Canais de Comunicação

September 21, 2008

Sumário

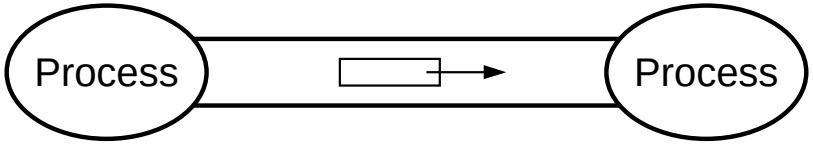
Aplicação Distribuída

O que é?

> *É uma aplicação que consiste em 2 ou mais processos que executam em diferentes processadores que não partilham memória.*

Um corolário desta definição é:

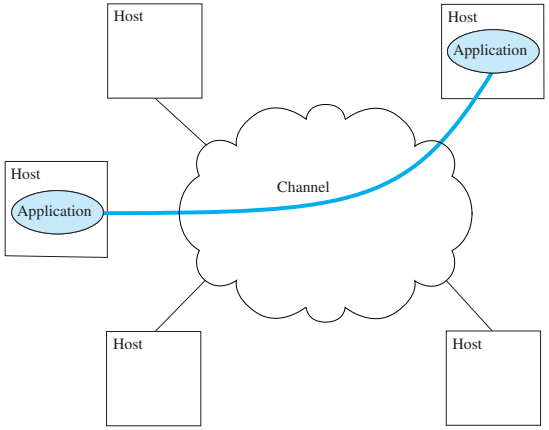
> *Os processos duma aplicação distribuída comunicam entre si através da troca de mensagens.*



- Uma *mensagem* é uma sequência de bits **indivísivel**:
  - O formato e o significado duma mensagem são especificados pelo *protocolo de comunicação*.

Comunicação com *Mensagens* e Redes

- O transporte duma mensagem da fonte ao destino é assegurado por uma rede de comunicação de dados.



- A rede pode ser abstraída como um canal de comunicação:
  - Quais as propriedades deste canal?

Propriedades dum Canal de Comunicação

- Baseado ou não em conexão.
- Fiável ou não (perda e duplicação).
- Ordenado ou não.
- Baseado em mensagens ou em *streams*.
- Com ou sem controlo de fluxo.
- Número de “extremidades” do canal.
- Identificação das entidades comunicantes.

Baseado em Conexão *vs.* Sem Conexão

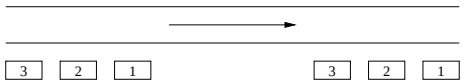
Baseado em conexão: os processos têm que estabelecer o canal antes de iniciarem a transferência de informação – análogo ao telefone;

Sem conexão: os processos não têm que estabelecer o canal – análogo ao correio.

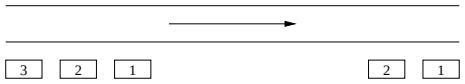
Fiabilidade (perda)

Fiável: garante que a informação enviada pelo remetente é entregue ao destinatário(s)

- assume determinadas condições;
- caso contrário, notifica os processos comunicantes.

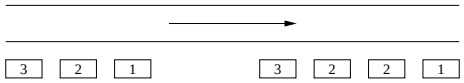


Não fiável: cabe aos processos que comunicam detectar problemas com a comunicação e actuar de acordo com os requisitos da aplicação:

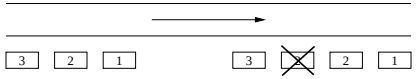


Fiabilidade(duplicação)

“Gera” duplicados: o canal pode entregar mensagens duplicadas ao destinatário(s) – cabe a este último detectá-los:

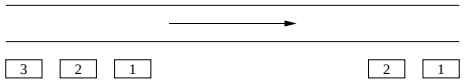


Não “gera” duplicados: o canal de comunicações não entrega mensagens duplicadas ao destinatário(s):

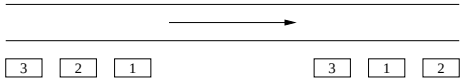


Ordem

Ordenado: garante que a informação é entregue ao destinatário(s) na ordem em que foi enviada



Não ordenado: se a preservação da ordem fôr importante cabe à aplicação detectar a não ordenação, e eventualmente re-ordenar a informação

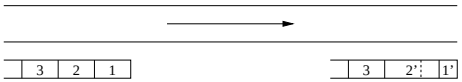


IMP ordem e fiabilidade são propriedades ortogonais

Abstracções da Comunicação

Mensagem (*datagram*): o canal suporta o transporte de mensagens – sequências de *bytes* processadas como um *todo indivísivel*: análogo ao correio.

Sequência de bytes (*stream*): o canal não suporta o conceito de mensagem. Essencialmente, funciona como uma conduta de *bytes*: análogo a *pipes* em Unix.



Outras Características do Canal de Comunicação

Controlo de fluxo: evitar que transmissores "rápidos" inundem receptores "lentos":

- não significa necessariamente que os transmissores têm uma capacidade de processamento superior.

- Número de "extremidades" do canal de comunicação:
  unicast (ponto-a-ponto) apenas 2 extremidades;
  broadcast (difusão) todos os nós dum "universo";
  multicast (multi-ponto) alguns nós dum "universo".
- Identificação das entidades comunicantes.
  - Identificar a entidade em si, p.ex. processo, pessoa.
  - Identificar a extremidade do canal, p.ex. o número telefónico.

(Identificação dos Processos na Internet (IPv4))

Problema Como se identificam os processos na Internet?

Resposta Através dum par de identificadores:

Endereço IP (IP Address) :
- Identifica o computador na Internet;
- É um inteiro com 32 bits, representando-se normalmente na forma *dotted decimal*, p.ex.: `193.136.28.31`.

Número do Porto (Port Number) :
- Identifica uma extremidade dum canal de comunicação num dado computador (atributo da camada transporte);
- É um inteiro de 16 bits (entre 0 e 65535);
- Por convenção, certos serviços usam portos fixos.

Esclarecimento O par (*IPAddress*, *PortNumber*) identifica uma extremidade dum canal de comunicação, o processo pode variar.

Isto é semelhante a um número de telefone, a pessoa que o atende pode variar.

Protocolos da Internet

Aplicação	Serviços de comunicação específicos.
Transporte	Comunicação entre 2 processos.
Rede	Comunicação entre 2 computadores **não** ligados directamente.
Interface	Comunicação entre 2 computadores ligados directamente.

- Na Internet, as propriedades do canal de comunicação entre processos dependem do protocolo de transporte usado (UDP ou TCP):
  - A concepção duma aplicação distribuída depende das propriedades suportadas pelo protocolo de transporte escolhido.

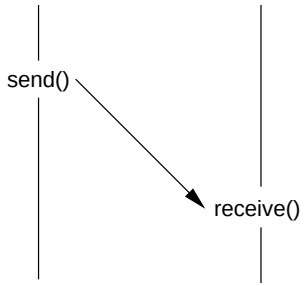
Resumo das Propriedades de UDP e de TCP

Propriedade	UDP	TCP
Abstracção	Mens.	*Stream*
Baseado em Conexão	N	S
Fiabilidade (perda & duplicação)	N	S
Ordem	N	S
Controlo de Fluxo	N	S
Número de Receptores	n	1

- A abstracção fornecida por TCP é apenas resultante da interface de programação ou é intrínseca ao protocolo?

UDP: Observações (1/3)

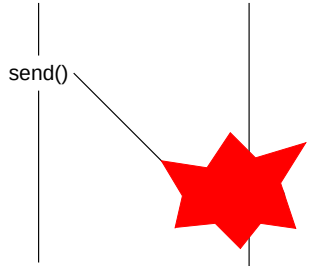
- Canais UDP são canais que suportam o transporte de mensagens – *datagramas UDP*:
  - Basicamente, a sua API oferece duas operações *send()* e *receive()*;
  - Cada mensagem é enviada invocando `send()` uma única vez;
  - Se entregue, uma mensagem sê-lo-à por inteiro e invocando `receive()` uma única vez.



- *Datagramas* têm um tamanho máximo – 65535 *bytes*:
  - As aplicações podem ter que *fragmentar* mensagens em datagramas e posteriormente *reconstruí-las*.

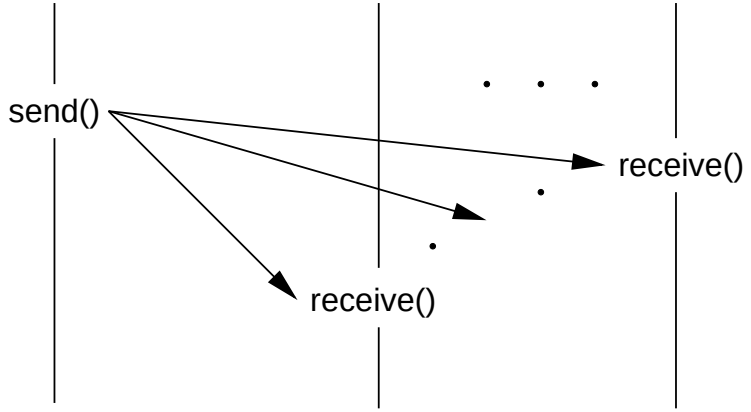
UDP: Observações(2/3)

- UDP, não sendo baseado em conexão:
  - \+ Permite transmitir dados sem espera;
  - − Requer a especificação completa da outra extremidade do canal em cada invocação de *send()*.
- UDP não dá qualquer garantia de fiabilidade:
  - Mensagens enviadas usando UDP podem perder-se ou ser duplicadas;
  - Se a aplicação não puder tolerar a perda de mensagens ou a sua duplicação, terá ela própria de detectar e recuperar dessas situações.



UDP: Observações(3/3)

- UDP não fornece controlo de fluxo, pelo que um receptor pode ser *inundado de pedidos*, ficando sem recursos (*buffers* de recepção) para receber outras mensagens.
- UDP suporta *multicast*, i.e. invocando `send()` uma única vez, pode enviar-se uma cópia da mesma mensagem para vários processos.



TCP: Observações(2/3)

- TCP é baseado em conexão. A comunicação com TCP tem 3 fases:
  1. Estabelecimento duma conexão TCP;
  2. Transferência da informação;
  3. Terminação da conexão.
- TCP garante fiabilidade (contra perda e duplicação de *bytes*):
  - Em caso de problemas na comunicação, a conexão é abortada, e os processos na sua extremidade notificados.
- TCP fornece controlo de fluxo:
  - Evita perda de dados devido a falta de recursos;
  - Contudo, TCP ainda é vulnerável a ataques de *denial-of-service* (nomeadamente, *SYN attacks*).

TCP: Observações (1/3)

- Canais TCP são canais do tipo *stream*, i.e.:
  - São semelhantes a *pipes* em Unix, excepto:
    - Permitem a comunicação entre processos em computadores distintos;
    - São canais bidireccionais – i.e. pode enviar-se informação nos 2 sentidos, simultaneamente.
- Embora a sua API ofereça duas operações `send()` e `receive()` para enviar dados,
  - TCP não preserva a separação entre *bytes* enviados em invocações de `send()`, distintas;
  - `write()` e `read()` são mais intuitivas.
- Se fôr importante manter a separação entre mensagens, a aplicação terá ela própria que o garantir. Como?

TCP: Observações(3/3)

- Canais TCP têm apenas duas extremidades, suportando comunicação entre 2 processos apenas.
- Ao contrário do que acontece com UDP, canais TCP no mesmo computador podem ter o mesmo número de porto:
  - Um canal TCP é identificado pelos pares *(IP Address, TCP Port)* das duas extremidades;
  - Permite o atendimento concorrente de vários clientes, em aplicações cliente-servidor, p.ex. *Web*: