

TECNOLOGIAS *INTERNET*

Camada de transporte TCP/IP

O protocolo IP:

- Identifica máquinas, mas não aplicações;
- Não garante que os pacotes cheguem ao destino na mesma ordem em que foram transmitidos;
- Não garante a entrega de pacotes nem faz retransmissões em caso de erros ou falhas;
- Não adequa a velocidade de envio à velocidade do recetor nem à velocidade da rede;

Os serviços e protocolos de transporte oferecem uma comunicação lógica entre processos em execução em diferentes máquinas.

São executados no lado emissor: as mensagens são divididas em segmentos e são passados à camada de rede; e no lado recetor: os segmentos são reagrupados e passados à camada de aplicação.

Os protocolos de transporte mais usados são o TCP (*Transmission Control Protocol*) e o UDP (*User Datagram Protocol*).



Camada de Transporte

- Principais Objectivos de um protocolo de transporte
 - Garantir a integridade do fluxo de dados (orientação à ligação).
 - Mecanismos de detecção e recuperação de pacotes fora de ordem.
 - Mecanismos de recuperação de erros e perdas.
 - Garantir a entrega dos pacotes à aplicação correcta.
 - Mecanismos de identificação da aplicação de destino dos pacotes.
 - Garantir a eficiência da transmissão de dados.
 - Mecanismos que garantam um atraso baixo sem degradar o overhead da rede.
- Controlo de Fluxo e Controlo de Congestão
 - Evitar congestionar o Receptor
 - Evitar congestionar a Rede

Mecanismos de recuperação de erros e perdas

Uso de somas de controlo (*checksums*) para deteção de pacotes corrompidos.

Uso de mecanismos de retransmissão:

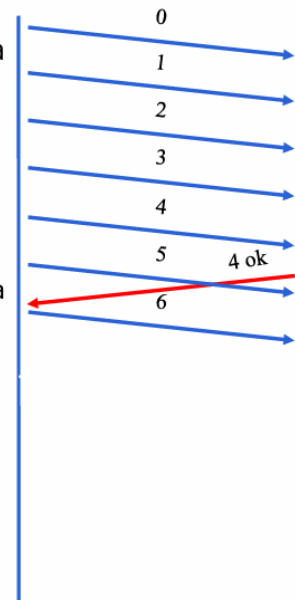
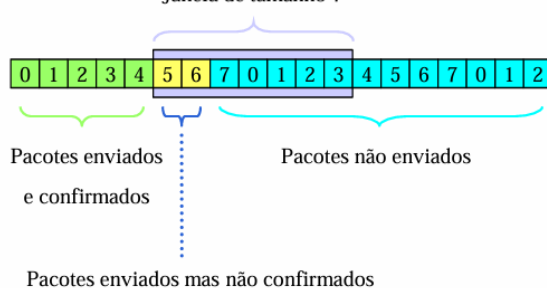
- uso de números de confirmação (*acknowledgment*) e *timers* de retransmissão (RTO): todos os pacotes enviados devem ser confirmados. Se um ACK não for recebido dentro de um certo intervalo de tempo – RTO, o emissor retransmite o pacote.
- princípio da janela deslizante:



Camada de Transporte: Mecanismos de recuperação de erros e perdas

■ Princípio de Janela deslizante

- No exemplo anterior, apesar de garantir fiabilidade, não garante uma boa utilização da largura de banda disponível
- Solução: utilização de uma janela deslizante!
 - Podem ser enviados *n* (neste exemplo *n*=7) pacotes sem confirmação, até à dimensão da janela
 - Um RTO timer é associado a cada pacote enviado
 - O emissor desliza a janela para a frente após a recepção de um ACK



11

Mecanismo de entrega à aplicação destino

Capacidade de distinguir múltiplos destinos (aplicações) numa mesma máquina. Cada pacote é associado a uma determinada aplicação através de *ports*.

Cada datagrama possui um endereço IP fonte e um destino, e um *port* fonte e um destino.

Cada datagrama transporta um segmento da camada de transporte.

UDP (*User Datagram Protocol*)

Protocolo de transporte simples, não orientado à ligação.

Não existe *handshaking* entre os emissores e os recetores, mas também não são introduzidos atrasos.

Cada segmento UDP é manipulado independentemente dos outros.

Não existe confirmação de receção de datagramas.

Fornece um serviço *best-effort*, não fiável (tal como o protocolo IP). Os segmentos UDP podem ser perdidos, sem mecanismo de controlo de erros, ou ser entregues fora de ordem ou duplicados.

Por ser mais simples e por não memorizar o estado das ligações nos nós, o *header* dos segmentos UDP é relativamente curto.

É mais usado em serviços que não exigem fiabilidade total como, por exemplo: DNS, VoIP, videoconferência, etc...

TCP (*Transport Control Protocol*)

Protocolo de transporte complexo, orientado à ligação em três fases: estabelecimento da ligação, transferência de dados e fecho da ligação.

Fornece um serviço fiável *end-to-end*, com mecanismos de numeração, de controlo de erros, de controlo de fluxo e de controlo de congestão.

Comunicação ponto-a-ponto: um emissor e um recetor.

Full-duplex.

Multiplexagem de canais lógicos utilizando o conceito de *port*.

É o protocolo de transporte da maior parte das aplicações da *Internet* (HTTP, *E-mail*, FTP, API, etc...).



Gestão de Ligações TCP (estabelecimento de nova sessão)

- Estabelecimento de uma ligação TCP: É usado um *Three way handshake* (aperto de mão triplo)

Passo 1: O Cliente TCP envia um segmento TCP SYN ao servidor:

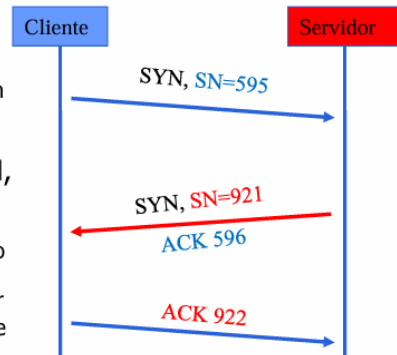
- Especifica o número de sequência inicial e o porto do servidor ao qual se pretende ligar.
- Um segmento SYN não transporta dados, mas consome um número de sequência.

Passo 2: O servidor TCP que recebe o segmento SYN, responde com um SYN/ACK segment

- Confirma a recepção do segmento SYN (os segmentos SYN consomem um número) activando a Flag ACK e indicando o próximo nr de seq a receber.
- Especifica o número de sequência inicial SN=# do servidor
- Um segmento SYNACK não transporta dados, mas consome um número de sequência.
- O servidor aloca um buffer de recepção.

Passo 3: O Cliente recebe o segmento SYN,ACK e confirma a sua recepção com um segmento ACK.

- Um segmento ACK pode ou não transportar dados. Senão transportar dados não consome um número de sequência.



Nota: SN = Seq
SN = **Sequence Number**

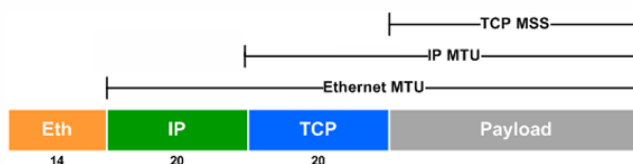
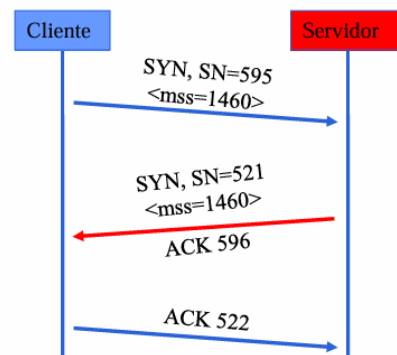
31



Gestão de Ligações TCP (Anúncio do MSS)

- Anúncio do MSS (Maximum Segment Size)

- É o tamanho máximo do campo de dados de um segmento TCP
- Quando a ligação é estabelecida, cada extremo anuncia o seu MSS.
- As estações usam a opção MSS
- A opção MSS apenas pode aparecer nos segmentos SYN
- Se o MSS não for anunciado, o outro extremo assume MSS=536byte (pois o tamanho predefinido de um pacote IP é 576bytes)
- Na Ethernet, o MSS é 1460bytes pois o MTU (maximum transmission unit) Ethernet=1500bytes



32



Gestão de Ligações TCP (3)

■ Fecho de uma ligação TCP

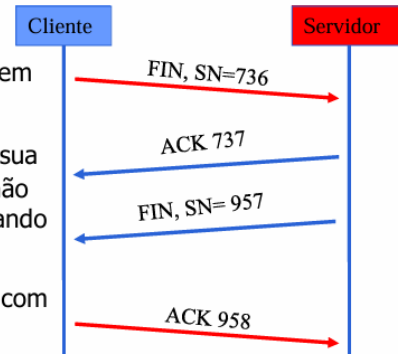
Passo 1: Quando um dos nós (neste caso o cliente) não tem mais dados para transmitir, envia um segmento FIN.

Passo 2: O servidor recebe o segmento FIN e confirma a sua recepção com um segmento ACK. Quando o servidor não tem dados para enviar, solicita o fecho da ligação enviando também um segmento FIN.

Passo 3: O cliente confirma a recepção do segmento FIN com um segmento ACK.

Passo 4: O servidor recebe o segmento ACK e termina o processo de fecho de ligação.

- Uma vez que a transmissão de dados é feita em full-duplex, este processo termina a ligação em cada um dos sentidos de uma forma independente.
 - A este processo chama-se "*half-close*".



33



Transmissão de Dados

- Fluxo de Dados Interactivos (TCP Interactive Data Flow)
 - Existe troca frequente de mensagens curtas entre os dois nós.
 - Exemplo: sessão de telnet ou aplicação de 'chat'.
 - Requisitos: atraso baixo, taxa de transferência não é importante.
- Fluxo de Dados Não Interactivos (TCP Bulk Data Flow)
 - Fluxo de dados, tipicamente num único sentido, com o objectivo de transmitir uma elevada quantidade de informação.
 - Exemplo: transferência de ficheiros.
 - Requisitos: taxa de transferência elevada, atraso não é importante.

(Capítulos 19 e 20 do Livro TCP/IP Illustrated Vol. 1)

41

Algoritmo de Nagle – Fluxo de dados interativo

Determina que só se deve enviar um segmento após a receção da confirmação do anterior.

Permite juntar vários blocos de dados num só segmento, diminuindo o *overhead* na rede pelo aumento do atraso de entrega de dados.

Quanto mais rápido chegarem as confirmações, mais rápido são enviados os dados. Se a rede possuir um atraso baixo, como numa rede de área local, as confirmações chegam rapidamente e o algoritmo não traz vantagens. Noutros casos, como em WANs, com a ajuda deste algoritmo são gerados menos segmentos, mas maiores, o que leva a menor congestão na rede.

Fluxo de dados não interativos

Nestes casos, durante o estabelecimento da ligação, é negociado o valor MSS (*Maximum Segment Size*).

As confirmações são cumulativas e confirmam a receção correta de dados até uma determinada posição. Por vezes, as confirmações estão um pouco atrasadas relativamente ao último segmento recebido.

É usado o mecanismo de janela deslizante.

Controlo de fluxo

Regula a quantidade de dados que uma fonte pode enviar antes da receção de uma confirmação do destino.

É definida uma “variável” janela imposta sobre o *buffer* do emissor, cujo valor depende da velocidade do recetor, sendo menor ou igual.

O recetor aloca outro *buffer*. A aplicação consome dados desse *buffer* a uma determinada velocidade, podendo obrigar ao abrandamento ou suspensão da transmissão. Idealmente, a velocidade média de envio deverá ser idêntica à velocidade média de leitura de dados pela aplicação consumidora.

O recetor anuncia o espaço livre no *buffer* através do campo *Rcvwindow*. À medida que estes dados vão sendo lidos, o tamanho da janela anunciado pode aumentar. O emissor considera o valor *Rcvwindow* como o tamanho máximo de dados transmitidos não confirmados. Assim, garante-se que a capacidade máxima do *buffer* do recetor não será ultrapassada.

À medida que são confirmados segmentos, o limite inferior da janela deslizante do emissor vai sendo incrementado.

Controlo de congestão

Congestão de rede ocorre, inevitavelmente, mais concretamente quando demasiadas fontes enviam demasiados dados demasiado depressa tendo em conta a velocidade com que a rede consegue processá-los.

Causa perda de pacotes e longos RTT (*Round Trip Time*), devido a *overflow* nos *routers* e atrasos nos encaminhamentos.



Controlo de Congestão TCP

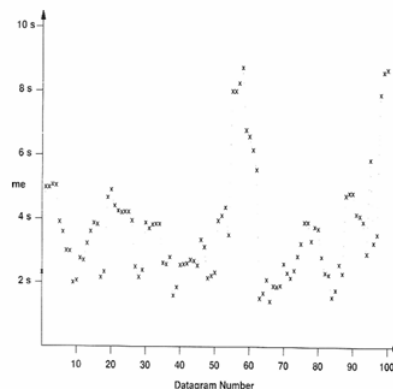
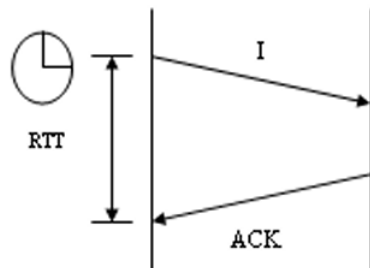
■ Efeitos da Congestão da rede (outra Vez)

- Aumento do tempo de trânsito dos pacotes (RTT) devido ao aumento dos tempos de atraso nas filas (buffers)

Os *timers* associados aos pacotes expiram, provocando um aumento das retransmissões, que por sua vez vão agravar ainda mais o congestionamento.

- Aumento das perdas de pacotes

↑ Tráfego => ↑ carga => ↑ RTTs (& ↑ Perdas) => ↑ Retransmissões => ↑ ↑ Tráfego



54

Cada fonte define uma variável *Cwnd* com o valor da janela de congestão. Este valor será ajustado em função da congestão da rede – quanto maior for a congestão, menor será o valor de *Cwnd*. A transmissão de dados não confirmados é limitada ao tamanho da janela de congestão.

O valor da janela de transmissão é o valor mínimo entre *Rcvwindow* e *Cwnd*. Ou seja, enviam à taxa da componente mais lenta – ou a rede ou o recetor.

Ocorre congestão na rede quando:

- são recebidos três confirmações duplicadas com um valor antigo;
- ocorre um *timeout* – ou o segmento não chegou ao destino, ou a confirmação não chegou ao emissor (pode haver congestão num ou nos dois sentidos).

RTT e *timeout*

O RTO deve ser superior ao RTT. Não deverá ser demasiado pequeno, podendo conduzir a retransmissões de pacotes desnecessárias; nem ser demasiado longo, de modo a não reagir tarde a perdas de segmentos.

Mecanismos de controlo de congestão

No algoritmo *Slow Start*, o tamanho da janela de congestão aumenta exponencialmente até atingir um *threshold*.

Este mecanismo foi criado para evitar a congestão em WANs (o *slow start* impede um *fast start*), e para fornecer um aumento exponencial dos valores iniciais do tamanho da *Cwnd*, isto é, após o estabelecimento de uma ligação, um aumento linear da *Cwnd* é demasiado lento.



Controlo de Congestão TCP: Mecanismo Slow Start (Tahoe, Reno)

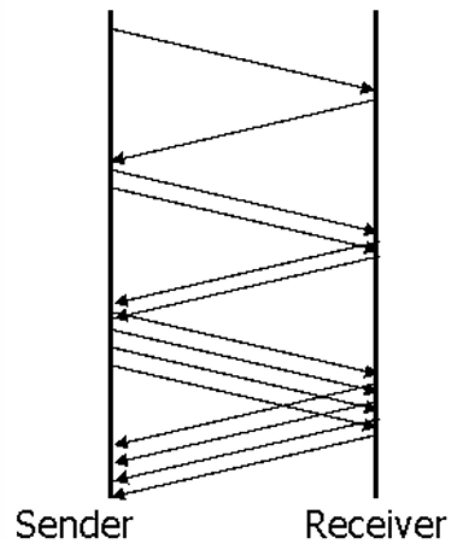
- O emissor começa com *Cwnd* de 1 (ou 2 MSS) (Maximum Segment Size);
- Ou seja, envia apenas um (ou dois) segmentos e espera pela confirmação;

[Sempre que um ACK chega, a *cwnd* is duplicada]

- De seguida envia 2 (ou 4) segmentos e espera pela confirmação;
- Depois, 4 (ou 8) segmentos e espera pela confirmação... até conhecer a dimensão da janela de congestão (a *cwnd*) [Ganha-se confiança sobre o débito da rede]

[A *cwnd* duplicada em cada "época" RTT]

- O crescimento exponencial do número de segmentos a enviar continua até:
 - Até ao *slow start threshold* (*ssthreshold*) -> fase *Congestion Avoidance*.
 - Até ser detetada congestão (perda de pacotes)



No algoritmo *Congestion Avoidance*, o tamanho da janela de congestão aumenta linearmente até ser detetada congestão. Começa quando termina o *Slow Start*, isto é, até atingir um *threshold* (quando $Cwnd \geq threshold$).

A partir desse momento, após a receção de cada confirmação, *Cwnd* é incrementada uma unidade.

Quando for detetada congestão:

- se for por deteção de três confirmações duplicadas – $threshold = Cwnd/2$ e $Cwnd = threshold$, e é iniciada uma nova fase *deste algoritmo*);
- se for por *timeout*, $Cwnd = 1$, e é iniciada uma nova fase *slow start*.

DNS (*Domain Name System*)

Serviço que pretende obter um endereço IP a partir de um nome, através de uma base de dados distribuída.

Pode, ainda, fornecer informações para encaminhamento de correio eletrónico.

Os nomes são traduzidos em endereços IP através da consulta de servidores de nomes.

O espaço de nomes é organizado de forma hierárquica, a partir de um domínio de raiz, ao qual se sucedem domínios de topo que, por sua vez, conterão subdomínios e assim sucessivamente.

Os domínios de topo podiam ser de três tipos diferentes: genéricos ou organizacionais (.com, .edu, .gov, .net, .org, etc...), geográficos (.pt, .es, .us, ...) ou arpa (domínio especial para mapeamento inverso). Atualmente, há uma liberalização dos domínios de topo.

Não há qualquer relação entre o nome DNS e a sua localização geográfica. Por exemplo, computadores com nomes DNS que terminem em “.pt” não estão necessariamente em Portugal.

Os computadores cujos nomes estão no mesmo domínio não precisam de estar na mesma rede nem na mesma zona geográfica.

Computadores numa rede não precisam de ter nomes no mesmo domínio.

Um endereço IP pode estar associado a vários nomes diferentes.

Um nome pode estar associado a vários endereços IP – para balanceamento de carga.

A definição de domínios e a delegação de responsabilidade de nomeação de um domínio nos seus subdomínios permite gerir todo o espaço de nomeação sem ter de recorrer a uma única base de dados centralizada – o DNS é um serviço distribuído.

Tipos de servidores DNS: primários (onde são configurados os domínios e os registos), secundários (distribuidores de carga e servidores *backup* dos primários) e de *caching* (estes não possuem qualquer ficheiro de configuração, baseiam o seu funcionamento no mecanismo de *cache*).

Por uma questão de redundância, cada domínio tem, pelo menos, um servidor de nomes secundário para além do primário. Os servidores secundários são atualizados, (normalmente) de 3 em 3 horas, pelos primários.

Por uma questão de eficiência, sempre que um servidor recebe informação de um dado mapeamento, essa informação é mantida em *cache* durante um determinado tempo – caso seja feito um novo pedido, evitam-se novas

consultas a outros servidores. O tempo máximo de armazenamento em *cache* é configurável através do valor TTL (*Time to Live*).

Um domínio é uma ramificação completa no espaço de nomes que inclui todos os subdomínios, caso existam.

Uma zona é uma parte de um domínio registada num ficheiro da base de dados de um servidor primário. A divisão do espaço de nomes em zonas permite repartir a informação por diversos servidores, evitando sobrecargas.

Um servidor pode administrar uma ou mais zonas. Para cada zona deve existir um servidor de nomes primário (e, pelo menos, um secundário).

Um servidor, ao receber um pedido de resolução de um nome que não pertence a nenhuma das suas zonas, deve contactar os servidores de nomes respetivos. Para isso, deve ser percorrida a hierarquia de nomes, desde a raiz até ao domínio desse nome.

A tradução de um nome num endereço IP é conseguida por um *resolver* (processo que corre no sistema operativo), o qual se encarrega de interrogar os servidores de nomes respetivos.

O processo ocorre por dois funcionamentos: o funcionamento recursivo (o mais comum), no qual o servidor percorre toda a hierarquia de nomes e devolve a resposta ao *resolver*, e o funcionamento iterativo, no qual o servidor indica ao *resolver* um servidor de raiz que deve ser consultado, o qual indica um servidor de topo e assim sucessivamente até ser percorrida a hierarquia de nomes. (Ou seja, o processo é igual em ambos os funcionamentos, mas no recursivo quem percorre a hierarquia de nomes é o servidor; no iterativo quem percorre a hierarquia é o *resolver*).

Ficheiros de configuração: */etc/resolv.conf* (contém a informação que permite ao cliente saber qual é o domínio predefinido e os servidores de nomes que devem ser usados) e */etc/nsswitch.conf* (contém a informação sobre serviços DNS na máquina e a ordem pela qual devem ser executados quando se pretende encontrar uma determinada informação).

Ficheiro de configuração servidor primário

Servidor DNS deste domínio

E-mail do responsável (joao@ipcb.pt)

Exemplo de ficheiro de zona - configuração do domínio *ipcb.pt*

@	IN	SOA	dns.ipcb.pt.	joao.ipcb.pt. (
199803237			; serial number	
		28800		; refresh após 8 horas
		7200		; retry após 2 horas
		604800		; expire após 7 dias
		86400)	; TTL de 24 horas
		NS	dns	
dns	A	193.137.234.1		
Mail	A	193.137.234.1		
www	A	193.137.234.2		
www2	A	212.55.157.5		

Cabeçalho (registo SOA)

Registos de servidores DNS

Outros registos

Serial number: número de série que deve ser incrementado sempre que se fazem alterações na configuração.

Refresh: intervalo de tempo (em segundos) entre *polling* do DNS secundário ao DNS primário.

Retry: tempo de espera até à próxima tentativa de *refresh*, se a primeira falhar.

Expire: período de tempo após o qual o DNS secundário deve, se não conseguir contactar o servidor primário, descartar a informação (partir do princípio que a zona foi cancelada).

TTL: período de tempo máximo que a informação deste domínio pode ser armazenada na cache de outros servidores DNS.

HTTP (*Hypertext Transfer Protocol*)

Protocolo do tipo cliente/servidor para transferência de ficheiros HTML, através de ligações TCP/IP (geralmente, no *port* 80).

Existem dois tipos de mensagens HTTP: pedidos (enviados pelos clientes) e respostas (enviadas pelos servidores, em resposta aos pedidos).

Uma mensagem é enviada através de métodos HTTP, obedece a uma determinada formatação e suporta *headers* específicos.

Os ficheiros HTML (documentos), possuem ligações para outros documentos, isto é, hiperligações.

Selecionada uma hiperligação, é carregado o documento apontado por essa ligação.

Um URI (*Uniform Resource Identifier*) é um identificador de um recurso. Um URL (*Uniform Resource Locator*) é uma extensão de URI que o precede com o tipo de protocolo usado para aceder a esse recurso. Exemplo:

est.ipcb.pt
URI

http://est.ipcb.pt
URL

Características dos métodos HTTP

Method	Safe	Idempotent	Cacheable
GET	Yes	Yes	Yes
HEAD	Yes	Yes	Yes
OPTIONS	Yes	Yes	No
TRACE	Yes	Yes	No
PUT	No	Yes	No
DELETE	No	Yes	No
POST	No	No	Conditional*
PATCH	No	No	Conditional*
CONNECT	No	No	No

Safe: um método HTTP é seguro se não alterar o estado do servidor. Por outras palavras, um método é seguro se conduzir a uma operação só de leitura no servidor.

Idempotent: um método HTTP é idempotente se o efeito de fazer vários pedidos idênticos no servidor, é o mesmo de fazer um único pedido.

Cacheable: métodos cuja resposta HTTP pode ser armazenada em cache, ou seja, armazenada para ser recuperada e utilizada mais tarde, poupando um novo pedido ao servidor.

Os *headers* dos pedidos contêm o método usado, o URI do pedido e a versão do HTTP em uso, entre outras informações. Exemplo:

GET /disciplinas/TI/frequencia.html HTTP/1.1
ACCEPT: text/html

Os *headers* das respostas contêm a versão do HTTP em uso, o código do estado da resposta e a mensagem relativa ao estado, entre outras informações. Exemplo:

HTTP/1.1 200 OK

Status-Codes

• 1xx : códigos informativos	Exemplos dos códigos mais comuns:
• 2xx : códigos de confirmação positiva	101 - switching protocols 200 - OK
• 3xx : códigos de redireccionamento; alguma acção tem que ser desenvolvida de modo a completar o pedido	202 - Accepted 304 - Not modified 305 - Use proxy 400 - Bad request
• 4xx : códigos de erro; o pedido está mal feito (sintaxe, etc)	402 - Unauthorized 403 - Forbidden 404 - Not found
• 5xx : códigos de erro; o pedido está correcto mas o servidor falhou	407 - Proxy Authentication Required 500 - Internal Server Error

Um *cookie* HTTP é um conjunto de dados que um servidor envia para o *browser* de um utilizador.

O *browser* pode armazenar *cookies*, criar novos *cookies*, modificar os existentes e enviá-los de volta ao mesmo servidor com pedidos posteriores.

Os *cookies* permitem as aplicações web armazenem quantidades limitadas de dados e recordem informações de estado. Por omissão, o HTTP é *stateless*, isto é, não tem estado.

Caching HTTP

A *cache* HTTP armazena uma resposta associada a um pedido e reutiliza a resposta armazenada para pedidos posteriores. Um exemplo de *cache* HTTP é aquela armazenada pelo próprio *browser* (*cache* local). Também pode ser guardada *cache* HTTP em servidores *web* e *proxies*.

A função da *cache* local é permitir a melhoria de desempenho. Quando um pedido é feito a um servidor, a resposta é armazenada em *cache*. Posteriormente, se existirem novos pedidos iguais, as respostas armazenadas em *cache* podem ser repescadas.

Caching HTTP num servidor *proxy*: todos os pedidos passam por esse servidor, no qual são guardados os ficheiros mais acedidos. Este servidor serve de intermediário entre o *browser* e os servidores *web*. Este cenário é muito útil, quando existem muitos clientes na mesma rede a aceder à *Internet* (por exemplo, no meio empresarial).

Caching HTTP num servidor *web*: todos os pedidos passam pelo *cache accelerator*. Os ficheiros em *cache* válidos são enviados para o cliente, não sendo necessário consultar o servidor *web*.

Para se poder utilizar *caching* HTTP, é necessário garantir que as respostas armazenadas estão dentro do seu prazo de validade. Esta informação está presente no *header* da mensagem.

HTTP 1.1: O problema do *Head of Line Blocking*

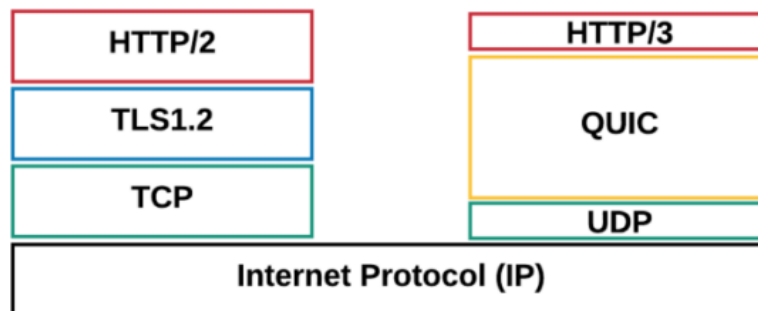
No HTTP 1.1, os pedidos são enviados em série – um pedido só é enviado após a receção da resposta do pedido anterior. Se o envio de uma resposta se atrasar, os pedidos seguintes serão enviados mais tarde.

No HTTP 2.0, os pedidos podem ser enviados em paralelo, ou seja, na mesma sessão TCP. Se o envio de uma resposta se atrasar, o envio de pedidos seguintes não será condicionado.

O HTTP 2.0 resolve este problema, mas continua a basear-se no mecanismo do protocolo TCP. Este protocolo é lento a iniciar ligações; o seu mecanismo de controlo de congestão da rede reduz a velocidade de transmissão quando são detetadas colisões; e, em caso de perda de pacotes, o protocolo espera pela respetiva retransmissão antes de entregar os pacotes seguintes. Por isso, foi criado o protocolo QUIC.

QUIC: principais características

- O protocolo QUIC faz praticamente o mesmo que o protocolo TCP+TLS, com uma grande diferença: a latência é muito menor
 - Reduz o overhead no estabelecimento das ligações ao fazer logo o handshake inicial de segurança TLS
 - Baseia-se no UDP e o controlo de pacotes perdidos é feito ao nível de cada stream, pelo que um pacote perdido num stream não vai atrasar os outros streams
 - A encriptação é feita pacote a pacote, pelo que não é necessário esperar pelos pacotes seguintes para poder descriptar o conteúdo de cada pacote



Servidores *web*

Disponibilizam conteúdos *web* aos *browsers*.

Alguns conteúdos são estáticos, como imagens ou vídeos, e outros são dinâmicos – são gerados quando uma sessão é criada.

A criação dinâmica de conteúdos nos servidores (*backend*) é feita com linguagens *server-side*, como PHP ou ASP. Já a interatividade local com o utilizador (*frontend*) é conseguida com linguagens *client-side*, como *JavaScript*.

Exemplos de tecnologias de servidores *web*: *Apache* e *NGINX*.

Apache

Software para servidores web, open source, lançado em 1995.

Configurável nos sistemas operativos *Linux* e *Windows*.

Suporta: ligações TLS/SSL; *virtual hosting*, isto é, é possível configurar vários servidores virtuais na mesma máquina; balanceamento de carga e tolerância a falhas.

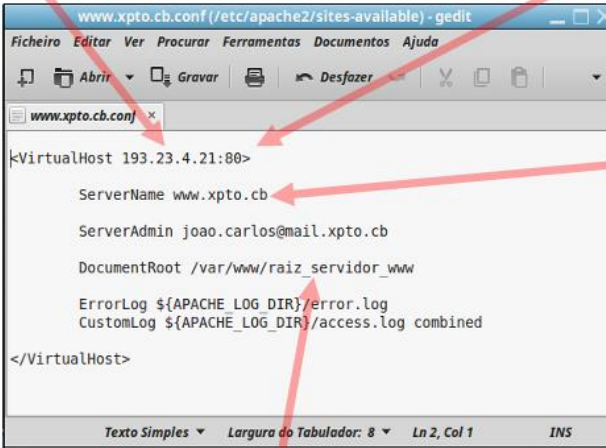
Compatível com protocolo IPv6.

Configuração Apache: criar um novo Virtual Host

Passo 3: editar o ficheiro de configuração do Virtual Host

Endereço IP do servidor onde o Virtual Host vai escutar

porto TCP onde o Virtual Host vai escutar



```
<VirtualHost 193.23.4.21:80>
    ServerName www.xpto.cb
    ServerAdmin joao.carlos@mail.xpto.cb
    DocumentRoot /var/www/raiz_servidor_www
    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

Nome DNS do Virtual Host

Definição da pasta do computador que vai ser a raiz do Virtual Host

IMPORTANTE: é necessário criar uma pasta para cada Virtual Host

NGINX

Software para servidores (*reverse proxy*) web.

Usa uma arquitetura *asynchronous event-driven* em vez de *threads*, o que melhora o seu desempenho.

Configurável nos sistemas operativos *Linux* e *Windows*.

Suporta balanceamento de carga.

Ficheiros e pastas importantes

- `/etc/nginx` : esta é a pasta onde estão todas as configurações
- `/etc/nginx/nginx.conf` : o ficheiro de configuração principal onde são aplicadas as configurações globais
- `/etc/nginx/sites-available` : a pasta onde são configurados os "blocos de servidor" (virtual hosts). Não ficam ativos automaticamente.
- `/etc/nginx/sites-enabled` : a pasta onde são criados os links para os "blocos de servidor" configurados no sites-available que se pretendem ativar. Ao criar um link para uma configuração, estamos a ativar esse "bloco de servidor". Ao apagar o link, estamos a desativar esse "bloco de servidor"
- `/etc/nginx/snippets` : esta pasta contém fragmentos de configuração que podem ser incluídos em outras partes da configuração do Nginx. Ideal para definir pedaços de configuração potencialmente repetíveis.
- `/var/log/nginx/access.log` : ficheiro de log onde ficam registados os acessos
- `/var/log/nginx/error.log` : ficheiros de log onde ficam registados os eventuais erros

Balanceamento de carga e tolerância a falhas

Quality of Experience (QoE) é uma métrica do grau de satisfação de um utilizador relativo à sua experiência com um serviço *Internet*.

Para garantir uma elevada QoE, foram criadas arquiteturas tolerantes a falhas e de balanceamento de carga.

Arquiteturas de balanceamento de carga distribuem os pedidos por vários servidores, de modo a suportar um número elevado de sessões em simultâneo; reduzir o tempo de resposta a pedidos; melhorar a tolerância a falhas dos serviços; e promover escalabilidade para responder ao crescimento exponencial do número de utilizadores da *Internet*.

Arquiteturas tolerantes a falhas usam redundância em todos os níveis e garantem alta disponibilidade.

Mecanismos de balanceamento de carga

Ferramentas ou métodos técnicos como, por exemplo: configurações em servidores DNS (*Round Robin*); configurações em servidores *web*; hardware específico; *software* específico; e “ARP gratuito”.

Configurações em servidores DNS (*Round Robin*)

Um servidor DNS é configurado para balanceamento de carga através da associação de um nome a vários endereços IP. A distribuição de pedidos pelos vários servidores é homogénea.

O servidor não sabe se os servidores estão operacionais.

A utilização de *cache* local compromete o balanceamento de carga por este mecanismo, uma vez que todos os pedidos de computadores na mesma rede serão encaminhados para o mesmo servidor, cujo endereço IP está armazenado em *cache*.

Hardware específico

Solução cara, mas eficaz, com baixa latência.

Detetam servidores com falhas e impedem o envio de pedidos para estes.

Baixa escalabilidade – quando o seu limite é atingido, é necessária a instalação de mais *hardware*.

Baixa interoperabilidade em ambientes de virtualização.

Software específico

Pode ser executado em *bare metal* (servidores físicos que operam sem camadas de virtualização ou abstração), máquinas virtuais, *containers* ou serviços *cloud*.

Solução mais barata e relativamente eficaz.

Detetam servidores com falhas e impedem o envio de pedidos para estes.

Alta escalabilidade – quando o seu limite é atingido, é necessário o aumento de recursos computacionais e/ou o número de instâncias do balanceador de carga.

Alta interoperabilidade em ambientes de virtualização.

Estratégias de balanceamento de carga

Regras ou “algoritmos” como, por exemplo: *Round Robin* (com ou sem pesos associados), *Least Connections*, *Least Response Time*, *Least Bandwith* e *IP Hash*.

Round Robin: os pedidos são enviados em sequência circular entre os servidores, de forma homogénea.

Weighted Round Robin (com pesos): semelhante ao anterior. No entanto, a distribuição não é homogénea – os servidores com mais peso recebem mais pedidos, proporcionalmente, tendo em conta os valores dos pesos de cada servidor.

Least Connections: os pedidos são enviados para o servidor com o menor número de sessões ativas.

Least Response Time: os pedidos são enviados para o servidor com o menor tempo de resposta, ou seja, a menor latência; não obstante, também é considerado o número de sessões ativas desse servidor.

Least Bandwith: os pedidos são enviados para o servidor cuja ligação à rede está menos saturada, ou seja, com maior aproveitamento da largura de banda.

IP Hash: (estratégia de persistência de sessões)...

Quando uma aplicação num computador faz vários pedidos sequenciais, estes podem ser encaminhados para servidores diferentes. Se a aplicação usar dados de sessão, todos os servidores que recebem os pedidos têm de possuir os dados dessa sessão.

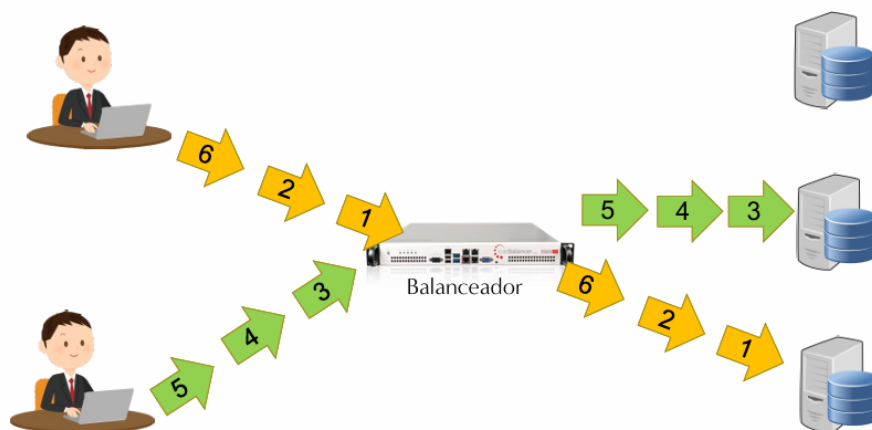
Para isso, os dados da sessão devem ser sincronizados em tempo real entre os servidores – esta sincronização é complexa e exigente do ponto de vista computacional.

Uma solução mais fácil é encaminhar os vários pedidos de uma mesma sessão sempre para o mesmo servidor. Desta forma não é necessária sincronização de dados de sessão, mas um servidor pode ser sobrecarregado.

IP Hash – Estratégia de persistência de sessões

Persistência de sessões: estratégia IP Hash

- Esta estratégia calcula um hash a partir do endereço IP do cliente
- O hash determina o servidor de destino
- Desta forma, os pedidos de um determinado cliente são sempre direcionados para o mesmo servidor



TI - Osvaldo Santos

19

Alta disponibilidade

Requer que a arquitetura do sistema seja tolerante a falhas a todos os níveis (rede, servidores, máquinas virtuais e aplicações).

Deverão ser eliminados todos os SPF (*Single Point of Failure*), através da criação de rotas redundantes. Um SPF é um componente da arquitetura (por exemplo, um *router*) que, em caso de falha, deixará todo o sistema indisponível.

Para haver redundância são necessários sempre, pelo menos, dois componentes (dois sistemas) que executem as mesmas funções.

Numa arquitetura tolerante a falhas, os sistemas redundantes verificam continuamente o “estado de saúde” dos seus semelhantes – normalmente, trocam *heartbeats*. Quando um sistema deixa de enviar o seu *heartbeat*, os seus semelhantes sabem que este já não está operacional.

Os sistemas redundantes podem ser configurados de dois modos:

Active-active failover: todos os sistemas estão operacionais em simultâneo e efetuado balanceamento de carga entre eles. Se algum sistema falhar, os restantes assumem o balanceamento da carga, incluindo os pedidos dirigidos ao sistema que falhou.

Active-passive failover: apenas um sistema está operacional num determinado instante, enquanto os restantes estão adormecidos (no entanto, mantém-se a troca de *heartbeats* entre eles). Se algum sistema falhar, um dos restantes assume o respetivo serviço.

Em *Cloud Computing*... Os serviços *cloud* possuem componentes para balanceamento de carga. É possível instanciar automaticamente mais máquinas virtuais (*autoscaling*), se a procura aumentar subitamente.

Servidores *proxy*

Intermediários entre clientes e servidores em pedidos e respostas HTTP. Um *proxy* é, ao mesmo tempo, um cliente e um servidor. Os pedidos deixam de ser feitos diretamente aos servidores HTTP – passa a ser o *proxy* a responder aos clientes.

São fundamentais sobretudo em ambientes empresariais: aumento do desempenho relativamente aos servidores, uma vez que é usada uma *cache* global numa rede; melhoria da segurança, por exemplo, no controlo de acesso dos utilizadores a determinadas páginas *web* (permitem configurações diferentes para utilizadores diferentes); e, isolamento da rede perante a *Internet*, através da anonimização pelo uso de endereços IP internos.

Operam ao nível da camada de aplicação.

Introduzem latência nas trocas de mensagens.

Squid

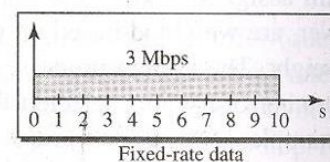
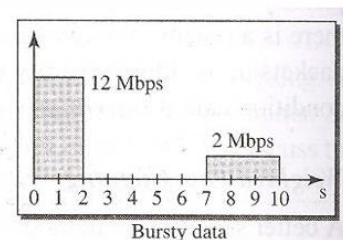
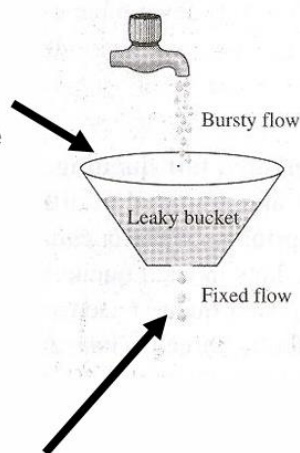
Servidor *proxy*, com *cache* e *open source*.

O controlo de acessos ao *proxy* pode basear-se em: endereço IP e *port* ou nome DNS de origem/destino, data, hora ou dia da semana, protocolo (HTTP, FTP, SSL, ...), entre outros...

Método leaky bucket (balde furado)

O Squid usa o método leaky bucket para controlar a velocidade de acesso

O tamanho do balde representa a quantidade de dados que pode ser processada acima da velocidade média, embora de uma forma esporádica (picos de velocidade)



A taxa de fuga representa a velocidade média admitida

Configuração squid: o básico

Configuração básica

<code>visible_hostname barreira_central</code>	← Nome do proxy server nas mensagens
<code>http_port 192.168.4.5:3128</code>	← Endereço IP e porto TCP onde o proxy vai receber pedidos
<code>acl rede_estcb src 10.6.0.0/16</code>	← Access control list (acl) que representa todos os computadores da rede da estcb
<code>http_access allow rede_estcb</code>	← Diretiva que permite o acesso aos computadores representados pela acl de nome "rede_estcb"

Todas as diretivas de configuração podem ser consultadas em:

<http://www.visolve.com/squid/squid30/contents.php>

Prof. Osvaldo Santos 13

Configuração squid: access control lists (acl)

Uma acl é uma forma de identificar um determinado tipo de tráfego

Sintaxe: `acl <nome> <tipo_de_acl> <parâmetro #1> <parâmetro #2> <parâmetro #n>`

<code>acl todos src 0.0.0.0/0</code>	← Todos os computadores clientes
<code>acl rede_estcb src 10.6.0.0/16</code>	← Os computadores clientes de uma rede
<code>acl servidor_da_ana dst 10.6.7.23/32</code>	← Um servidor específico (destino)
<code>acl sitesSociais dstdomain .facebook.com</code>	← Sites de um domínio DNS (destino)
<code>acl horarioTrabalho time MTWHF 9:00-17:00</code>	← Um determinado horário
<code>acl users proxy_auth REQUIRED</code>	← Utilizadores autenticados através de um programa externo

Prof. Osvaldo Santos 14

Taxa de transferência é o volume de dados transferido por unidade de tempo.

Atraso (ou latência) é o tempo desde o envio até à receção de um pacote.

Jitter é a soma das variações dos atrasos ao longo do tempo. Se o atraso for constante, o *jitter* é nulo. Idealmente, o atraso na entrega dos pacotes numa *stream* deve ser constante, o que equivale a um *jitter* nulo.

RTP (*Real-time Transport Protocol*)

Protocolo de transporte adequado para aplicações multimédia.

Indica o tipo de conteúdo que está a ser transportado; garante sequenciamento de pacotes; permite sincronização, cálculo do *jitter* com *time stamping*, e monitorização de entregas de pacotes.

Não garante entregas de pacotes em tempo útil nem *Quality of Service* (QoS).

SSRC e CSRC são valores que constam no *header* de um pacote RTP. SSRC identifica um fluxo sincronizado de pacotes. CSRC identifica as várias fontes que transmitiram os dados.

RTCP (*Real-time Transport Control Protocol*)

Protocolo usado para transmitir dados de controlo sobre o fluxo RTP.

Tipos de pacotes RTCP

- Receiver report packet
 - Um pacote que o receptor envia ao emissor, com estatísticas sobre os pacotes recebidos (número de pacotes perdidos, atraso, jitter, etc)
- Sender report packet
 - Semelhante ao receiver report, mas neste caso o receptor também actua como emissor pelo que existem campos adicionais para enviar também estatísticas sobre os pacotes enviados
- Source Description RTCP Packet
 - Descreve a fonte de pacotes
- Goodbye RTCP Packet
 - Indica que uma ou mais fontes já não estão activas
- Application Specific RTCP packets
 - Dependem de cada aplicação em particular

O RTP usa um *port* par; as mensagens de controlo de RTCP usam o *port* seguinte. Não se sabe à priori quais os *ports* usados, o que representa um problema na coexistência de configurações de *firewalls*.

RTP e NAT (*Network Address Translation*)

Tipos de NAT

Full Cone: Todos os pedidos do mesmo IP/port interno são mapeados no mesmo IP/port externo. Além disso, podem ser enviados pacotes da Internet para a máquina interna, enviando-os para o IP/port externo mapeado, mesmo sem qualquer “ligação” previamente estabelecida.

Restricted Cone: Semelhante ao Full Cone, mas um computador X só pode enviar pacotes da Internet para a máquina interna, enviando-os para o IP/port externo mapeado, se a máquina interna estabelecer previamente uma “ligação” ao computador X.

Port Restricted Cone: Semelhante ao Restricted Cone, mas o computador X tem que ter sempre um source port igual ao destination port dos pacotes enviados a partir da máquina interna. (no restricted cone o source port pode ser diferente).

Symmetric: Todos os pedidos do mesmo IP/port interno para um determinado IP/port de destino são mapeados no mesmo IP/port externo. Pedidos do mesmo IP/port interno, mas para um destino diferente têm um mapeamento diferente. Um computador X só pode enviar pacotes da Internet para a máquina interna, enviando-os para o IP/port externo mapeado, se a máquina interna estabelecer previamente uma “ligação” ao computador X.

STUN (*Simple Traversal of UDP through NATs*)

Serviço que permite a um cliente saber o endereço IP e o *port* exteriores que lhe foi atribuído pelo protocolo NAT e o tipo de NAT aplicado.

Após ambos os computadores saberem os respetivos endereços IP e *port* externos, a ligação será realizada diretamente entre os dois.

TURN (*Traversal Using Relay NAT*)

Serviço utilizado quando é impossível dois computadores estabelecerem uma ligação direta entre eles (por exemplo, quando o tipo de NAT de ambos é *symmetric*).

Todos os dados passam pelo servidor TURN, pelo que existe um atraso adicional.

ICE (*Interactive Connectivity Establishment*)

Framework que une os serviços STUN e TURN.

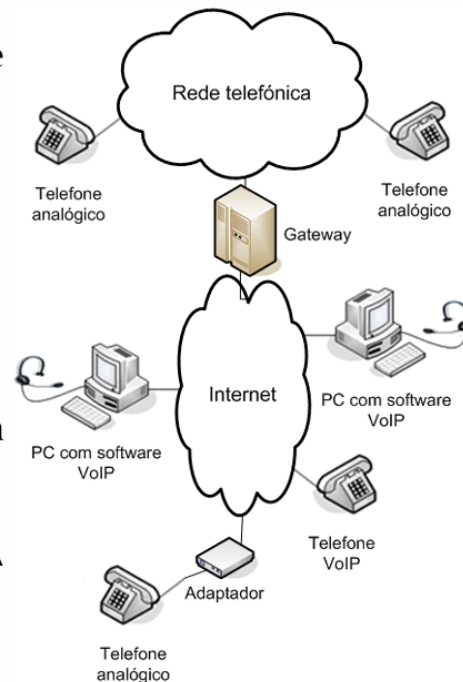
Proposta de norma IETF para resolver o problema do atravessamento de NATs.

Identifica todas as possibilidades de ligação de rede e seleciona a melhor.

Voice over IP (VoIP)

VOIP – Voice Over Internet Protocol

- Permite conversações de voz sobre a Internet
- Interoperável com as redes telefónicas tradicionais
- A voz é digitalizada, comprimida e enviada em pacotes IP (normalmente RTP)
- O desempenho da rede influencia a qualidade das ligações VoIP
 - Internet: Best Effort
 - Redes IP geridas: serviço baseado em SLA
- A sua maior vantagem é a redução de custos



Osvaldo Santos

2

O sistema telefónico tradicional – PTSN (*Public Switch Telephone Network*), usa uma rede própria só para ligações de voz. As ligações são estabelecidas através de um sistema hierárquico de numeração.

É usada comutação de circuitos.

Tipicamente, é cobrado o tempo de utilização.

O VoIP permite reduzir custos com chamadas telefónicas: VoIP para VoIP é, geralmente, gratuito, desde que exista uma ligação à *Internet*; VoIP para rede telefónica tradicional tem custos reduzidos.

Normalmente, existe um *gateway* que permite interligar a rede VoIP com a rede telefónica tradicional.

É independente da localização geográfica, isto é, não há diferenças nem custos adicionais em ligações locais nem internacionais.

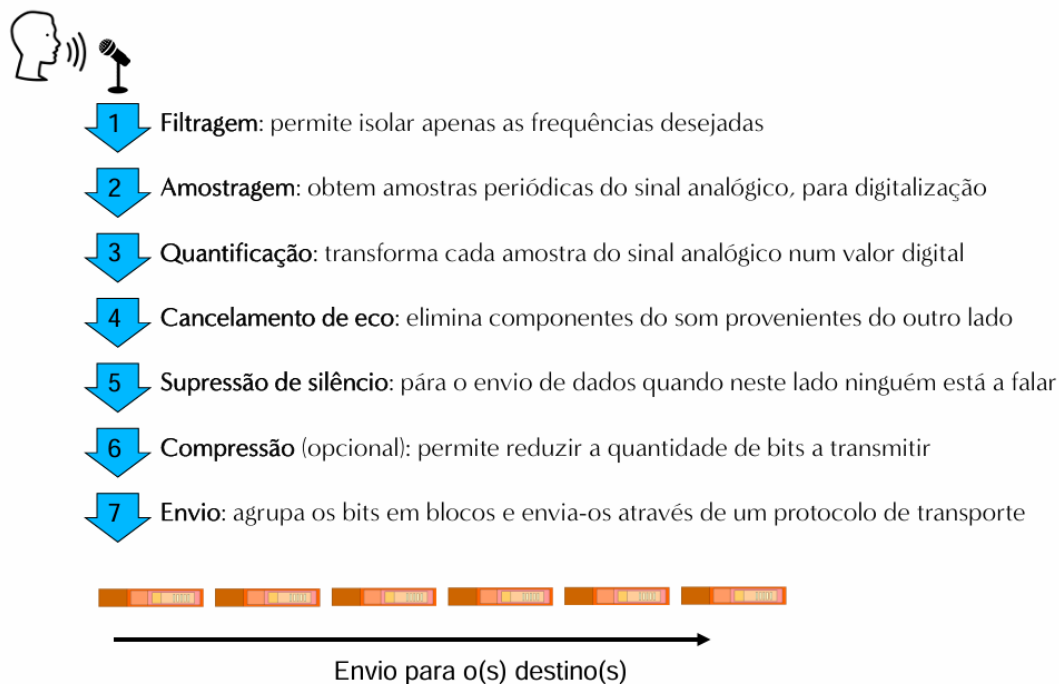
Uma única conexão à *Internet* permite centenas ou mesmo milhares de chamadas em simultâneo.

A combinação com outros serviços como transmissão de vídeo, transferência de ficheiros ou troca de mensagens, permite centralizar serviços de comunicação numa única rede, tornando-os mais fácil de gerir.

No entanto, o uso deste protocolo está sujeito a atrasos superiores e *jitter*, bem como à perda de pacotes de IP e perda de qualidade de áudio por compressão. É, também, mais complexo, e depende do bom funcionamento de vários subsistemas.

Por vezes, é difícil, também, adotar um sistema universal de numeração, assim como encaminhar o tráfego através de *firewalls* e NATs.

VoIP: da voz humana até aos pacotes de dados



Oswaldo Santos

5

Em redes IP, é possível reservar largura de banda exclusivamente para ligações VoIP, ou atribuir prioridade máxima ao tratamento dos pacotes VoIP, evitando perda de dados e atrasos elevados. Em redes locais, é comum criar VLANs próprias para VoIP, atribuindo-lhes valores de prioridade mais altos nas configurações dos *switches*.

Componentes de um sistema VoIP: terminais (telefones tradicionais, telefones IP ou *soft phones*); conversores (de VoIP para PSTN e vice-versa); processador (*media gateway Controller*, servidor *proxy*); *signaling gateway*; entre outros.

Power over Ethernet (PoE)

Permite alimentar dispositivos através da própria cablagem *Ethernet*.

Protocolos para VoIP



SIP (*Session Initiation Protocol*)

Protocolo do tipo cliente/servidor que permite gerir serviços multimédia sobre IP, usando o protocolo HTTP como referência.

Permite iniciar, modificar e terminar sessões entre dois ou mais utilizadores.

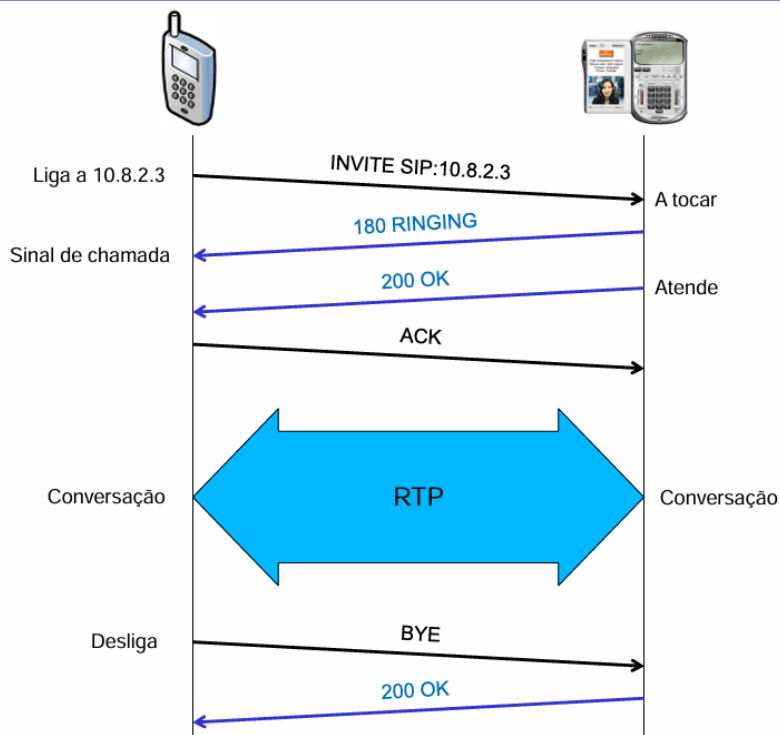
Pode ser usado para aplicações de voz, vídeo, mensagens, etc...

Tipos de servidores SIP

- **Redirect Server**
 - Indica ao cliente um novo set de URIs a usar, redireccionando-o
- **Proxy Server**
 - Servidor intermédio entre o cliente e outro componente SIP
- **Registrar Server**
 - Fornece um serviço de localização que permite registar URIs e associá-los a endereços IP e portos
- **Location Server**
 - Usado para determinar a localização de um determinado URI

Muitos destes papéis coexistem no mesmo servidor

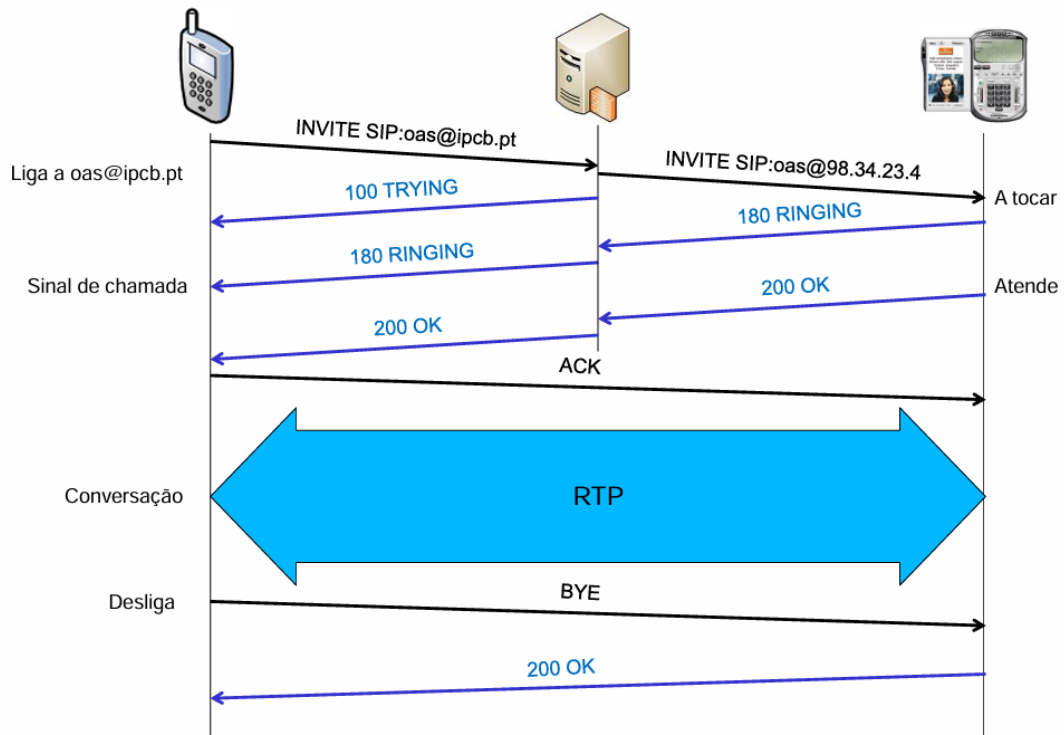
Uma ligação SIP básica



Osvaldo Santos

32

Uma ligação SIP com proxy server



Osvaldo Santos

33

Encaminhamento IP

Unicast: um único destinatário; é usado o endereço IP que identifica esse destinatário.

Broadcast: todos os dispositivos numa subrede; é usado um endereço IP especial (endereço de *broadcast*) que representa todos os dispositivos de uma subrede.

Anycast: pelo menos um dispositivo de um grupo; é útil quando existem vários servidores semelhantes e se pretende selecionar um deles (por exemplo, para balanceamento de carga)

Multicast: todos os dispositivos de um grupo; é usado um endereço IP especial (endereço de *multicast*) que representa todos os dispositivos de um grupo; os fluxos de dados só são clonados para os nós da rede necessários, isto é, numa transmissão, só receberão dados os dispositivos que “pedem” para os receber.

Hosts: níveis de conformidade

Os dispositivos podem estar em 3 categorias diferentes de conformidade:

- Nível 0
 - Não há qualquer suporte para multicasting
- Nível 1
 - Suporta o envio de datagramas multicasting, mas não a receção
 - O host não consegue juntar-se a grupos multicasting
- Nível 2
 - Suporte total a multicasting
 - Suporta o envio e receção de datagramas multicasting
 - Pode juntar-se a um ou mais grupos multicasting

Endereços IPv4 de multicasting

- Os endereços da antiga Classe D (desde 224.0.0.0 a 239.255.255.255) estão reservados para multicasting
- Os endereços de 224.0.0.0 a 224.0.0.255 foram reservados pela IANA para serem usados apenas na rede local. Pacotes com estes endereços de destino não devem ser encaminhados pelos routers
- Exemplos de grupos de multicasting locais predefinidos
 - 224.0.0.1 : todos os sistemas desta sub-rede
 - 224.0.0.2 : todos os routers desta sub-rede
 - 224.0.0.5 : todos os routers OSPF
 - 224.0.0.6 : todos os routers OSPF DR (designated routers)
 - 224.0.0.12 : Agente DHCP server/relay
- <http://www.iana.org/assignments/multicast-addresses>

IGMP e PIM

IGMP (*Internet Group Management Protocol*) – permite aos dispositivos negociarem de forma dinâmica a sua participação num grupo *multicasting*.

PIM (*Protocol Independent Multicast*) – é usado para construir árvores de distribuição de dados *multicasting*.