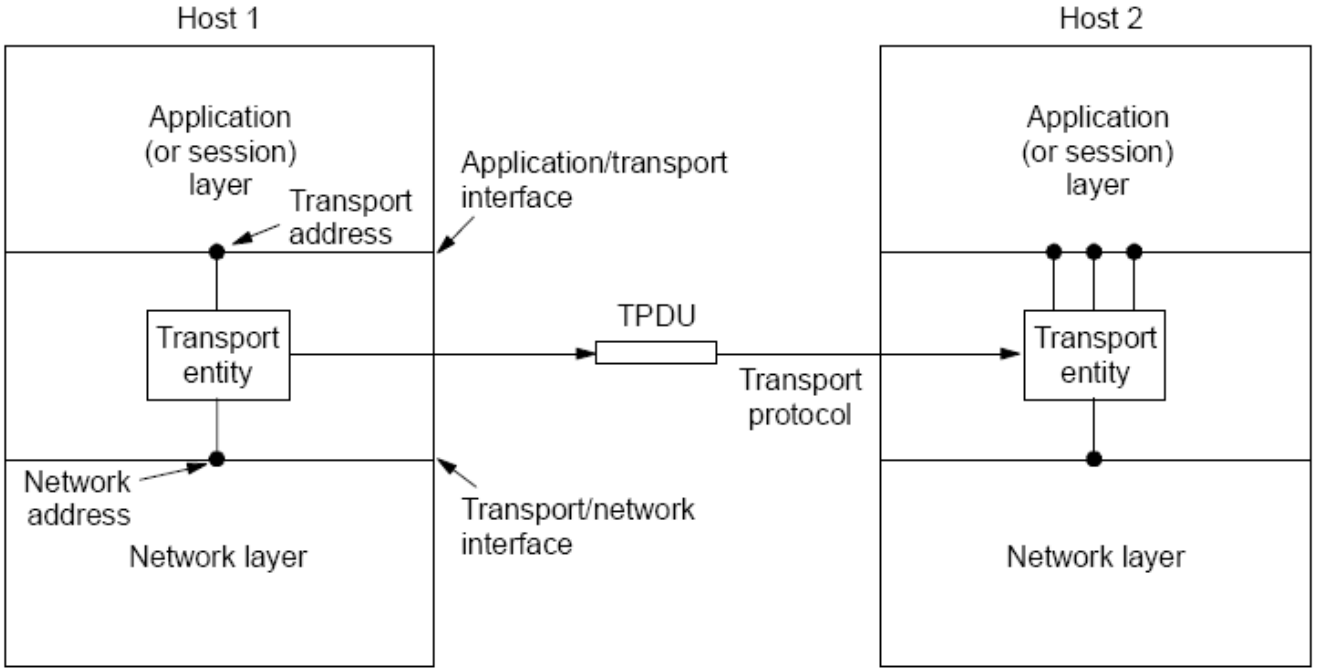


- O hardware e/ou software da Camada de Transporte que executa o trabalho é chamado de **entidade de transporte** (*transport entity*).
- Esta entidade de transporte por estar…
  - …no kernel do sistema operacional;
  - … em um outro processo de usuário;
  - …em uma biblioteca vinculada a aplicações de rede;
  - …ou em uma placa de interface de rede.

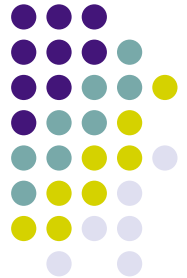
Camada de Transporte



- Como já mencionado – existem dois tipos de serviços na camada de transporte – Orientados a conexão e sem conexão.

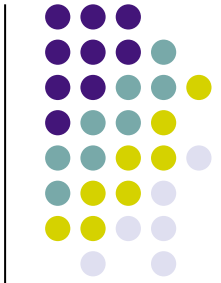


Camada de Transporte



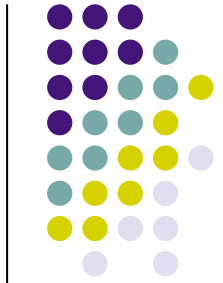
- O serviço da camada de transporte orientado à conexão é semelhante ao serviço de rede orientado à conexão – existem semelhanças no que diz respeito ao processo de conexão (que acontece em 3 fases), endereçamento e controle de fluxo.
- Analogamente, o serviço de transporte sem conexão também é semelhante ao serviço da camada de rede sem conexão.

Camada de Transporte



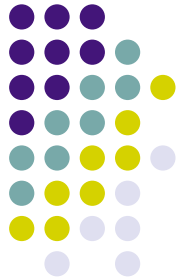
- Pergunta-se:
  - “Se o serviço da camada de transporte é tão semelhante ao serviço da camada de rede, porque há duas camadas distintas?”
- A resposta é sutil porém de grande importância:
  - A camada de rede faz parte da sub-rede de comunicação e, geralmente é controlada pelas concessionárias de telecomunicação (EMBRATEL, telefônica, entre outros…). Em geral os processos usuários não têm controle sobre esta camada.

Camada de Transporte



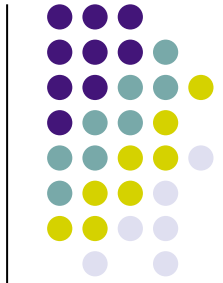
- Então, o que acontece se:
  - A camada de rede oferecer um serviço orientado à conexão, não confiável?
  - Se a camada de rede perder pacotes com freqüência?
  - Os roteadores apresentarem falhas de vez em quando?

Camada de Transporte



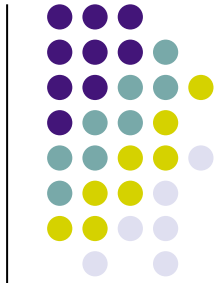
- Os processos usuários não possuem controle sobre a sub-rede – não podem resolver o problema de um serviço ineficaz (não podem trocar os roteadores ou aumentar o controle de erro da camada de enlace).
- Única possibilidade – colocar sobre a camada de rede um outra camada que melhore a qualidade dos serviços.

Camada de Transporte



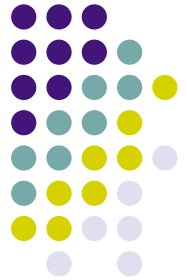
- Em resumo:
  - A existência da camada de transporte torna o serviço de transporte entre processos muito mais confiável do que o serviço de rede adjacente.
  - A ocorrência de pacotes perdidos e/ou de dados corrompidos pode ser percebida e compensada pela camada de transporte.

Camada de Transporte



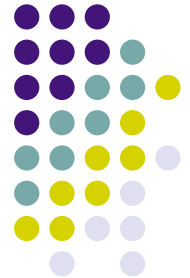
- Primitivas do serviço de transporte – podem ser projetadas para serem independentes das primitivas do serviço de rede, que podem variar muito de sub-rede (ou de rede) para sub-rede.
- Por exemplo: um serviço LAN sem conexão, pode ser muito diferente de um serviço WAN orientado à conexão.

Camada de Transporte



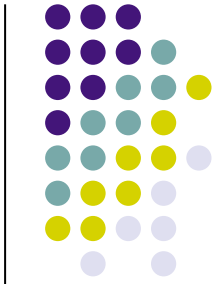
- Graças à Camada de Transporte – os programas aplicativos podem ser desenvolvidos utilizando-se um conjunto de primitivas-padrão, podendo funcionar em uma ampla variedade de redes, sem a preocupação de se lidar com diferentes interfaces de sub-redes e com uma eventual transmissão duvidosa.
  - Se todas as redes reais fossem iguais e perfeitas não seria preciso camada de transporte.

Protocolos de Transporte da Internet



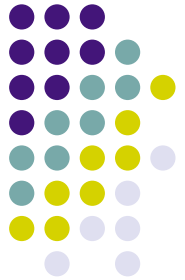
- Internet – possui 2 protocolos principais na camada de transporte: orientado a conexão (TCP) e sem conexão (UDP).

TCP



- Projetado para oferecer um fluxo de bytes fim a fim, confiável, em uma inter-rede (*internet*) não-confiável
- Definido pelo RFC-793, com esclarecimentos e revisões no RFC-1122 e extensões e melhoramentos no RFC-1323.

TCP

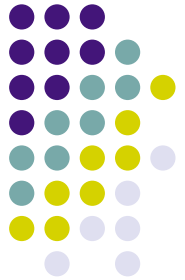


- Nas internets (inter-redes) – muitas partes podem ter topologias, larguras de banda, tamanhos de pacotes, e outros parâmetros completamente diferentes.
- TCP – projetado para se adaptar dinamicamente às propriedades das inter-redes, e para ser robusto diante de muitos tipos de falhas que podem ocorrer.

TCP

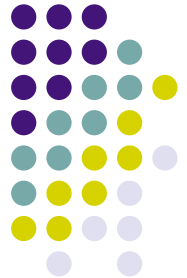
- Cada máquina compatível com o TCP – possui uma entidade de transporte TCP, que pode ser uma parte do *kernel* ou um processo de usuário.
- Esta entidade de transporte TCP – gerencia os fluxos e interfaces TCP para a camada IP.
- Uma entidade TCP aceita fluxos de dados dos usuários, vindos dos processos locais, divide-os em partes de no máximo 64 Kb (na prática o normal é 1.500 bytes) e envia cada parte em um datagrama IP separado.

TCP



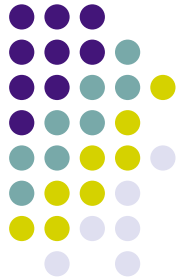
- Quando os datagramas que contêm dados TCP chegam ao seu destino – são enviados à entidade TCP, a qual restaura os fluxos originais.
- Camada IP – não fornece qualquer garantia de que os datagramas serão entregues de forma apropriada – cabe à camada TCP administrar os temporizadores, e re-transmitir os datagramas, sempre que necessário.

TCP



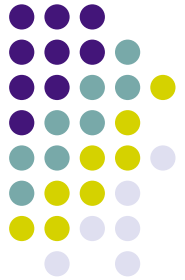
- Os datagramas também podem chegar fora de ordem – o TCP também terá que reorganizá-los em mensagens na seqüência correta.

Endereçamento



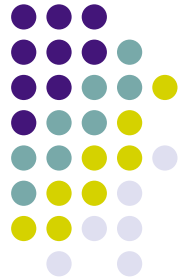
- Quando processo de aplicação (sessão) deseja estabelecer uma conexão com um processo de aplicação remoto – é preciso especificar a aplicação com quem se conectar.
- Se for sem conexão – há o mesmo problema – a quem cada mensagem deve ser enviada.
- Método – definir os endereços de transporte, os quais os processos podem ouvir, para receber solicitações de conexão.

Endereçamento



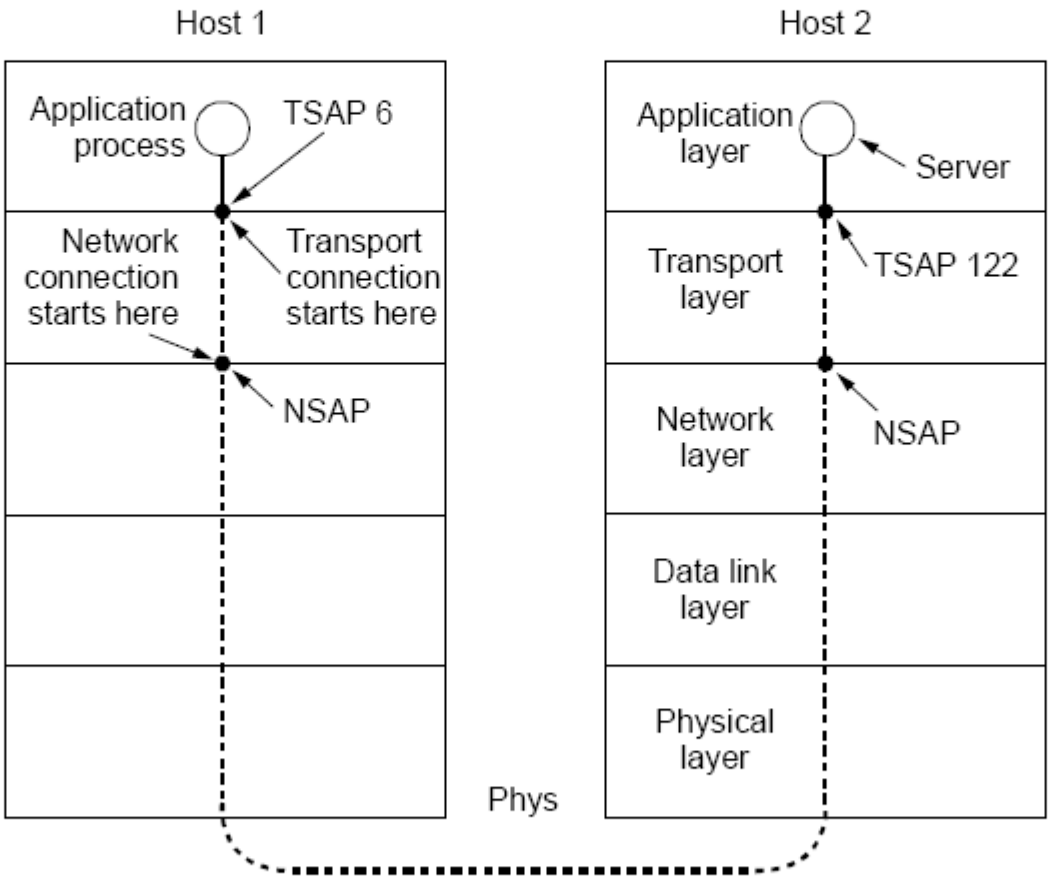
- Na Internet – esses endereços formam pares – (endereço IP, porta local).
- Eles são os TSAPs – *Transpot Service Access Point*
- Os pontos finais análogos na camada de rede, ou seja, os endereços da camada de rede são os NSAPs – *Network Service Access Point* – exemplo: endereço IP

Relação entre o NSAP e TSAP



- A entidade de transporte pode aceitar vários TSAPs.
- Em determinadas máquinas, podem existir vários NSAPs, no entanto, em outras, cada máquina tem um único NSAP – por exemplo, um endereço IP.

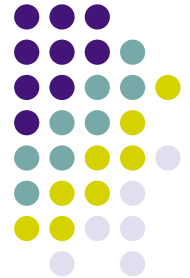
Relação entre o NSAP e TSAP



Cenário para uma conexão de transporte

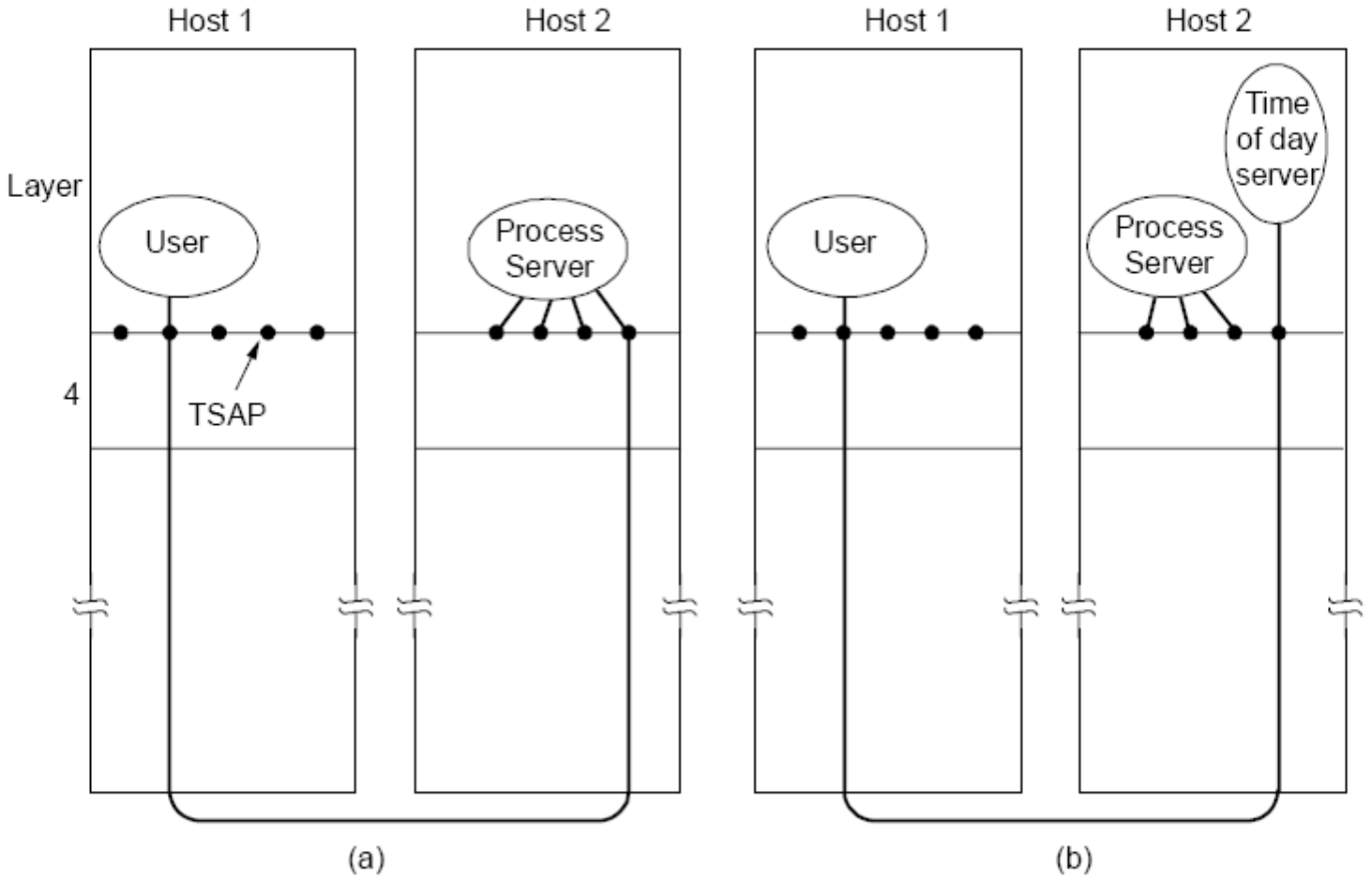
- Um processo servidor de resposta para hora do dia no Host 2 – associa-se ao TSAP 122 e aguarda a chegada de uma chamada.
- Um processo de aplicação no Host 1, deseja saber a hora do dia e transmite uma solicitação CONNECT – especificando o TSAP 6 como origem, e o TSAP 122 como destino.
- A entidade de transporte do Host 1 – seleciona um endereço de rede (NSAP) em sua máquina (se houver mais de 1) e estabelece uma conexão de rede entre eles (em um rede sem-conexão não seria possível estabelecer essa conexão da camada de rede).
- Usando esta conexão de rede – a entidade de transporte do Host 1 pode se comunicar com a entidade de transporte do Host 2.

Cenário para uma conexão de transporte

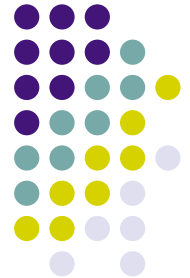


- Entidade de transporte no Host 1 diz a sua correspondente no Host 2
  - Olá. Eu gostaria de estabelecer uma conexão de transporte entre o meu TSAP 6 e o seu TSAP 122. É possível?
- A entidade de transporte do Host 2 – pergunta ao seu servidor de hora do dia no TSAP 122 se ele está disposto a aceitar uma nova conexão. Se ele concordar, a conexão de transporte é estabelecida, e os demais protocolos entram em ação para realizarem a operação desejada.

Cenário para uma conexão de transporte

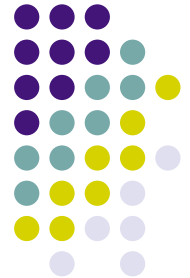


Cenário para uma conexão de transporte



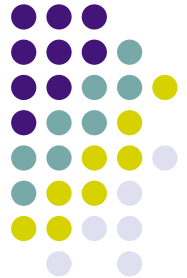
- Problema – como o processo de usuário do Host 1 sabe que o servidor de hora do dia está associado ao TSAP 122?
- Possibilidade – o endereço deste serviço é estável, e tradicionalmente mantido neste TSAP.
- Não é uma boa opção – só um pequeno número nunca muda; limita-se a administração dos recursos e de configurações e, além disso, em geral os processos de usuário desejam se comunicar com outros processos de usuário que existam por um curto período de tempo e, assim, não tenham endereço TSAP previamente conhecido.

Cenário para uma conexão de transporte



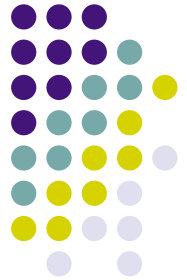
- Esquema melhor – Protocolo de Conexão Inicial (*Initial Connection Protocol*).
- Ao invés de ter todos os servidores associados a um TSAP conhecido – cada máquina tem um servidor de processos (*process server*) – ele atende uma série de portas ao mesmo tempo, aguardando um conexão TCP.
- Os usuários começam com uma solicitação CONNECT especificando o endereço TSAP (porta TCP) do serviço que necessitam. Se nenhum servidor estiver aguardando nesta porta – eles estabelecem uma conexão com o servidor de processos.

Cenário para uma conexão de transporte



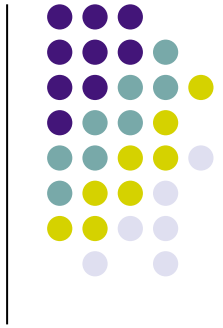
- Depois de receber a solicitação – o servidor de processos “acorda” o serviço necessário, coloca o processo solicitante em contato com o TSAP do servidor, e encerra a conexão com o servidor solicitado, permitindo que ele “herde” a conexão já existente com o usuário.

O serviço TCP



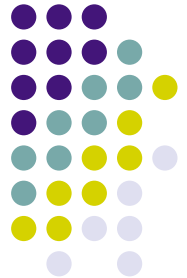
- Serviço TCP – é obtido quando tanto o receptor quanto o transmissor criam pontos terminais de interconexões chamados *sockets*.
- Cada *socket* tem um número (endereço) que consiste no endereço IP do host, mais um número local de 16 bits para este host – trata-se da PORTA.
- Para que um serviço TCP funcione – é necessário que uma conexão seja explicitamente estabelecida entre um *socket* de uma máquina transmissora e um *socket* de uma máquina receptora.

O serviço TCP



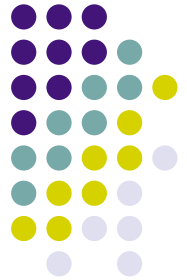
- Um *socket* pode ser usado por várias conexões ao mesmo tempo – isto é, duas ou mais conexões podem terminar no mesmo *socket*.
- Conexões – são identificadas nas duas extremidades pelos identificados de *sockets* (*socket1, socket2*)

O serviço TCP



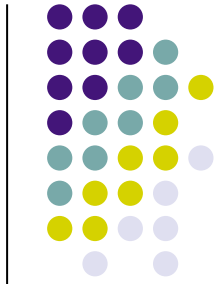
- Todas as conexões TCp são *full-duplex* e ponto a ponto
  - *Full duplex* – porque o tráfego pode ser feito em ambas as direções ao mesmo tempo
  - Ponto-a-ponto: porque cada conexão possui exatamente dois pontos terminais únicos.
- Uma conexão TCP é um fluxo de dados e não um fluxo de mensagens. As fronteiras das mensagens não são preservadas de uma extremidade a outra – se por exemplo um processo transmissões executar quatro escritas de 512 bytes cada no fluxo TCP, estes dados poderão ser entregues ao processo receptor em 4 partes de 512, em 2 de 1024 ou uma de 1048, ou em qualquer outra divisão – não há um meio do receptor detectar as unidades em que os dados foram escritos.

O serviço TCP



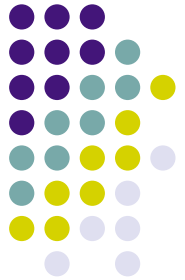
- O software TCP não tem idéia do significado dos bytes. Ele apenas cuida da entrega.
- Quando uma aplicação passa dados para a entidade TCP – ele (TCP) pode enviá-los imediatamente, ou armazená-los em um *buffer* (para aguardar outros dados e enviar um volume maior de uma só vez), de acordo com as necessidade e condições.

O serviço TCP



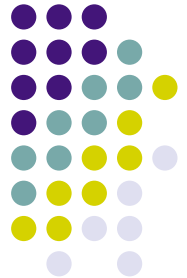
- Um recurso interessante – dados urgentes (*urgent data)*. Quando um usuário interativo pressiona uma tecla <CTRL-C> ou <CTRL-Q> para interromper um processo remoto já iniciado – a aplicação transmissora adiciona uma informação de controle à cadeia de dados, e a entrega juntamente com um *flag URGENT*.

O serviço TCP



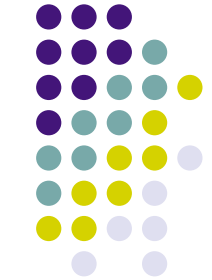
- Isso faz com que o serviço TCP para de acumular dados e transmita tudo imediatamente. Quando os dados urgentes são recebidos no destino – a aplicação receptora é interrompida (em terminologia UNIX ela recebe um “signal”) e pára tudo o que estiver fazendo para ler o luxo de dados e encontrar os dados urgentes.

Especificação do Protocolo



- As entidades TCP transmissoras e receptoras – trocam dados na forma de segmentos.
- Segmento – consiste de um cabeçalho fixo de 20 bytes (mais uma parte opcional), seguido de zero ou mais bytes de dados.
- O software TCP decide qual deve ser o tamanho dos segmentos. Ele pode acumular dados de várias escritas em um único segmento, ou dividir os dados de uma única escrita em vários segmentos.

Especificação do Protocolo

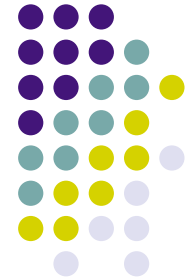


- Dois fatores restringem o tamanho do segmento:
  - Cada segmento, incluindo o cabeçalho TCP, deve caber na carga útil do IP, que [e 65.535 bytes.
  - Como já discutido cada tipo de rede tem uma MTU – *Maximum Transfer Unit*, e cada segmento deve caber na MTU. Na prática, a MTU tem alguns milhares de bytes (aprox. 1500bytes) e, portanto, define o limite superior em termos de tamanho de segmento.

Especificação do Protocolo

- Se um segmento atravessar uma seqüência de redes sem ser fragmentado, e depois chegar em uma rede cuja MTU não o comporta – o roteador localizado na fronteira fragmentará o segmento em dois ou mais segmentos menores. Cada novo segmento receberá seus próprios cabeçalhos TCP e IP – portanto a fragmentação realizada pelos roteadores aumenta o *overhead* total – cada novo segmento adiciona 40 bytes de informação de *header*.

O cabeçalho do segmento TCP

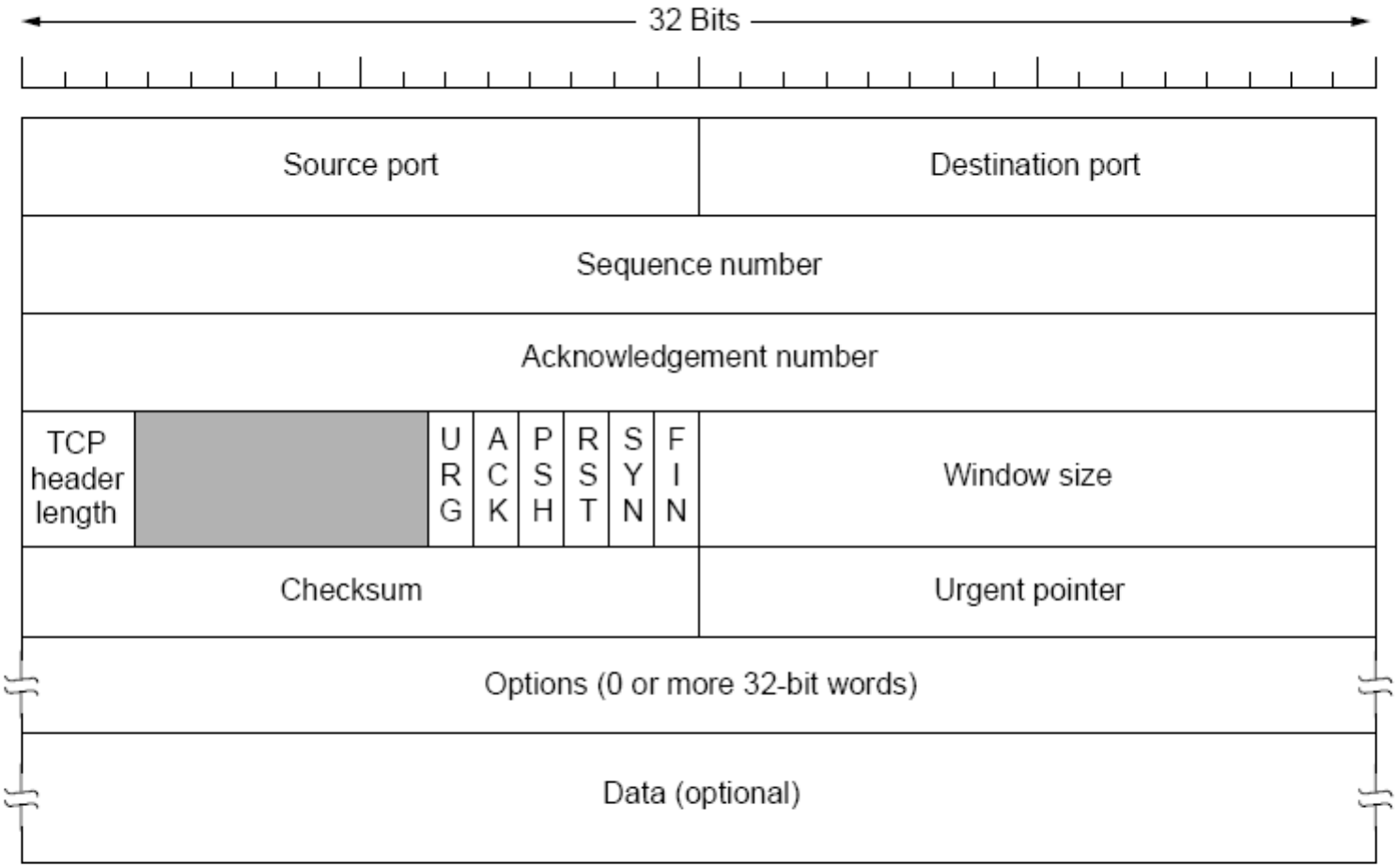


- Cada segmento TCP – Começa com um cabeçalho de formato fixo de 20 bytes.
- Em seguida ao cabeçalho fixo – podem haver opções de cabeçalho.
- Depois destas opções (se houver alguma) pode haver:
  - 65535 – 20-20 = 65515 bytes de dados
- Os 20 primeiros bytes se referem ao cabeçalho IP, e os outros 20 ao cabeçalho TCP

O cabeçalho do segmento TCP

- Os segmentos sem dados são válidos e, em geral são utilizados para confirmação e mensagens de controle.

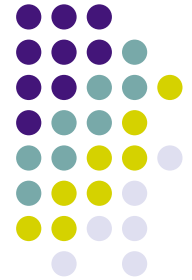
O cabeçalho do segmento TCP



O cabeçalho do segmento TCP

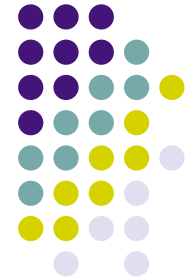
- *Souce port* e *Destination port* – identificam os pontos terminais de conexão. Cada host decide como alocar suas próprias portas, começando em 256.
  - Uma porta e o endereço IP de seu host – formam um TSAP único de 48 bits.
  - O número de *sockets* de origem e destino juntos – identificam a conexão.
- *Sequence number* e *acknoledgement number* – montagem e verificação dos segmentos. Cada um em 32 bits.
  - Note que o *Acknoledgement Number* especifica o último segmento aguardado e não o último segmento recebido corretamente

O cabeçalho do segmento TCP



- TCP *header length* – informa quantas palavras de 32 bits existem no cabeçalho TCP. Essa informação é necessária pois o campo OPTIONS tem tamanho variável portanto o mesmo acontece com o cabeçalho.
  - Tecnicamente – este campo indica o início dos dados dentro do segmento, com base em palavras de 32 bits. Este número é o tamanho do cabeçalho, em palavras.

O cabeçalho do segmento TCP



- Em seguida há um campo de 6 bits que não é atualmente utilizado.
- Depois há seis *flags* de 1 bit. Estes *flags* são usados da seguinte forma:
  - ACK - Valor 1 atribuído a este bit para indicar que o *acknolegement number* é válido (ou seja, que o ACK *number* deve ser considerado). Se for igual a zero, isso significa que o segmento não contem uma confirmação e, portanto, o campo *acknoledgement number* não deve ser considerado.

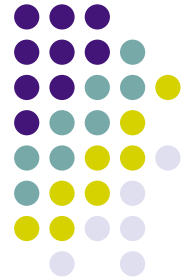
O cabeçalho do segmento TCP

- PSH – indica dados exigindo o *flag* PUSH, Significa que o receptor é solicitado a entregar os dados à aplicação imediatamente após sua chegada, ao invés de armazená-la até encher o *buffer*.
- RST – usado para re-inicializar uma conexão que tenha ficado confusa ou paralisada devido a uma falha no *host*, ou por qualquer outra razão. Ele também é usado para rejeitar um segmento inválido, ou para recusar uma tentativa de conexão.

O cabeçalho do segmento TCP

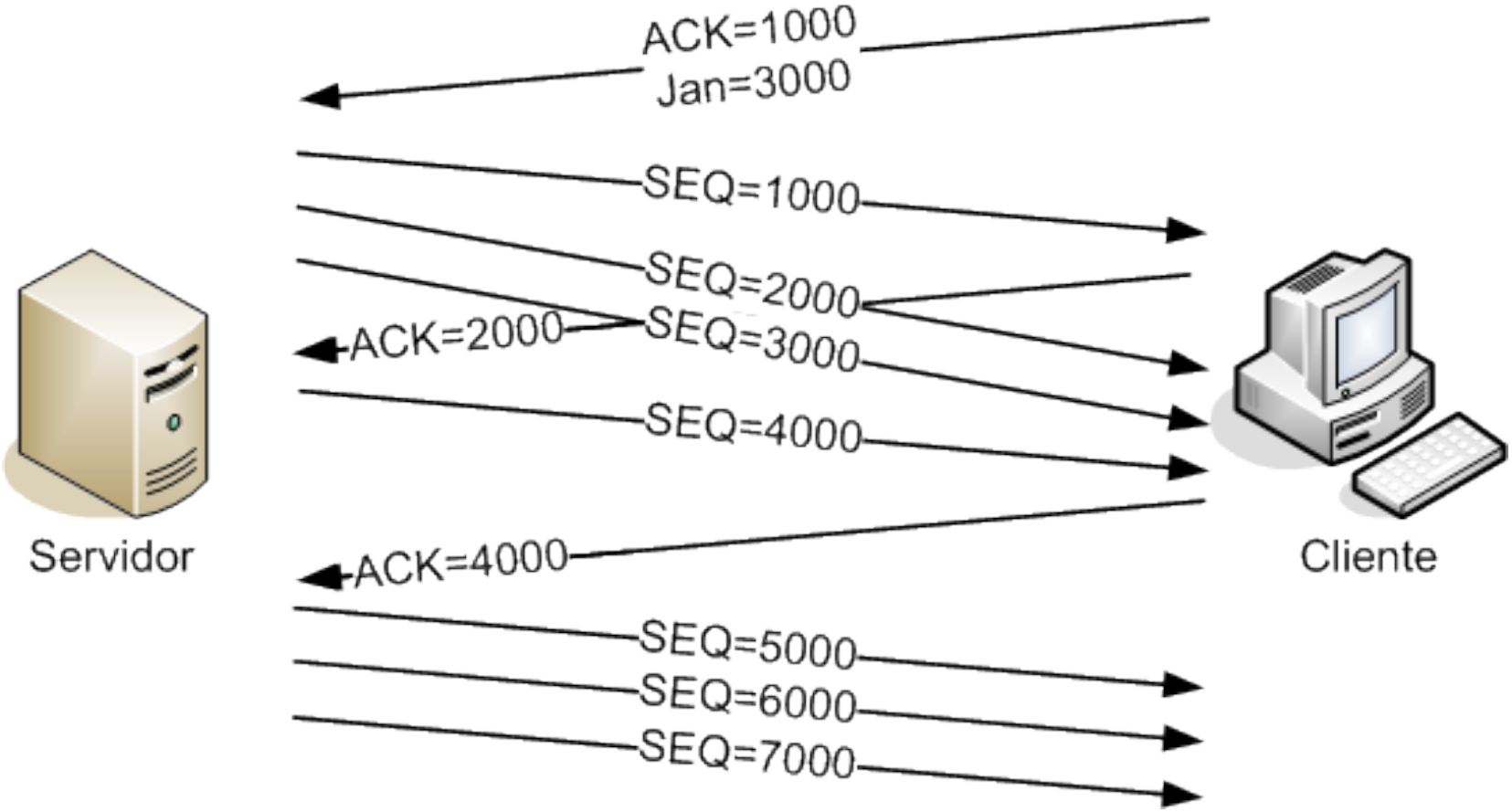
- SYN – é usado para estabelecer conexões
  - A solicitação de conexão tem SYN=1 e ACK=0 para indicar que o campo de confirmação não está sendo utilizado.
  - A resposta contém uma confirmação – portanto tem SYN=1 e ACK=1
  - Basicamente o bit SYN é usado para denotar CONNECTION REQUEST e CONNECTION ACCEPTED, enquanto o bit ACK combinado com o SYN é usado para distinguir entre estar duas possibilidades
- FYN – usado para encerrar uma conexão. Ele indica que o transmissor não tem mais dados para enviar.

Controle de Fluxo



- O TCP implementa o controle de fluxo utilizando os dados dos campos Número de Seqüência, Número de Confirmação e Tamanho da Janela.
- O controle de fluxo no TCP – é gerenciado por uma técnica denominada janela que pode utilizar uma janela de tamanho fixo ou uma janela deslizante.
- O campo Tamanho da Janela indica, em tempo real, o número máximo de bytes sem confirmação que podem ser enviados. Com a utilização de janelas um emissor só poderá enviar o número de bytes, previsto na janela, antes de receber alguma confirmação.

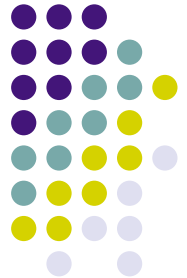
Comunicação Sem Janela Fixa



Comunicação Sem Janela Fixa

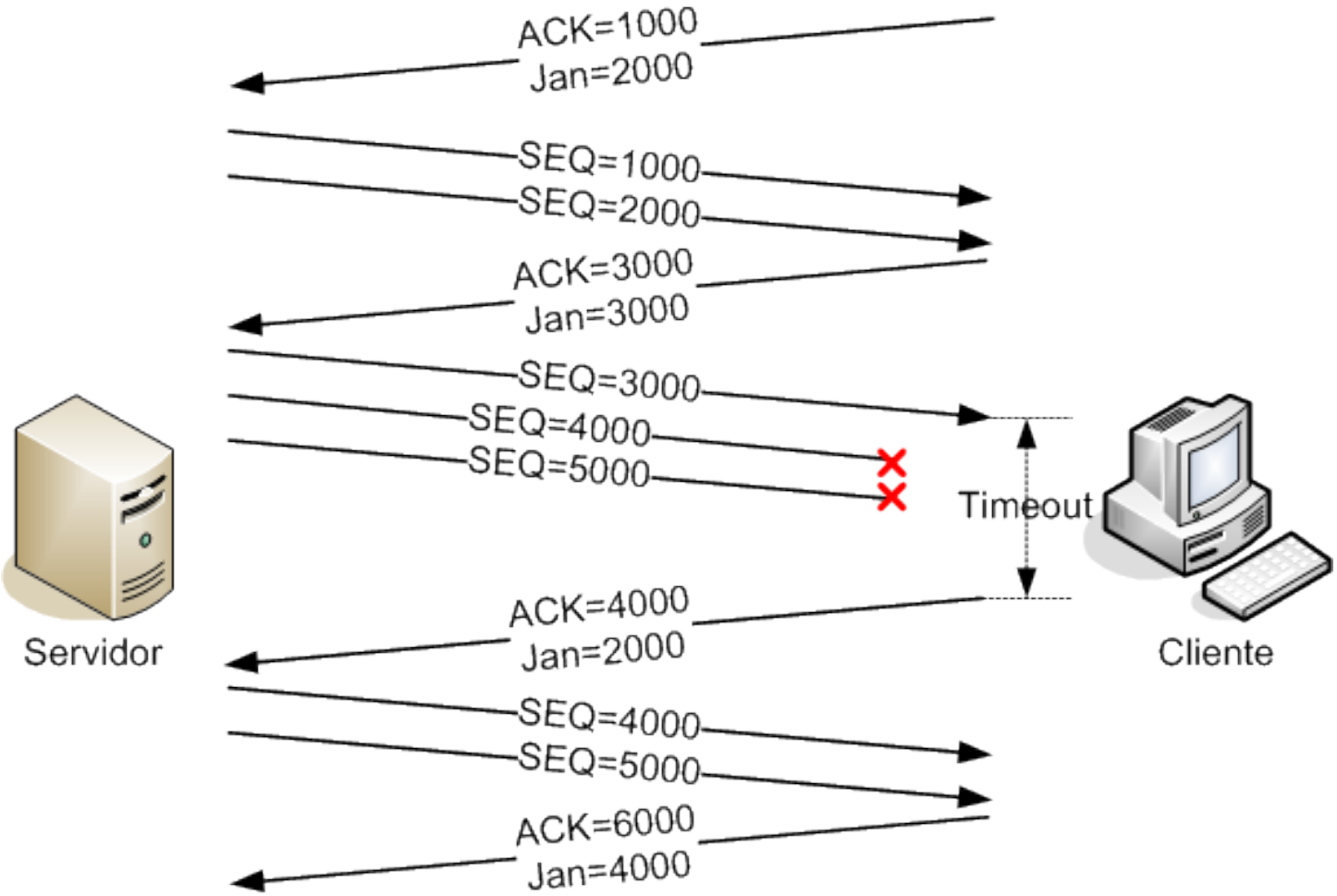
- O Cliente fala pro servidor: Servidor, estou um pouco ocupado mas quero continuar esse *download*. Por favor, não me envia mais que 3000 (tamanho da janela) bytes não confirmados OK??
- O servidor enviar 3 pacotes cada um com 1000 bytes, mas por algum motivo o primeiro chega mais rápido e os dois últimos demoraram um pouco.
- O cliente recebe o pacote de seqüência 1000 espera e não recebe mais nada. Então ele envia uma confirmação: “Vamos Servidor!! Eu falei 3000!! Só recebi 1000, manda o próximo (ACK)”
- O servidor recebe esse ACK e verifica que enviou 3000 mas só 1000 foram confirmados, ou seja tem 2000 não confirmados e 1000 de espaço livre. Então ele envia mais 1000.
- O cliente recebe todos os pacotes, então ele responde: “Ok, recebi até o 4000, pode enviar o 5000!!! (ACK)”, e dessa forma o servidor manda mais 3000 bytes.

Comunicação com Janelas Deslizantes

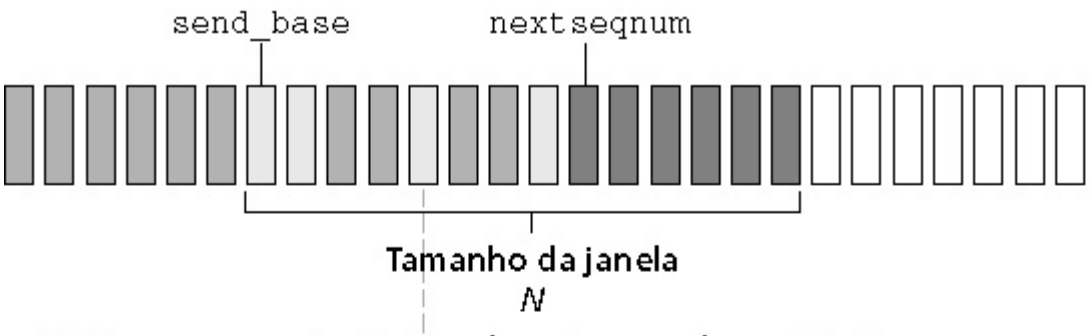


- Caso o protocolo TCP esteja utilizando janelas deslizantes o tamanho da janela irá variar ao longo de uma transmissão.
- Ao iniciar uma conexão a janela começa pequena e aumentará gradativamente até que ocorram erros ou o destinatário seja sobrecarregado.
- Ao serem detectados erros a janela diminui, após um tempo o tamanho da janela começa a aumentar novamente.
- Caso o destinatário perceba uma sobrecarga, no próximo ACK enviado por ele haverá um novo tamanho de janela, o qual ele acredita ser apropriado para sua recuperação.
- Caso seja enviado um valor igual à zero o destinatário estará informando que não possui condições de processar mais dados e a comunicação estará suspensa até que o remetente receba um tamanho de janela diferente de zero.

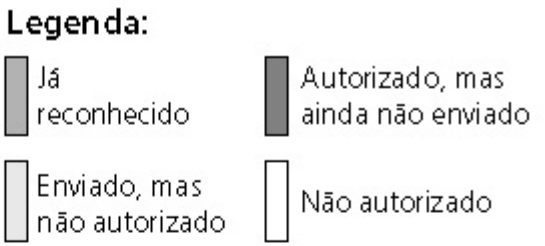
Comunicação com Janelas Deslizantes

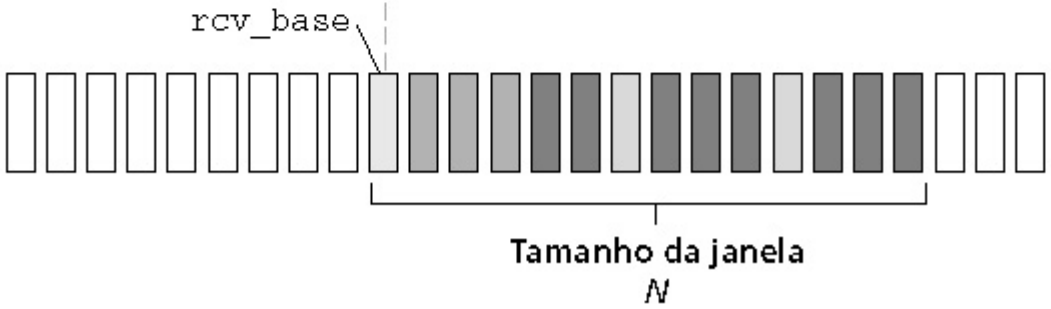


Comunicação com Janelas Deslizantes

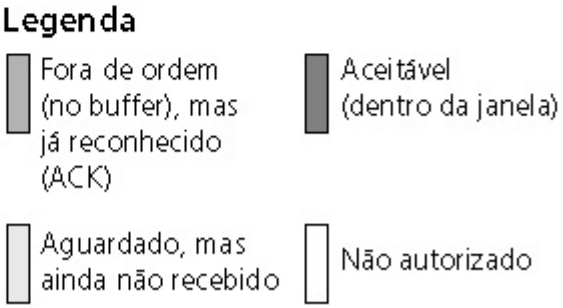


a. Visão que o remetente tem dos números de seqüência

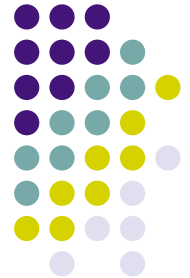




b. Visão que o destinatário tem dos números de seqüência



Comunicação com Janelas

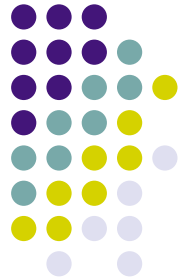


- Campo *Window* – está relacionado com o controle de congestionamento, indica quantos bytes podem ser enviados a partir do último byte confirmado.
  - Campo *window* igual a 0 é válido – informa que todos os dados até *ack number* -1, inclusive, foram recebidos, mas que o receptor precisa de uma folga no momento, e pede para interromper o envio de dados. A permissão para retomar a transmissão pode ser dada depois, com o mesmo *ack number* e com um campo *Windows* diferente de zero.

Comunicação com Janelas

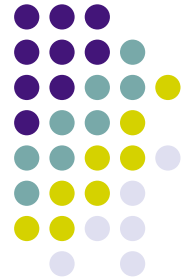
- Campo *checksum* – também é fornecido para aumentar a confiabilidade. Ele confere a soma de verificação do cabeçalho, dos dados e do pseudo-cabeçalho (formador e originador do IP) incluído.
- Metodologia – ao efetuar este cálculo o campo *checksum* é definido como zero, e o campo de dados é preenchido com um byte zero adicional, caso seu tamanho seja um número ímpar.

Comunicação com Janelas



- Em seguida – o algoritmo de soma de verificação simplesmente soma todas as palavras de 16 bits em complementos de 1, e depois tira os complementos de 1 da soma.
- Quando o receptor efetuar o cálculo no segmento inteiro, incluindo o campo *checksum*, o resultado deve ser zero.

Comunicação com Janelas



- O pseudo-cabeçalho – contém dados IP de 32 bits das máquinas origem e destino, o número de protocolo para o TCP (que é igual ao decimal 6), e a contagem de bytes para o tamanho do segmento TCP, incluindo o cabeçalho

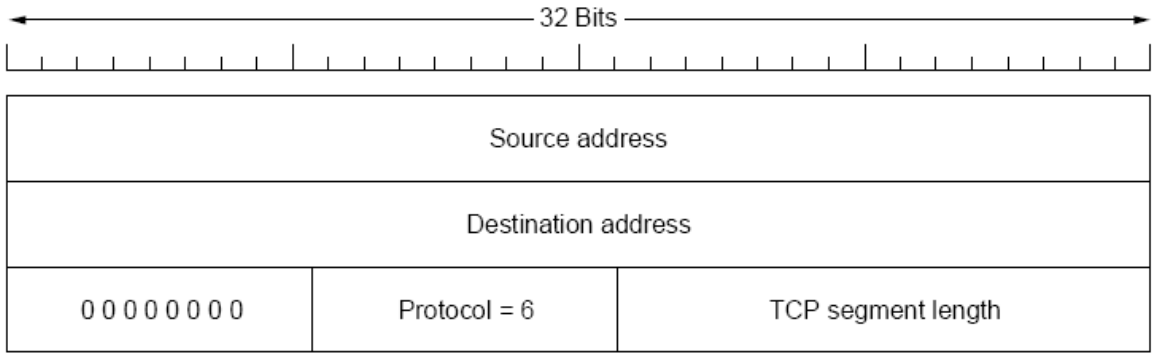
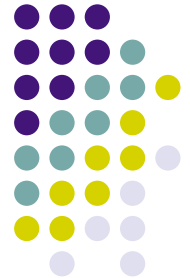


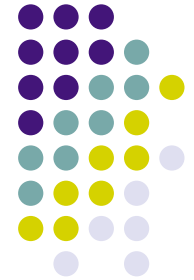
Figura 5.5: O *pseudo-header* usado no *checksum* do TCP (Fig. A.S.T. 6-25)

Comunicação com Janelas



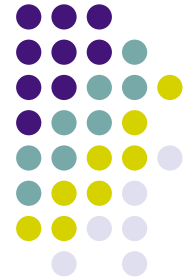
- Violação do protocolo – incluir o pseudo-cabeçalho na soma de verificação ajuda a detectar pacotes extraviados. Entretanto, essa estratégia viola a hierarquia do protocolo, pois os endereços IP nele contidos deveriam estar na camada de Rede e não na camada de Transporte.

Campos do Pseudo-cabeçalho



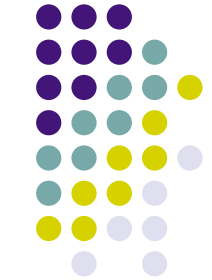
- Campo *options* – projetado para oferecer recursos extra, não previsto no cabeçalho normal do TCP.
  - Opção mais importante – a que permite a cada host estipular o máximo de carga útil do TCP que está disposto a receber.
  - O uso de segmentos grandes é mais eficiente que segmentos pequenos – cabeçalho de 20 bytes pode ser diluído em um maior volume de dados.
  - Entretanto, possível que hosts de pouca capacidade de processamento não consigam administrar segmentos muito grandes.

Campos do Pseudo-cabeçalho



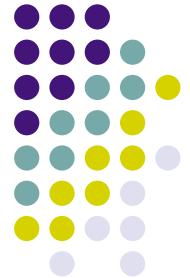
- Durante a conexão, cada lado pode anunciar sua capacidade máxima e avaliar a do seu parceiro. Os dois valores mais baixos são os escolhidos.
- Se o host não adotar esta opção – o valor padrão 536 bytes será estipulado (*default*). Todos os hosts Internet devem aceitar segmentos TCP de 536+20=556 bytes no mínimo.

Determinação do tempo



- RTT - *Round Trip Time* - tempo aproximado de viagem.
- É calculado dinamicamente, baseado na tomada de tempo entre o envio do pacote e o recebimento da confirmação.
- Problemas:
  - Tempo fixo curto/longo
  - Tempo perdido na retransmissão

TCP Round Trip Time e Temporização



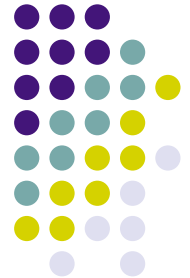
Q: como escolher o valor da temporização do TCP?

- maior que o RTT
  - nota: RTT varia
- muito curto: temporização prematura
  - retransmissões desnecessárias
- muito longo: a reação à perda de segmento fica lenta

Q: Como estimar o RTT?

- `SampleRTT`: tempo medido da transmissão de um segmento até a respectiva confirmação
  - ignora retransmissões e segmentos reconhecidos de forma cumulativa
- **`SampleRTT`** varia de forma rápida, é desejável um amortecedor para a estimativa do RTT
  - usar várias medidas recentes, não apenas o último **`SampleRTT`** obtido

Princípios de Controle de Congestionamento



Congestionamento:

- informalmente: “muitas fontes enviando dados acima da capacidade da *rede* tratá-los”
- diferente de controle de fluxo!
- sintomas:
  - perda de pacotes (saturação de buffer nos roteadores)
  - atrasos grandes (filas nos buffers dos roteadores)
- um dos 10 problemas mais importantes na Internet!

Abordagens do problema de controle de congestionamento

Existem duas abordagens gerais para o problema de controle de congestionamento:

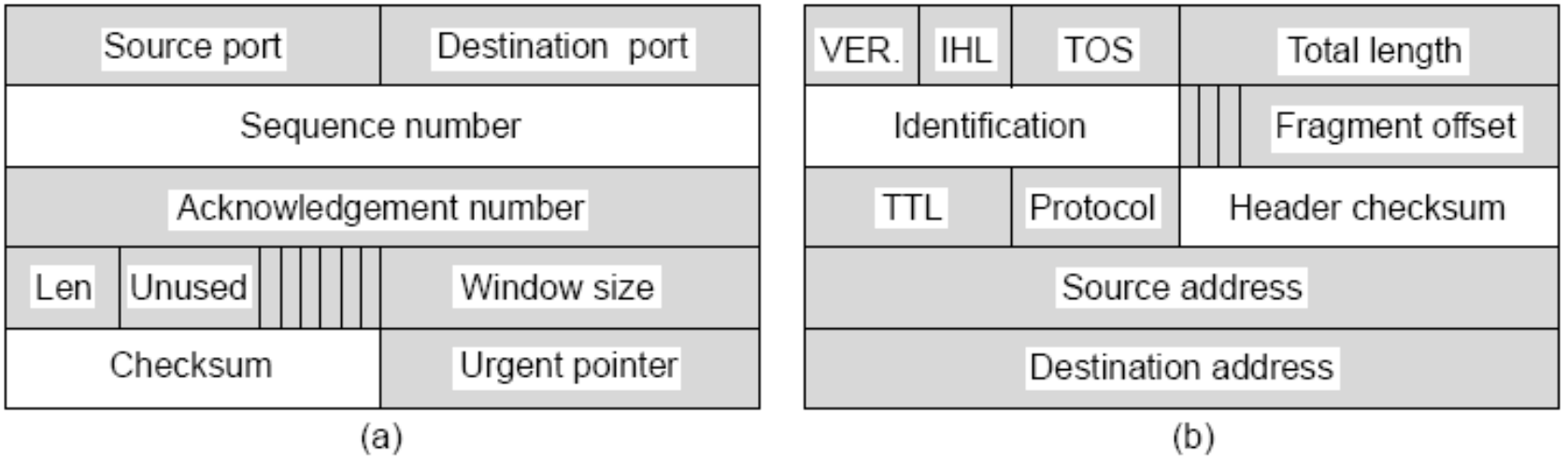
Controle de congestionamento fim-a-fim:

- não usa realimentação explícita da rede
- congestionamento é inferido a partir das perdas e dos atrasos observados nos sistemas finais
- abordagem usada pelo TCP

Controle de congestionamento assistido pela rede:

- roteadores enviam informações para os sistemas finais
  - bit único indicando o congestionamento (SNA, DECbit, TCP/IP ECN, ATM)
  - taxa explícita do transmissor poderia ser enviada

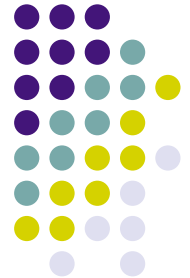
Cabeçalhos TCP e IP



Estabelecimento e Gerenciamento de uma Conexão TCP

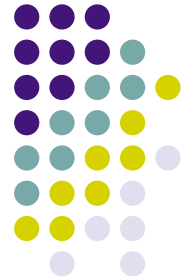
- Utilizam *Handshake* de 3 vias.
- Para estabelecer uma conexão – o servidor aguarda passivamente executando uma primitiva LISTEN e ACCEPT, por meio da especificação ou não, de uma determinada origem.
- Cliente – executa uma primitiva CONNECT especificando:
  - Endereço IP e a porta onde deseja conectar.
  - Tamanho máximo do TCP que está disposto a aceitar.
  - Alguns eventuais dados opcionais (por exemplo uma senha)

Estabelecimento e Gerenciamento de uma Conexão TCP



- A primitiva CONNECT – envia um segmento TCP com um bit SYN ativado e um bit ACK desativado, e aguarda uma resposta.
- SYN – significa *Synchronize sequence number*
- Neste pacote – ele avisa qual número inicial será usado como ponto de partida para numeração de seus segmentos.
- Estes números são usados para manter os segmentos em ordem.

Estabelecimento e Gerenciamento de uma Conexão TCP

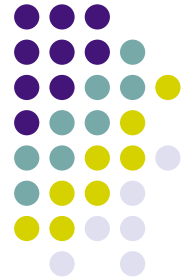


- Quando segmento chega do destino:
  - Entidade TCP verifica se existe um processo que tenha executado uma primitiva LISTEN na porta informada pela origem no campo *Destination Port*.
  - Se não houver, ele manda uma resposta com um bit RST (*reset*) ativado para rejeitar a conexão.
  - Se um processo estiver na escuta da porta – ele receberá o segmento TCP que chegou. Em seguida ele poderá aceitar ou rejeitar a conexão.

Estabelecimento e Gerenciamento de uma Conexão TCP

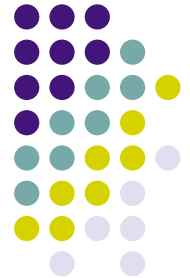
- Se o processo aceitar – um segmento de confirmação será retornado
  - (ACK ajustado para 1 e também com o bit SYN para 1)
- Além da confirmação de aceita da conexão, neste pacote segue também o *sequence number* que o host destino usará como início da numeração de seus segmentos.
- Finalmente – o host que iniciou a chamada confirma o recebimento do segmento do host destino e transfere os primeiros dados.

Estabelecimento e Gerenciamento de uma Conexão TCP



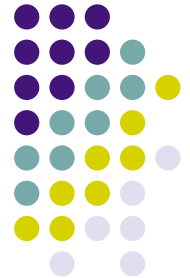
- Como anteriormente discutido – TCP enxerga os dados como um fluxo seqüencial de bytes e não como pacotes independentes.
- Entretanto – O TCP toma cuidado em manter a seqüência na qual os bytes são enviados e recebidos – para isso os números de segmentos são usados.
- O padrão TCP NÃO requer que cada sistema possua um determinado número de sequencia inicial. Cada sistema escolhe que número usar como ponto de partida.

Estabelecimento e Gerenciamento de uma Conexão TCP

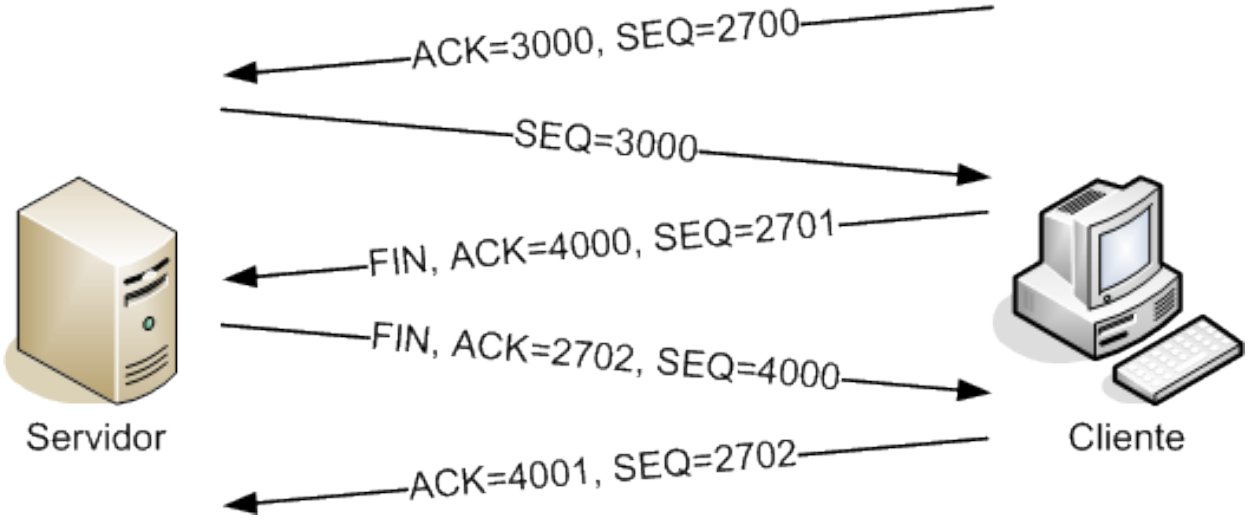


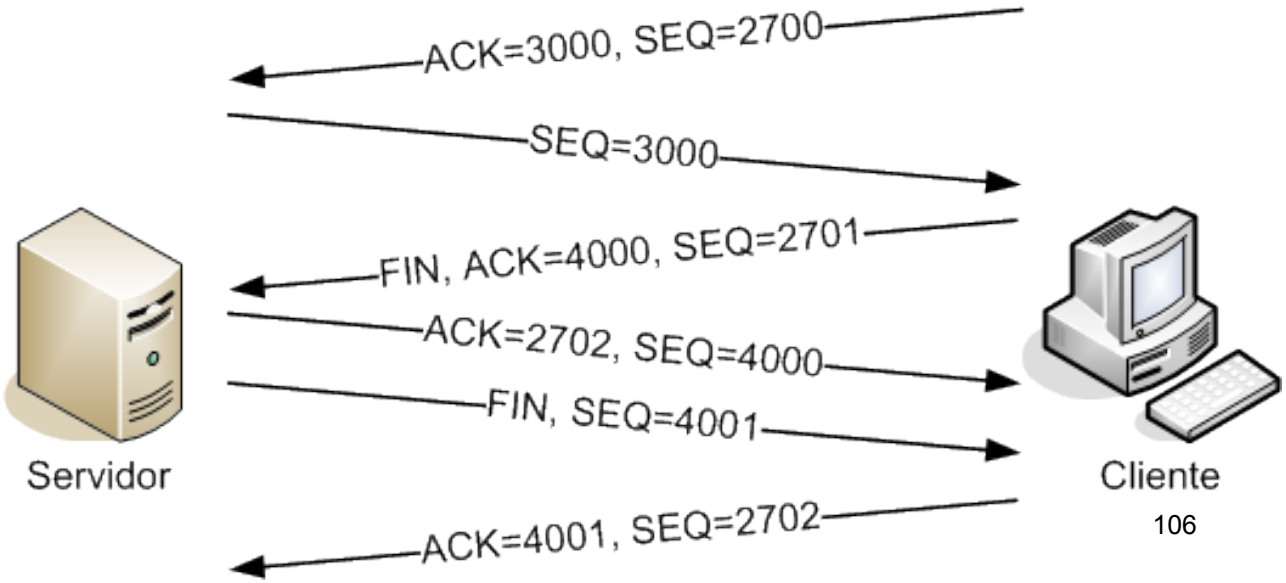
- No caso de dois hosts tentarem estabelecer uma conexão ao mesmo tempo – entre os mesmo *sockets* simultaneamente – somente uma conexão será estabelecida e não duas, pois as conexões são identificadas por suas extremidades.
- Se a primeira configuração resultar em uma conexão identificada por (x,y) – os números de seqüência de uma delas não será coerente e não será confirmada.

Finalizando a Comunicação

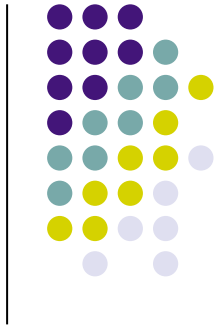


- A finalização de uma conexão TCP é feita por meio de uma confirmação de três ou quatro vias.
- É utiliza a *flag* FIN para indicar um pedido de desconexão. Este procedimento deve ser feito em ambas as direções.


ACK=3000, SEQ=2700
SEQ=3000
FIN, ACK=4000, SEQ=2701
FIN, ACK=2702, SEQ=4000
ACK=4001, SEQ=2702
Servidor
Cliente


ACK=3000, SEQ=2700
SEQ=3000
FIN, ACK=4000, SEQ=2701
ACK=2702, SEQ=4000
FIN, SEQ=4001
ACK=4001, SEQ=2702
Servidor
Cliente

Resumo



- princípios por trás dos serviços da camada de transporte:
  - multiplexação/demultiplexação
  - transferência de dados confiável
  - controle de fluxo
  - controle de congestionamento
- instanciação e implementação na
  - UDP
  - TCP

A seguir:

- Começamos a “borda” da rede (camada de transporte)
- vamos avançar para a parte mais externa (camada de aplicação)