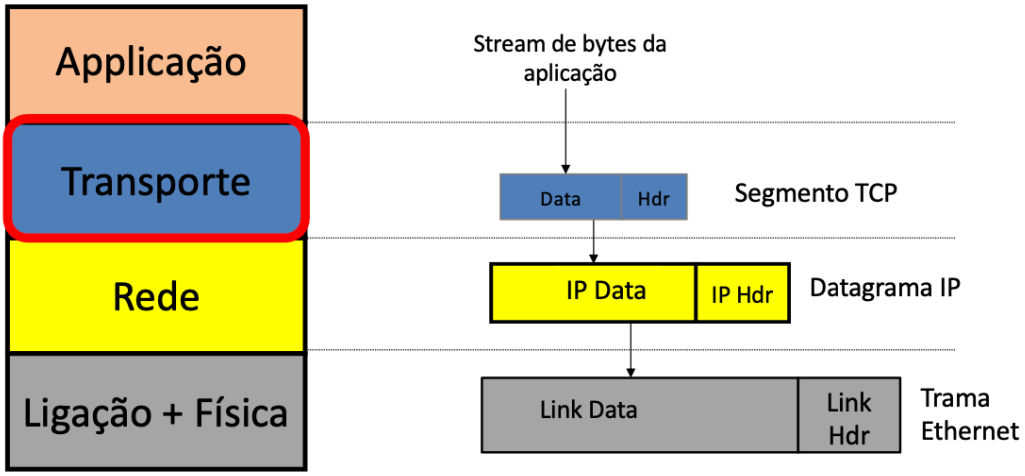
T03 - Camada de transporte

Subjects RC

Serviços da camada de transporte



- Oferece um canal logico de comunicação entre processos remotos
- Os protocolos de transporte correm apenas nos computadores terminais :
  - Não na infrastrutura da rede (routers e switches)
- As aplicações têm vários protocolos de transporte a sua disposição (ex: TCP , UDP)

Camada de transporte vs camada de rede

- Camada de transporte :
  - Comunicação logica entre processos
  - Usa e aumenta os serviços da camada de rede

- Camada de rede:

- Comunicação logica entre computadores

Protocolos camada de transporte

- TCP
  - Entrega fiável e ordenada de pacotes
  - Controlo de congestão e de fluxo
  - Estabelecimento de ligação
- UDP
  - Entrega não-fiável e não ordenada de pacotes
  - Extensão simples do serviço “best-effort” IP
- Serviços não fornecidos por nenhum destes protocolos
  - Garantias de atraso
  - Garantias de largura de banda

Multiplexagem e desmultiplexagem

Multiplexagem/desmultiplexagem

- Como distinguir entre processos?
  - Multiplexagem na origem → Recolhe dados de vários sockets e adiciona cabeçalho de transporte com informação sobre origem
  - Desmultiplexagem no destino → Informação no cabeçalho usada para entregar os segmentos ao socket correto

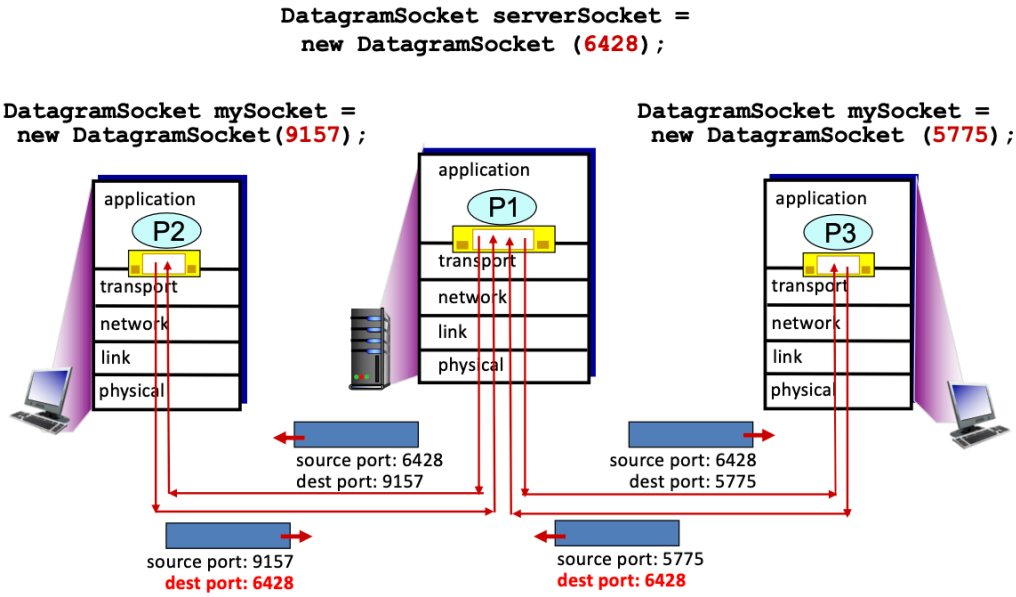
Desmultiplexagem UDP

- Serviço de transporte UDP não requer ligação
- Quando se cria um socket este vai receber um numero de porto local
- Quando se cria um datagrama para ser enviado para um socket UDP deve ser especificado:
  - Endereço IP de destino
  - Numero de porto destino
- Quando o computador destino recebe um segmento UDP:

- Verifica o porto destino e direciona o segmento UDP para o socket com esse numero de porto
- Consequencia : datagrama IP com o mesmo porto destino , mas diferentes IP ou porto origem são direcionado para o mesmo socket no destino

---

Desmultiplexagem UDP: exemplo

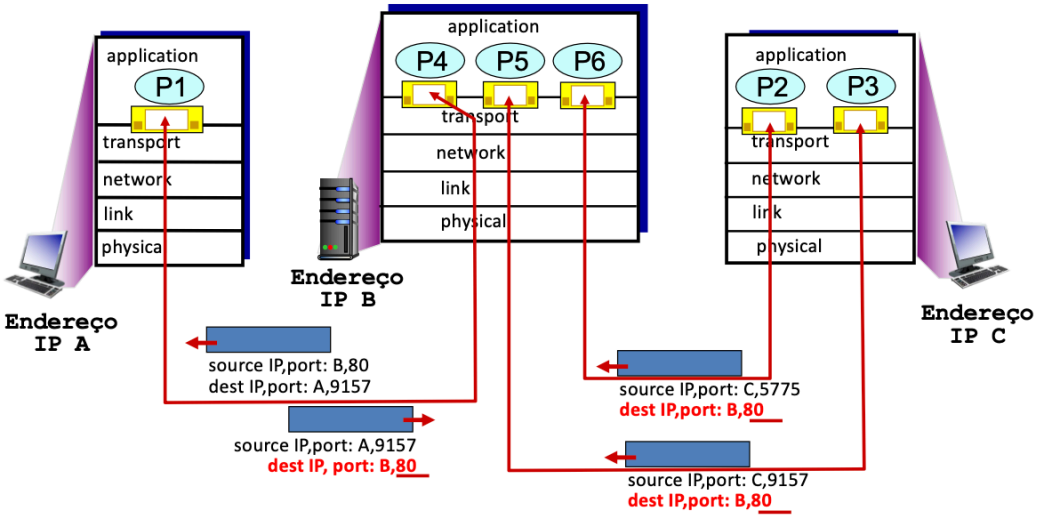


- Um socket TCP é identificado pelo tuplo:

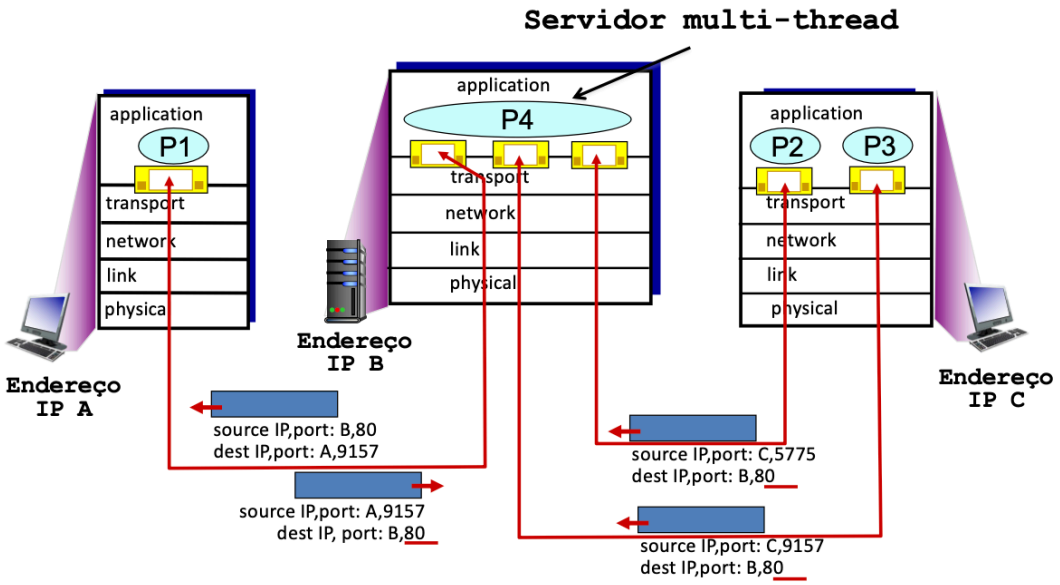
  <Endereço IP origem, porto origem, IP destino porto destino>

- Consequencia: o servidor pode suportar multiplos sockets TCp para o mesmo porto destino

Desmultiplexagem TCP: exemplo



Desmultiplexagem TCP: exemplo



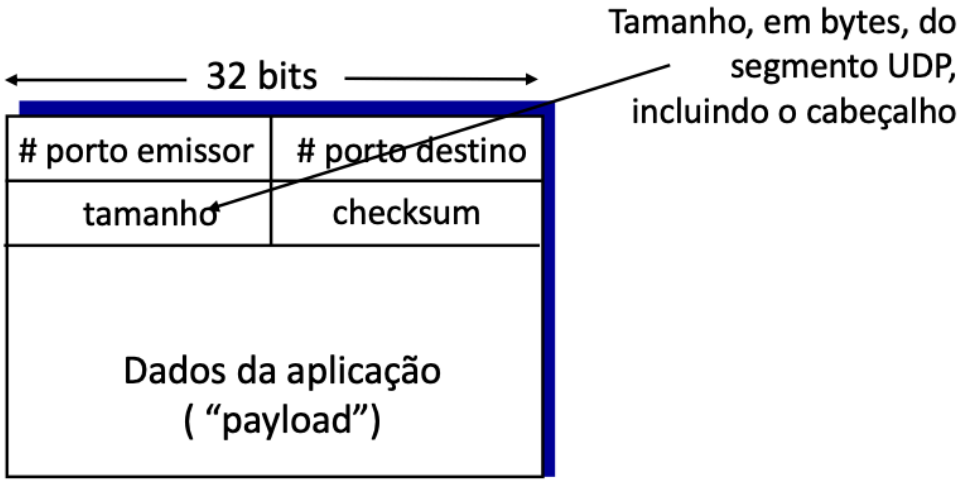
Transporte não fiável: UDP

- O protocolo de transporte internet main simples
- Oferece um serviço “best effort”

    - Segmentos UDP podem perder-se ou ser entregues à aplicação fora de ordem
- Como conseguir fiabilidade quando se usa UDP?
    - Adiciona-se ao nivel da camada de aplicação , ou na própria aplicação
- Protocolo “connectionless”:
    - Não há handshaking emissor-recetor
    - Cada segmento UDP tratado de forma independente
- Uso do UDP:
    - Em aplicações de streaming multimedia
    - DNS e SNMP

Vantagens do UDP

- Não estabelecer ligação → não tem atraso de estabelecer ligação
- Simples(stateless)
- Cabeçalho pequeno
- Não inclui mecanismos de controlo de fluxo ou congestão



Checksum UDP

- Objetivo: detetar erros no segmento que foi transmitido

- Algoritmo no emissor
  - Conteudos do segmento→ sequecia de inteiros de 16 bits
  - Checksum = adição (cp1) dos conteudos
  - Checksum colocado no campo checksum do cabeçalho UDP
- Algoritmo no recetor:
- Calcular o checksum do segmento recebido da mesma forma
- Verifica se o checksum que calculou é igual ao que está no campo checksum do cabeçalho.

Transferência fiável de dados: princípios

Transferência fiável de dados:

- Importante em todas as camadas
- A complexidade do protocolo de transmissão depende das caracteristicas do canal não fiavel

---

Deteção e recuperação de erros

- O canal de comunicação pode trocar alguns bits do pacote (bit flipping) → O checksum e usado para detetar estes erros
- O pacote pode perder-se (fila estar cheia)
- Como recuperar dos erros?
  - Principio 1: Atraves de feedback do recetor
  - Opção 1: acknowledgements (ACK) → o recetor comunica explicitamente ao emissor que recebeu bem o pacote
  - Opção 2: negative acknowledgements (NACK) → o recetor comunica explicitamente ao emissor que o pacote tinha erros ou foi perdido
- OK, há erros, depois?
  - Principio 2: Retransmitir o capote
  - O emissor retransmite o segmento se receber um NACK
  - O emissor retransmite segmento se não receber ACK num intervalo de tempo previamente definido
- No TCP se usa ACK

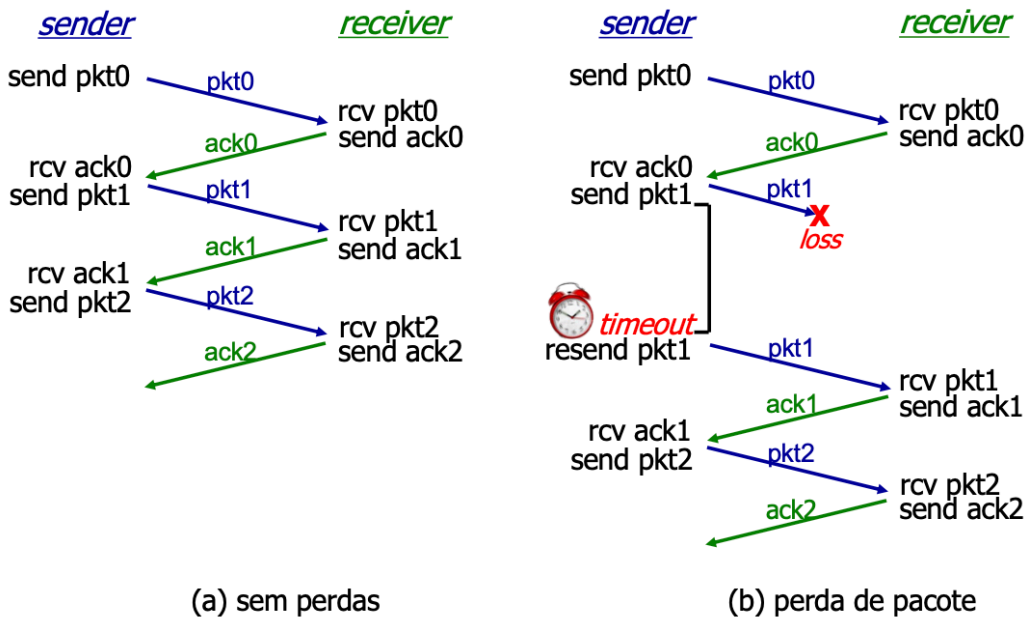
Número de sequência

- Se o erro estiver no ACK\NACK ou se estes se perderem?
  - O emissor fica sem saber o que se passou
  - Principio 2: retransmitir o pacote
  - Cria uma nova problema : duplicados
- Como tratar esta problema ?
  - Principio 3: pacotes incluem um numero de sequencia
  - O recetor ignora todos os segmentos que já recebeu
- No TCP usa-se como numero de sequencia o numero de bytes ja transferidos

---

Protocolo stop-and-wait

- Emissor envia um pacote, para e espera pela resposta do recetor



(a) sem perdas

(b) perda de pacote

Princípio #4: Pipelining

- O emissor permite que haja multipos pacotes “in-flight”
  - pacotes “in-flight”: pacotes que o emissor enviou mas dos quais ainda não recebeu ACK
- Custo: e necessario armazenar estes pacotes em buffers no emissor e por vezes no destinatario

Protocolos pipelined: Go-back-N

- Emissor pode ter ate N pacotes in-flight→N , para não inundar o recetor e a rede
- O recetor envia ACK cumulativos→ ACk do ultimo pacote recebido na ordem correta
- O emissor mantem temporizador apenas para segmentos mais antigos sem ACK
  - Simplificando emissor
  - Quando o timer expira : retransmite todos os pacotes

Protocolos pipelined: Selective Repeat

- Emissor pode ter ate N pacotes “in-flight”
- O destinatario envia ACKs individualizados(um por pacote recebido)
- O emissor mantem um timer para cada pacote sem ACK
  - Vantagem: Quando o timer expira , manda so aquele pacote
  - Desavantagem ：Não tem a vantagem dos ACK cumulativos quando a perdidos

Comparação

- Go-Back-N e mais simples de concretizar
- Selective repeat e mais eficiente , poupando recursos da rede

Transporte fiável: sumário

- Checksum → detetar erros de bit
- ACK → avisa emissor de que o pacote foi corretamente recebido
- NACK → Avisa emissor de que o pacote foi não recebido corretamente
- Timer → Usado para retransmitir pacote quando ele se perde
- Numero de sequencia → Usado para detetar pacotes em falta ou duplicados
- Pipelining → Permite enviar multiplos pacotes em receber confirmação, aumentando a utilização da rede e portanto a taxa efetiva de transferencia de dados

Transporte fiável: TCP

TCP: visão geral do serviço oferecido

- Dados em full duplex

- Pontoa ponto (1:1)
  - 1:N e o UDP
- Connection oriented (trebuie sa faca mai intai handshaking apoi se conecteaza)
- Entrega fiavele ordenada de streams de bytes
- Controlo de fluxo
- Controlo de congestão
- Protocolo pipelined

TCP: transferência fiável de dados

- O TCp cria um serviço de transferência de dados fiavel por cima do serviço não-fiável IP
  - Protocolo pipelined
  - Usa ACK cumulativos
  - Usaa apenas um timer para retransmissões
- Quando e que pode haver retransmissões :
  - Timeout
  - ACKs duplicados

	Pipelined	ACK	Timer emissor	Retransmissão	Buffer out of order
Stop and wait	Não	individualizado	1	ACK	Não
Go back N	Sim	cumulativos	1	ACK+seguintes	Não
Selective repeat	Sim	individualizados	1 por pacote	ACK	Sim
TCP	Sim	cumulativos	1	ACK	Depende

Controlo de fluxo (flow control)

- Control flow → o recetor controla o emissor para que este não encha o seu buffer por transmitir muito rapido
- O recetor publica no cabeçalho o espaço livre no buffer
- O emisor limita o numero de pacotes in-flight
- Garantia de que o buffer não vai encher

Controlo de congestão: princípios

Congestão

- Termo informal: Muitos emissores enviam muitos dados muito rapido para a rede e a rede não consegue lidar com esse carga
- O control - flow previne entupir(sabota) o recetor; o controlo de congestão previne entupir a rede
- Como se manifesta :
  - Perda de pacotes
  - Atrasos

Custos da congestão

- Atraso: aumenta a medida que a taxa de envio de pacotes se aproxima da capacidade maxima
- Perda de pacotes: a capacidade de transmissão usada anteriormente é desperdiçada
- Retransmissão: o emissor tem de retransmitir pacotes perdidos

---

Controlo de congestão: 2 abordagens

- Controlo de congestão fim a fim (end-to-end)
  - A rede não envia feedback explicito a indicar congestão
  - A congestão é inferida pelo computador terminal, quando observa perda de pacotes
  - Esta abordagem usada pelo TCP
- Controlo de congestão com auxilio da rede
  - os routers envia explicitamente feedback aos computadores terminais

---

Controlo de congestão TCP

- Como o emissor deteta qua ha congestão → Atraves da perda de pacotes (rede IP não envia pacotes) ou 3 ACK duplicados
- Como e que o emissor controla a congestão → limitando a janela de congestão (cwnd = numero de segmentos in flight permitidos)
- A rede informa qual o tamanho da janela ?
  - não
  - Emissor vai testando: aumenta a janela ate detetar perdas , reduz a janela, aumenta , reduz , etc

---

Algoritmo de controlo de congestão

- #1 fase → slow start
  - janela começa pequena e aumenta exponencialmente
- #2 fase → congestion avoidance
  - A partir de dado limite (o slow start threshold) a janela aumenta linearmente
- #3 fase → recuperação
  - pode ser mais lenta (janela e reduzida para 1 quando deteta perdas) ou mais rapida (janela reduzida para metade)