Instituto Politécnico de Bragança

Escola Superior de Tecnologia e de Gestão

Departamento de Informática e Comunicações

Comunicações por Computador II

Engenharia Informática – 4º ano

Nuno Rodrigues

nuno@ipb.pt

2002

Índice

Capítulo 1 - Aplicações, Serviços e Protocolos da Camada de Aplicação

- Os **protocolos de alto nível** (protocolos de aplicação) comunicam com aplicações em outros *hosts*, desempenhando o papel da interacção com os utilizadores, na pilha protocolar TCP/IP
- Em geral, os protocolos de alto nível possuem um conjunto de características comuns:
  - Podem-se traduzir em aplicações desenvolvidas pelos utilizadores, ou aplicações standard incluídas na própria pilha TCP/IP (p.e. Telnet, FTP, etc)
  - Utilizam UDP ou TCP como mecanismo de transporte
  - Muitos utilizam como meio de interacção o modelo cliente/servidor

1.1. O Modelo Cliente/Servidor

- O TCP é um protocolo orientado à conexão, não funcionando com base em relações do tipo **master/slave**
- No entanto, as aplicações utilizam o **modelo cliente/servidor**, para a comunicação entre si
- Um **Servidor** (*server*) é uma aplicação que oferece serviços a utilizadores
- Um **Cliente** (*client*) é um programa que requisita serviços
- Uma **Aplicação** normalmente engloba as duas partes, Cliente e Servidor, que podem correr na mesma ou em diferentes máquinas

?? Os utilizadores invocam a parte Cliente da aplicação, que formula um pedido para um determinado serviço e o envia à parte Servidor, através da rede TCP/IP

?? O Servidor recebe os pedidos dos clientes, processa-os e reenvia de volta os resultados

?? Um Servidor, em geral, pode processar vários pedidos (atender vários clientes) ao mesmo tempo

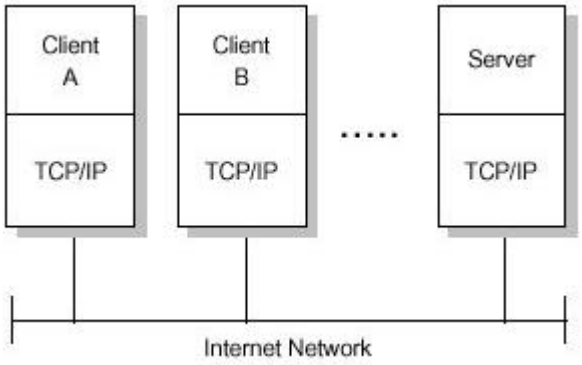


Figura 1.1 – Modelo Cliente Servidor

?? Nas próximas secções serão apresentados os protocolos aplicacionais mais utilizados

1.2. *Domain Name System* (DNS)

- O DNS é um protocolo *standard*, com o STD 13. Encontra-se descrito nos RFC's 1034 e 1035
- Permite efectuar o mapeamento de endereços IP para nomes de máquinas e vice-versa

1.2.1. A estrutura hierárquica do espaço de nomes

- As organizações são, na maior parte dos casos, caracterizadas por estruturas internas desenvolvidas de uma forma hierárquica, de acordo com múltiplos critérios
- Os nomes de domínio do DNS são formados da mesma forma, reflectindo ao mesmo tempo a delegação hierárquica de autoridade sobre esses mesmos nomes

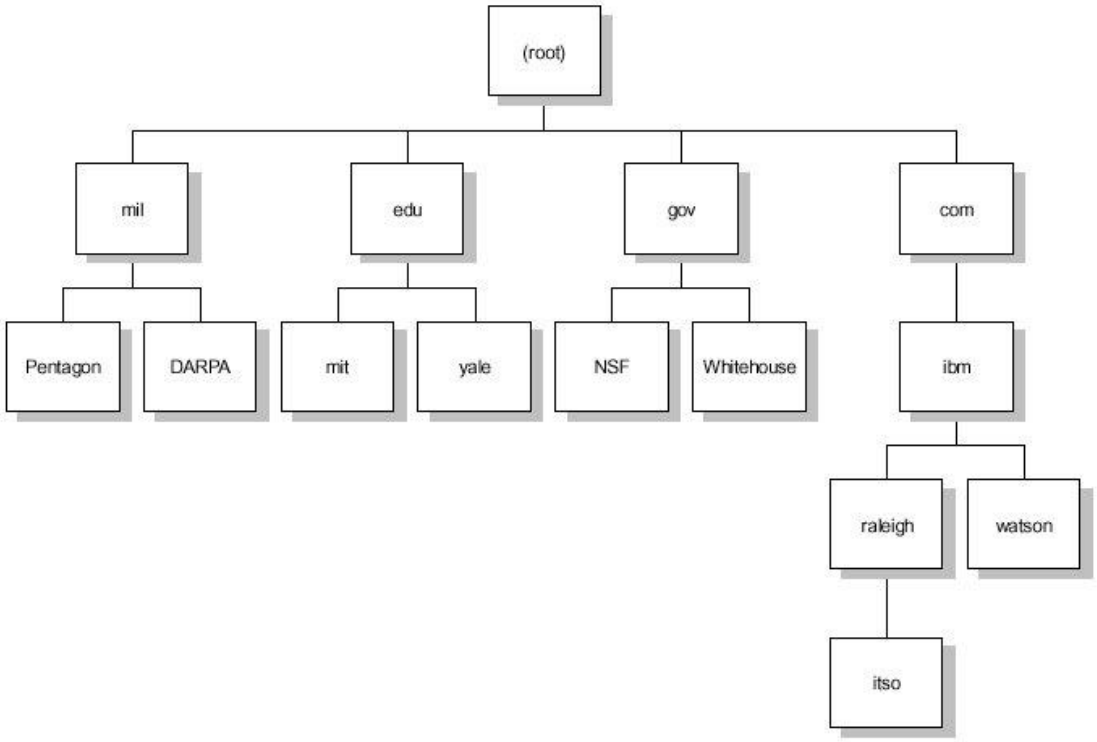


Figura 1.2 – Espaço Hierárquico de Nomes

- O nome completo de um *host* designa-se por *Fully Qualified Domain Name* – FQDN

?? Estamos na presença de um FQDN sempre que o nome inclui toda a hierarquia de sub-domínios até à raíz da estrutura

?? Esta raíz da hierarquia é representada por um **ponto final** (**.**)

?? Exemplo de um FQDN: "**elara.ipb.pt.**", onde "**ipb.pt.**" corresponde ao sub-domínio e **elara** corresponde ao nome (local) atribuído ao *host*

?? Por questões de facilidade de manuseamento dos nomes, o ponto final normalmente não é representado pelos utilizadores, sendo no entanto sempre considerado pelas aplicações

?? Imediatamente a seguir à raíz da hierarquia DNS surgem os **domínios de topo** (*top-level domains*)

?? Estes incluem os códigos de dois caracteres ISO 3166 de cada país, mais um conjunto de domínios genéricos de topo, que caracterizam diferentes tipos de organizações (de acordo com classificação definida nos EUA)

?? O domínio de topo correspondente a Portugal, de acordo com a norma ISO 3166, é **.pt**, sendo gerido pela Fundação para a Computação Científica Nacional – FCCN (www.fccn.pt)

Table 4. DNS - Some Top-Level Internet Domains

Domain Name	Meaning
com	Commercial organizations
edu	Educational institutions
gov	Government institutions
int	International organizations
mil	U.S. Military
net	Major network support centers
org	Non-profit organizations
country code	ISO standard 2-letter identifier for country-specific domains

Figura 1.3 – Principais domínios DNS de topo

?? Durante o ano 2000 foram propostos mais sete domínios de topo, que deverão estar disponíveis para registo durante o ano de 2001:

- **.aero**: *Air-transport industry*

- **.biz**: *Businesses*
- **.coop**: *Cooperatives*
- **.info**: *Unrestricted use*
- ***.museum***: *Museums*
- ***.name***: For registration by individuals
- **.pro**: *Accountants, lawyers, and physicians*

1.2.2. Mapeamento de nomes de domínio em endereços IP e vice-versa

- O mapeamento de nomes em endereços IP e vice-versa é efectuado por programas específicos, denominados **Servidores de Nomes** (*name servers*)
- Cada servidor mantém uma base de dados com as correspondências entre endereços IP e nomes, que partilha com outros servidores e com clientes que solicitam a resolução de endereços
- Conceptualmente, os servidores de nomes estão organizados numa estrutura em árvore, que corresponde à estrutura hierárquica de nomes
- Enquanto que a procura de um endereço IP a partir do nome é relativamente simples, dada a estrutura hierárquica dos nomes, o processo inverso já não pode seguir esta hierarquia
- Assim, para resolver este problema, foi criada outra hierarquia de nomes, que faz o mapeamento inverso (*reverse mapping*) e cujo domínio de topo é **in-addr.arpa**
- Dado que os nomes de domínio têm a parte menos significativa do nome primeiro mas os endereços IP contém os bytes mais significativos primeiro, o endereço decimal pontuado é representado em ordem inversa (*reverse order*)
- P.e. o nome de domínio que corresponde ao endereço IP 193.136.195.220 é **220.195.136.193.in-addr.arpa**

- Uma *query* para encontrar um nome associado a um endereço IP é denominada ***pointer query***

1.2.3. A natureza distribuída do espaço de nomes

- O DNS utiliza o conceito de **espaço de nomes distribuído**
- Os nomes são agrupados em **zonas de autoridade**
- Em cada uma desta zonas, um ou mais servidores tem a tarefa de manter uma base de dados de endereços e nomes, além de responder a pedidos de resolução de nomes e endereços de clientes
- Estes servidores locais de nomes encontram-se logicamente interconectados, numa estrutura hierárquica de domínios
- Cada zona contém uma parte da árvore hierárquica, sendo os nomes no interior de cada zona administrados de forma independente das outras zonas
- A delegação das zonas de autoridade é efectuada nos servidores de nomes
- A divisão do espaço de nomes em zonas é conseguido usando **resource records**, manuseados pelo DNS

1.2.4. O processo de resolução de nomes

- O processo de resolução de nomes pode ser resumido nos seguintes passos:

    1. Uma aplicação formula um pedido de resolução de nome, recorrendo, p.e., à chamada *gethostbyname()*
    2. O *resolver* formula uma *query* ao servidor de nomes
    3. O servidor de nomes verifica se a resposta se encontra na base de dados da zona de autoridade ou *cache* local, e, em caso positivo, retorna esta resposta ao cliente. Em caso contrário, este vai interrogar outros servidores disponíveis, começando pela raíz da

estrutura DNS, ou pelo ponto mais alto possível dessa estrutura

4. A aplicação do utilizador obtém o endereço IP correspondente (ou o nome, dependendo da *query*), ou um erro, se a *query* não puder ser respondida

?? As mensagens de *query/reply* são transportadas em TCP e/ou UDP

?? A resolução de nomes de domínio é um processo cliente/servidor

?? A função cliente (denominada *resolver* ou *name resolver*) é transparente para o utilizador e é chamada por uma aplicação para resolver nomes em endereços IP ou vice-versa

?? O Servidor de Nomes é a aplicação servidora do sistema, fornecendo a tradução entre nomes e endereços

?? O *Resolver* pode ser de dois tipos:

- *Full Resolver*: programa distinto das aplicações dos utilizadores, que encaminha todos os pedidos de resolução para o servidor de nomes

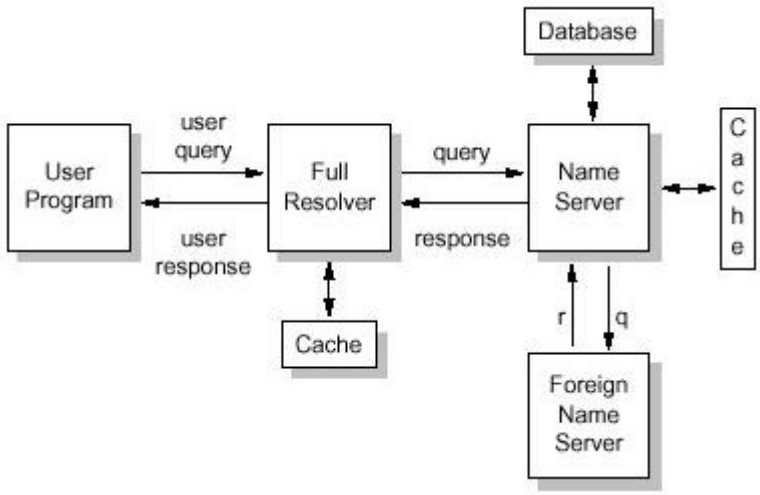


Figura 1.4 – Processo de Resolução de Nomes usando um *Full Resolver*

- *Stub Resolver*: é uma rotina incluída nos próprios programas dos utilizadores, que encaminha as *queries* para o servidor de nomes. Em UNIX, o *Stub Resolver* é implementado por duas rotinas: *gethostbyname()* e *gethostbyaddr()*

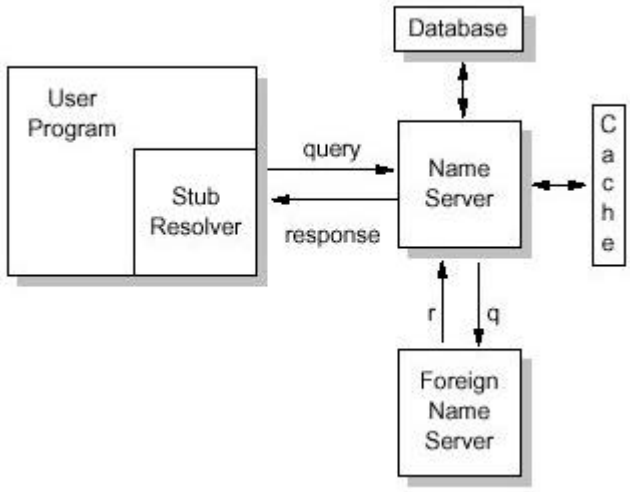


Figura 1.5 – Processo de Resolução de Nomes usando um *Stub Resolver*

- As **Queries** podem ser de dois tipos (a escolha do tipo de *query* é assinalada através de uma *flag bit* na própria *query*):
  - **Recursivas**: O servidor de nomes contactado processa a *query* por forma a determinar a totalidade da informação necessária e responde ao cliente
  - **Iterativas**: O servidor de nomes pode retornar apenas a informação que tem disponível, fornecendo ainda uma lista de servidores adicionais para o cliente contactar e completar a *query*
- As **Respostas** de pedidos de resolução podem também ser de dois tipos (também assinaladas com uma *flag bit* na mensagem de resposta):
  - *Authoritative*: respostas à resolução de nomes ou endereços relativos a zonas cuja **autoridade está delegada** ao servidor que as está a dar

- *Non-Authoritative*: respostas à resolução de nomes ou endereços relativos a zonas cuja **autoridade não está delegada** ao servidor que as está a dar

?? Quando um servidor (ou um programa *full resolver*) recebe uma resposta, armazena-a em *cache*, para aumentar a performance em *queries* repetidas

?? Cada entrada na *cache* é armazenada por um período máximo de tempo, especificado pelo originador no campo *time-to-live* - TTL (32 bits), da mensagem de resposta. 86400 segundos (um dia) é um TTL típico

?? Existem três tipos de **Servidores de Nomes**:

- **Primários**: um servidor primário de determinada zona carrega a informação das mesmas a partir do disco rígido, possuindo autoridade sobre essas zonas
- **Secundários**: um servidor secundário de determinada zona possui autoridade sobre a mesma, mas obtém a informação a partir de um servidor primário, utilizando um processo denominado *zone transfer*. Para manter a informação actualizada é usado um processo de sincronização com o servidor primário
- **Cache (***Caching-only*): servidor de nomes que não possui autoridade sobre qualquer zona, limitando-se a contactar servidores primários e/ou secundários para obter a informação DNS

?? Um servidor de nomes pode operar como primário ou secundário para múltiplos domínios, ou ainda como primário para alguns domínios e como secundário para outros

?? Um servidor primário ou secundário executa todas as funções de um servidor de cache

?? Quando é registado um domínio correspondente a uma zona de autoridade diferente e que inclua a raíz (.), aplicam-se as seguintes regras:

- O domínio tem de ser registado pelo administrador da raíz (ou pelo administrador do nível imediatamente acima)
- Tem de existir um administrador identificado para o domínio
- Tem de existir pelo menos dois servidores com autoridade sobre a zona a criar

1.2.5. Os Resource Records (RRs) DNS

- A base de dados distribuída do DNS é composta por *resource records* (RRs), que fornecem o mapeamento entre nomes de domínio e objectos de rede (endereços IP, etc)
- O início de uma zona é marcado pelo registo **Start of Authority (SOA),** terminando esta com o registo **NS** (*Name Server Record*), que aponta para um servidor de nomes que detém a autoridade sobre essa zona
- A autoridade sobre a raíz da estrutura do DNS é mantida por um conjunto de servidores de nomes (actualmente 13), denominados *root name servers*
- A lista actual destes servidores pode ser obtida em ftp.rc.internic.net, no ficheiro domain/named.root
- O formato geral de um *resource record* é:

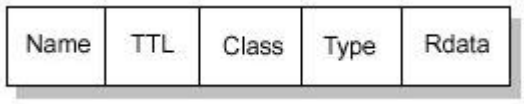


Figura 1.6 – Formato genérico de um *RR*

- Onde:
  - *Name*: onde o nome do domínio é definido

- *TTL*: período de vida (*time-to-live*), em segundos, que este registo pode ser válido, na cache de um servidor de nomes

- *Class*: Identifica a família de protocolos usada. O único valor normalmente usado é *IN* (*Internet system*)

- *Type*: Identifica o tipo de recurso, neste registo. Cada tipo tem um nome e um valor, estando os mais comuns representados na tabela seguinte:

Type	Value	Meaning
A	1	A host address.
CNAME	5	Canonical name of an alias; specifies an alias name for a host.
HINFO	13	The CPU and OS used by the host; this is only a comment field.
MX	15	A mail exchange for the domain; specifies a domain name for a mailbox. This is used by SMTP (see 4.7.2, "SMTP and the Domain Name System" on page 191 for more information).
NS	2	The authoritative name server for a domain.
PTR	12	A pointer to another part of the domain name space.
SOA	6	The start of a zone of authority in the domain name space.
WKS	11	Well-known services; specifies that certain services (for instance SMTP) are expected to be always active on this host.

Figura 1.7 – Alguns tipos de RR's

- *Rdata*: Este valor depende do tipo. Por exemplo:

  - *A:* endereço IP de 32 bits

  - *CNAME:* nome de domínio

  - *MX*: valor de preferência de 16 bits, seguido do nome de domínio (os valores mais baixos são prioritários)

  - *NS*: nome de um *host*

  - *PTR*: nome de domínio

```
options {
     directory "/var/named/";
     allow-transfer {
          193.136.194.0/24;
          193.136.195.0/24;
          193.136.231.0/24;
          193.136.0.1;
          193.136.1.158;
     };
     allow-query {any; };
};
zone "." {
     type hint;
     file "db.cache";
};
zone "0.0.127.in-addr.arpa." {
     type master;
     file "db.local";
};
zone "194.136.193.in-addr.arpa."{
     type master;
     file "primary/db.194.136.193.rev";
};
zone "195.136.193.in-addr.arpa."{
     type master;
     file "primary/db.195.136.193.rev";
};
zone "ipb.pt."{
     type master;
     file "primary/db.ipb.pt";
};
zone "estig.ipb.pt."{
     type master;
     file "primary/db.estig.ipb.pt";
};
zone "bcd.pt."{
     type master;
     file "primary/db.bcd.pt";
};
zone "ruralnet.pt."{
     type slave;
     file "secondary/db.ruralnet.pt";
     masters{
          195.23.73.250;
     };
};
zone "cm-braganca.pt."{
     type master;
     file "primary/db.cm-braganca.pt";
};
```

Figura 1.8 – Exemplo de configuração de algumas zonas no servidor de DNS do IPB

```
; Sub-domínio do IPB
$TTL 86400
$ORIGIN ipb.pt.
@               IN      SOA     elara.ipb.pt.   root.ipb.pt. (
                        2001030701 ; serial
                        10800 ; refresh
                        3600 ; retry
                        604800 ; expire
                        86400 ; default_ttl
                        )
ipb.pt.         IN      NS      elara.ipb.pt.
ipb.pt.         IN      NS      io.ipb.pt.
$ORIGIN ipb.pt.
io              IN      A       193.136.195.219
proxy           IN      CNAME   io
elara           IN      A       193.136.195.220
dns             IN      CNAME   elara
ieee            IN      CNAME   elara
mail            IN      CNAME   elara
pop             IN      CNAME   elara
www             IN      CNAME   elara
www.teb         IN      CNAME   elara
pan             IN      A       193.136.195.222
news            IN      CNAME   pan
puck            IN      A       193.136.195.225
adm             IN      CNAME   puck
ftp             IN      CNAME   puck
linux           IN      CNAME   puck
linuxberg       IN      CNAME   puck
tucows          IN      CNAME   puck
elara2          IN      A       193.136.195.226
gtipb           IN      CNAME   router3.eth0.ccom.ipb.pt.
quake           IN      CNAME   moon.ccom.ipb.pt.
unreal          IN      CNAME   moon.ccom.ipb.pt.
ipb.pt.         IN      A       193.136.195.220
www.jornadas-clima      IN      CNAME   phobos.alunos.ipb.pt.
gtdial          IN      CNAME   router1.eth0.ccom.ipb.pt.
fti             IN      CNAME   gimonde.estig.ipb.pt.
www.fti         IN      CNAME   gimonde.estig.ipb.pt.
ipb.pt.         IN      MX      10      mail.ipb.pt.
ipb.pt.         IN      MX      15      pan.ipb.pt.
```

Figura 1.9 – Exemplo da configuração do sub-domínio ipb.pt

1.2.6. Mensagens DNS

?? Todas as mensagens utilizadas pelo protocolo DNS utilizam um formato idêntico ao da figura seguinte

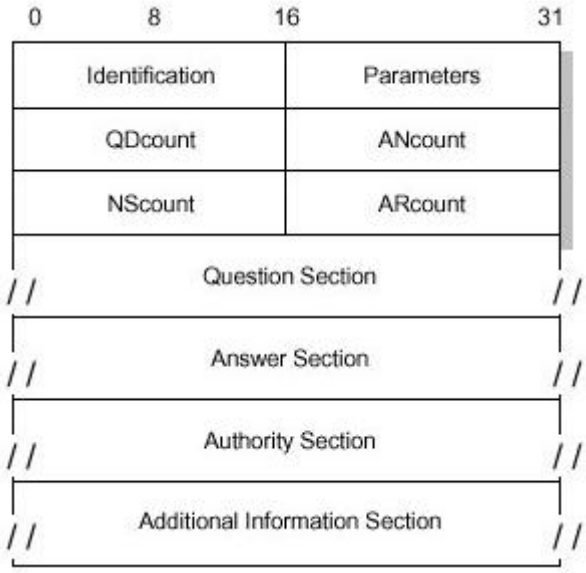


Figura 1.10 – Formato de uma Mensagem DNS

??O cabeçalho é usado em todos os tipos de mensagens

??Cada uma das restantes secções é usada apenas de acordo com as circunstâncias (*queries, replies*, etc)

1.2.7. DNS em IPv6

?? Com a introdução de endereços de 128 bits, o IPv6 torna mais difícil para os utilizadores a utilização dos endereços IP para acesso a recursos da rede

?? Neste sentido, o DNS assume um papel ainda mais essencial, no funcionamento de uma rede baseada em IPv6

?? Foi assim desenvolvido um conjunto de extensões ao DNS, para permitir a utilização deste com endereços IPv6 (RFC 1886 – *DNS Extensions to Support IP Version 6*)

?? Foram definidas as seguintes extensões:

- O novo *Resource Record* **AAAA**, que faz o mapeamento de um nome de domínio para um endereço IPv6
- Um novo domínio, usado para suportar a resolução de nomes do tipo endereço-para-domínio
- Uma alteração na definição das *queries* existentes, para que processem correctamente *RR's* **A** e **AAAA**

?? O *RR* AAAA é similar ao *RR* A, variando no entanto a secção *Data*, que passa a acomodar um endereço de 128 bits e o campo *Type*, que toma o valor decimal 28

?? O domínio **IP6.INT** é usado para fazer o *inverse lookup* dos endereços para nomes (de forma similar ao domínio *in-addr.arpa* do IPv4).

?? Por exemplo, o endereço *2222:0:1:2:3:4:5678:9abc*

tem como nome inverso:

c.b.a.9.8.7.6.5.4.0.0.0.3.0.0.0.2.0.0.0.1.0.0.0.0.0.0.0.2.2.2.2.IP6.INT

1.3. Serviços de Directoria

- Os serviços de directoria são serviços de rede que **identificam os recursos nessa mesma rede** e os tornam acessíveis para os utilizadores e as aplicações
- Os recursos incluem endereços de e-mail, computadores, periféricos, etc
- Idealmente, os serviços de directoria devem tornar a topologia física das redes e os protocolos transparentes, para que cada utilizador possa aceder a qualquer recurso sem precisar de saber como e onde está fisicamente ligado
- Dois serviços de directoria destacam-se, actualmente, pela sua utilização:
  - LDAP – *Lightweight Directory Access Protocol*
  - NDS – *Novell Directory Services*: serviço proprietário da Novell, utilizado em redes Novel Netware
- Estes dois serviços (tal como a maior parte dos serviços de directoria) baseiam-se no protocolo standard X.500

1.3.1. O Protocolo X.500

- O protocolo X.500 é um standard do CCITT, tendo-se tornado também uma norma ISO com o número ISO 9594
- O X.500 organiza as entradas no **directório** num **espaço de nomes hierárquico**, com capacidade para suportar grandes quantidades de informação
- Define ainda poderosas capacidades para extracção da informação, a partir do directório
- Especifica também a forma de comunicação entre o cliente e o servidor de directório, utilizando o protocolo **DAP** – *Directory Access Protocol*
- Implementa o serviço de directoria global de uma forma **distribuída**

?? A informação de cada organização é mantida localmente em um ou mais **DSA**'s (*Directory System Agents*)

?? É possível que um único DSA armazene informação de mais do que uma organização

?? Ao mesmo tempo, também é possível que a informação de uma grande organização se encontre distribuída por mais do que um DSA

?? Um DSA é essencialmente uma base de dados:

- onde a informação é armazenada de acordo com a estrutura definida no modelo de informação do X.500
- que tem a capacidade de, quando necessário, trocar dados com outros DSA's, recorrendo ao protocolo DSP (*Directory System Protocol*)

?? Todos os DSA's num serviço de directoria X.500 estão interligados de acordo com a *Directory Information Tree* (DIT)

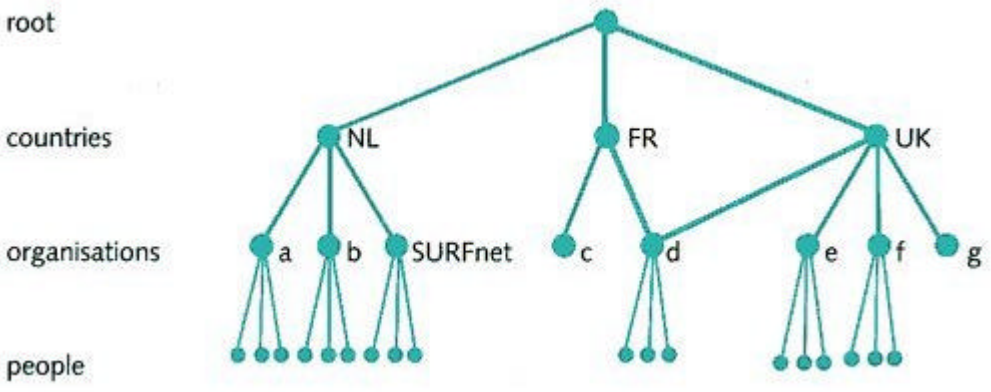


Figura 1.11 – Esquema simplificado de uma DIT, do X.500

?? A **DIT** é uma estrutura hierárquica e virtual de dados, que corresponde na prática à **raíz**, seguida dos códigos dos países.

?? A seguir aos países normalmente definem-se organizações, às quais se podem seguir unidades (*units*) ou recursos individuais

- De acordo com o modelo de informação do X.500, todos os dados são armazenados em entradas (*entries*), pertencendo cada uma a pelo menos uma **classe de objectos** (*object-class)*

- Nas implementações, as principais classes de objectos são *country*, *organization*, *organizational unit* e *person*

- A informação é determinada, em cada entrada, de acordo com os **atributos** aí referidos

- A definição da classe do objecto identifica automaticamente que tipos de atributos podem ser usados em cada entrada

```
Attribute type Attribute value
Object Class: top
person
Common Name: Peter Jurg
P. Jurg
Surname: Jurg
Postal Address: SURFnet bv
Postbus 19035
NL-3501 DA Utrecht
Telephone Number: +31 30 305305
Facsimile Telephone Number: +31 30 305329
Mail: jurg@surfnet.nl
```

Figura 1.12 – Exemplo de um entrada num directório X.500, referente a um objecto do tipo *person*

- Devido às complexidades de implementação, não se desenvolveram, ao longo dos anos, implementações suficientemente atractivas para a sua utilização em massa. O X.500 É assim usado essencialmente como base de outros protocolos deste tipo

1.3.2. Lightweight Directory Access Protocol (LDAP)

- O LDAP define um método standard de **acesso e actualização de informação num directório**

- Foi desenvolvido como uma alternativa mais leve ao protocolo DAP do X.500

- Enquanto o protocolo X.500 requer toda a pesada pilha protocolar do modelo OSI, o LDAP funciona sobre a mais leve pilha protocolar TCP/IP

?? A primeira versão do LDAP foi definida no RFC 1487 – *X.500 Lightweight Access*, que foi actualizado pelo RFC 1777 – *Lightweight Directory Access Protocol*, dando origem à versão 2 deste protocolo (em conjunto com os RFC's 1778, 1779, 1959 e 1960)

?? A versão 3 do LDAP está definida no RFC 2251 – *Lightweight Directory Access Protocol (v3)*

?? O LDAP define o transporte e o formato das mensagens usadas por clientes para aceder a directórios compatíveis X.500

?? Como um Servidor X.500 não entende as mensagens LDAP (o 1ºfunciona sobre OSI e o 2º sobre TCP/IP), torna-se necessário recorrer a um gateway para um cliente LDAP aceder a um servidor X.500

?? A esse gateway dá-se o nome de Servidor LDAP

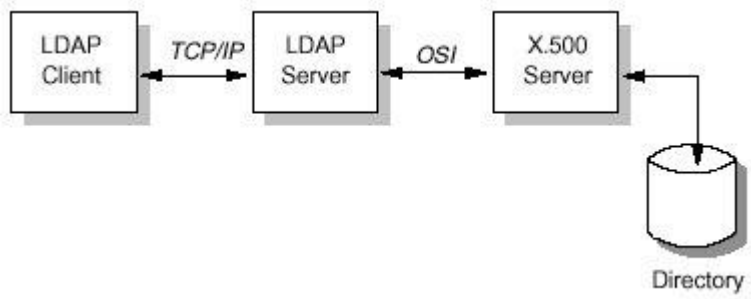


Figura 1.13 – Servidor LDAP operando como Gateway para um Servidor de Directoria X.500

?? Quando não há a necessidade do acesso a servidores X.500, o próprio LDAP permite a criação de Servidores de Directório LDAP, que precisam apenas de suportar as capacidades requeridas por este protocolo

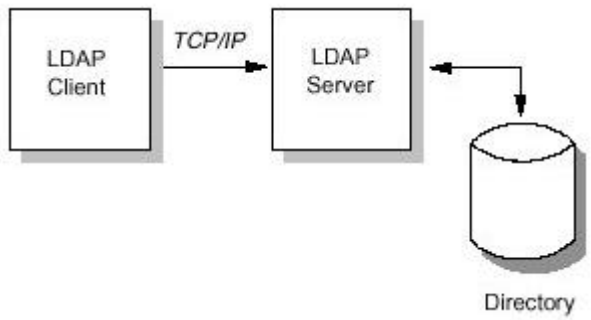


Figura 1.14 – Servidor LDAP *Sand-Alone*

- Genericamente, a interacção entre um cliente LDAP e um servidor LDAP funciona da seguinte forma:
  - O cliente LDAP estabelece uma sessão com o servidor (após especificar o nome ou o endereço IP e o porto onde o servidor está à escuta)
  - O cliente pode indicar um *username* e uma *password* para se autenticar perante o servidor, ou então pode estabelecer uma sessão anónima, com direitos de acesso por defeito
  - De seguida, o cliente realiza operações sobre os dados do directório (de leitura e actualização)
  - Quando o cliente termina os seus pedidos, termina a sessão com o servidor
- Como o LDAP foi desenvolvido inicialmente com o objectivo de ser uma alternativa mais leve ao DAP, para acesso a directórios X.500, os servidores LDAP seguem o modelo X.500
- Assim, o directório armazena e organiza as estruturas de dados em **entradas** (*entries*)
- Uma entrada do directório normalmente **descreve objectos**, como pessoas, impressoras, servidores, etc
- Cada entrada tem um nome, conhecido por *distinguished name* (**DN**), que a identifica de forma unívoca
- Cada DN consiste numa sequência de partes, denominadas *relative distinguished name* (**RDN**)
- As entradas podem organizam-se segundo uma estrutura em árvore hierárquica, baseada nos seus DN's
- Esta árvore de entradas do directório denomina-se *Directory Information Tree* (**DIT**)
- O LDAP define **operações** para aceder e modificar as entradas no directório, entre as quais:
  - **Procura** por entrada de acordo com critérios definidos pelos utilizadores

- **Adicionar** uma entrada
- **Eliminar** uma entrada
- **Modificar** uma entrada
- **Modificar o DN ou o RDN** de uma entrada (operação de mover)
- **Comparar** uma entrada

?? O LDAP é estruturado de acordo com **quatro modelos**:

- **de Informação** (*Information*): descreve a estrutura da informação armazenada no directório LDAP
- **de Nomeação** (*Naming*): descreve como a informação é organizada e identificada no directório
- **Funcional** (*Functional*): descreve as operações que podem ser realizadas sobre a informação armazenada no directório
- **de Segurança** (*Security*): descreve como a informação armazenada num directório LDAP pode ser protegida de acessos não autorizados

1.3.2.1. Modelo de Informação

?? Cada entrada no directório contém um ou mais **atributos** que a descrevem

?? Cada atributo têm um **tipo** e um ou mais **valores**

?? O tipo é definido de acordo com um conjunto de regras de **sintaxe**, que o caracterizam

Table 15 (Page 1 of 2). Common LDAP Attribute Syntaxes

Syntax	Description
bin	Binary information.
ces	Case exact string, also known as a directory string. Case is significant during comparisons.
cis	Case ignore string. Case is not significant during comparisons.
tel	Telephone number. The numbers are treated as text, but all blanks and dashes are ignored.

Table 15 (Page 2 of 2). Common LDAP Attribute Syntaxes

Syntax	Description
dn	Distinguished name.
Generalized Time	Year, month, day, and time represented as a printable string.
Postal Address	Postal address with lines separated by "$" characters.

Figura 1.15 – Tipos de sintaxe dos atributos do LDAP

Table 16. Common LDAP Attributes

Attribute, Alias	Syntax	Description	Example
commonName, cn	cis	Common name of an entry	John Smith
surname, sn	cis	Surname (last name) of a person	Smith
telephoneNumber	tel	Telephone number	512-838-6008

Figura 1.16 – Alguns atributos comuns do LDAP

?? Uma classe de objecto é uma descrição geral (também designada por *template*) de um objecto

?? Um **Esquema** define que classes de objectos são permitidas num directório, que atributos estes podem conter, quais os atributos opcionais e qual a sintaxe de cada atributo

?? A verificação de um Esquema assegura que todos os atributos requeridos estão presentes numa entrada, antes de esta ser armazenada

Table 17. Object Classes and Required Attributes

Object Class	Description	Required Attributes
InetOrgPerson	Defines entries for a person	commonName (cn) surname (sn) objectClass
organizationalUnit	Defines entries for organizational units	ou objectClass
organization	Defines entries for organizations	o objectClass

Figura 1.17 – Classes de objectos

?? Cada servidor define os seus próprios esquemas, no entanto, por questões de interoperabilidade existe um conjunto base de esquemas padrão

1.3.2.2. Modelo de Nomes

?? O modelo de nomes define como as **entradas são identificadas e organizadas**

?? Estas são organizadas na *Directory Information Tree* (DIT), de acordo com os *distinguished names* (DN) e *relative distinguished names* (RDN)

?? Cada RDN de um DN corresponde a um braço da DIT, sendo derivados dos atributos da entrada do directório

?? Genericamente, um RDN tem o formato <attribute-name>=<value>

?? Um DN é composto por uma sequência de RDN's, separados por virgulas

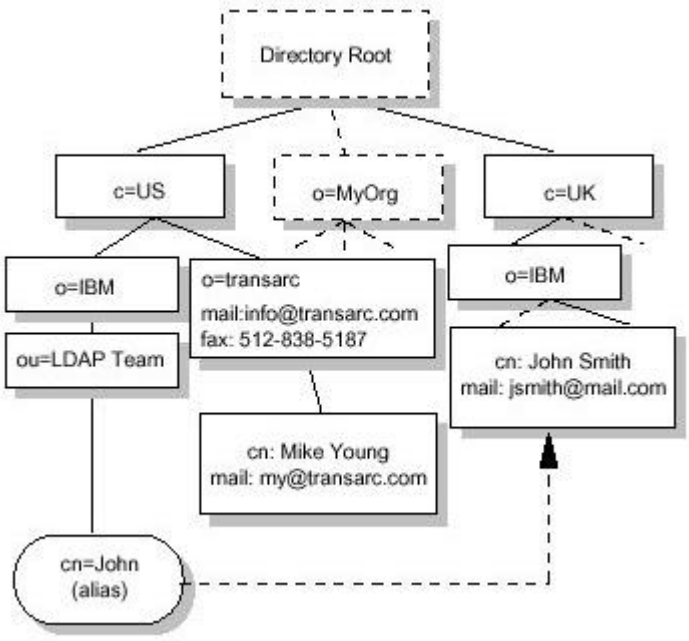


Figura 1.18 – Exemplo de uma *Directory Information Tree* (DIT)

?? As entradas são identificadas de acordo com a sua posição na DIT

- P.e. a entrada no directório mais em baixo à direita (na figura acima) tem o DN **cn=John Smith,o=IBM,c=UK**

1.3.2.3. Modelo Funcional

- As operações que permitem o acesso e a alteração de dados num directório LDAP dividem-se em três categorias:
  - *Query*: inclui operações de pesquisa e de comparação, utilizadas para obter informação do directório
  - *Update*: inclui as operações de adição, eliminação e actualização, usadas para actualizar a informação do directório
  - *Autentication*: inclui as operações de descoberta de um servidor, autenticação, e abandono de uma sessão, que permitem o estabelecimento dos direitos de acesso e de protecção da informação

1.3.2.4. Modelo de Segurança

- O termo segurança deve ser considerado de acordo com os seguintes aspectos:
  - *Autenticação*: assegura que a outra parte (máquina ou pessoa) realmente é quem diz que é
  - *Integridade*: assegura que a informação que chega é exactamente a mesma que partiu do outro lado da comunicação
  - *Confidencialidade*: protege a informação contra olhares indiscretos, ao longo do canal de comunicação
  - *Autorização*: assegura que a outra parte tem permissões para fazer aquilo que solicita. Normalmente o processo de autorização segue-se ao de autenticação
- O LDAP define três **mecanismos** diferentes de autenticação, integridade e confidencialidade

- Não autenticação: é usado quando a segurança dos dados não é uma questão importante
- Autenticação básica: o cliente identifica-se perante o servidor através do envio, através da rede, de um DN e da password correspondente, em claro (ou utilizando cifras fracas)
- *Simple Authentication and Security Layer* (SASL): trata-se de um pacote que permite a implementação de mecanismos adicionais de autenticação a protocolos orientados à conexão (definido no RFC 2222 - *Simple Authentication and Security Layer*)

?? A escolha do mecanismo de segurança a ser usado é negociada quando é estabelecida a conexão entre o cliente e o servidor

1.3.2.5. Esquema de URL's do LDAP

?? Dada a importância crescente do LDAP na Internet, foi definido um formato para acesso aos recursos, baseado num URL (RFC 2255 – *The LDAP URL Format*)

?? Alguns exemplos:

- ldap://saturn.itso.austin.ibm.com/
- ldap://saturn.itso.austin.ibm.com:389/o=Transarc,c=US
- ldap://saturn.itso.austin.ibm.com/cn=John%20Smith,ou=Austin,o=IBM,c=US

1.3.3. A WWW (World Wide Web)

- A *World Wide Web* é um **sistema global de hipertexto**
- Foi desenvolvido inicialmente em 1989, por **Tim Berners Lee**, no Centro Europeu de Física das Partículas (CERN)
- Tinha como objectivo inicial facilitar a edição e partilha de documentos de investigação, entre grupos geograficamente dispersos de cientistas
- Em 1993, a WWW começou verdadeiramente o seu crescimento exponencial, com o desenvolvimento, no *National Center for Supercomputing Applications* (NCSA), do primeiro **navegador Web** com interface gráfico, denominado **Mosaic**

1.3.3.1. Servidores e Navegadores WWW

- Um **Navegador** (*Browser*) WWW é genericamente uma aplicação que permite o acesso a informação hipertexto, disponibilizada por um **Servidor WWW**
- No mínimo, é constituído por um **interpretador de HTML** (*hipertext Markup Language*) e por um **cliente HTTP** (*HiperText Transfer Protocol*) usado para aceder as páginas hipertexto em HTML
- Para além destes requisitos mínimos, muitos navegadores suportam ainda o acesso a outros serviços da Internet, como FTP, NNTP, E-Mail, em interfaces gráficos caracterizados pela facilidade de utilização
- Tal como outros serviços Internet, a WWW funciona com base no **modelo Cliente/Servidor**, onde o Navegador WWW efectua as **operações do cliente**
- Existem vários navegadores disponíveis, a maior parte deles de utilização e distribuição gratuita: *Netscape Navigator, Microsoft Internet Explorer, Mosaic, Lusitano*, etc

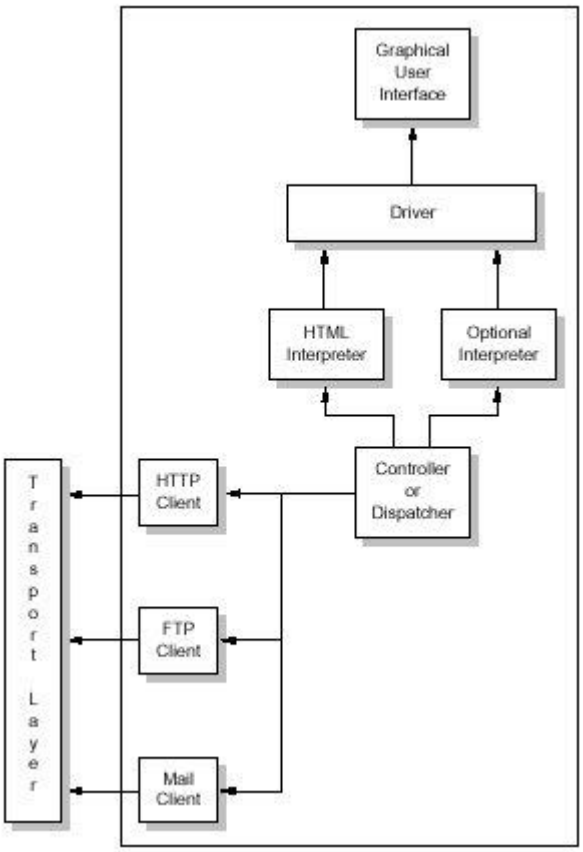


Figura 1.19 – Estrutura de um Navegador WWW

?? Os navegadores WWW são responsáveis por formatar e apresentar a informação, interagir com os utilizadores e executar funções externas (p.e. visualizadores externos de tipos de dados que os navegadores não suportam)

?? Os **Servidores WWW** (*Web Servers*) são responsáveis por **disponibilizar informação hipertexto aos navegadores WWW**

?? A **informação** pode ter **origem num ficheiro de hipertexto** (páginas HTML) armazenado no disco local do servidor, ou pode ser **gerada por um programa executado pelo servidor** para realizar determinada função (informação dinâmica)

?? Existe um conjunto muito alargado de **Servidores WWW**, para as mais variadas plataformas operativas (incluindo alguns *public domain*):

- *Apache* (http://www.apache.org): o mais utilizado a nível mundial; é *public domain*, funcionando em vários

sistemas operativos da família UNIX e em sistemas operativos da família Windows, da Microsoft

- Microsoft *Internet Information Server*: servidor comercial ; só funciona sobre os sistemas operativos servidor da Microsoft (NT e 2000)

- *Domino Weserver*: servidor da IBM, integrado com a ferramenta Lotus Notes

?? Existe actualmente um conjunto alargado de tecnologias que permitem a criação de **conteúdos dinâmicos**:

- *Common Gateway Interface* (CGI): é uma forma de permitir que um Servidor WWW execute um programa indicado pelo administrador. Os CGI's permitem ao servidor gerar respostas dinâmicas, normalmente de acordo com o *input* dos clientes

- *API's Server-Specific*: Alguns servidores fornecem API's específicas, que disponibilizam ferramentas de desenvolvimento de conteúdos interactivos aos programadores. Normalmente são ferramentas proprietárias, o que impede a portabilidade entre servidores

- *Servlets*: Tecnologia baseada em Java, que permite a invocação de programas desenvolvidos nesta linguagem, na memória dos servidores

- *Java Server Pages* (JSP): Constitui uma forma fácil de gerar páginas HTML, com conteúdo dinâmico. Um ficheiro JSP contém combinações de *tags* HTML, *tags* <SERVLET> e sintaxe JSP

1.3.3.2. O Protocolo HTTP (HyperText Transfer Protocol)

?? O **HTTP 1.1** é um protocolo *proposed standard*, descrito no RFC 2068

- A versão anterior (HTTP 1.0) é um protocolo *informational*, descrito no RFC 1945

- O **protocolo HTTP** tem a função de efectuar a transferência de documentos HTML (e outros ficheiros associados a estes) entre um Servidor e um Cliente WWW

- Uma **transação HTTP** é dividida em quatro passos:

    1. o navegador abre uma conexão
    2. de seguida envia um pedido ao servidor
    3. o servidor envia a resposta ao navegador
    4. a conexão é terminada pelo servidor

- Na Internet, o HTTP normalmente **funciona sobre conexões TCP**

- Por defeito é utilizado o **porto 80**, apesar de outros poderem ser usados

- O HTTP é um protocolo *stateless*, já que as conexões são independentes umas das outras

- P.e. quando um navegador carrega uma página que tem duas imagens, são abertas três conexões independentes: uma para a página propriamente dita e uma para cada imagem

- A maior parte dos navegadores têm capacidade para abrir várias conexões simultaneamente

- Este comportamento pode consumir muitos recursos, se uma página for constituída por muitos elementos independentes

- O HTTP 1.1 alivia esta sobrecarga permitindo que seja estabelecida apenas **uma conexão TCP por tipo de elemento** da página a carregar (p.e. uma única conexão para todos os ficheiros do tipo *jpeg*, outra para os ficheiros *gif*, etc)

- Se há a necessidade de um pedido depender de informação trocada previamente, torna-se necessário utilizar **cookies**

- Um **cookie** é uma parcela de informação que é trocada entre o servidor e o cliente, durante uma **transacção HTTP**; Pode ter um **tamanho máximo de 4 KB**.

- Essa informação (cookie) é armazenada num único ficheiro na directoria do navegador, podendo ser acedida pelo servidor em conexões subsequentes

- Dado que os **cookies** se podem traduzir em potenciais falhas de segurança, os navegadores permitem que os utilizadores definam a possibilidade ou não de utilização deste mecanismo

- **Parâmetros do protocolo**

    - *HTTP Version*: HTTP-Version = “HTTP” “/” 1*DIGIT “.” 1*DIGIT

    - *Uniform Resource Identifiers (URI):* são constituídos por combinações de *Uniform Resource Locators* (URL) e *Uniform Resource Names* (URN); na prática são strings que indicam a localização e o nome da origem, num servidor

        - *HTTP URL*: permite a localização de recursos de rede através do protocolo HTTP

            - HTTP_URL = “http” “//” host [“:” port] [abs_path] onde o porto é opcional (se não especificado assume-se o valor 80)

1.3.3.2.1. Mensagens HTTP

- As mensagens HTTP são constituídas pelos seguintes campos:

- **Tipos da mensagem:**

    - HTTP-message = Request | Response

- **Cabeçalho da Mensagem:** pode ser um dos seguintes:

    - *General Header*

    - *Request Header*

    - *Response Header*

- *Entity Header*

- **Corpo da mensagem**

- **Tamanho da mensagem**

1.3.3.2.1. Métodos do HTTP

- *OPTIONS*: permite ao cliente determinar opções ou requisitos associados a uma fonte ou um servidor

- *GET*: este método permite ao cliente obter os dados determinados pelo URI do pedido

- *HEAD*: permite ao cliente a obtenção de meta-informação sem necessidade de efectuar a transferência completa de um recurso

- *POST*: esta função é determinada pelo servidor

- *PUT*: similar ao *POST*

- *DELETE*: solicita ao servidor que elimine a fonte definida no URI do pedido

- *TRACE*: permite ao cliente verificar como a mensagem é obtida do outro lado, para testes e diagnósticos

1.3.3.2.3. Autenticação de acesso a recursos HTTP

- O HTTP disponibiliza um **mecanismo de autenticação** que permite aos servidores **definirem as permissões de acesso a recursos** e que clientes o podem fazer

- Existem os seguintes métodos de autenticação:

  - *Basic Authentication Scheme*: é baseado em *user ID's* e *passwords*, onde o servidor permite a conexão se estes dois parâmetros são validados. Neste tipo de autenticação o *user ID* e a *password* não são encriptados (utilizam codificação "*Base64 Encoding*")

- *Digest Authentication Scheme*: é uma extensão ao HTTP, descrita no RFC 2069. Este esquema cifra a *password* de autenticação e envia uma função da mesma para o servidor, oferecendo elevado nível de confidencialidade

1.3.3.2.4. HTTP Caching

- A possibilidade de implementação de **mecanismos de caching** no HTTP, aumenta de forma significativa a performance, na distribuição dos documentos hipertexto
- O HTTP 1.1. disponibiliza um conjunto de **funções** que permitem a **utilização eficiente de mecanismos de cache**
- Genericamente este mecanismo permite o armazenamento de pedidos de clientes e respostas de servidores, em dispositivos de armazenamento temporários, durante um período limitado de tempo
- Assim, se uma resposta se encontra na cache (do host que formula um pedido ou de um servidor intermédio mais próximo), e se encontra dentro do tempo de validade, **não há necessidade de contactar o servidor de origem**
- A utilização de mecanismos de cache permite não só reduzir a utilização de largura de banda da rede, mas também reduzir o tempo de resposta
- O servidor de cache estima um **tempo mínimo** em que uma **mensagem de resposta é válida** (um tempo de expiração)
- Durante esse tempo, a mensagem pode ser usada sem haver necessidade de consultar o servidor original
- Para verificação da alteração ou não dos dados de uma mensagem, após a expiração do tempo, o HTTP 1.1 define um **mecanismo de validação**, que assenta no seguinte:
  - *Expiration Mechanism*: permite a definição do tempo de expiração de uma mensagem. Este tempo pode ser definido no Servidor de origem. Se tal não acontecer, pode-se estimar/calcular este tempo, recorrendo a

várias formas (p.e. tendo em atenção ao tempo da última modificação)

- *Validation Mechanism*: Se o tempo da mensagem expirou, este mecanismo permite verificar, junto da origem ou de outros servidores de cache intermédios, se os dados já estão desactualizados

1.3.3.3. A linguagem HTML (HyperText Markup Language)

- A linguagem HTML é um dos componentes base e mais importantes da Web
- É constituída por um conjunto de *tags* que têm de ser entendidas, quer pelos **Navegadores**, quer pelos **Servidores WWW**
- Estas *tags* são **independentes** dos dispositivos e dos sistemas operativos que as vão interpretar
- Descrevem **elementos básicos** de um documento WWW, como sejam **cabeçalhos**, **parágrafos**, **estilos de texto**, **listas**, etc
- Existem ainda *tags* mais sofisticadas, que permitem a criação de **tabelas** e a inclusão de documentos interactivos, como formulários, *scrits*, *applets Java*, etc
- Dado que o HTML suporta hipertexto, permite a inclusão, em documentos deste tipo, de ligações para outros documentos HTML
- Estes documentos podem estar na mesma máquina que o original, ou em qualquer outra máquina na mesma ou em outra rede: **ligações HTML**

1.3.3.4. A linguagem XML (Extensible Markup Language)

- A **linguagem XML** descreve **classes de objectos** de dados, denominados documentos XML, que são armazenados em computadores
- Descreve ainda parcialmente o **comportamento dos programas** que processam estes objectos
- Encontra-se ainda num estágio inicial de implementação, esperando-se que venha a introduzir outra dimensão à interactividade e desenvolvimento de aplicações comerciais para a Web

1.3.4. O protocolo FTP (File Transfer Protocol)

- O FTP (*File Transfer Protocol*) é um protocolo standard, com o STD número 9. Está descrito no RFC 959 – *File Transfer Protocol (FTP)* e actualizado no RFC 2228 – *FTP Security Extentions*
- Permite a **transferência de ficheiros** entre hosts, em redes TCP/IP
- Funciona em ambiente cliente/servidor, podendo a **transferência** dos dados ser feita em **ambos os sentidos** (do servidor para o cliente e vice-versa)
- O FTP utiliza o **protocolo TCP** ao nível da camada de transporte, por forma a obter **conexões fiáveis** ponto-a-ponto
- O servidor mantém-se disponível para conexões nos **portos 20 e 21**
- O **porto 20** é utilizado para a transferência dos dados, baseando-se no protocolo Telnet, e o **porto 21** para o estabelecimento e controlo da ligação
- Para um cliente estabelecer uma sessão com um servidor é necessário proceder a uma **identificação e autenticação**, através de um *username* e de uma *password*

?? De ambos os lados da ligação, a aplicação FTP é constituída por um *Protocol Interpreter* (**PI**), um *Data Transfer Process* (**DTP**) e uma interface de utilizador

?? A interface do utilizador comunica com o *Protocol Interpreter*, que controla a conexão

?? Este troca informação com o file system do sistema local

?? No lado oposto, o *Protocol Interpreter* tem como função **responder ao protocolo Telnet**

?? Durante a transferência dos dados, **a gestão é feita pelos DTP's**

?? Depois do pedido do utilizador estar realizado, os *Protocol Interpreter's* terminam a conexão de controlo

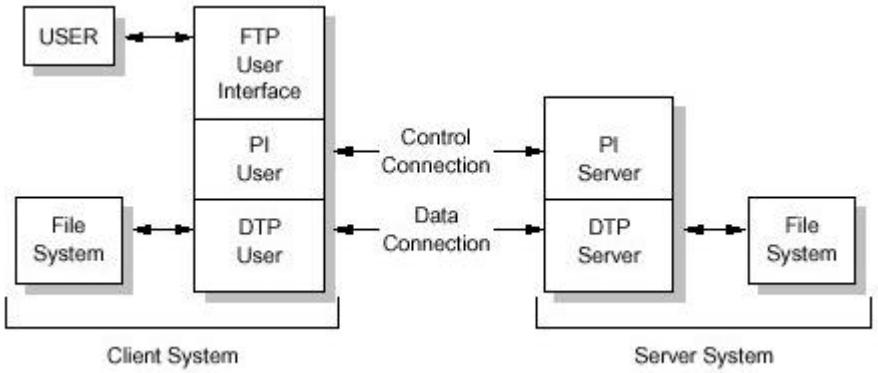


Figura 1.19 – Princípio de funcionamento do FTP

1.3.4.1. Operações FTP

?? Na utilização do FTP, o utilizador pode realizar as seguintes **operações**:

- Estabelecer a ligação ao host remoto. São usados os comandos:
  - *Open*: selecciona o host remoto e inicia a sessão de login
  - *User*: identifica o utilizador remoto, através de um ID
  - *Pass*: autentica o utilizador
- Seleccionar uma directoria: após o estabelecimento da ligação, o utilizador pode utilizar o comando cd (*change* directory) para seleccionar a directoria remota em que pretende trabalhar. Obviamente, o utilizador só poderá aceder a directorias para as quais tem permissão. Pode ainda mudar de directoria local, com o comando lcd (*local change directory*)
- Listar os ficheiros disponíveis para transferência: possível com os comandos ls ou dir
- Definir o modo de transferência: a transferência de dados entre sistemas diferentes requer transformações nesses mesmos dados. Assim, o utilizador pode decidir o **Modo** como os dados são transferidos e o **Tipo** de caracteres usados na transferência:
  - *Mode*: especifica como o ficheiro vai ser tratado: ***Block*** ou ***Stream***
  - *Type:* **ASCII**, **EBCDIC** ou **Image**
- Copiar os ficheiros do/(para o) host remoto: com os seguintes comandos:
  - *Get*: copia um ficheiro do host remoto para o host local
  - *Mget*: copia vários ficheiros do host remoto para o host local

- ✍*Put*: copia um ficheiro do host local para o host remoto

- ✍*Mput*: copia múltiplos ficheiros do host local para o host remoto

o Terminar a ligação com o host remoto: com os comandos: **Quit** (termina a conexão ao host remoto e termina a utilização do cliente FTP); **Close** (termina a conexão com o host remoto mas mantém o cliente FTP a funcionar)

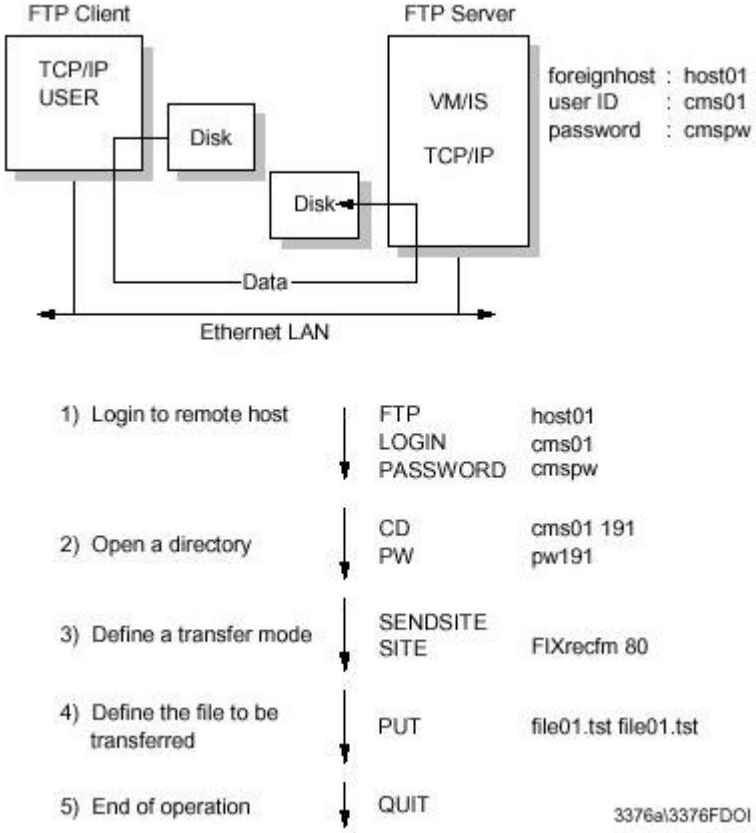


Figura 1.20 – Cenário FTP

1.3.4.2. Anonymous FTP

?? O **FTP anónimo** (*anonymous FTP*) **permite o acesso público a servidores de ficheiros**, de forma simplificada

?? O utilizador remoto necessita apenas de se identificar com o login *anonymous* e a password *guest* (também é comum utilizar o endereço de e-mail como password)

```
[C:\SAMPLES]ftp host01.itsc.raleigh.ibm.com
Connected to host01.itsc.raleigh.ibm.com.
220 host01 FTP server (Version 4.1 Sat Nov 23 12:52:09 CST 1991) ready.
Name (rs60002): cms01
331 Password required for cms01.
Password: xxxxxx
230 User cms01 logged in.
ftp> put file01.tst file01.tst
200 PORT command successful.
150 Opening data connection for file01.tst (1252 bytes).
226 Transfer complete.
local: file01.tst remote: file01.tst
1285 bytes received in 0.062 seconds (20 Kbytes/s)
ftp> close
221 Goodbye.
ftp> quit
```

Figura 1.21 – Exemplo de utilização do FTP

1.3.5. O Serviço de Correio Electrónico

- O Correio Electrónico (*E-Mail*) é um dos serviços mais utilizados da Internet

- Permite, através de um conjunto de protocolos, **efectuar a troca de mensagens entre hosts** e, em último lugar, **entre utilizadores**

- O principal destes protocolos designa-se **SMTP** – *Simple Mail Transfer Protocol*, que tem a responsabilidade de efectuar a troca de mensagens entre Servidores de E-Mail

- Adicionalmente, os protocolos **POP-3** – *Post Office Protocol, version 3* e **IMAP** – *Internet Message Access Protocol* complementam o primeiro, na tarefa de disponibilizar este serviço aos utilizadores

- A transferência de mensagens de correio electrónico, em redes TCP/IP está descrita em três *Standards*:

  - Um standard para a troca de mensagens entre dois computadores (STD 10/RFC 821), que especifica o protocolo usado para enviar mail entre hosts TCP/IP: **Protocolo SMTP**

  - Um standard (STD 11), que define o formato das mensagens de mail:

    - O **RFC 822** descreve a sintaxe dos campos do cabeçalho das mensagens, além de definir o conjunto de campos do cabeçalho e a sua interpretação

    - O **RFC 1049** descreve como um conjunto de tipos de documentos não-ASCII podem ser transmitidos no corpo das mensagens de mail (nomeadamente PostScript, Scribe, SGML, TEX, TROFF e DVI)

    - O nome oficial deste protocolo standard é **MAIL**

  - Um standard para o encaminhamento do mail utilizando o DNS – *Domain Name System*, descrito no RFC 974. O nome oficial deste standard é **DNS-MX**

1.3.5.1. O protocolo SMTP (Simple Mail Transfer Protocol)

- O STD 10/RFC 821 **define a transmissão de dados por SMTP**, utilizando a representação de caracteres ASCII, de 7 bits

- Trata-se de uma representação suficiente para texto em Inglês, mas inadequada para a maioria de outras línguas que não a inglesa

- Existem duas **alternativas** para resolver este problema:

  - Utilizando **extensões MIME** – *Multipurpose Internet Mail Extensions* (ver próximo sub-capítulo)

  - Utilizando *SMTP Service Extensions*, descritas em três RFC's:

    - RFC 1869, que define os mecanismos que permitem a um cliente SMTP solicitar a lista das extensões suportadas pelo Servidor, e, em função da resposta, definir o tipo de caracteres disponibilizados para o envio das mensagens

    - RFC 1652, que define um protocolo para transmissão de mensagens utilizando representação de caracteres de 8 bits

    - RFC 1870, que define um protocolo para o servidor poder informar o cliente sobre o tamanho máximo das mensagens que aceita

- As **mensagens SMTP** são constituídas por:

  - Um **Cabeçalho** (ou envelope), cuja estrutura é definida no RFC 822. Este cabeçalho é terminado por uma linha nula (uma linha vazia antes da sequência <CRLF>)

  - **Conteúdo**: todo o conteúdo antes de uma linha em branco, no final da mensagem, pertence ao corpo desta (constituído por sequências de linhas contendo caracteres ASCII)

- O RFC 821 define um protocolo do tipo Cliente/Servidor
- O **cliente SMTP é** aquele que inicia a sessão (o que envia) e o **servidor SMTP** é o que responde ao pedido de estabelecimento de uma sessão (o que recebe a mensagem)

1.3.5.1.1. Formato do Cabeçalho de uma Mensagem

- A **sintaxe do cabeçalho** das mensagens de correio electrónico (*Mail* Header) encontra-se descrita no RFC 822
- Esta **sintaxe** é escrita de acordo com uma forma conhecida por *Augmented Backus-Naur Form* – **BNF** (Forma Backus-Naur aumentada)
- Resumidamente, o cabeçalho é constituído por uma lista de linhas, na seguinte forma:
  - **`field-name: field-value`**
- Alguns dos **campos** mais importantes **do cabeçalho** (por exemplo os campos **To** ou **From**) são *mailboxes*, que podem tomar várias formas:
  - `nuno@ipb.pt`
  - `Nuno Rodrigues <nuno@ipb.pt>`
  - `“Nuno Rodrigues” <nuno@ipb.pt>`
- Alguns dos **campos mais usados**:
  - *to*: recipiente primário de uma mensagem
  - *cc*: recipiente secundário (*carbon-copy*) de uma mensagem
  - *from*: identifica o emissor
  - *reply-to*: identifica a *mailbox* para onde as respostas devem ser encaminhadas

- o *return-path*: endereço e rota de volta ao originador. Este campo é adicionado pelo sistema final de transporte que envia o e-mail
- o *Subject*: sumário da mensagem, normalmente fornecido pelo utilizador

1.3.5.1.2. Troca de Mensagens de E-Mail

?? O funcionamento do SMTP é baseado no modelo de comunicação apresentado na figura seguinte:

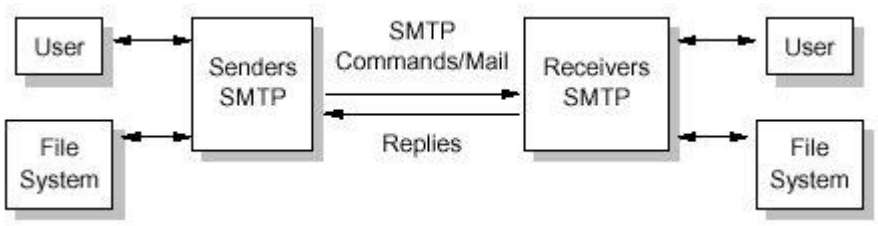


Figura 1.22 – Modelo de funcionamento do SMTP

?? Como resultado de um pedido de um utilizador, o **SMTP emissor** estabelece uma conexão bidireccional com o **SMTP receptor**

?? Este pode ser o último destinatário ou um intermediário (*mail gateway*)

?? O SMTP emissor gera comandos que são respondidos pelo SMTP receptor

?? **Fluxos de uma transacção SMTP** (ver figura seguinte): os comandos/respostas/dados trocados durante a transmissão de uma mensagem são constituídos por linhas de texto, delimitados por um <CRLF>

?? Todas as **respostas** têm um código numérico no início da linha

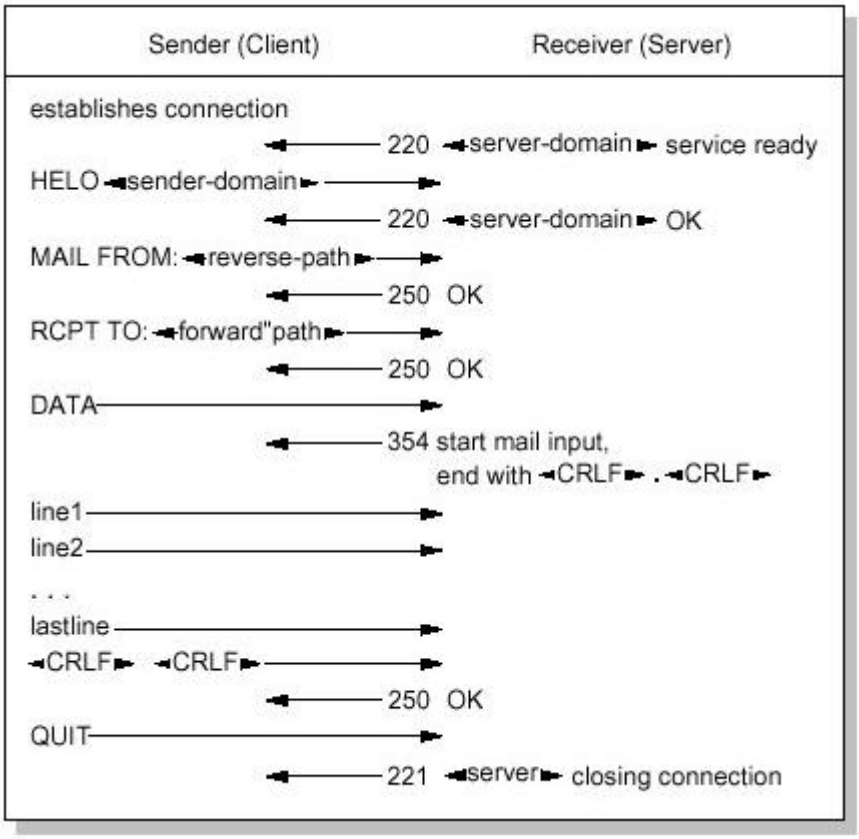


Figura 1.23 – Fluxos de Dados do SMTP

?? **Endereço SMTP de Destino**: toma a forma geral parte-local@nome-de-domínio, podendo tomar várias formas:

- *user@host*: para um destinatário directo, na mesma rede TCP/IP
- *user%remote-host@gateway-host*: para um destinatário numa rede não SMTP, via gateway de mail *gateway-host*
- *@host-a, @host-b:user@host-c*: para uma mensagem reencaminhada. Este endereço contém informação explícita de encaminhamento. A mensagem é enviada inicialmente para o *host-a*, que a vai reenviar para o *host-b*. Este vai finalmente encaminhá-la para o destino final – *host-c*

```
R: 220 delta.aus.edu Simple Mail Transfer Service Ready
S: HELO stockholm.ibm.com
R: 250 delta.aus.edu

S: MAIL FROM:<abc@stockholm.ibm.com>
R: 250 OK

S: RCPT TO:<xyz@delta.aus.edu>
R: 250 OK
S: RCPT TO:<opq@delta.aus.edu>
R: 550 No such user here
S: RCPT TO:<rst@delta.aus.edu>
R: 250 OK

S: DATA
R: 354 Start mail input, end with <CRLF>.<CRLF>
S: Date: 23 Jan 89  18:05:23
S: From: Alex B. Carver <abc@stockholm.ibm.com>
S: Subject: Important meeting
S: To:  <xyz@delta.aus.edu>
S: To:  <opq@delta.aus.edu>
S: cc:  <rst@delta.aus.edu>
S:
S: Blah blah blah
S: etc.....
S: .
R: 250 OK

S: QUIT
R: 221 delta.aus.edu Service closing transmission channel
```

Figura 1.24 – Exemplo de Operação SMTP

1.3.5.1.3. Interacção do SMTP com o DNS

?? Se determinada rede utiliza os recursos do DNS, um Servidor SMTP pode enviar mensagens de e-mail para utilizadores de um domínio sem ser necessário referenciar, no endereço, o nome do servidor receptor

?? Para que isso seja possível, os servidores de nomes DNS utilizam os *resource-records* MX (*mail exchange*)

?? Estes efectuam o mapeamento de um nome de domínio para dois valores:

- **Um valor de preferência**. No caso de existirem múltiplos *resource-records* MX para o mesmo domínio, este valor define a preferência do servidor de entrega das mensagens. Assim, se o servidor principal de SMTP não está disponível, os emissores vão

procurando efectuar a entrega seguindo esta lista de preferência

- **O nome de um host** (Servidor de SMTP)

?? Exemplo dos *resource-records* MX do domínio ipb.pt:

```
...
ipb.pt.    IN      MX    10      mail.ipb.pt.
           IN      MX    15      pan.ipb.pt.
...
```

1.3.5.2. Multiple Internet Mail Extensions (MIME)

- Como referido na secção anterior, o **SMTP** (STD 10/RFC 821) está limitado ao transporte de texto ASCII de 7 bits, com um máximo de 1000 caracteres por linha

- Estas características traduzem-se num conjunto de **limitações para o SMTP**:

  - Não pode transmitir ficheiros executáveis, ou outros binários. Existem no entanto alguns métodos que permitem o encapsulamento de ficheiros binários em mensagens SMTP:

    - Codificando o ficheiro em hexadecimal

    - Os utilitários do UNIX UUencode e UUdecode servem para codificar dados binários no sistema de e-mail UUCP, que ultrapassa algumas da limitações de um transporte a 7 bits

    - A representação *Andrew Toolkit*

  - Não pode transmitir texto que inclua caracteres de linguagens que contenham uma codificação ASCII superior a 128

  - Os servidores SMTP podem rejeitar mensagens a partir de determinado tamanho (este limite pode ser configurado nos servidores)

  - Gateways SMTP que traduzem mensagens de ASCII para EBCDIC e vice-versa não usam mapeamentos de códigos consistentes, que resultam em problemas de tradução

- O MIME é um standard que **inclui mecanismos para resolver estes problemas**, com grande compatibilidade com os standards definidos no RFC 822

- De uma forma genérica, uma mensagem MIME pode ser encaminhada através de qualquer número de redes que sejam compatíveis com o RFC 821 e que tenham de transmitir mensagens que estejam de acordo com esse RFC

- O MIME encontra-se **descrito em cinco partes**:

- **Protocolos para inclusão de objectos não ASCII**, no corpo de mensagens RFC 822 (descritos no RFC 2045)
- **A estrutura geral dos tipos MIME** e um conjunto inicial de tipos, (RFC 2046)
- **Um protocolo para codificação de texto não ASCII em campos do cabeçalho de mensagens** compatíveis com o RFC 822 (RFC 2047)
- **Critérios de conformidade MIME** (RFC 2049)

?? O standard MIME foi desenhado de acordo com a seguinte **ordem de prioridades**:

1. Compatibilidade com os standards existentes, nomeadamente o RFC 822
2. Robustez no suporte de implementações de MTA's (*Mail Transfer Agents*) existentes, que não sejam completamente compatíveis com os standards
3. Facilidade de extensão, através da definição de novos objectos suportados

??O MIME é um protocolo de alto nível, que funciona completamente no interior das fronteiras dos STD 10 e STD 11, não interagindo directamente com a camada de transporte

?? Uma mensagem MIME-compatível contém um campo no cabeçalho com os seguintes **parâmetros**:

- **`MIME-Version: 1.0`**

??A **sintaxe geral dos campos** de uma cabeçalho MIME é a mesma definida no RFC 822, sendo definidos cinco campos:

- *MIME-Version*: define a versão, tomando actualmente o valor 1.0
- *Content-Type*: descreve a forma como os objectos no corpo da mensagem devem ser interpretados. O valor por defeito é *text/plain; charset=us-asci*. Toma a seguinte forma:

✍ `Content-Type: type/subtype ;parameter=value`

Existem sete tipos definidos:

✍*text*: contém apenas o sub-tipo *plain* definido; trata-se de texto ASCII ou iso-8859-x não-formatado

✍*multipart*: mensagens deste tipo contém múltiplos objectos de tipos de dados independentes; neste caso o corpo da mensagem é dividido em partes, através de linhas chamadas *encapsulation boundaries*

✍*message*: o corpo destes e-mails contém mensagens encapsuladas, ou partes de uma; existem três sub-tipos definidos: *rfc822*, *partial* (utilizado para permitir fragmentação de mensagens grandes) e *external-body* (contém um apontador para um objecto que existe algures noutro lado

✍*image*: o corpo da mensagem contém imagens, utilizando um dos dois sub-tipos definidos: *jpeg* e *gif*

✍*video*: o corpo da mensagem transporta imagens em movimento, com sub-tipo *mpeg* definido

✍*audio*: o corpo da mensagem transporta dados de áudio, com o sub-tipo *basic*

✍*application*: é utilizado por tipos de dados que não se enquadram em outras categorias, particularmente para dados que precisam de ser processados por aplicações externas

```
MIME-Version: 1.0
From: Steve Hayes <steve@hayessj.bedfont.uk.ibm.com>
To:   Matthias Enders <enders@itsol80.itso.ral.ibm.com>
Subject: Multipart message
Content-type: multipart/mixed; boundary="1995021309105517"

This section is called the preamble.  It is after the header but before the
first boundary.  Mail readers which understand multipart messages must
ignore this.
--1995021309105517

The first part.  There is no header, so this is text/plain with
charset=us-ascii by default.  The immediately preceding <CRLF> is part of
the <CRLF><CRLF> sequence that ends the null header.  The one at the end is
part of the next boundary, so this part consists of five lines of text with
four <CRLF>s.
--1995021309105517
Content-type: text/plain; charset=us-ascii
Comments: this header explicitly states the defaults

One line of text this time, but it ends in a line break.

--1995021309105517
Content-Type: multipart/alternative; boundary=_
Comments:     An encapsulated multipart message!

Again, this preamble is ignored.  The multipart body contains a still image
and a video image encoded in Base64.  See4.8.3.5, "Base64 Encoding" on page 204
One feature is that the character "_" which is allowed in multipart
boundaries never occurs in Base64 encoding so we can use a very simple
boundary!
--_
Content-type: text/plain
```

Figura 1.25 – Exemplo de uma mensagem MIME complexa

```
This message contains images which cannot be displayed at your terminal.
This is a shame because they're very nice.

--_
Content-type: image/jpeg
Content-transfer-encoding: base64
Comments: This photograph is to be shown if the user's system cannot display
          MPEG videos.  Only part of the data is shown in this book because
          the reader is unlikely to be wearing MIME-compliant spectacles.

Qk1OAAAAAAAAE4EAABAAAAAQAEAAPAAAAABAAgAAAAAAAAAAAAAAAAAAAAAAAABAAAAAQAAAAAA
AAAAAAAAAAAAAAAAAAAAAAB4VjQSAAAAAAAAgAAAkgAAAJKAAKoAAACqAIAAqpIAAMHBwQDJyckA
/9uqAKpJAAD/SQAAAG0AAFVtAACqbQAA/20AAAAkAABVkgAAqiQAAP+SAAAAtgAAVbYAAKq2AAD/
<base64 data continues for another 1365 lines>
--_
Content-type: video/mpeg
Content-transfer-encoding: base64

AAABswoAeBn//+CEAAABsgAAAOgAAAG4AAAAAAAAAQAAT/////wAAAGy//8AAAEBQ/ZlIwwBGWCX
+pqMiJQDjAKywS/1NRrtXcTCLgzVQymqqHAf0sL1sMgMq4SWLCwOTYRdgyAyrhNYsLhhF3DLjAGg
BdwDXBv3yMV8/4tzrp3zsAWIGAJg1IBKTeFFI2IsgutIdfuSaAGCTsBVnWdz8afdMMAMgKgMEkPE
<base64 data continues for another 1839 lines>
--_--
That was the end of the nested multipart message.  This is the epilogue.
Like the preamble it is ignored.
--1995021309105517--
And that was the end of the main multipart message.  That's all folks!
```

Figura 1.26 – Exemplo de uma mensagem MIME complexa

- *Content-Transfer-Encoding*: descreve a forma como os objectos no corpo de mensagem são codificados:
  - 7-bit (codificação por defeito)
  - 8-bit

- Binary
- Quoted-Printable
- Base64

Base64 value	ASCII char.	Base64 value	ASCII char.	Base64 value	ASCII char.	Base64 value	ASCII char.
0	A	16	Q	32	g	48	w
1	B	17	R	33	h	49	x
2	C	18	S	34	i	50	y
3	D	19	T	35	j	51	z
4	E	20	U	36	k	52	0
5	F	21	V	37	l	53	1
6	G	22	W	38	m	54	2
7	H	23	X	39	n	55	3
8	I	24	Y	40	o	56	4
9	J	25	Z	41	p	57	5
10	K	26	a	42	q	58	6
11	L	27	b	43	r	59	7
12	M	28	c	44	s	60	8
13	N	29	d	45	t	61	9
14	O	30	e	46	u	62	+
15	P	31	f	47	v	63	/

Figura 1.27 – Alfabeto Base64

- *Content-Description*: descrição, em texto plano, do objecto contido na mensagem. Útil quando o objecto não é legível (p.e. dados de áudio)
- *Content-ID*: um valor único, que especifica o conteúdo desta parte da mensagem

1.3.5.3. Post Office Protocol (POP)

- O **Post Office Protocol, Version 3** (**POP3**) é um protocolo standard (STD 53), descrito no RFC 1939
- A versão anterior deste protocolo (Post Office Protocol, version 2 – POP2) é um *historic protocol*, descrito no RFC 937
- O **POP3** é um protocolo que **suporta funções básicas** (download e eliminação) **na obtenção de mensagens de correio electrónico a partir de um servidor**
- Os clientes POP3 estabelecem conexões TCP com o Servidor usando o **porto 110**
- Quando a conexão é estabelecida, o servidor POP3 envia uma mensagem de saudação ao cliente, entrando a **sessão** em modo *authentication state*
- De seguida, o cliente tem de enviar a identificação para o servidor
- Se este verifica o ID com sucesso, a sessão entra em modo *transaction state*
- A partir deste momento, o cliente pode aceder à *mailbox*
- Quando o cliente envia o comando **QUIT**, a sessão entra em *Update state*, e a conexão é terminada
- Na interacção com o Servidor, o Cliente envia **comandos**, aos quais o primeiro reage com **respostas**
- As respostas podem conter até 512 caracteres e começam com um indicador de estado, que traduz se a resposta é positiva (**+OK**) ou negativa (**-ERR**)

- **Comandos POP3** mais importantes:
  - **USER** name: username para autenticação

- **PASS** password: password para autenticação
- **STAT**: obtém o número e o tamanho total das mensagens
- **LIST** [msg]: se for especificado o número da mensagem, é indicado o seu tamanho; se nenhum número não for indicado, são listados os tamanhos de todas as mensagens
- **RETR** msg: envia a mensagem com o número msg para o cliente
- **DELE** msg: elimina a mensagem especificada
- **NOOP**: O Servidor não faz nada, apenas envia uma resposta positiva
- **RSET**: este comando cancela pedidos anteriores de eliminação de mensagens, se eles existirem (dentro de uma mesma sessão)
- **QUIT**: se enviado em modo *authorization state*, termina apenas a conexão TCP; se enviado em *transaction state*, actualiza a mailbox (elimina as mensagens de pedidos anteriores) e por fim termina a conexão TCP

1.3.5.4. Internet Message Access Protocol version 4 (IMAP4)

- O **Internet Message Access Protocol, Version 4** (**IMAP4**) encontra-se no estado *proposed standard protocol*, com o estado *elective* (descrito no RFC 2060)
- O **IMAP4** é um **protocolo de manuseamento de caixas de correio electrónico**, similar ao POP3
- Os servidores IMAP4 armazenam mensagens de múltiplos utilizadores, que **podem ser acedidas por pedidos de clientes**
- Comparativamente com o POP3, os clientes IMAP4 apresentam mais capacidades de interacção com os servidores

- O IMAP4 permite que os clientes acedam e controlem múltiplas mailboxes ao mesmo tempo

- Os clientes IMAP4 podem especificar **critérios** para o **carregamento de mensagens** (p.e. tamanho máximo, etc)

- O IMAP4 mantém as mensagens no Servidor, **replicando-as** para os clientes e **sincronizando** as mailboxes entre os clientes e o servidor

- O protocolo IMAP4 suporta **três modelos** de gestão das mailboxes:

    - **Offline**: o cliente liga-se periodicamente ao servidor e copia as mensagens, sendo de seguida eliminadas do servidor (modelo utilizado pelo POP3)

    - **Online**: o cliente efectua alterações no servidor, ou seja, o e-mail é processado remotamente neste

    - **Disconnected**: é uma mistura dos dois modelos anteriores. O cliente copia as mensagens e efectua alterações localmente; mais tarde, faz o upload destas alterações para o servidor

- Os servidores IMAP4 normalmente respondem aos comandos dos clientes no **porto 143**

- Após o período de estabelecimento da conexão TCP, ambos trocam dados interactivamente, com o cliente a enviar **comandos** e o servidor a enviar **respostas**

- Da mesma forma que no POP3, uma sessão IMAP4 passa por diferentes **estados**:

    - **Non-Authenticated**: o cliente envia identificação para o servidor

    - **Authenticated**: o cliente tem de seleccionar a mailbox que vai utilizar

    - **Selected**: a mailbox foi seleccionada com sucesso

    - **Logout**: a conexão é terminada, após um pedido do cliente, ou por qualquer outro motivo

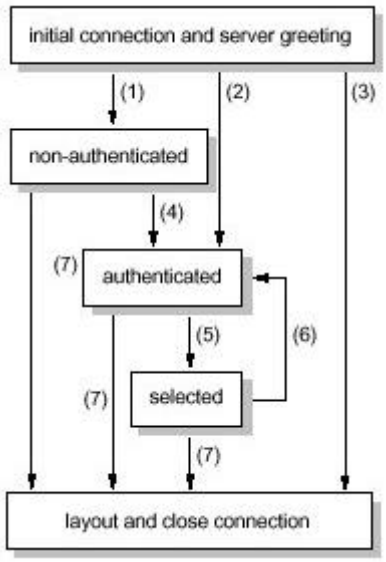


Figura 1.28 – diagrama de fluxos do IMAP4

(1)	Connection without pre-authentication (OK greeting)
(2)	Pre-authenticated connection (PREAUTH greeting)
(3)	Rejected connection (BYE greeting)
(4)	Successful LOGIN or AUTHENTICATE command
(5)	Successful SELECT or EXAMINE command
(6)	CLOSE command, or failed SELECT or EXAMINE command
(7)	LOGOUT command, server shutdown, or connection closed

?? A lista seguinte mostra os **principais comandos** disponíveis para os clientes IMAP4, em conjunto com os estados onde estão disponíveis:

In Any State

The following commands are valid for this state:

CAPABILITY This command sends a request a list of functions that the server supports.

NOOP This command does nothing. It can be used to reset the inactivity autologout timer on the server.

LOGOUT This command sends a request to end the connection.

In Non-Authenticated State

All commands in any state and the following commands are valid for this state:

AUTHENTICATE This command requests a special authentication mechanism with an argument from the server. If the server does not support that mechanism, the server sends an error message.

LOGIN This command sends the username and password in plain text.

In Authenticated State

All commands in Any State and the following commands are valid for this state:

SELECT This command selects a mailbox.

EXAMINE This command also selects a mailbox but access to the mailbox with this command is read-only.

CREATE This command creates a mailbox with a given name. It is not allowed to create INBOX or any other existing mailbox.

DELETE This command permanently removes the mailbox with the given name.

RENAME This command changes the name of the mailbox.

SUBSCRIBE This command adds the specified mailbox to the subscription list which can be obtained by the LSUB command.

UNSUBSCRIBE This command removes the specified mailbox name from the subscription list.

LIST This command requests a subset of names from the complete set of all names available from the server.

LSUB This command requests a subset of names from the subscription list.

STATUS This command requests the status of the given mailbox name. The server checks the flags and send the status according to the status of the flags.

APPEND This command appends a message text to the given mailbox as a new message.

In Selected State

All commands in any state, all commands in authenticated state and the following commands are valid for this state:

CHECK This command requests resolution of the state of the selected mailbox in the memory and the disk.

CLOSE This command permanently removes all messages from the currently selected mailbox that were previously marked as deleted and returns to authenticated state from selected state.

EXPUNGE This command permanently removes all messages from the currently selected mailbox that were previously marked as deleted.

SEARCH This command searches the mailbox for the messages that match given searching criteria.

FETCH This command retrieves data associated with a message in the selected mailbox.

STORE This command updates the message with the data which was retrieved by a FETCH command.

COPY This command copies the specified message to the end of the specified destination mailbox.

UID This command returns unique identifier instead of message sequence numbers. This command is used with other commands.

1.3.6. NetBIOS over TCP/IP

1.3.6.1. Enquadramento

- Durante muito tempo, o serviço **NetBIOS (Network Basic Input Output System)** foi um dos principais mecanismos de interligação de computadores pessoais em ambientes de rede local

- O NetBIOS **não é um protocolo**, mas sim **uma API de software independente** dos fabricantes

- Não existe uma especificação oficial para o mesmo, no entanto, na prática, a descrição efectuada na publicação da IBM *SC30-3587 LAN Technical Reference: 802.2 and NetBIOS APIs* é **usada como referência**

- Este serviço apresenta algumas **características específicas**, que limitam a sua utilização em alguns ambientes de rede de área alargada:

    - As Estações NetBIOS identificam-se através de **nomes NetBIOS**; O seu registo e procura é efectuado recorrendo à técnica de *broadcasting*

    - O NetBIOS utiliza um **espaço de nomes curto**, sem qualquer relação com o DNS – *Domain Name System*

    - Geralmente é utilizado em interfaces 802.2, que não são sujeitos a decisões de encaminhamento

- Uma solução para ultrapassar estas limitações encontra-se no **RFC 1001** – *Protocol Standard for a NetBIOS Service on a TCP/UDP Transport: Concepts and Methods* e no RFC 1002 – *Protocol Standard for a NetBIOS Service on a TCP/UDP Transport: Detailed Specifications*

- Estes RFC não definem uma técnica de encapsulamento do NetBIOS sobre TCP ou UDP, mas sim o **mapeamento de serviços NetBIOS para UDP/IP e TCP/IP**

- Por exemplo, quando uma sessão NetBIOS é estabelecida, os **sockets** de comunicação são usados sobre conexões TCP para **enviar dados de sessões NetBIOS**

?? Do ponto de vista das aplicações de rede escritas para o interface NetBIOS, não há alterações

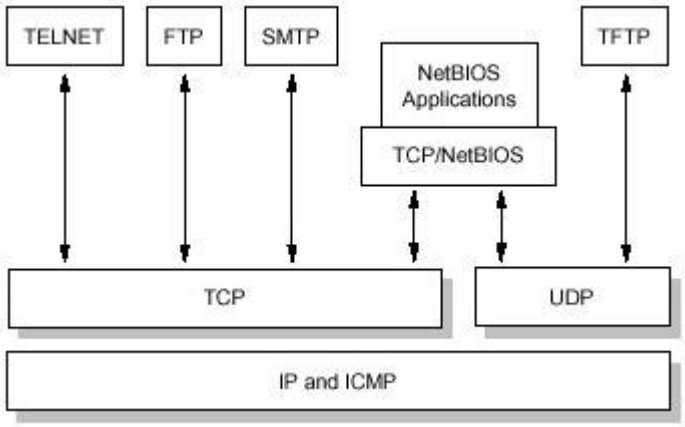


Figura 1.29 – TCP/IP e NetBIOS

?? O RFC 1001 introduz um conjunto de **novos conceitos**, para facilitar a operação do NetBIOS em ambiente TCP/IP:

- **NetBIOS Scope**: é um grupo de computadores, com um nome NetBIOS conhecido.

  Numa configuração NetBIOS standard, todos os computadores num grupo têm capacidade de se contactar, através de técnicas de *broadcasting*

  O NetBIOS sobre TCP/IP tem de fornecer um método alternativo para que cada grupo possa comunicar com outros, particularmente quando estes se situam em sub-redes diferentes

  Uma Internetwork pode suportar múltiplos *NetBIOS Scopes*, onde cada um tem um *scope identifier*, constituído por uma string de caracteres, num formato compatível com o DNS

- **NetBIOS End Node**: computadores que usam NetBIOS sobre TCP/IP são conhecidos por *NetBIOS End Nodes*. São definidos **três tipos**:

  - *Broadcast (B) node*: utiliza **datagramas UDP** locais de broadcast, para estabelecer conexões e transmitir datagramas

Dado que os datagramas de *broadcast* tipicamente não atravessam os routers, os **nodos B** podem apenas estabelecer comunicações no interior de simples segmentos de redes

- *Point-to-Point (P) node*: Confia a tarefa de estabelecer conexões com outros nodos a um *NetBIOS name server* (**NBNS**) e a tarefa de transmitir os datagramas a um *NetBIOS Datagram Distributor* (**NBDD**). Estes nodos usam apenas pacotes unicast directos

- *Mixed (M) node*: é uma combinação de nodos P e B, utilizando tanto pacotes unicast (para um NBNS) como de broadcast

Além destes três tipos costuma ser também implementado um quarto tipo, que não se encontra definido nos RFC's:

- *Hybrid (H) node*: trata-se também de uma combinação de nodos B e P. No entanto, tenta sempre contactar em primeiro lugar um NBNS, usando pacotes unicast, e só se este contacto não for possível, é que recorre a pacotes de broadcast

o **NetBIOS Name Server (NBNS)**: regista o nome NetBIOS/Endereço IP, para cada nodo P, M ou H, quando estes são inicializados

Quando um nodo precisa de contactar outro, através do nome NetBIOS, vai consultar o NBNS para obter o endereço IP correspondente

o **NetBIOS Datagram Distributor (NBDD)**: Um servidor NBDD estende o serviço de comunicação por datagramas NetBIOS ao ambiente Internet

Quando um nodo final precisa de efectuar o broadcast de um datagrama, este envia um datagrama unicast, que contém o alcance do datagrama de broadcast, para o NBDD

O NBDD envia o datagrama NetBIOS para cada nodo final que pertença ao *NetBIOS scope* especificado, usando os serviços do NBNS para obter os endereços IP relevantes

- Tomando por base o NetBIOS, a IBM desenvolveu o protocolo **NetBEUI** (*NetBIOS Enhanced User Interface*), para o seu sistema operativo OS/2 Warp

- Mais tarde este protocolo (**NetBEUI**) **foi adoptado pela Microsoft para os seus produtos de rede**

- O NetBEUI utiliza o protocolo de ligação de dados **IEEE 802.2**

- A par de comunicação não orientada à conexão, este protocolo oferece envio garantido de mensagens com tamanho até **64 Kb**, em modo orientado à conexão

1.3.6.2. *NetBIOS over TCP/IP* em sistemas Microsoft Windows

- Quando clientes Windows NT 4, Windows 95 ou Windows 98 são configurados para usar o protocolo TCP/IP, as partilhas de ficheiros e de impressão são actualmente suportadas por **NetBIOS over TCP/IP** (também conhecido por **TCPBEUI**)

- O NetBIOS over TCP/IP em ambiente Windows é baseado nos RFC's 1001/1002, com extensões de encaminhamento

- Estes sistemas costumam usar o ficheiro **LMHOSTS** para efectuar o **mapeamento de nomes** NetBIOS para Endereços IP e vice-versa

- Este ficheiro localiza-se na directoria \WINDOWS, no caso do Win95 e Win98, e na directoria \WINNT\SYSTEM32\DRIVERS\ETC, no caso do WinNT4

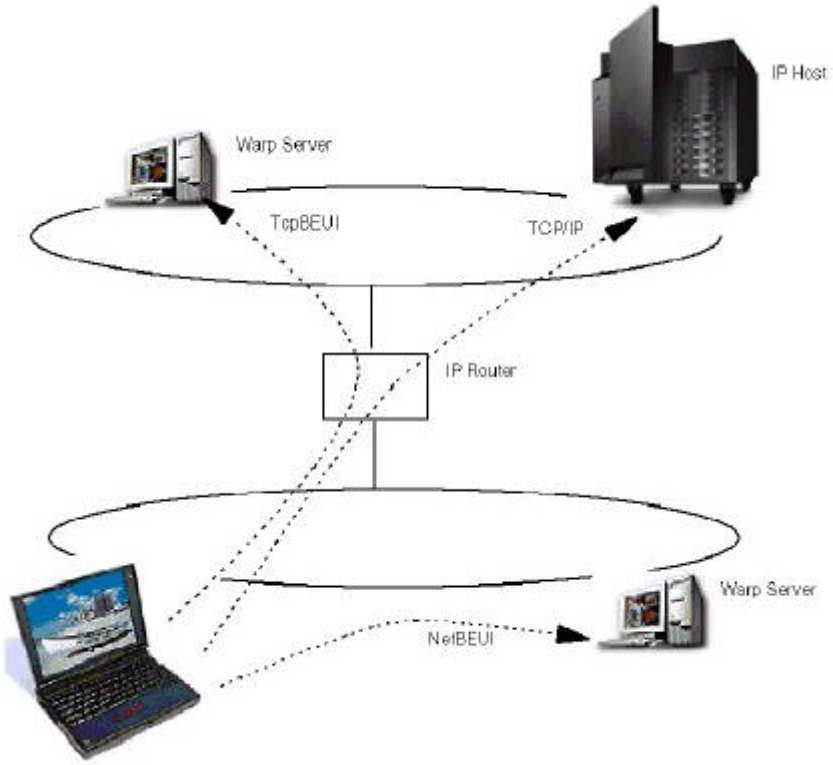


Figura 1.30 – Exemplo de comunicação com múltiplos protocolos

?? Uma forma alternativa de estações Windows resolverem nomes NetBIOS para endereços IP é recorrendo a **Servidores de nomes NetBIOS** (NBNS)

1.3.6.2.1. Windows Internet Name Service (WINS)

?? O **WINS** é a implementação da Microsoft de um NBNS

?? Os servidores WINS apenas suportam clientes proprietários da Microsoft, com esta implementação nativa do NetBIOS e NetBIOS over TCP/IP

?? Cada cliente Microsoft é configurado com o endereço IP do servidor WINS primário, e eventualmente com um secundário

?? Sempre que o cliente é iniciado, tenta registar o seu nome NetBIOS e o seu endereço IP junto do servidor primário WINS

- Se o nome não se encontrar já registado por outro cliente, o servidor responde com uma mensagem onde indica o nome registado e o *time-to-live* – TTL associado

- O registo dos nomes funciona numa base temporária, sendo do cliente a responsabilidade da renovação do mesmo

Capítulo 2 - Multicast e Multimédia

2.1. Multicasting

- O conceito do **IP Multicasting** foi desenvolvido em 1988, por Steve Deering na Universidade de Standford
- O IP multicast utiliza **endereçamento IP da classe D**, entre o endereço **224.0.0.0** e o **239.255.255.255**
- Para cada endereço multicast existe um conjunto de zero ou mais hosts que respondem
- Este conjunto é conhecido por **grupo de hosts**
- Um host que envia informação para um grupo pode não ser membro desse grupo
- Existem **dois tipos** de grupos de hosts multicast:
    - **Permanentes**: constituídos por endereços IP permanentes, atribuídos pela IANA

      A associação de um host a um grupo não é permanente; estes podem deixar ou juntar-se a um grupo, por sua vontade

      Alguns dos endereços deste tipo são:

        - **224.0.0.0** - Endereço base reservado
        - **224.0.0.1** – Todos os sistema nesta sub-rede
        - **224.0.0.2** – Todos os encaminhadores nesta sub-rede
        - **224.0.0.5** – Todos os encaminhadores OSPF
        - **224.0.0.6** – Encaminhadores designados OSPF
        - **224.0.0.9** – Todos os encaminhadores RIP2

      Uma aplicação pode também recuperar um endereço IP de um grupo permanente de hosts, através do DNS,

usando o domínio **mcast.net**, ou determinar o grupo permanente de um endereço usando uma *pointer query*, no domínio **224.in-addr.arpa**

Os grupos permanentes existem mesmo se não contiverem membros

- **Transitórios**: Os grupos que não são permanentes são considerados transitórios, estando disponíveis para atribuição dinâmica, quando necessário

  Os grupos transitórios deixam de existir quando o seu número de membros chega a zero

?? O Multicasting numa mesma rede física que o suporte é simples

?? Para efectuar uma adição a um grupo, um processo a correr num host tem de algum modo informar os seus drivers de rede que deseja ser membro do grupo especificado

?? O software do driver de rede tem de mapear o endereço IP multicast para o endereço físico multicast, e activar a recepção de pacotes para esse endereço

?? As redes **Ethernet** (IEEE 802.3) suportam multicasting se o **byte mais significativo** do endereço de 48 bits for **X'01'**

?? A **IANA reservou** o bloco de endereços Ethernet entre **X'01005E000000' e X'01005EFFFFFF'**

?? A parte mais baixa deste conjunto foi atribuída pela IANA para endereçamento multicast

?? Assim, nas LAN's Ethernet o conjunto de endereços **X'01005E000000'** até **X'01005E7FFFFF'** é usado para IP multicasting, ocupando um total de **23 bits**

?? Os **endereços IP multicast** de 32 bits são **mapeados** para **endereços Ethernet** substituindo os 23 bits de mais baixa ordem dos endereços de classe D pelos 23 bits de mais baixa ordem do bloco reservado pela IANA

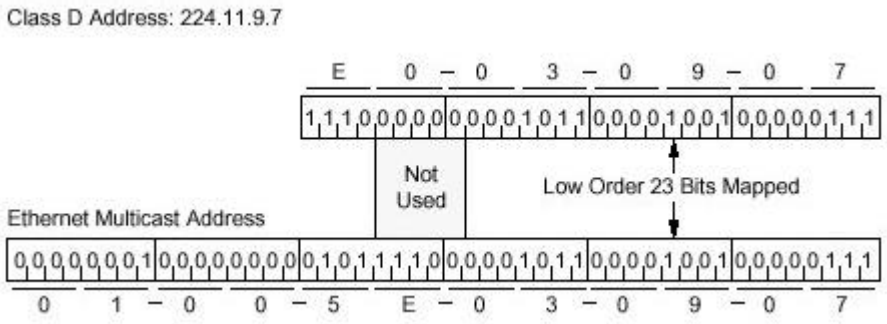


Figura 2.1 – Mapeamento de Endereços IP de Classe D para Endereços IEEE 802.3 Ethernet

- O **multicasting** não está limitado a um única rede física
- Existem dois aspectos a ter em conta para o **informação multicast** poder **atravessar diferentes redes físicas**:
  - Um **mecanismo** para decidir qual o alcance do tráfego multicast

    Ao contrário dos endereços Unicast e Broadcast, os endereços Multicast cobrem a totalidade da Internet
  - Um **mecanismo** para decidir o que um datagrama multicast precisa para ser encaminhado para uma particular rede
- O primeiro problema é resolvido através da utilização do campo *Time to Live* (**TTL**)
- O mecanismo para os encaminhadores decidirem quando devem encaminhar um datagrama é denominado *Internet Group Management Protocol* (**IGMP**)

2.2. O protocolo IGMP (Internet Group Management Protocol)

- O protocolo IGMP é um standard Internet, incluído no STD número 5, a par do IP e do ICMP
- Encontra-se descrito no RFC 1112, com actualizações no RFC 2236

?? É usado pelos hosts que pretendem **juntar-se** ou **abandonar** um **grupo** de hosts multicast

?? O IGMP também é visto como uma extensão do protocolo ICMP, ocupando a mesma posição que este na pilha protocolar TCP/IP

2.2.1. Mensagens IGMP

?? As mensagens IGMP são encapsuladas em datagramas IP, onde o campo *Protocol* toma o valor 2

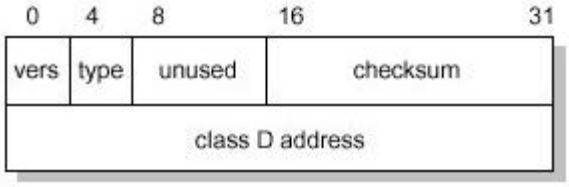


Figura 2.2 – Formato das Mensagens IGMP

?? O campo de dados do datagrama IP transporta a mensagem IGMP, ocupando um total de 8 bytes (ver figura), onde:

- *Vers*: versão do IP. Toma sempre o valor 1
- *Type*: especifica se é uma *query* ou um *report*:
  - 1: define uma *query* enviada por um encaminhador multicast
  - 2: define um *report* enviado por um host
- *Checksum*: soma de controlo, de 16 bits, calculada da mesma forma que para o ICMP
- *Class D Address*: toma o valor zero para um pedido (*query*) e transporta um endereço multicast válido no caso de um *report*

2.2.2. Funcionamento do IGMP

?? Os **sistemas** que utilizam o IGMP dividem-se em dois tipos: **hosts** e **encaminhadores multicast**

- Para receber datagramas multicast, um host tem de se "**juntar**" a um **grupo multicast**
- Para se **adicionar a um grupo**, os hosts enviam um *report* IGMP, através de um interface de rede, com **destino ao endereço de multicast** do grupo de interesse
- Os **encaminhadores multicast** na mesma sub-rede recebem o *report* e activam uma *flag*, que indica que **pelo menos um host** nessa sub-rede **é membro** do grupo especificado
- A inscrição no grupo "*all-hosts*" de uma sub-rede (224.0.0.1) é automática, não necessitando do envio de qualquer pedido por parte dos hosts
- Os **encaminhadores multicast** enviam regularmente *queryes* para o endereço multicast *all-hosts*
- Cada host que pretende continuar membro de um ou mais grupos responde uma vez por cada grupo de interesse
- Cada resposta é enviada após um **período aleatório** de tempo, para assegurar que o tráfego IGMP não vai sobrecarregar a rede
- Se não houver qualquer host a anunciar que faz parte de um grupo durante um intervalo predefinido de tempo, o encaminhador multicast assume que não há qualquer host na sub-rede membro desse grupo
- O **IGMP** apenas **regula** a **comunicação entre os hosts e o último encaminhador multicast**, numa árvore multicast
- O **encaminhamento** dos pacotes **entre os encaminhadores multicast** é **controlado** pelos **protocolos de encaminhamento multicast**
- Na figura seguinte podem ver-se as diferentes secções onde intervêm os protocolos de encaminhamento multicast e o protocolo IGMP, numa árvore multicast
- A figura mostra **pacotes multicast** a ser **enviados** de uma **origem**, através de uma **árvore de distribuição**, até aos **destinatários**

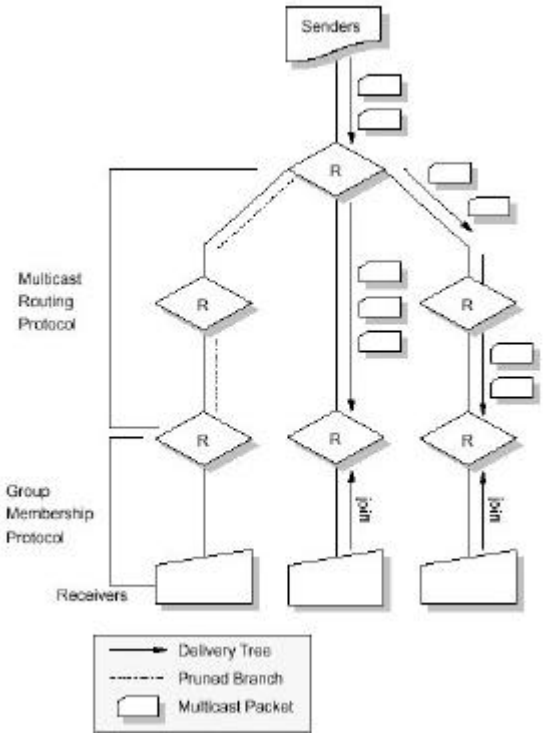


Figura 2.3 – Encaminhamento IP Multicast

?? Se nenhum dos hosts ligados directamente a um encaminhador multicast fizer parte do grupo ao qual pertencem os pacotes, estes não são enviados para esse mesmo ramo da árvore

?? Quando um host ou um encaminhador multicast recebe um **datagrama multicast**, a sua **acção** está **dependente** do **valor do TTL** e do **endereço IP de destino**:

- **0**: o datagrama enviado com o valor zero no TTL **fica restrito ao host de origem**
- **1**: um datagrama com TTL 1 destina-se a **todos os hosts da sub-rede** que são **membros do grupo**

  Os encaminhadores multicast **decrementam** este valor até zero mas, ao contrário dos datagramas unicast, **não emitem a mensagem ICMP** *Time Exceeded Message*
- **2+**: **todos os hosts membros do grupo e todos os encaminhadores multicast** recebem o datagrama

2.3. Encaminhamento Multicast

- Numa internet, um **encaminhador multicast** deve conhecer:
  - **Informação** de modo a determinar a **melhor rota até ao grupo**
  - As **sub-redes** com **elementos activos**
- Numa internet, o encaminhamento multicast tem de ser feito usando quer o **endereço da fonte** quer o **endereço do grupo destinatário**
- Tomando o endereço fonte como referência, as **árvores de distribuição** a partir de cada nodo **são idênticas**, evitando-se a recepção múltipla da mesma informação
- **Objectivos** genéricos, a nível de encaminhamento:
  - escolher uma **rota óptima**, usando opcionalmente diferentes funções de custo
  - **optimizar recursos**; ter em atenção a eliminação de ciclos e distribuição balanceada da carga
  - **minimizar informação de estado** nos encaminhadores
  - **minimizar informação em trânsito**
  - proporcionar suporte para **comunicação fiável**
  - **adaptar-se à dinâmica** dos grupos de multicast
  - ser **estável** e **eficiente** (topologia, dinâmica, dispersão)
  - ...

2.3.1. Algoritmos de Encaminhamento Multicast

- Existem diversos algoritmos dedicados à criação de árvores de distribuição:

- *Flooding, Spanning Trees*
- *Reverse Path Forwarding (RPF), Flood and Prune (RPM), Steiner Trees*
- *Center-Based Trees*

Flooding

- Se 1ª recepção, envia pacote para todas as interfaces de saída, senão ignora
- **Informação de estado**:
  - **Não** usa tabela de encaminhamento
  - **Mantém informação** sobre os **pacotes mais recentes**
- **Pouco eficiente**:
  - Duplicação de pacotes
  - Uso de todos os links
- Apresenta **problemas de escala**

Spanning Trees

- Entre dois encaminhadores apenas existe um ***path*** activo
- A **árvore** é determinada **a partir** de um **nodo raíz**
- Fácil de implementar
- Todas as **fontes partilham** a **mesma árvore**:
  - Concentração de tráfego
  - Rotas não óptimas
- Problemas de escala

?? **RPF (Reverse Path Forwarding)**

- *Shortest-path spanning tree* por fonte
- Simples
- Maior eficiência
  - Distribuição
  - Balanceamento do tráfego
- Indiferente à existência ou não de membros activos

?? **Flood and Prune**

- Também designado por ***Reverse Path Multicast*** (RPM)
- Os datagramas **seguem a árvore** de entrega multicast a **partir da origem**, para **todos os membros do grupo** multicast, **replicando-os apenas nos ramos necessários**
- O envio a **encaminhadores folha** de ***prunes*** ou ***grafts*** (enxertos) **depende da existência** de **membros** do grupo nas **sub-redes** em causa
- Os **encaminhadores intermédios** também têm de ser **notificados** se os encaminhadores folha ficam "inactivos" (*prune*)
- A **informação de estado** é periodicamente **refrescada** e o processo de *flood and prune* reiniciado
- Cada encaminhador tem de ter **informação de encaminhamento** ou de ***prunning*** para **cada par** (fonte, grupo)
- Apresenta problemas de escala

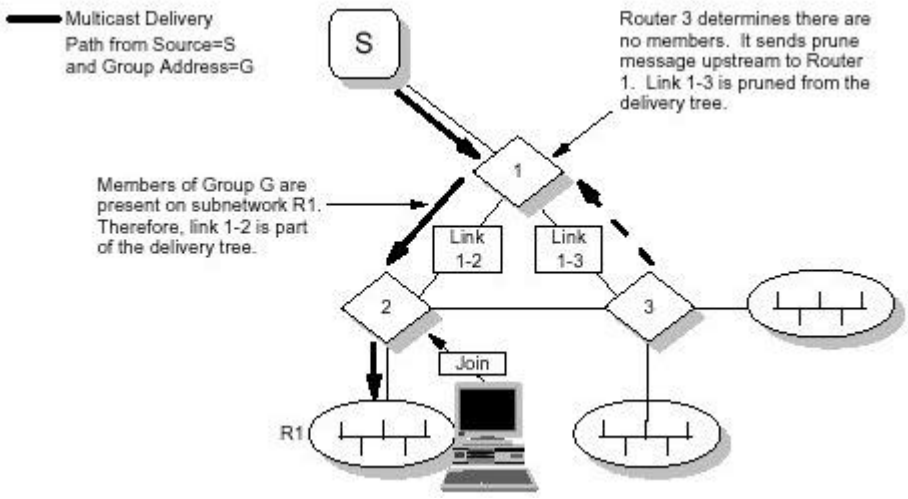


Figura 2.4 – *Reverse Path Multicasting* (RPM)

?? **Steiner Trees**

- **Algoritmo centralizado** para cálculo de uma **árvore de distribuição óptima** incluindo todos os grupos activos
- Trata-se de um **algoritmo instável**
- **Aumento exponencial da complexidade** com o aumento do número de grupos
- Algoritmo **não implementado**

?? **Center-based trees**

- Escolhe um **nodo central** (NC) como centro do grupo
- *Rendez-vous* de potenciais membros com o NC; os **nodos intermédios registam a interface usada** na recepção
- Orientado para **múltiplos pares** (fonte, grupos)
- Construída uma ***spanning tree* por grupo** (partilhada)
- Podem existir **vários nodos centrais** (tolerância a falhas)

- **Informação de estado** mantida **por grupo**, não por fonte
- A **árvore partilhada não** proporciona **rotas óptimas** globais
- Possível **congestão** em links próximos do nodo central

?? Com tanta variedade de algoritmos, **qual a solução óptima**?

?? Não existe uma resposta única para esta pergunta, sendo importante ter em atenção algumas **questões** para chegar a uma resposta, para cada caso:

- Que aplicações?
- Que tipo de serviço?
- Qual o número de nodos da rede?
- Qual a densidade e dispersão dos grupos?

2.3.2. Protocolos de Encaminhamento Multicast

- Os protocolos de encaminhamento multicast são **responsáveis** pela **criação de árvores de distribuição** e pelo **encaminhamento de pacotes multicast**
- Os **mais utilizados** nesta função são:
  - *Distance Vector Multicast Protocol* (DVMRP)
  - *Multicast OSPF* (MOSPF)
  - *Protocol Independent Multicast* (PIM)
  - *Core-Based Trees* (CBT)

2.3.2.1. Distance Vector Multicast Routing Protocol (DVMRP)

- Protocolo de encaminhamento IP multicast mais antigo, definido originalmente em 1988, no RFC 1075
- Foi entretanto melhorado para suportar o mecanismo ***Reverse Path Multicasting*** (RPM)
- Protocolo **IGP** (*Interior Gateway Protocol*), sendo usado exclusivamente para encaminhamento multicast
- Permite a construção de **árvores de entrega** de grupos multicast **baseados na origem**
- Implementa o algoritmo *flood and prune*
- Suporta **Encapsulamento** de pacotes multicast através do estabelecimento de **túneis** em redes que não suportam encaminhamento multicast nativo

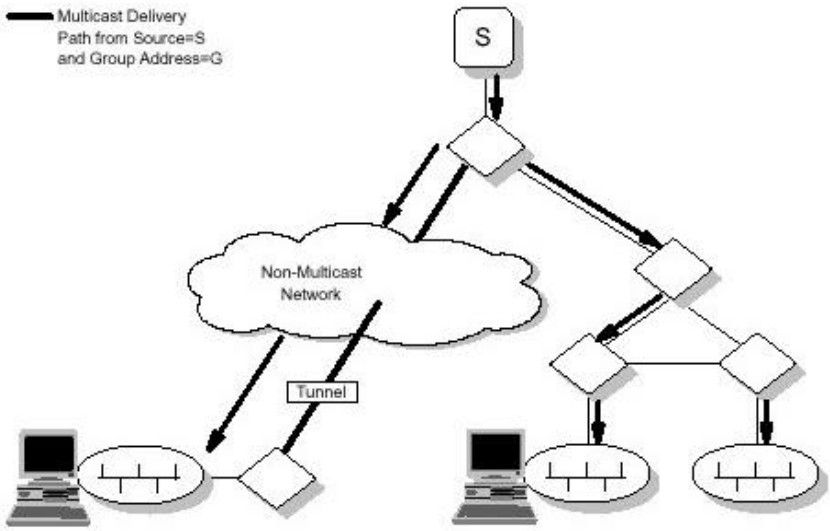


Figura 2.5 – Túnel DVMRP

?? **Modo de operação**: cada processo DVMRP

- **Troca informação** de encaminhamento com encaminhadores DVMRP vizinhos
- **Mantém uma tabela de encaminhamento** com informação sobre as árvores de distribuição óptimas para as possíveis origens
- **Mantém uma tabela de *forwarding*** com base na tabela de encaminhamento, nos grupos conhecidos e estado de *prunning*
- Efectua **regeneração** da informação de estado

?? Em **resumo**:

- Cada processo DVMRP participa na construção de uma árvore de distribuição, de forma a incluir apenas:
  - ?? Sub-redes com grupos de participação não nulos
  - ?? Encaminhadores ao longo do caminho óptimo até essas sub-redes
- As **entradas** na **tabela de *forwarding*** são criadas quando um datagrama novo (fonte, grupo) chega a um encaminhador (após ser efectuado o *reverse-path check*)

2.3.2.2. Multicast OSPF (MOSPF)

?? Conjunto de **extensões multicast** (RFCs 1584 e 1585) adicionadas ao protocolo de encaminhamento OSPF:

- o É destinado a um único sistema autónomo
- o Permite o cálculo rápido de rotas e minimiza o tráfego de encaminhamento
- o Suporta encaminhamento hierárquico e balanceamento de carga

?? Usa o **protocolo IGMP** para gerir grupos de multicast (conceito de *Designated Router* – DR)

?? A **informação de estado** (*link state database*) é idêntica à do **protocolo OSPF** + **informação de grupo**

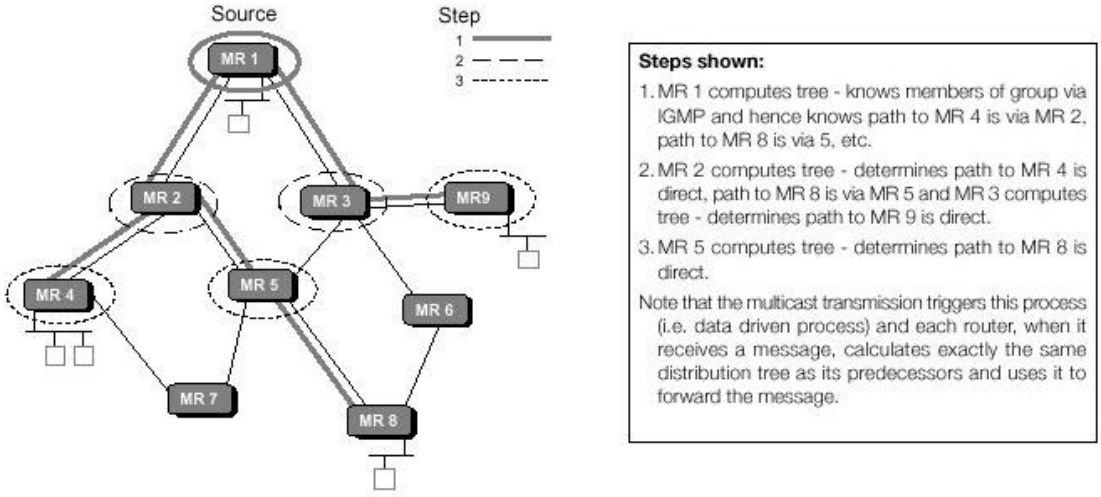


Figura 2.6 – Construção de uma árvore MOSPF

?? Não inclui *tunneling* – coexistência de encaminhadores (M)OSPF

?? Cada encaminhador constrói (*on demand*) uma ***shortest-path tree*** por par (fonte, grupo) usando o **algoritmo de Dijkstra**

?? Existe uma **árvore única** para o **mesmo par** (fonte, grupo)

- Requer **elevados recursos computacionais** por cada computação (fonte, grupo): problemas de escala!
- **Complexidade acrescida** de forma a prever:
  - Caminhos alternativos de igual custo
  - Inter-área e inter-AS multicast
- As **decisões de encaminhamento** são feitas com base numa ***forwarding cache***, construída com base na localização do encaminhador na árvore de distribuição e informação contida na *local group database*
- ***MOSPF vs DVMRP**:*
  - Não segue o algoritmo RPF
  - A informação da *forwarding cache* só é **renovada** se:
    - Topologia é alterada
    - Falta de recursos no encaminhador
    - A distribuição dos grupos é alterada
  - **Não obriga** ao **conhecimento prévio** da **identificação do grupo** (*wilcard multicast receivers*)
  - Evoluiu a partir de um protocolo existente

2.3.2.3. Protocol Independent Multicast (PIM)

- Apresenta um arquitectura orientada para encaminhamento em WAN's
- **Usa informação de encaminhamento** gerada por **qualquer** protocolo de encaminhamento unicast existente (*protocol independent*)
- **Modo de operação** (função da dispersão dos membros):
  - ***Dense mode*** (DM):
    - *Intra*-AS

- *Flood and prune*
- Similar ao DVMRP, mas mais simples

- ***Sparse mode*** (SM):
  - *Inter*-AS
  - *Center-based trees*

- **PIM-SM**
  - Definido no RFC 2362
  - Usa IGMP
  - Um encaminhador PIM tem de **pedir explicitamente** (*rendez-vouz point*) **inclusão** na árvore de distribuição
  - Antes de receber datagramas, um encaminhador PIM mantém **informação de estado reduzida** (associação (*, grupo)-RP)
  - Por defeito, o protocolo usa árvores partilhadas com raíz no RP (*Rendez-vouz Point*)
  - Um encaminhador PIM com receptores locais, após a recepção de datagramas, tem a **possibilidade** de **abandonar a árvore partilhada** e criar uma *shortest-path tree* a partir da fonte

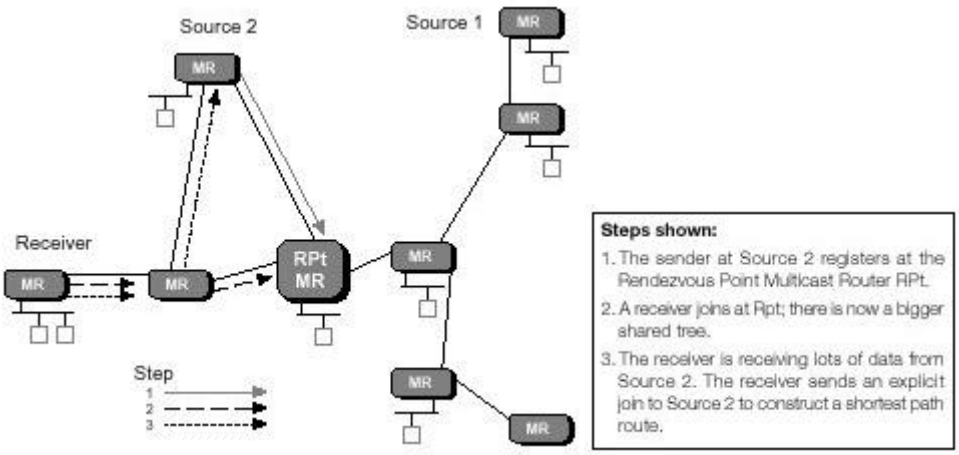


Figura 2.7 – Processo de *join* do PIM-SM

2.3.2.4. Core-Based Trees (CBT)

- Definido nos RFC's 2189 e 2201
- Arquitectura *multicast* é baseada numa **árvore de entrega partilhada**
- **Objectivo principal**: resolver problemas de escala

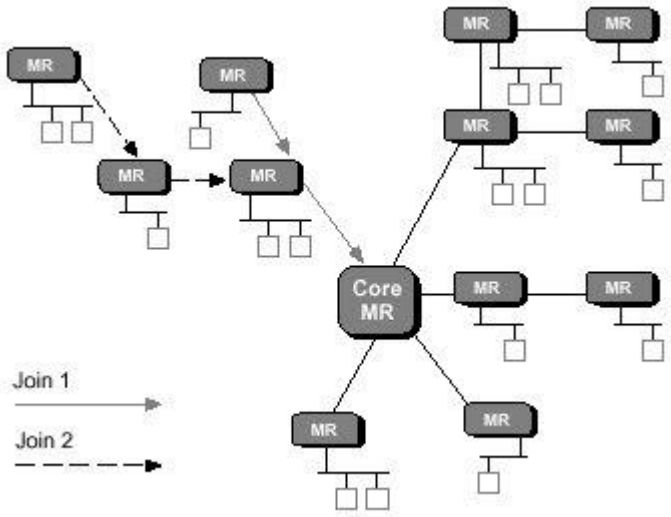


Figura 2.8 – Árvore partilhada do CBT

- *Protocol independent* (tal como o PIM-SM)
- Prevê a existência de um **único *core router*** por grupo, podendo este ser *core* de vários grupos
- Principal **diferença** em relação ao **PIM-SM**: **não permite a optimização do tempo de trânsito** (ST -> SPT) por forma a minimizar a informação de estado nos encaminhadores

2.4. A rede MBONE (Multicast Backbone)

- Começou a funcionar em **1992**
- **Objectivo inicial**: fornecer transmissão da emissão de áudio em tempo real de um *meeting* do IETF, via multicasting, para a Internet
- Trata-se de uma **rede virtual à escala mundial**, que usa a infra-estrutura existente da Internet
- Dado que muitos encaminhadores da Internet não suportam multicasting, a MBONE é "**nivelada**" **no topo dos protocolos Internet** existentes
- As redes ligadas à MBONE têm de cumprir **requisitos especiais** relativamente à **largura de banda disponível**
- Para transmissões de **vídeo**: mínimo de **128 kbps**
- Para transmissões de **áudio**: mínimo de **9-16 kbps**

2.4.1. Encaminhamento MBONE

- O MBONE consiste em várias **ilhas de redes** que suportam **IP multicasting**
- Estas ilhas estão ligadas por **ligações virtuais ponto-a-ponto** (túneis), que fazem a "ponte" sobre as áreas da Internet que não suportam multicasting
- O **envio de pacotes multicast** sobre túneis é efectuado **encapsulando-os em pacotes unicast**, com o endereço de destino do outro lado do túnel

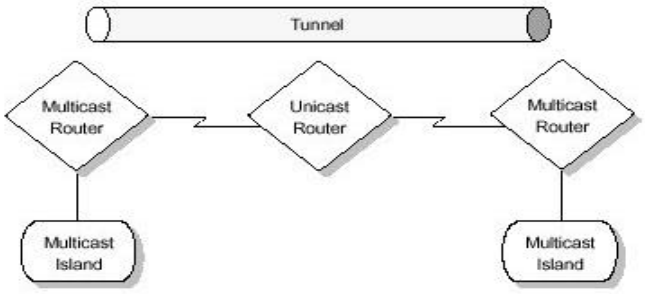


Figura 2.9 – Túnel MBONE

- O encaminhador do destino do túnel retira o cabeçalho do pacote unicast, encaminhando o pacote multicast que vinha no interior – **processo de *tunneling***

2.4.2. Aplicações MBONE

- O primeiro conjunto de **ferramentas de conferência** usadas no MBONE foi desenvolvido no *Network Research Group at Lawrence Berkeley National Laboratory* (LBNL)
- As mais conhecidas são a ferramenta ***vic***, para videoconferência e a ***vat*** para emissões de áudio
- As principais ferramentas disponíveis actualmente são:
  - *SDR Session Directory*: usado para anunciar sessões MBONE
  - *NV Netvideo*: um utilitário de videoconferência
  - Sistema de videoconferência *VIC*
  - Sistema de videoconferência *IVS Inria*
  - *VAT Visual Audio Tool*: utilitário de áudio conferência
  - *RAT Robust Audio Tool*: utilitário de áudio conferência
  - WB Whiteboard
- Muitos dos **utilitários** do MBONE usam o **UDP** em vez do TCP, porque os mecanismos de fiabilidade na transmissão e de controlo de fluxo do TCP não são muito práticos para transmissões broadcast multimédia em tempo real
- Por cima do UDP, algumas aplicações MBONE usam o **protocolo RTP** – *Real-Time Transport Protocol*, descrito a secção seguinte

2.5. Protocolos de Tempo Real RTP e RTCP

2.5.1. Real-Time Transport Protocol (RTP)

- Desenvolvido pelo *Audio/Video Transport Working Group* do IETF
- Definido nos **RFC's 1889 e 1890**
- Usado em **aplicações de tempo real**, como **videoconferência**, **broadcasting de áudio**, **telefonia sobre a Internet**, etc
- Normalmente **implementado** directamente **nestas aplicações**
- Fornece **mecanismos de transporte** que permitem a **sincronização de streams** de dados multimédia de diferentes aplicações
- Se um utilizador utiliza uma ferramenta de vídeo para criar uma sessão de vídeo e outra ferramenta diferente para criar uma sessão de áudio, o **RTP** pode **marcar** os pacotes das duas aplicações com um ***payload type*** e um ***time stamp*** específico, que permite uma **sincronização** do áudio e do vídeo, no receptor
- Neste caso, os **dados** de vídeo e de áudio são **encapsulados** em pacotes RTP, e de seguida transmitidos

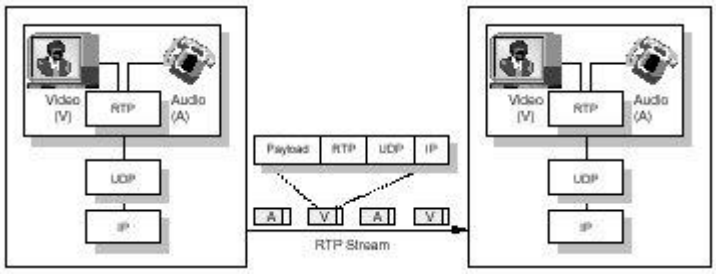


Figura 2.10 – Operação RTP

?? Sem a utilização do RTP, os **dados** de vídeo e de áudio **poderiam** eventualmente **não chegar ao mesmo tempo** ao receptor

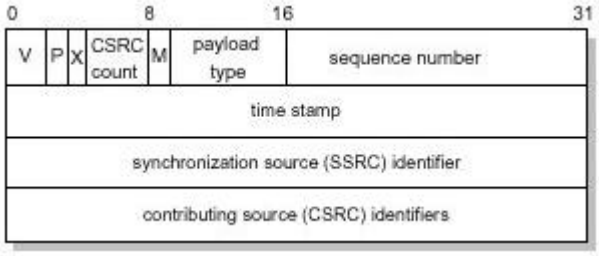


Figura 2.11 – Cabeçalho do RTP

?? O cabeçalho do RTP é constituído pelos **seguintes campos**:

- **V**: versão do RTP (2 bits)
- **P**: bit de *padding*. Se activo, existe um ou mais octetos adicionais de *padding*, usados por alguns algoritmos de cifragem
- **X**: bit de extensão. Se activo, existe um cabeçalho de extensão, que segue o cabeçalho fixo
- ***CSRC count***: contém o número de identificação das fontes que seguem o cabeçalho fixo
- ***M***: *Marker bit*, para utilizações especiais
- **Payload type**: especifica o formato do payload, no pacote RTP
- **Sequence number**: número usado pelo receptor para restaurar a sequência dos pacotes e detectar pacotes perdidos
- **Timestamp**: valor que representa a data em que os dados do pacote RTP foram amostrados
- **SSRC identifier**: campo *Synchronization Source*; identificação (gerada de forma aleatória) da fonte de uma stream de pacotes RTP, que permite ao(s) receptor(es) agrupar os pacotes por fonte sincronizada

- **CSRC list**: contém as fontes que contribuem para o *payload* no pacote actual

?? **Funções** principais do RTP:

- Identificação do *payload type*
- Numeração de sequência
- *Time stamping*

?? O *payload* de um pacote RTP **identifica a codificação** usada pelo pacote áudio ou vídeo

Audio		Video	
Payload Type	Encoding Name	Payload Type	Encoding Name
0	PCMU	24	HDCC
1	1016	25	CellB
2	G.721	26	JPEG
3	GSM	27	CUSM
4	G.723	28	NV
5	DVI4 (8 KHz)	29	PicW
6	DVI4 (16 KHz)	30	CPV
7	LPC	31	H.261
8	PCMA	32	MPV
9	G.722	33	MP2T
10	L16 Stereo	34	H.263
11	L16 Mono	35-71	Unassigned
12	TPSO	72-76	Reserved
13	VSC	77-95	Unassigned
14	MPA	96-127	Dynamic
15	G.728		
16	DVI4		
17	DVI4		
18	G.725		
19	CN		
20	Unassigned		
21	Unassigned		
22	Unassigned		
23	Unassigned		

Figura 2.12 – Tipos de *Payload* RTP

?? Embora o **RTP** tenha sido **desenvolvido** inicialmente para **transportar dados multimédia**, qualquer outra aplicação que produza streams contínuas de dados pode usar este protocolo

?? A **Numeração de sequência** é usada para **restaurar a ordem** dos pacotes no receptor e para **detectar pacotes perdidos**

?? O *Timestamp* é usado no RTP para **sincronizar** os pacotes de diferentes fontes

?? Representa o tempo de criação do primeiro byte num pacote RTP

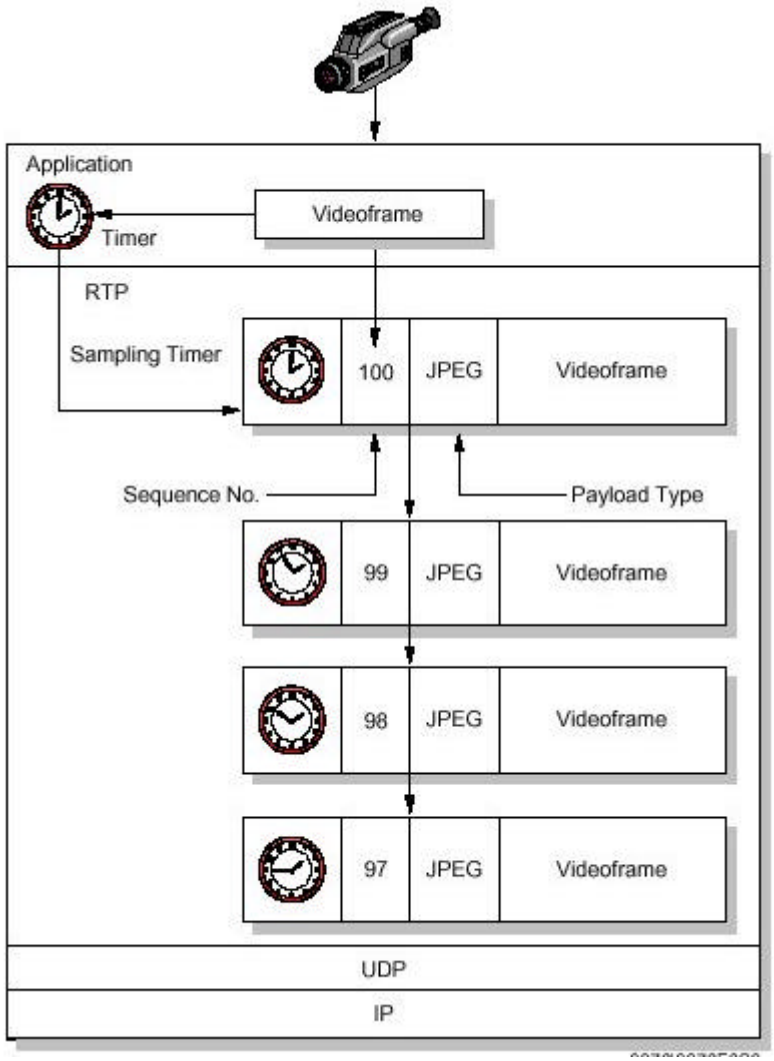


Figura 2.13 – Geração de pacotes RTP numa aplicação de Vídeo

2.5.2. Real-Time Control Protocol (RTCP)

?? Tem como **objectivo** principal **fornecer um *feedback* da qualidade** de uma transmissão RTP

- **Funções** equivalentes às de **controlo de fluxo e de congestão** de outros protocolos de transporte
- Produz *reports* no emissor e no receptor, como estatísticas das streams e contagem de pacotes
- Utiliza o porto UDP (**porto RTP + 1**)

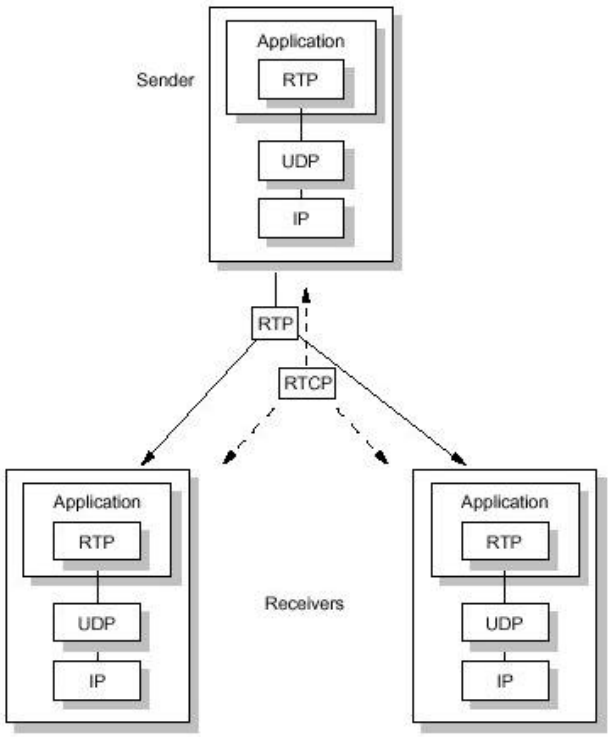


Figura 2.14 – Entrega de pacotes RTP e RTCP

- Como os pacotes RTCP têm **diferentes funções**, existem **diferentes formatos** definidos:
  - **SR**: *Sender report*
  - **RR**: *Receiver report*
  - **SDES**: *Source description items*
  - **BYE**: Indica fim da participação
  - **APP**: Funções *application-specific*

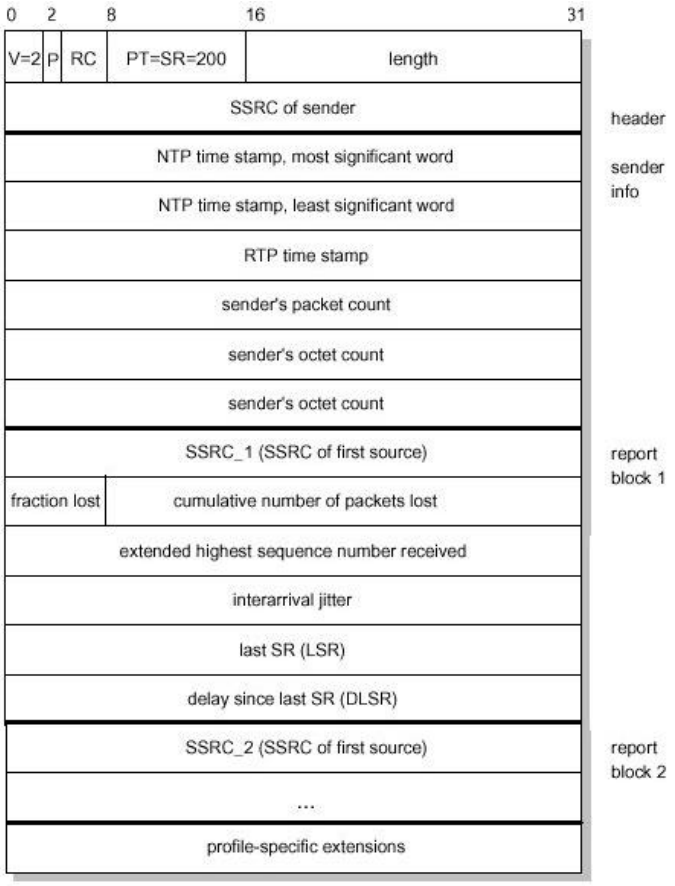


Figura 2.15 – Pacote RTCP *Sender Report*

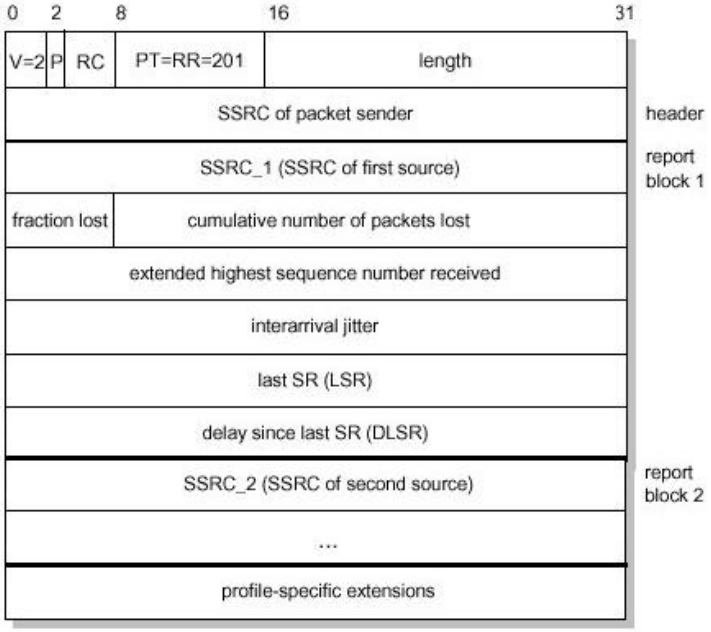


Figura 2.16 – Pacote RTCP *Receiver Report*

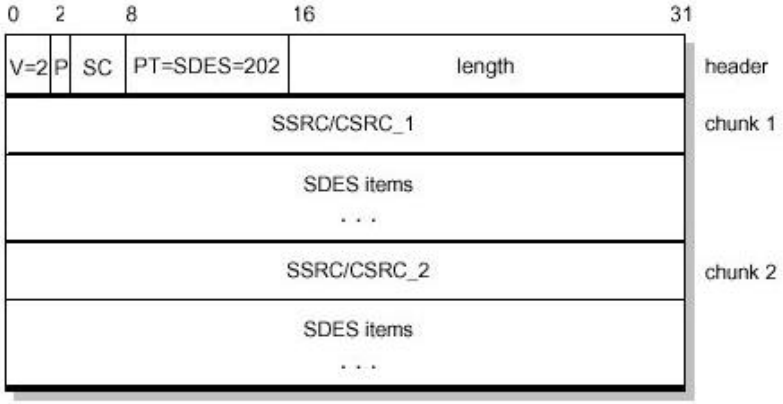


Figura 2.17 – Pacote RTCP SDES

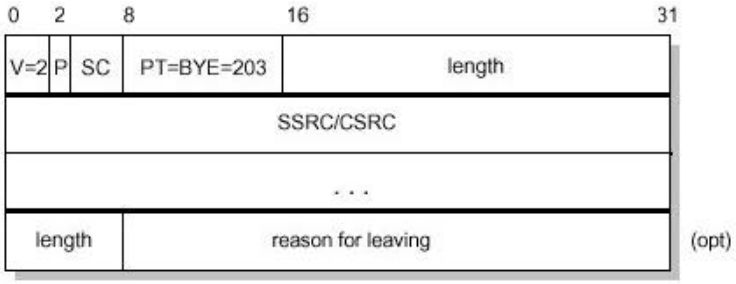


Figura 2.18 – Pacote RTCP BYE

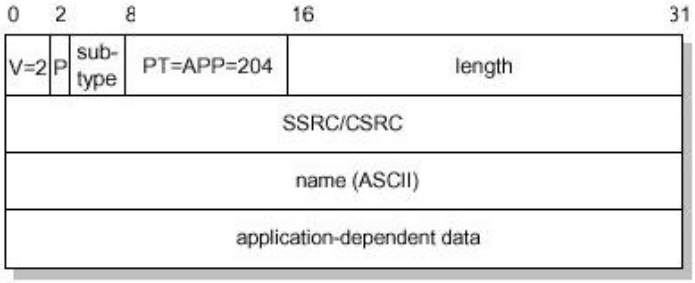


Figura 2.19 – Pacote RTCP APP

2.5.3. Tradutores e Misturadores RTP

?? O RTP suporta a utilização de **tradutores** (Translators) e de **misturadores** (Mixers) para modificar uma stream de pacotes RTP

?? Um **Tradutor RTP** efectua a **conversão da codificação** de pacotes RTP entre diferentes tipos de *payload* (p.e. de MPEG para H.261)

?? Todo o **processo de sincronização mantém-se intacto** com esta tradução

?? Exemplo:

- o 3 estações ligadas a uma rede de 100 Mbps, participam numa videoconferência, trocando dados MPEG a 1.5 Mbps
- o Se outra estação que está ligada a esta rede por um link a 1 Mbps pretende participar na videoconferência, surge um problema, já que não tem largura de banda suficiente para receber uma única stream de vídeo
- o Uma solução passa pela codificação do vídeo noutro formato (p.e. H.261 a 256 Kbps), alterando o *payload type*
- o Esta alteração é feita apenas para a quarta estação, resultando para esta uma qualidade inferior, mas mantendo-se a qualidade da transmissão entre as outras três estações

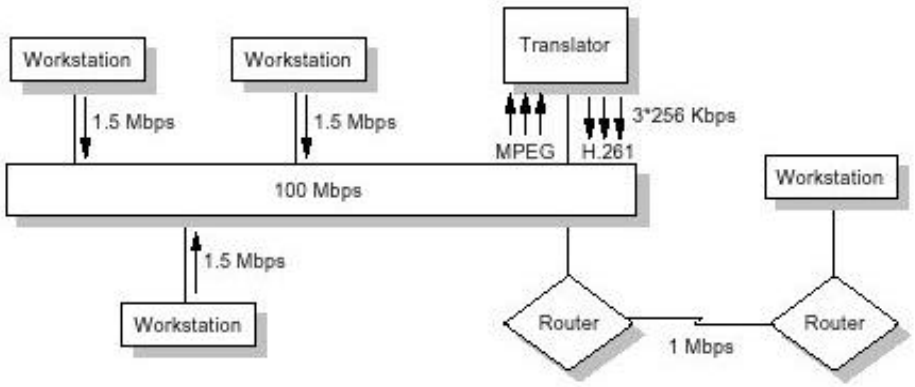


Figura 2.20 – Tradutor RTP

?? Um **Misturador efectua a "fusão" de diferentes *streams*** de dados RTP numa única *stream*

?? Faz sentido a sua utilização apenas para **tráfego de áudio**

?? Exemplo:

- o 3 participantes numa áudio-conferência ligados a 1 Mbps, onde cada um produz uma *stream* PCM áudio de 64 Kbps

- Se outra estação pretende juntar-se à conferência, precisaria de 192 Kbps de largura de banda, para receber os outros participantes

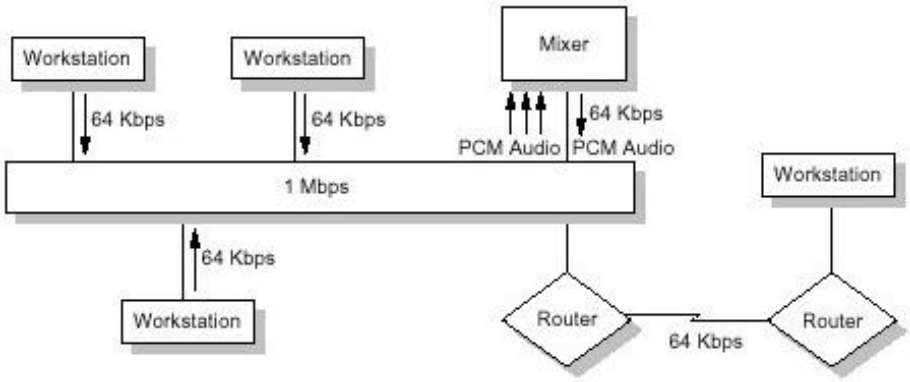


Figura 2.21 – Misturador RTP

- Se estiver ligada por apenas 64 Kbps, um Misturador RTP pode "juntar" as *streams* dos outros 3 participantes numa única *stream* de 64 Kbps

?? No processo de mistura, **não é necessário alterar o formato dos dados** da stream "misturada", podendo manter-se o mesmo *payload type*

?? Apenas será necessário alterar o tipo do *payload* se a largura de banda disponível à saída for inferior à largura de banda mínima para a transmissão de uma *stream* com determinado formato

?? Nesse caso, o Misturador poderá usar uma codificação diferente

2.6. Voice over IP

- A possibilidade de transmitir **voz sobre a Internet** é bastante atractiva, dado o **baixo custo da tecnologia IP e da largura de banda**, quando **comparado** com as **tecnologias tradicionais**
- Existem actualmente várias **aplicações de tempo real**, com capacidade para suportar este serviço
- O problema está na **utilização**, na maior parte dos casos, de **protocolos e algoritmos de compressão próprios**
- Este facto torna **incompatível a sua inter-operação**
- *Voice over IP Forum* – tem por objectivos a **definição de standards** para este serviço
- *VoIP Forum Service Interoperability Implementation Agreement* (VoIP IA): **especificação** que inclui um **conjunto de standards** para garantir um **serviço completo de telefonia pela Internet**
- O **standard** base **recomendado** pelo Fórum VoIP é o standard **H.323**, da ITU
- **Elementos chave** do VoIP IA:
  - **H.323**, usado para o **estabelecimento de chamadas** e **negociação de capacidade**
  - Recomendação ITU-T **Q.931** (faz parte do H.323), que faz a **sinalização de chamada** entre elementos do serviço
  - Mecanismos de **resolução dinâmica de endereços IP**, através do RAS H.323 (*Registration Admission, Status*)
- O objectivo base do VoIP IA é **ajudar a garantir a interoperabilidade** entre equipamento de diferentes fabricantes

2.6.1. Recomendação ITU-T H.323

?? **Standard para comunicação multimédia** através de **redes de comutação de pacotes**

?? Título formal: *Visual Telephone Systems and Equipment for Local Area Networks Which Provide a Non-Guaranteed Quality of Service*

?? **Componentes** de uma rede H.323:

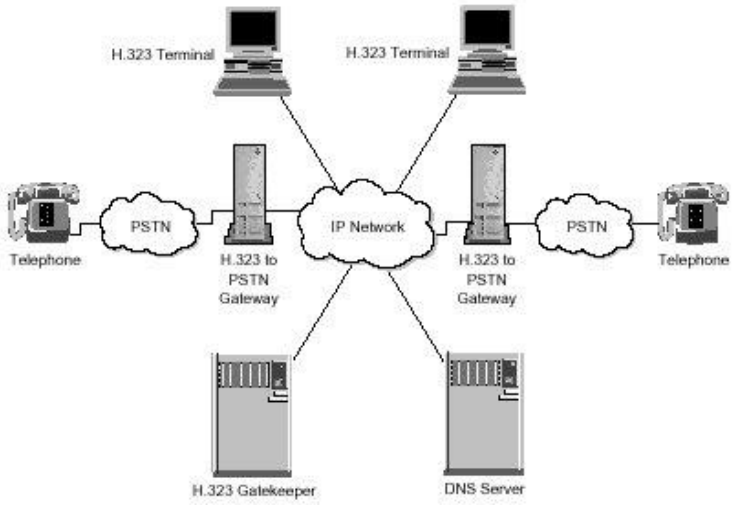


Figura 2.22 – Elementos de uma Rede VoIP

- **Terminais**:
  - *Corporate Network Terminal*: suporta alta qualidade e múltiplas funções
  - *Internet Terminal*: optimizado para utilização com largura de banda mínima
- ***Gateways***: fornecem interligação entre terminais H.323 ligados a redes IP e outros dispositivos de áudio (p.e. telefones normais) ligados por outras redes
- ***Gatekeepers***: fornece funções de Servidor de Directório e funções de controlo do sistema
- ***Multipoint Control Units*** (MCUs). Um MCU consiste num *Multipoint Controler* (MC) e num *Multipoint Processor* (MP). Fornecem gestão de conferências multiponto (em modos unicast e multicast)

2.6.2. Compressão de voz (G.723.1 e G729)

- As recomendações da ITU G723.1 e G729 **especificam a operação de Codecs** em **ambiente** de rede **de baixa qualidade**, similares aos encontrados na Internet

- **G723.1**:
  - Oferece um **nível** relativamente **elevado de compressão**, com um **bit rate de saída** entre os **5.3** e os **6.4 Kbps**
  - **Pacotes** com **tamanho** dos dados entre **20** e **24 bytes**
  - Significa que **cada pacote transporta 30 ms** de **sinal de voz** em tempo real
  - **Atraso** fim-a-fim aproximado de **135 ms**

- **G.729**:
  - Divide-se em **duas recomendações**: **G.729** e **G.729A**
  - Diferem na carga de processamento requerida para implementar o algoritmo
  - O **bit rate** resultante da compressão é de **8 Kbps** com **10 ms de voz** em cada pacote
  - **Tamanho** dos dados do pacote resultante: **10 bytes**
  - Usa o **algoritmo** *Conjugate Structured Algebraic Code Excited Linear Predictive* (CS-ACELP)
  - Atraso fim-a-fim de aproximadamente **50 ms**

- **GSM** (*Group Special Mobile*):
  - O **algoritmo de compressão** de áudio GSM foi originalmente **definido para utilização com telefones móveis digitais**
  - Standard Europeu (não é standard ITU)
  - Trata-se de uma **alternativa** para o VoIP

?? Dado que **cada pacote** com compressão G.729 tem **10 bytes** de tamanho, **não é muito eficiente a sua utilização em redes TCP/IP**, devido ao tamanho do cabeçalho que tem de ser adicionado por cada 10 bytes

?? Por isso, o **G.723.1** é muito **mais eficiente** na utilização da largura de banda

?? Por outro lado, o **G.729** apresenta **atrasos bastante menos significativos**

2.6.3. A pilha protocolar VoIP

?? A **VoIP IA** é constituída por um **conjunto de protocolos** interligados entre si, para produzir o resultado final

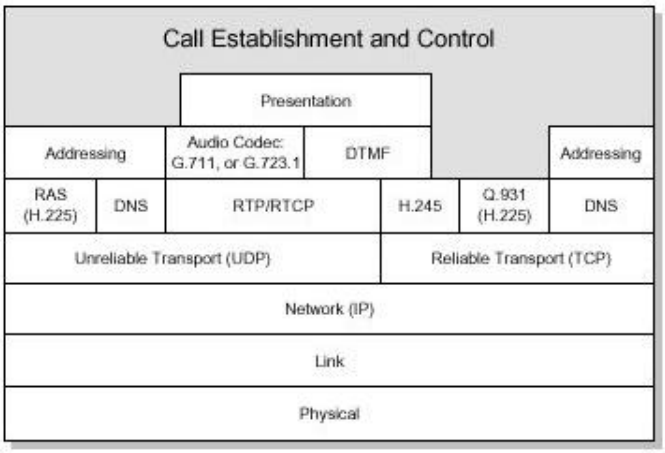


Figura 2.23 – Pilha protocolar VoIP

?? A **pilha protocolar VoIP** é muito **semelhante** à pilha **H.323**, no entanto apresenta bastantes **funções adicionais**

?? **Call Establishment and Control**: determina a sequência e o *timing* de:

- Estabelecimento da chamada
- Terminação da chamada
- Controlo da sessão H.323, após seu estabelecimento

?? **Presentation**: interpreta a sintaxe dos sinais de áudio transmitidos e da informação de controlo "*in-band*", como

sejam os sinais de ligação *dual tone multiple frequency* (DTMF)

- **Addressing**:
  - Utilizado devido ao dinamismo do endereçamento IP em algumas circunstâncias, e dado que não é muito fácil os utilizadores decorarem endereços IP
  - mecanismo dinâmico (tipo Agenda telefónica), que se adapta à mobilidade dos utilizadores (p.e. recorrendo ao RAS)
- **RAS** (Registration, Admission, Status): protocolo que especifica como as entidades H.323 pode aceder a *gatekeepers*, para realizar tradução de endereços
- **Audio Codec** (Coder/Decoder): usados para converter áudio analógico para/de sinais digitais
- **H.245**:
  - protocolo de controlo para comunicações multimédia
  - Na pilha VoIP especifica como os canais das chamadas telefónicas são estabelecidos e controlados
  - Inclui métodos para envio e recepção de *dial tones* DTMF
- **H.225.0**:
  - Conhecido por *Media Stream Packetisation and Synchronization on Non-Guaranteed Quality of Service LANs*
  - Define a forma como os pacotes de áudio, pacotes de controlo e informação de ligação DTMF são transmitidos através de redes IP
- **Q.931**: protocolo de sinalização, usado para ligar, terminar e controlar sessões de comunicação H.323
- **RTP/RTCP**: protocolos de tempo real usados na pilha VoIP para controlar e monitorizar a entrega de pacotes

3. Segurança e Privacidade

3.1. As questões de segurança nas redes informáticas

- **Anos 50** (inícios)
  - Número reduzido de computadores
  - Número limitado de pessoas com acesso
  - Computadores usados principalmente para tratar informação confidencial
  - Ameaça: espionagem e invasão da privacidade
  - Segurança informática <-> segurança da informação
- **Anos 60**
  - Proliferação das redes (partilha da informação e de recursos computacionais)
  - Necessidade de sistemas operativos seguros
  - Ao roubo de informação acrescenta-se o manter a integridade da informação
- O únicos **computadores seguros** são os que estão **desligados**
- No entanto, um computador desligado **não é muito útil**
- Assim, o **principal objectivo** da segurança informática é encontrar um **equilíbrio** entre a manutenção dos computadores **utilizáveis** e **limitação** do seu **uso errado** ou **abusivo**
- O **tempo e dinheiro** investidos na segurança dos computadores e das redes informáticas deve ser **proporcional às perdas**, em caso de utilização abusiva dos mesmos

- **Definições de segurança informática**:
  - Impedir que alguém ou algo faça algo que não queremos **a**, **como**, **com**, **em** ou **a partir de** computadores ou outros dispositivos periféricos
  - Um computador é **seguro** se **podemos depender dele** e o seu software se comporta como é esperado (confiança)
- **Definição do CERT/CC** (www.cert.org):
  - **Segurança informática** é **impedir os agressores** de alcançar os seus objectivos através do acesso não autorizado, ou do uso não autorizado de computadores ou de redes de computadores
  - **Exclui**:
    - Roubo do equipamento
    - Ameaças ambientais
    - Questões de software, a não ser que possam ser aproveitadas para fornecer acesso ou uso não autorizado
- **Alvos – Proteger o quê?**
  - Processos que estão a ser executados em computadores
  - Informação armazenada nos computadores
  - Informação em trânsito nas redes
- **Agentes agressores e motivações – proteger de quem?**
  - **Hackers** -> desafio, notoriedade
  - **Espiões** -> benefícios políticos
  - **Terroristas** ->benefícios políticos
  - **Criminosos** -> benefícios financeiros
  - **Vândalos** -> danos

- O **acesso não autorizado** consiste na exploração de **Vulnerabilidades**:
  - Vulnerabilidade de implementação (bug de software)
  - Vulnerabilidade de concepção
  - Vulnerabilidade de configuração
- **Principais objectivos**:
  - Corrupção da informação
  - Divulgação da informação
  - Roubo de serviço
  - Rejeição de serviço (*denial of service*)

3.2. Problemas de segurança mais comuns e respectivas soluções

- **Problemas de segurança** mais comuns, nas redes informáticas:
  - **Escuta da rede**: obtenção de acesso a dados ou passwords
  - **Disfarce**: para obtenção de acesso não autorizado a dados ou criação não autorizada de e-mails, ordens, etc
  - **Rejeição de serviço** (*Denial-of-service*): tornar recursos de rede não funcionais
  - **Resposta a mensagens**: para obter acesso a informação em trânsito e alterá-la
  - **Obtenção indevida de passwords**: para ter acesso a informação e serviços que normalmente seriam negados (ataques de dicionário)
  - **Obtenção indevida de chaves**: para ter acesso a dados e passwords cifrados (ataques de força bruta)

- **Vírus**: para destruição de dados

?? Com o mesmo zelo com que os **intrusos procuram formas de entrar** em redes informáticas, os **administradores** destas redes **tentam protege-las**

?? Pegando nas **vulnerabilidades** referidas atrás, existe um **conjunto de soluções**, que defendem as redes contra esses ataques

?? De notar que cada uma destas soluções resolve apenas um ou um número limitado destes problemas

?? Apenas uma combinação das soluções propostas pode ser considerada para garantir algum nível de segurança

?? Principais soluções:

- **Cifragem**: para protecção de dados e passwords
- **Autenticação e autorização**: para prevenir acessos não autorizados
- **Verificação de integridade e códigos de autenticação de mensagens**: para protecção contra alterações impróprias de mensagens
- **Não repudiação**: para assegurar que uma acção não pode ser negada pela pessoa que a realiza
- **Assinaturas e certificados digitais**: para verificação de partes
- **Passwords *one-time* e chaves aleatórias *two-way***: autenticação mútua das partes de uma conversação
- **Refrescamento frequente de chaves e utilização de chaves fortes**: para protecção contra a quebra de chaves (*cryptanalysis*)
- **Dissimulação de endereços**: para protecção contra ataques de negação de serviço

?? A tabela seguinte apresenta os **principais problemas e vulnerabilidades** de segurança para as soluções listadas atrás:

Problemas/Vulnerabilidades	Remédios
Como prevenir “escutas da rede”?	Cifragem de mensagens, usando tipicamente uma chave secreta partilhada. (As chaves secretas oferecem performances muito melhores que as chaves publicas/privadas)
Como distribuir chaves de forma segura?	Usando uma técnica de cifragem diferente, tipicamente de chaves publicas/privadas
Como prevenir o “envelhecimento” das chaves e a dedução de futuras chaves pela descoberta das actuais?	Refrescar chaves com frequência, e não derivar novas chaves a partir das antigas
Como prevenir retransmissão de mensagens por um impostor (ataque de repetição)?	Usando números de sequência (*time stamps* normalmente não são de confiança, em termos de segurança)
Como ter a certeza que uma mensagem não foi alterada em trânsito?	Usando sumários de mensagens (funções de hash ou funções one-way)
Como ter a certeza que o sumário de uma mensagem (*message* digest) não foi comprometido?	Usando assinaturas digitais para cifragem do sumário com chave secreta ou privada (autenticação na origem, não repudiação)
Como ter a certeza que a mensagem e a assinatura tiveram origem no agente correcto?	Usando *two-way handshakes* envolvendo números aleatórios cifrados (autenticação mútua)
Como ter a certeza que as “handshakes” são trocadas entre os agentes certos (man-in-the-middle attack)?	Usando certificados digitais (associando chaves públicas a entidades fixas)
Como prevenir uso impróprio de serviços por utilizadores não correctamente	Usando modelos de controlo de acesso *multi-layer*

autenticados?	
Como realizar protecção contra vírus?	Restringindo o acesso a fontes exteriores; correr software anti-vírus (devidamente actualizado)
Como obter protecção contra mensagens não solicitadas ou maliciosas?	Restringindo o acesso à rede interna, usando filtros, firewalls, proxies, autenticação de pacotes, escondendo os endereços internos e a estrutura de nomes, etc

?? Em geral, devemos manter a **rede permanentemente vigiada** e **protegida em relação ao exterior**, mas mantendo sempre um **olho** bem **aberto** para o **interior**

?? Muitos dos **ataques** às redes informáticas **surgem no interior** das próprias organizações

3.3. Implementações das soluções de segurança

?? Principais protocolos e sistemas usados para fornecimento de segurança a diferentes níveis, nas redes TCP/IP:

- **IP filtering**
- **Network Address Translation** (NAT)
- **IP Security Architecture** (IPSec)
- **SOCKS**
- **Secure Sockets Layer** (SSL)
- **Proxies** Aplicacionais
- **Firewalls**
- **Kerberos** e outros sistemas de autenticação (servidores AAA)

- **Secure Electronic Transations** (SET)

?? A figura seguinte apresenta o posicionamento das principais soluções de segurança nos diferentes níveis do modelo TCP/IP

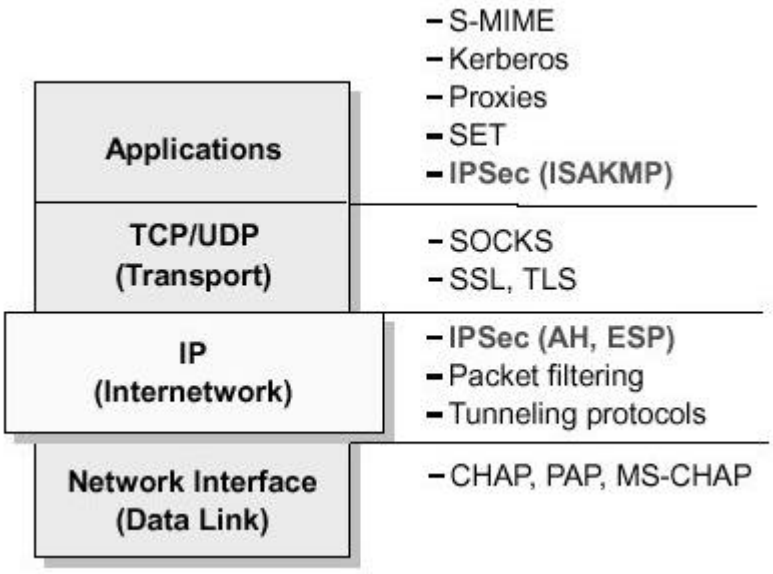


Figura 2.24 – Soluções de segurança nos diferentes níveis do TCP/IP

?? A figura seguinte sumariza as características de cada uma das soluções de segurança mencionadas atrás, comparando-as umas com as outras:

Table 11. Security Solution Implementations - A Comparison

	Access Control	Encryption	Authenti-cation	Integrity Checking	PFS	Address Conceal-ment	Session Monitoring
IP Filtering	y	n	n	n	n	n	n
NAT	y	n	n	n	n	y	y (connection)
IPSec	y	y (packet)	y (packet)	y (packet)	y	y	n
SOCKS	y	n	y (client/user)	n	n	y	y (connection)
SSL	y	y (data)	y (system/ user)	y		n	y
Application Proxy	y	normally no	y (user)	y	normally no	y	y (connection and data)
AAA servers	y (user)	n	y (user)	n	n	n	n

Figura 2.25 – Comparação entre diferentes soluções de segurança

- Como mencionado atrás, uma solução global de segurança pode, em muitos casos ser fornecida apenas com uma combinação das opções listadas, p.e. usando uma **Firewall**

- **Requisitos particulares** de segurança podem ser especificados em **políticas de segurança** (*security policies*)

3.4. Políticas de segurança da rede

- A **política global de segurança** da rede de uma organização deve ser **determinada** de acordo com uma análise da segurança e das necessidades do negócio

- Apesar de uma Firewall poder garantir a segurança de uma rede, esta tem pouco valor se a política global de segurança **não estiver correctamente definida**

- As políticas de segurança da rede **definem**:
  - que serviços são explicitamente permitidos ou negados
  - como esses serviços podem ser usados
  - e excepções a essas regras

- As **regras** da política de segurança devem ser implementadas na Firewall e/ou no gateway/servidor de acesso remoto (RAS)

- Em geral, uma firewall utiliza um dos seguintes **métodos**:
  - **Tudo que não é explicitamente permitido é proibido**:
    - Bloqueio de todo o tráfego entre duas redes excepto o dos serviços e aplicações que são explicitamente permitidas
    - Consequentemente, cada serviço ou aplicação permitida tem de ser implementado, um a um
    - Método mais seguro do ponto de vista da rede

- ✍Por outro lado, do ponto de vista dos utilizadores, trata-se do método mais restritivo e menos conveniente

- **Tudo que não é explicitamente proibido é permitido**:
  - ✍Todo o tráfego entre duas redes é permitido, excepto aquele que é explicitamente negado
  - ✍Em consequência, cada serviço ou aplicação não seguro ou potencialmente perigoso tem de ser proibido, um a um
  - ✍Apesar de se tratar do serviço mais flexível e conveniente para os utilizadores, pode ser potencialmente perigoso e originar mesmo sérios problemas de segurança

?? Os **Servidores de Acessos Remotos** (RAS) devem:

- fornecer **autenticação** dos utilizadores
- também fornecer uma forma de **limitar** certos utilizadores a certos sistemas e/ou redes, dentro da rede da organização (Intranet)
- determinar se um utilizador é considerado **móvel** (pode-se ligar de múltiplas localizações) ou **estacionário** (pode-se ligar apenas de uma única localização remota)

3.5. Introdução à Criptografia

3.5.1. Terminologia

?? **Criptografia**: ciência que permite manter os dados e as comunicações seguras

?? **Objectivo** base da criptografia: garantir comunicação segura entre agentes, em ambiente hostil

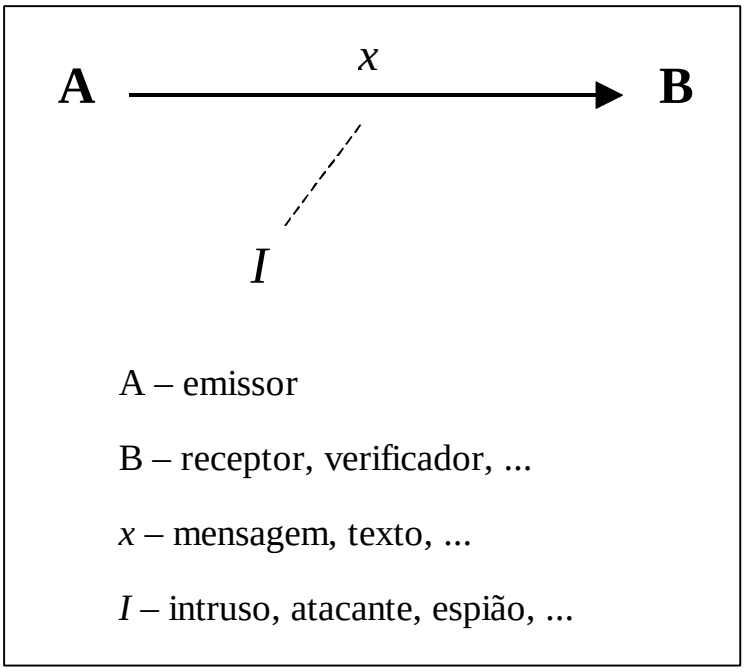


Figura 2.26 – Processo de comunicação entre agentes, em ambiente hostil

?? Para isso acontecer são usadas técnicas, como a **cifragem**, **decifragem** e **autenticação**

?? Qualquer **processo que efectua transformação de dados** baseada em **métodos que são difíceis de reverter**, é considerado **criptografia**

?? O factor chave da **criptografia forte** é a **dificuldade** de utilização de **engenharia inversa** (*reverse engineering*)

- **Cifragem**: transformação de **mensagens em claro** para uma **forma não legível**, com o objectivo de **esconder o seu conteúdo**

- **Decifragem**: transformação oposta à cifragem, que **devolve o texto em claro**

- A **função matemática** usada na cifragem e decifragem tem o nome de **cifra**

- Um cifra utiliza um **protocolo criptográfico**

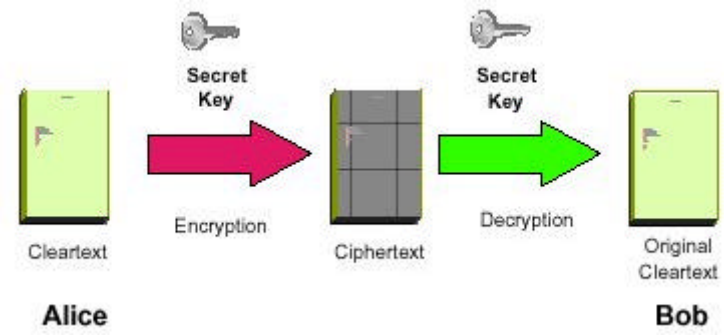


Figura 2.27 – Cifragem e decifragem, com base em chaves

- **Características da segurança**:

  - **Confidencialidade** - garante o controlo do conhecimento da mensagem:

    - após o seu envio, apenas *A* e *B* (e nenhum intruso) conhecem *x*

    - *B* deve ser o único receptor da mensagem

  - **Autenticidade**

    - garante a **integridade** da mensagem: a mensagem que chega a *B* é a mesma que foi emitida por *A*

    - garante a **originalidade** da mensagem: *B* tem a certeza da origem *A* da mensagem *x*

    - garante a *A* a **refutação** de mensagens fraudulentas e a *B* a **não-refutação** de mensagens legítimas

  - **Entidade**

- garante a **identificação** de um agente perante outro agente
- *B* e A devem ter garantias que comunicam um com o outro e nenhum deles pode ser substituído por *I*

?? **Requisitos de segurança**:

- **A segurança em canais fechados**, ou segurança física, baseia-se no obscurantismo nas comunicações e nas técnicas:
  - O conhecimento sobre o tráfego de informação e sobre as técnicas de segurança é controlado
  - Os sistemas de segurança, uma vez instalados, não são testados e se forem atacados não é fácil constatar a existência e a natureza do ataque
  - Não impõe requisitos especiais aos sistemas computacionais
- A **segurança em canais abertos**:
  - assume que todo o tráfego de informação é público e que as técnicas de segurança são publicamente conhecidas
  - O controlo do segredo restringe-se a certos items de informação: as **chaves**
  - Os sistemas de segurança podem ser testados pela comunidade científica em geral e eventuais quebras de segurança têm mais possibilidade de ser detectadas
  - Porque o conhecimento dos intrusos é maior, as técnicas de segurança devem ser mais exigentes e computacionalmente mais elaboradas dos que as usadas em canais fechados

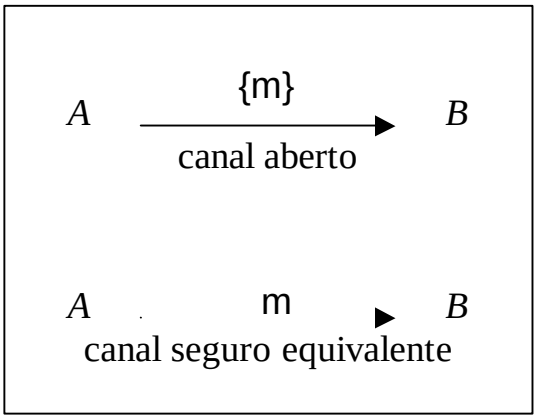
?? Do ponto de vista da criptografia, pode-se definir **segurança** como sendo a **ausência de ataques**

- Existem dois grandes **tipos de ataques**:
  - **Intrusão**: ocorre quando um agente (intruso) não devia – de acordo com os objectivos gerais do sistema – ter conhecimento de um determinado item de informação e acaba por obter esse conhecimento
  - **Pseudominia**: ocorre quando um agente altera fraudulentamente o estado de conhecimento de outro agente
- A maioria dos ataques reais têm aspectos tanto de intrusão como de pseudominia

3.5.2. Cifras

?? Para garantir a **confidencialidade** na comunicação usam-se **cifras**:

- o funções **invertíveis** transformando textos em **criptogramas**



M	espaço de textos
m ? M	texto claro
{m}	criptograma de m

?? **Condição de confidencialidade**:

- o Apenas o receptor **B** deve ser capaz de recuperar o texto **m** a partir do criptograma **{m}**

?? **Códigos**:

- o A **comunicação segura** exige uma **fase prévia de codificação** através de uma **função invertível** que converte textos em números inteiros (códigos)
- o O **processo de codificação** (e o seu inverso, **descodificação**) consideram-se **fora do sistema seguro**

X = espaço de códigos

- o **Cod: M ✍ X** função de codificação
- o **Cod-1: X ✍ M** função de descodificação
- o As **funções** de **codificação** e **descodificação** constituem uma **mudança de representação** da informação e são publicamente conhecidas
- o Sob o ponto de vista da segurança, **são irrelevantes**

?? **Noção Geral de Cifra**

- *X* = códigos do texto
- *Y* = códigos do criptograma
- **c**: *X* ✍ *Y* *função de cifragem*
- **d**: *Y* ✍ *X* *função de decifragem*
- **Condição de Correcção**:
  - ✍**d**(**c**(*x*)) = *x* ? *x* ? *X*
- **Não inversão**:
  - ✍Dado qualquer ***y*** ? ***Y*** é **computacionalmente intratável**, sem conhecimento de **d**, encontrar *x* tal que **c**(*x*) = *y*
- **Confidencialidade**:
  - ✍Apenas *B* conhece **d**(.)
  - ✍{*x*} = **c**(*x*)

?? Os códigos *x* ? *X* chamam-se **textos claros**, simplesmente **textos** ou **mensagens**

?? Os códigos *y* ? *Y* designam-se **criptogramas**

?? **c**(.) designa-se **função de cifragem**

?? **d**(.) designa-se **função de decifragem**

?? A **segurança contra intrusão** é feita através do controlo do conhecimento sobre as funções **c**(.) e **d**(.)

?? Estas funções dependem de parâmetros, designados por **chaves**

?? O controlo do conhecimento é feito escolhendo funções de cifragem e decifragem que, apesar de terem uma estrutura conhecida, só são completamente determinadas quando for conhecida a **chave** (ou as chaves)

?? Surge assim um novo espaço de códigos **R**

?? Os elementos $k \in R$ designam-se por **chaves**, que **parametrizam** as funções de cifragem e decifragem

?? c_k: $X \rightarrow Y$ d_k: $Y \rightarrow X$, para todo $k \in R$

?? Existem actualmente duas grandes classes de cifras:

- **Cifras Simétricas**: a mesma chave é usada para cifrar e decifrar
- **Cifras Assimétricas**: a cifragem e a decifragem usam chaves distintas

3.5.3. Cifras simétricas ou de chave-privada

?? Nas cifras simétricas, é usada a **mesma chave** $k \in K$, tanto para cifrar como para decifrar

?? c_k: $X \rightarrow Y$, $k \in K$ funções de cifragem

?? d_k: $Y \rightarrow X$, $k \in K$ funções de decifragem

?? **Condição de correcção**:

- $\boldsymbol{d_k(c_k(x)) = x}$ para todo $x \in X$ e $k \in K$

?? **Condições de segurança**:

- O conhecimento de $\boldsymbol{y = c_k(x)}$ não permite recuperar **k**, mesmo conhecendo **x**
- O conhecimento de $\boldsymbol{y = c_k(x)}$ não permite recuperar **x** sem o conhecimento de **k**

?? **Não inversão**:

- (não inversão da chave) Conhecidos $x \in X$ e $y \in Y$, é **computacionalmente intratável encontrar** $k \in K$ tal que $y = \boldsymbol{c_k(x)}$
- (não inversão do criptograma) Conhecido $y \in Y$, é **computacionalmente intratável determinar** $x \in X$ e $k \in K$ tal que $y = \boldsymbol{c_k(x)}$

?? Todas as cifras simétricas modernas pertencem a uma de duas **categorias**:

- **Cifras sequenciais**
- **Cifras por blocos**

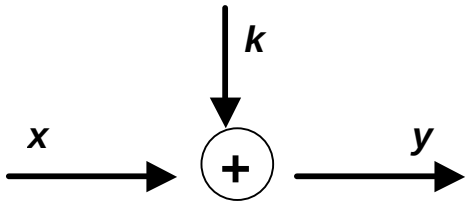
3.5.3.1. Cifras sequenciais

?? Têm por finalidade **cifrar sequências** ("*streams*") de bits em **tempo real**, normalmente derivadas de sinais de **vídeo**, **áudio**, etc

?? São quase **exclusivamente implementadas em hardware**, já que têm de ser muito **eficientes computacionalmente**

?? A **cifragem** é feita **bit a bit**, de modo diferente em cada bit

?? O espaço de códigos utilizado é $\mathbf{Z_2}$



?? A forma mais simples de **cifragem** de bits recorre apenas à adição com a chave

?? Supondo a cifragem de um único bit:

- $\mathbf{c_k(x) = x + k}$ $\quad$ $\mathbf{x,y, k \in Z_2}$

?? A **decifragem** é igual:

- $\mathbf{d_k(y) = y + k}$

?? Note-se que

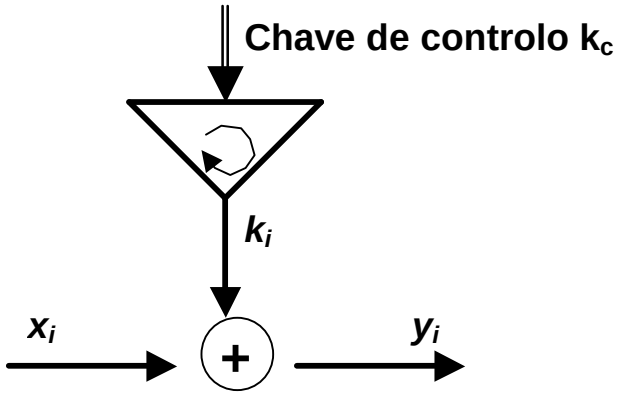
- $\mathbf{d_k(c_k(x)) = (x + k) + k = x}$, porque em $\mathbf{Z_2}$ $\mathbf{k + k = 0}$

?? Quando se quer cifrar não apenas um bit, mas uma sequência de bits $x_1, x_2, x_3, ..., x_i, ...$ temos de possuir uma **sequência "paralela" de chaves** $k_1, k_2, k_3, ..., k_i, ...$

?? O criptograma resultante é dado por $\mathbf{y_i = x_i + k_i}$

?? Portanto, a **essência** da cifra **está** na **geração da sequência de chaves** $k_1, k_2, ...$

?? Esta sequência é gerada por um algoritmo apropriado (o **gerador de chaves**), que é controlado por uma **chave de controlo k_c**



?? As várias cifras sequenciais **distinguem-se** apenas pela **forma do gerador de chaves**

?? Um **gerador** que gera uma **repetição infinita** de uma mesma secção de bits **diz-se periódico**

?? **Geradores periódicos** (ou de período curto) são **muito inseguros**

?? Isto porque **é possível** a um **intruso obter** num período (ou em dois períodos, dependendo da informação disponível) **a secção de chaves que é repetida**, e, a partir daí, determinar toda a restante sequência de chaves

?? Outra **condição essencial de segurança** é a **não invertibilidade** das chaves:

- A partir do **conhecimento da sequência** de chaves k_1, k_2, k_3, ..., k_i, ... (que se obtém facilmente depois de ter sido usada) **não deve ser possível recuperar a chave** de controlo **k_c** que determina toda a sequência
- A **chave k_c** permite ao receptor gerar a sequência k_1, k_2, k_3, ..., k_i, ... e recuperar o texto

3.5.3.2. Cifras por blocos

?? **Percorrem blocos de bits** de tamanho fixo, usando a mesma função de cifragem em cada um dos blocos

?? São normalmente **implementadas em software**

?? Tamanho de bloco mais frequente: **64 bits**

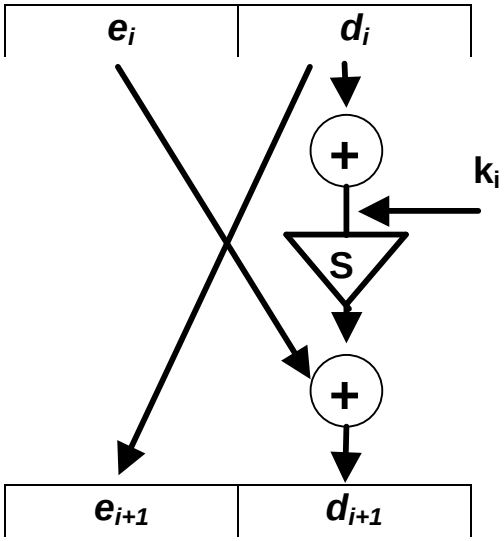
?? Os algoritmos deste tipo são essencialmente constituídos por uma repetição de operações de "**mistura de bits**", em que os **resultados de uma determinada iteração** são **misturados** com bits provenientes da chave para **produzir os bits da iteração seguinte**

?? Cada iteração é denominada ***round*** da cifra

?? Um exemplo típico destas cifras são os **Circuitos de Feistel**

?? Nestes circuitos cada bloco é dividido em dois sub-blocos iguais (de 32 bits, neste caso)

?? Os sub-blocos são "**trocados e misturados**" em cada round



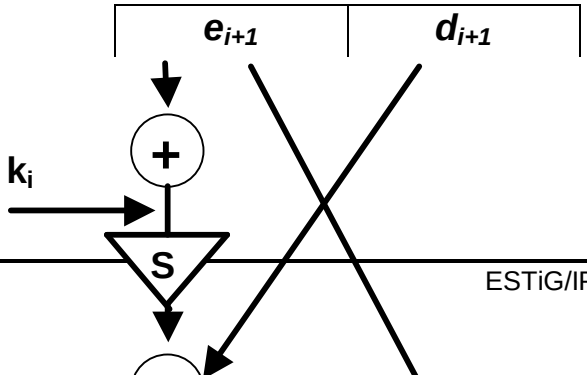
?? Ou seja:

o $$\begin{cases} e_{i+1} = d_i \\ d_{i+1} = e_i \oplus s(d_i \oplus k_i) \end{cases}$$

o sendo **s(.)** uma **transformação não linear** de bits, que determina as propriedades da cifra

?? Principal **vantagem** destas cifras: a **simplicidade da função de cifragem**

?? Assim, conhecido o $\mathbf{k_i}$, esta transformação inverte-se facilmente:



- $e_i = d_{i-1} \oplus s(e_{i-1} \oplus k_i)$
  $d_i = e_{i-1}$

  - é essencialmente a **mesma função que a cifragem**, mas antecedida e continuada com uma troca das metades esquerda e direita

- Enquanto que **um round** desta forma é **relativamente fraco** em termos de segurança, **vários rounds** seguidos tornam estas cifras **extremamente fortes**, porque vão **misturando** as duas metades, em cada round, de uma **forma não linear**

- Um exemplo típico destas cifras é o **DES** – *Data Encryption Standard*, desenvolvido no início dos anos 80

- Trata-se de uma **cifra standard**, **usada** actualmente em quase todas as **transacções financeiras**

- O DES usa **16 rounds**

- Um **problema** típico de qualquer cifra simétrica é o **tamanho da sua chave**

- Este vai condicionar o **limite máximo** do esforço para um **ataque de força bruta**

- O DES tem **56 bits** úteis na sua chave

- Actualmente uma **chave de 56 bits** é considerada **demasiadamente pequena**

- Com as **máquinas actualmente existentes**, é possível fazer **ataques de força bruta** em que $\mathbf{2^{56}}$ tentativas são consideradas **fazíveis**

- As **cifras mais recentes** usam chaves de tipicamente **128 bits**

- Por **força bruta**, um **ataque** a uma **chave de 128 bits** é considerado actualmente um **esforço computacionalmente intratável**

- Uma cifra mais recente que usa **128 bits** é o **IDEA** – *International Data Encryption Algorithm* (usada no **PGP** – *Pretty Good Privacy*)

- Usa **8 rounds** que processam **blocos de 64 bits**, divididos em **4 sub-blocos de 16 bits**

- **Outra classe de cifras** por blocos são as da **família RC5**

- Nestas, cada round usa as **operações** de "***shift de bits***" **para obter** as desejadas **não linearidades**

- O **tamanho do bloco** e da **chave** é um **múltiplo qualquer de 32 bits**

- Recentemente, tem vindo a ser discutida uma **nova família** de **Circuitos de Feistel**, de que são **exemplos**:

  - **CAST**: blocos e chave de 64 bits

  - **Blowfish**: 64 bits de bloco e chave de tamanho não limitado

- **Características** das cifras por blocos:

  - São caracterizadas por um **tamanho de bloco** e um **tamanho de chave** que **determina** o **esforço limite** num ataque por força bruta

  - São **formadas** por uma **repetição de vários rounds**, em que cada um pode ser inseguro mas a combinação de vários rounds torna a cifra segura

  - São **baseadas** em operações de "**troca e mistura de bits**", que pode ser feita muito facilmente de forma muito eficiente (em software ou mesmo em hardware)

  - A **decifragem** é normalmente uma **pequena variação** sobre a função usada na **cifragem**

3.5.4. Cifras assimétricas ou de chave-pública

?? A cifragem e decifragem usam **chaves distintas**

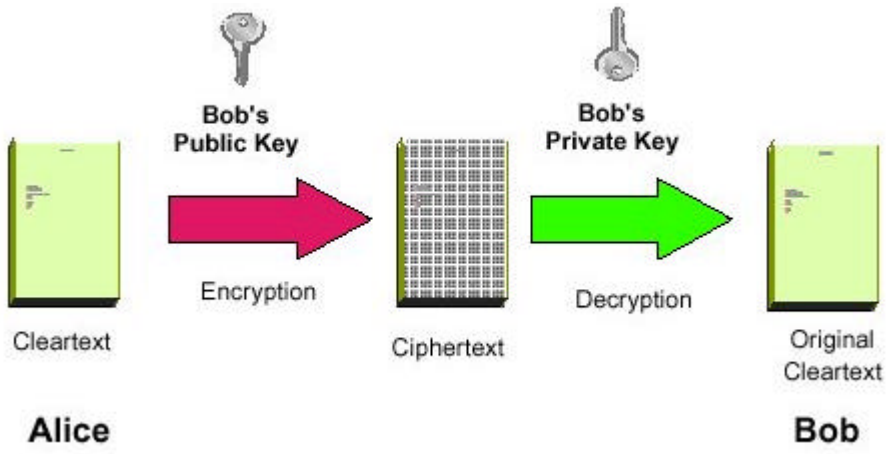


Figura 2.28 – Cifragem usando Chaves Públicas

?? A **chave pública**, como o próprio nome indica, é **conhecida** por toda a gente

?? A **chave privada** é mantida em **segredo** pelo seu dono

?? A chave privada **não pode ser determinada** a partir da chave pública

?? Em vez de "troca de bits" são usadas funções de cifragem e decifragem baseadas em **propriedades matemáticas dos números naturais**

?? Sendo necessário **relacionar matematicamente** as chaves públicas com as privadas (sem fornecer informação privada), a **relação** entre estas entidades **assume a forma** de "**one way functions**"

?? Isto é, **funções invertíveis**, que são **facilmente calculáveis numa direcção**, mas **dificilmente calculáveis na direcção oposta**

?? **Exemplos de funções deste tipo**:

1. **Factorização** de grandes números:
    - Dados dois grandes números primos **p** e **q**, o seu produto **p.q** é facilmente calculável

- Inversamente, dado o produto **m = p.q** (desconhecendo-se os factores **p** e **q**) sabemos que existe uma **única factorização** de **m** que **determina** estes factores
- **Computacionalmente**, porém, se **m** for **suficientemente grande** é **impossível** determinar esses factores

2. Números em $\mathbf{Z_p}$ com **p** primo:
   - Sabe-se que nestas circunstâncias, todos os elementos não zero de $\mathbf{Z_p}$ são invertíveis
   - Costuma-se representar por $\mathbf{Z^*_p}$ o conjunto de todos os elementos invertíveis de $\mathbf{Z_p}$ equipados com a operação de multiplicação
   - Sabe-se também que escolhendo apropriadamente um número ? , chamado **gerador**, as diversas potências

     $? ^{0}, ? ^{1}, ? ^{2}, \ldots ? ^{p-2} \mod p$

     geram **todos** os elementos de $\mathbf{Z^*_p}$
   - Computacionalmente a **exponenciação** $\mathbf{exp_{?,p}}$ é **facilmente calculável**
   - Existe um **algoritmo**, cuja complexidade é proporcional ao número de bits usado na representação do argumento **x**, que **calcula** $\mathbf{exp_{?,p}(x)}$ muito eficientemente
   - Em contrapartida, para uma escolha apropriada de **p**, o **logaritmo** é **computacionalmente intratável**
   - Assim,

     dado $? = ? ^{a} \mod p$,

     dados ? e **p**,
   - **é impraticável determinar a mesmo** sabendo que ele existe

?? A partir destas duas formas de “one way functions” constroem-se os **sistemas assimétricos** usados nas cifras deste tipo

?? Uma **cifra assimétrica** é **formada** pelo par de funções:

- **$C_k : X \rightarrow Y$**
- **$D_{k'} : Y \rightarrow X$**

?? Deve obedecer à **condição de correcção**:

- (1) $\forall k \in_1 k' : \forall x \in X : D_{k'}(C_k(x)) = x$

?? e às **condições de segurança**:

- (2) conhecido **k** deve ser **impossível** determinar **k’**
- (3) conhecido **k** e **$C_k(x)$** deve ser **impossível** determinar **x** sem conhecer **k’**

?? A condição **(1)** garante a **invertibilidade** da cifra

?? A condição **(2)** diz-nos que **não é possível recuperar a chave privada** a partir da chave pública

?? A condição **(3)** diz-nos que conhecendo o criptograma **$C_k(x)$** e a chave com que foi cifrado, **não deve ser possível recuperar a mensagem original**

?? Uma **propriedade** interessante dos algoritmos de chave pública é a possibilidade de **implementação** de **mecanismos** de **autenticação** e **não-refutação**

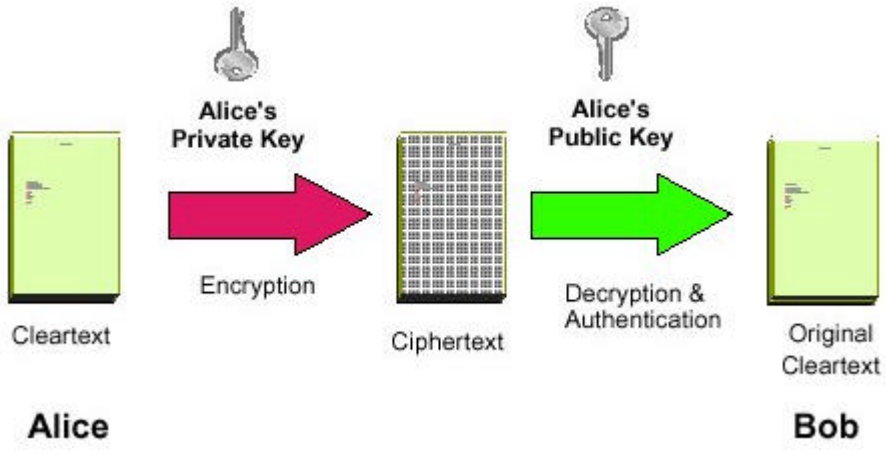


Figura 2.29 – Autenticação através da cifragem com a chave privada

?? Usando a **chave privada**, podemos **autenticar** uma mensagem, já que apenas o seu legítimo dono pode utilizar esta chave

?? Como qualquer pessoa pode ter acesso à **chave pública**, pode assim fazer a **verificação da assinatura** com esta

?? As principais **cifras assimétricas** actualmente em uso são:

- **RSA** (nome resultante das iniciais dos seus três inventores: *Ron **R**ivest, Adi **S**hamir* e *Leonard **A**dleman*)
- **El Gamal**
- **Diffie-Helman**

3.5.4.1. Cifra RSA

?? **Parâmetros da cifra RSA**:

- Escolhem-se dois números primos **p** e **q** grandes
- Calculam-se:

  ✍ **m = p.q** ? **= (p-1) . (q-1)**

- e **destroem-se** os números iniciais **p** e **q**
- **m** chama-se **módulo** da cifra
- **escolhe-se** um número **a** invertível em $\mathbf{Z_0}$ e calcula-se

  ✍ **b= a^{-1} mod** ?

- **destroi-se** ?
- A **chave pública** é o par **<m, a>**
- A **chave privada** é o par **<m, b>**
- A **função de cifragem** é dada por:

✍ **$c(x) = x^a \bmod m$**

- A **função de decifragem** é dada por:

  ✍ **$d(y) = y^b \bmod m$**

?? Note-se que **p**, **q** e ? foram **destruídos** e desconhecidos por todos

?? Os valores de **m** e **a** são públicos e o valor de **b** é conhecido pelo dono da chave

?? **Correcção do RSA** - deriva de um teorema que diz:

- **$a \cdot b = 1 \bmod (p-1).(q-1)$** implica
- **$x^{a.b} = x \bmod p.q$**, se **p** e **q** forem **primos**

?? **Condições de Segurança**:

- Conhecida a chave pública **<m, a>** para determinar **$b= a^{-1} \bmod$** ? é necessário conhecer ?

  Como a única informação sobre os números **p** e **q** originais provêm de **m** (que é igual a **p.q**), para determinar ? **= (p-1).(q-1)** precisávamos de determinar **p** e **q** através da factorização de **m**

  Computacionalmente isso **não é possível**, se **m** for **bem escolhido**

- Conhecido **<m, a>** e **$x^a \bmod m$** não é possível recuperar a base **x** desta exponenciação

?? O **RSA** não é atacável directamente, mas pode ser atacável por protocolos que levem o dono da cifra a ser ludibriado de modo a fornecer a sua chave privada

?? Os **principais ataques** deste tipo são:

- Ataque por **módulo comum**
- Ataque por **falso criptograma**

?? Alguns **conselhos** para evitar estes ataques:

- A **ambiguidade** decifragem/assinatura é uma má opção
- Nunca assinar directamente uma mensagem que nos é dada sem colocar nenhuma marca que evite uma decifragem a partir da assinatura

3.5.4.2. Cifra El Gamal

- **Parâmetros da cifra**
  - Escolhe-se um primo **p** grande, para o qual o problema da determinação do logaritmo discreto em $\mathbf{Z^*_p}$ seja intratável
  - Escolhe-se um gerador ? , um **a** ? $\mathbf{Z_{p-1}}$ e calcula-se

    ✍? **= ? ^{a} mod p**
  - A **chave pública** é o tripo **<p, ? , ? >**
  - A **chave privada** é **a**
- A **função de cifragem** é comandada por um parâmetro aleatório **w** (uma **chave de sessão**):
  - É gerado **w** ? $\mathbf{Z_{p-1}}$
  - Calcula-se

    ✍**h = ? ^{w} mod p** **s = x.? ^{w} mod p**
  - e destroi-se **w**
- O **criptograma** é o par **<h, s>**
- **Função de cifragem**:
  - **c(x) = <? ^{w} mod p, x.? ^{w} mod p>**
- **Função de decifragem**:
  - **d(<h, s>) = s$(h^a)^{-1}$ mod p**
- **Correcção da cifra**:

- **$d(<h, s>) = x.? ^{w}(? ^{a.w})^{-1} \bmod p$**

  $= x.(? ^{a.w}).(? ^{a.w})^{-1} \bmod p$

  $= x \bmod p$

?? **Condições de segurança**:

- Conhecendo **<p, ? , ? >** a determinação de **a** tal que **? = ? ^{a} mod p** equivaleria ao cálculo do algoritmo discreto
- Conhecido **<h, s>**, desconhecendo **w**, é impossível determinar **w** a partir de **h** (o logaritmo discreto de novo) e sem **w** não é possível calcular **? ^{w}** para recuperar **x** a partir da relação **s = x. ? ^{w} mod p**

3.5.4.3. Cifra Diffie-Hellman

?? Na fase inicial de uma **sessão de negociação de chaves** não estamos na presença de um canal seguro

?? O **algoritmo Diffie-Hellman** é usado nesta fase de troca de chaves secretas

?? **Funcionamento do algoritmo**:

- Ambas as partes partilham o **módulo de m** e um valor inteiro **g**
- **m** deve ser um **número primo** grande
- O **emissor** gera o **número aleatório** grande **a** e calcula:

  ✍ **$x = g^{a} \bmod m$**

- O **receptor** gera um **número aleatório** grande **b** e calcula:

  ✍ **$y = g^{b} \bmod m$**

- O emissor envia **x** para o receptor
- O receptor calcula:

$$K1 = x^b \bmod m$$

- e envia **y** para o emissor
- Este calcula $K2 = y^a \bmod m$

- **K1** e **K2** são iguais a $g^{ab} \bmod m$, que é a **chave secreta**
- Não é possível gerar este valor sem conhecer **a** ou **b**
- A **segurança da troca** é baseada na **extrema dificuldade de inversão da exponenciação** (cálculo do logaritmo discreto)

3.5.5. Cifras Simétricas versus Cifras Assimétricas

- **Cifras assimétricas**:
  - Só a **chave privada** precisa de ser mantida **confidencial**
  - A **gestão das chaves públicas** de uma comunidade de *N* agentes pode ser feita numa base de dados aberta, requerendo apenas a certificação das chaves por um **agente de confiança** (*"trusted agent*").

    Cada agente individualmente só se tem de preocupar com a **gestão de um única chave**: a sua chave privada
  - A **complexidade computacional** é várias ordens de grandeza **superior** à das cifras simétricas

    O **tamanho das chaves**, a **complexidade dos algoritmos**, a **complexidade de criação das chaves** são sempre **superiores** aos parâmetros equivalentes nas cifras simétricas
  - **Não é fácil alterar** um par de chave pública e privada; estas chaves tendem a ser **chaves de longa duração**
- **Cifras simétricas**:
  - Numa comunicação entre dois agentes, a chave deve ser mantida **confidencial** em ambos os extremos

- Numa comunidade de *N* agentes cada **par de agentes** necessita de uma **chave própria**

  São necessários globalmente **N x (N – 1)** chaves distintas

  Cada agente, individualmente, tem de garantir o segredo para **(N – 1)** dessas chaves

- Numa comunicação entre dois agentes é **possível** (recomendável) **alterar** a chave frequentemente

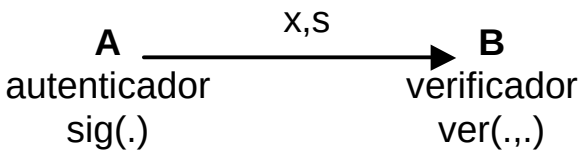
  Possivelmente uma chave só deve ser usada numa única comunicação; nesse caso chama-se uma **chave de sessão**

- As cifras simétricas são, **computacionalmente**, **extremamente eficientes** e a sua segurança pode ser garantida com **chaves substancialmente menores** do que as usadas nas cifras assimétricas

3.5.6. Assinaturas Digitais

?? Para garantir a autenticidade das comunicações usam-se **assinaturas digitais**

?? O **sistema de autenticação** é constituído por dois agentes: o **autenticador** (A) e o **verificador** (B)



?? **A** envia para **B** um texto **x** juntamente com informação adicional **s** chamada **assinatura**

?? **B** recebe o texto **x** e a assinatura **s** e deve ser capaz de autenticar o par **<x, s>** como proveniente de **A**

?? O verificador **B** pretende **assegurar-se**:

(1) O texto **x** que ele recebe **coincide** com o texto que foi enviado por **A**

(2) Apenas **A** tem **informação suficiente** para enviar o par **<x, s>** recebido por **B**

?? **Esquemas** de assinaturas **mais complexos asseguram** ainda:

(3) Uma vez recebido **x** por **B**, o autenticador **não pode negar** a terceiros, posteriormente, que realmente o **autenticou**

(4) Se o texto **x** foi enviado com uma função específica, **B** deve ser capaz de o distinguir de uma sua cópia, usada noutra função

?? Sob o ponto de vista do agente **A**, o autenticador, deve-se **garantir**:

(1) O verificador **B**, ao receber **<x, s>**, não deve adquirir informação que lhe permite assinar futuros textos como se fosse **A**

?? Adicionalmente, os **esquemas mais complexos permitem**:

(2) Na presença de um **texto que não autenticou**, **A** deve ser capaz de **provar** este facto a terceiros, sem violar as outras condições de segurança

(3) O **autenticador** deve poder **distinguir** entre um texto que assinou e uma sua cópia

?? Os **esquemas simples de assinaturas** são feitos para garantir as **condições (1)** e **(2)** do verificador e a **condição (1)** do autenticador

?? Por isso, o autenticador conhece uma **função de assinatura sig(.)** que determina:

- **s = sig(x)**

?? e o verificador conhece um **predicado de verificação** binário **ver(.,.)**, de tal modo que:

(a) ? **x,s . (s = sig(x)) sse ver(x,s)**

(b) A partir do conhecimento de **ver(.,.)** não é possível reconstruir a função **sig(.)**

- A função **sig(.)** é **privada** de **A** enquanto que **ver(.,.)** é **pública**
- Principais **esquemas de assinaturas** actuais:
  - **RSA**
  - **El Gamal**
  - **DSA (Digital Signature Algoritm)**

3.5.6.1 DSA (*Digital Signature Algoritm*)

- O **El Gamal** necessita de módulo de **p** de pelo menos **512 bits** e exige um **poder computacional** que não é compatível com a sua utilização em **smart-cards**
- É possível ter algo **tão seguro** como o El Gamal usando uma variante deste que usa uma aritmética de **160 bits**
- **Parâmetros**:
  - Chave pública: **<p, q,** ? **,** ? **>**
  - Chave privada: **a**
  - com ? **=** ? **^{a} mod p**
  - sendo **q** um **número primo** de 160 bits, divisor de **p – 1**
- **Assinatura**:
  - textos **x** ? **Z_q**
  - **chave de sessão w** ? **Z^*_q**
  - calcula-se:
    - **h =** ? **^{w} mod q**
    - **$s = w^{-1}.(x + a . h) \bmod q$**
  - e destroi-se **w**
  - **sig(x) = <h,s>**

- **Verificação**:
  - **ver(x, <h, s>)**
  - calcula-se:
    - **u = x . s^{-1} mod q**
    - **v = h . s^{-1} mod q**
    - **? = $?^{u} \cdot ?^{v}$ mod p**
  - e testa-se:
    - **ver(x, <h, s>) = (? = h mod q)**

3.5.7. Funções de *Hash*

- As **Funções de Hash** (também conhecidas como *message digests*) são fundamentais na criptografia, possibilitando a **implementação** de mecanismos de **assinaturas digitais**
- São **funções** que recebem dados de tamanho variável e produzem dados de saída com um tamanho fixo
- Estes dados de saída podem ser vistos como uma **impressão digital** (*fingerprint*) dos dados originais
- Se o resultado da **aplicação** de uma **função de hash** a duas mensagens condizer, temos elevada certeza de que **essas mensagens são a mesma**
- Uma função de hash **mapeia** um conjunto de **input** para um conjunto de **output mais pequeno**
- Isto leva à possibilidade da **existência de colisões**, ou seja, **mensagens diferentes** podem dar origem a uma **mesma** *fingerprint*
- As Funções de Hash devem ser **suficientemente fortes**, por forma a evitar estas situações

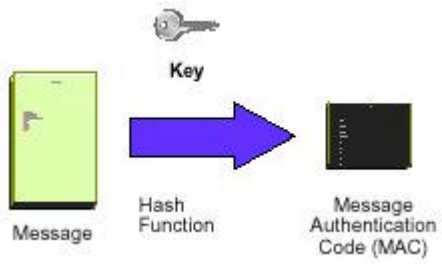


Figura 2.30 – Aplicação de uma Função de Hash

?? Estas funções são usadas essencialmente em **técnicas de verificação da integridade e de autenticação**:

- o O emissor **calcula** a *hash* da mensagem e **acrescenta-a** à mensagem a enviar
- o O receptor **calcula** a *hash* da mensagem recebida e **compara-a** com a *hash* que lhe foi enviada
- o Se as *hashes* forem **iguais**, então a mensagem **não foi alterada**

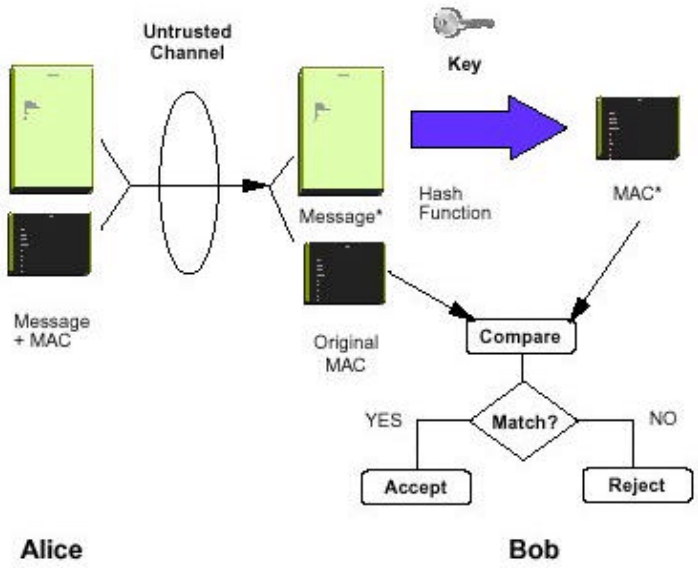


Figura 2.31 – Verificação de Integridade e Autenticidade

?? Como vimos anteriormente, a **verificação de integridade e de autenticidade** já é fornecida pela **cifragem**

?? No entanto, muitas vezes **apenas precisamos** de assegurar a **integridade** e/ou a **autenticação**, não necessitando da cifragem da totalidade dos dados a transmitir

- Como a **cifragem** consome muitos **recursos computacionais**, com as Funções de Hash garantimos a Integridade/Autenticação com **menos recursos**

- A **cifragem** de uma *Hash* com a **chave privada** dá origem a uma **Assinatura Digital**

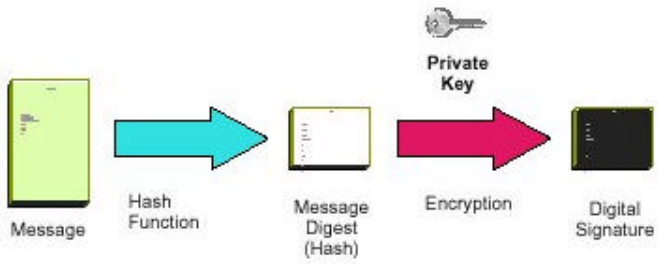


Figura 2.32 – Geração de uma Assinatura Digital

- A **cifragem** de **chaves secretas** com a chave pública denomina-se **Envelope Digital**

- Esta **técnica** é usada habitualmente para efectuar distribuição de chaves secretas para algoritmos **simétricos**

- Principais **exemplos** de **Funções de Hash**:

  - **MD5**:

    - desenvolvido pelo co-autor do RSA, Ron Rivest

    - produz uma Hash de **128 bits**

  - **SHA-1** (*Secure Hash Algoritm 1*)

    - inspirado no MD5

    - eleito pela NSA (*National Security Agency* Americana) para utilização com o DSS – *Digital Signature Standard* (que utiliza o algoritmo DSA, abordado atrás)

    - produz uma Hash de **160 bits**

    - considerado **mais seguro** que o MD5, devido ao tamanho das Hashes que produz

3.5.8. Autoridades de Certificação e Certificados Digitais

- Com **criptografia de chave pública** as partes de uma conversação **partilham** entre si as respectivas chaves públicas

- Se um **intruso** consegue **alterar** uma **chave pública** com a sua ou outra chave pública (ataque *man-in-the-midle*), pode **aceder** a **informação confidencial**

- Um **intruso** pode ainda **fazer-se passar por outro** e conseguir assim **acesso a informação** que de outra forma não lhe estaria acessível

- A **solução** para estas ameaças de segurança são os **Certificados Digitais**

- Um certificado digital **associa** uma **identidade** a uma **chave pública**

- Para se **garantir** a **confiança** nestes certificados, os mesmos devem ser **validados** por uma **Autoridade de Certificação** - **CA** (*Certification Authority*)

- Um certificado digital é **assinado** com a chave privada de uma autoridade de certificação, logo pode ser autenticado

- Esta **assinatura** é efectuada depois da **verificação** do **pretendente ao certificado**

- Além da **chave pública** e da **identificação**, um certificado digital contem ainda **outras informações**:

    - Data de Emissão
    - Data de Fim de Validade
    - Informação variada do CA

- Com a utilização dos certificados digitais, as partes têm **confiança** nas chaves públicas dos donos desses certificados, já que a sua **autenticidade** é **garantida** por uma **terceira entidade de confiança**

- Protocolos ISO **X.509**: Standard internacional para certificados digitais

3.6. Firewall's e Proxy's

?? Uma **Firewall** é um **sistema**, ou grupo de sistemas (baseado em Hardware ou Software), que **controla os acessos** entre uma rede interna segura e a rede externa, tipicamente insegura (p.e. a Internet)

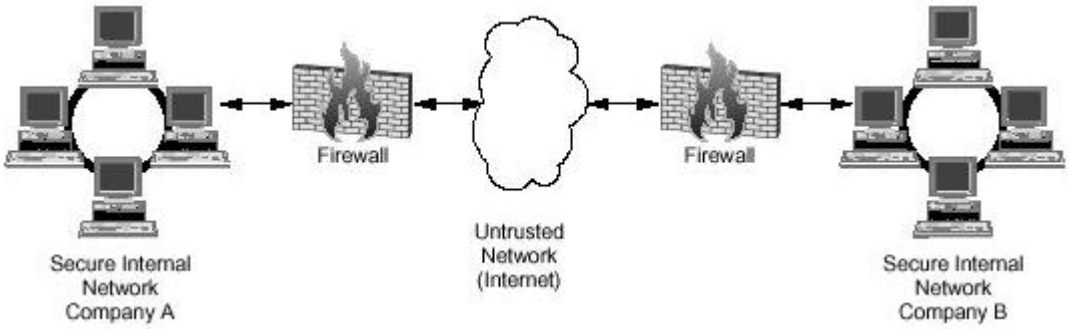


Figura 2.33 – Exemplo de utilização de Firewall's

?? Geralmente são utilizadas como **meio de protecção** entre as redes privadas e a Internet, **posicionando-se na fronteira** entre as duas redes

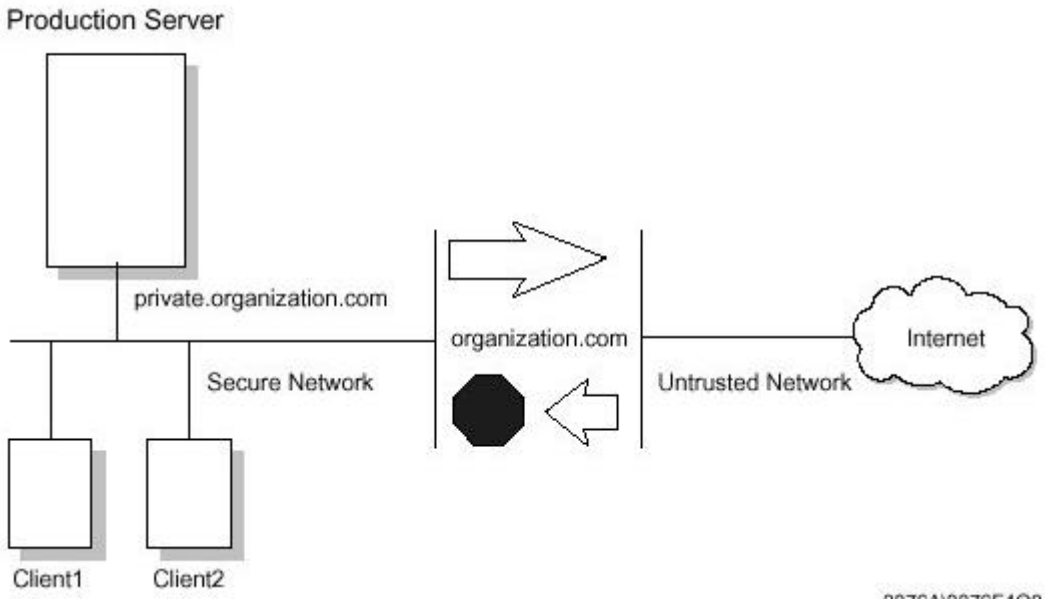


Figura 2.34 – Enquadramento da Firewall na Rede

- Uma **Firewall** pode ser um **PC**, um **Mainframe**, uma **Estação de Trabalho UNIX**, um **Router** ou uma combinação destes
- Dependendo dos requisitos, uma firewall é **constituída** por um ou mais dos seguintes **componentes funcionais**:
  - *Router* de filtragem de pacotes
  - *Application-Level Gateway* (Proxy)
  - *Circuit-Level Gateway*

3.6.1. Routers de Filtragem de Pacotes

- Normalmente estas **funções de filtragem** são **implementadas no router** que desempenha as funções de encaminhamento entre a rede da organização e a rede exterior
- Trata-se do **primeiro nível de defesa** da rede interna
- É efectuada a **análise do cabeçalho** dos pacotes que chegam ao router, e de seguida são **tomadas decisões** (pacote **reencaminhado** ou **descartado**) de acordo com as **regras de filtragem** definidas
- A **análise do cabeçalho** do pacote é baseada nas seguintes **informações**:
  - Endereço IP de Origem
  - Endereço IP de destino
  - Porto TCP/UDP de origem
  - Porto TCP/UDP de destino
  - Tipos de mensagens ICMP
  - Informação de protocolos encapsulados (TCP, UDP, ICMP ou túneis IP)

?? As **definição das regras de filtragem** de pacotes é baseada na **política de segurança** da rede

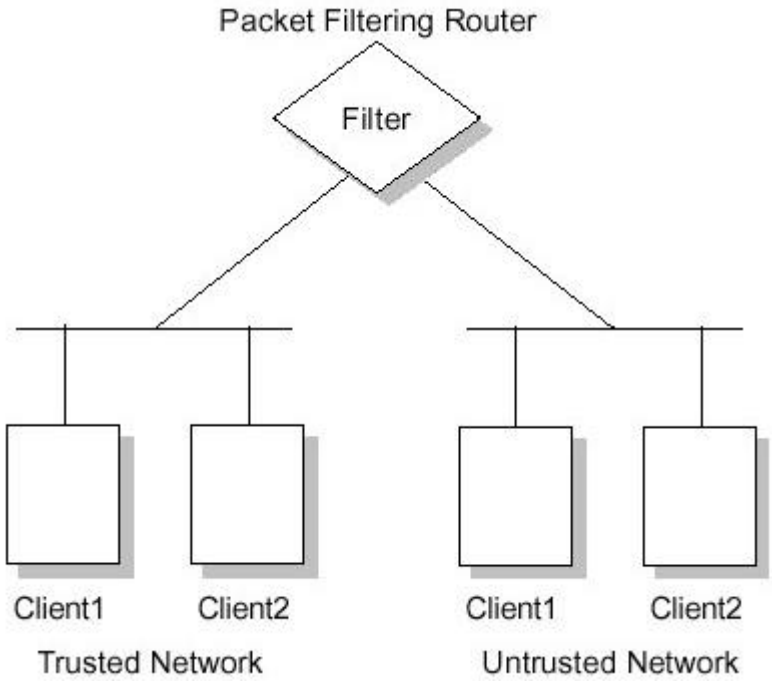


Figura 2.35 – Router de Filtragem de Pacotes

3.6.2. Application Level Gateway's (Proxy's)

?? Um **proxy** fornece elevado **controlo do tráfego aplicacional** entre duas redes

?? Para **cada aplicação** que se deseja aceder através do **proxy**, é necessário **instalar** neste o respectivo **suporte para esse serviço**

?? Um proxy **actua como servidor** para o cliente, e **como cliente** para o servidor de destino

?? É criada uma **conexão virtual** entre o cliente e o servidor de destino

?? Sendo **transparente** para o cliente e o servidor, o proxy deve ser capaz de **monitorizar e filtrar** qualquer tipo específico de dados, antes de os enviar até ao destino

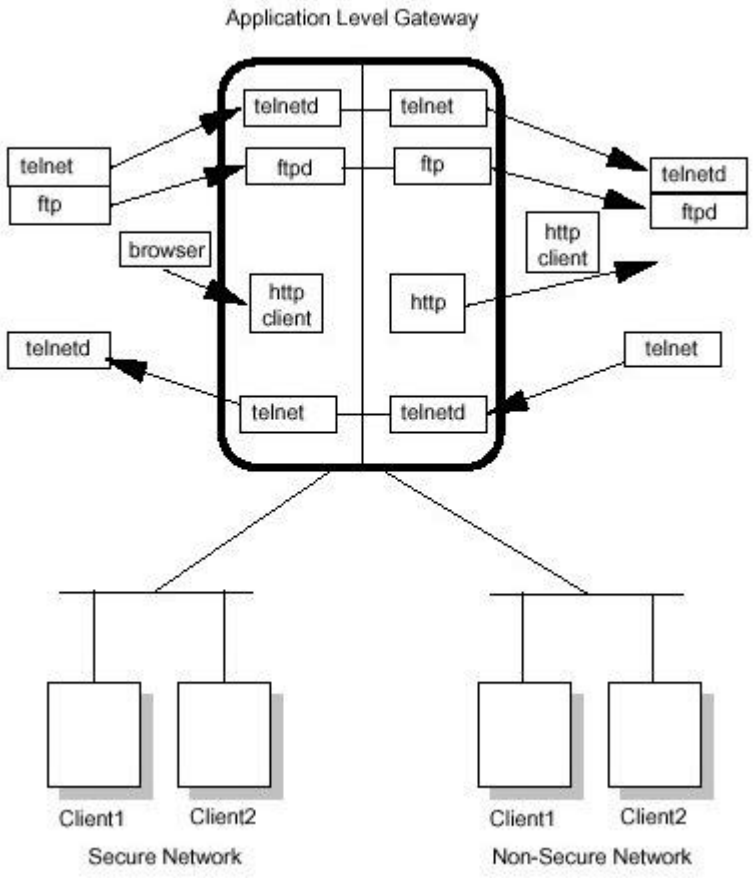


Figura 2.36 – *Application Level Gateway*

?? Por **exemplo**, um Servidor de FTP pode estar acessível a partir do exterior

Com o objectivo de o proteger contra ataques, o proxy FTP pode ser configurado para negar os comandos PUT e MPUT

?? Um servidor de Proxy é um ***relay application-specific***, que normalmente funciona no host que interliga uma rede segura com outra rede insegura

?? O seu **objectivo** é **controlar a troca de dados** entre as redes que conecta, **ao nível da camada de aplicação**, em vez da camada de rede

?? Para um **cliente** poder utilizar um Servidor de Proxy, tem de **suportar** o uso de proxy's e ser especialmente **configurado** para o utilizar

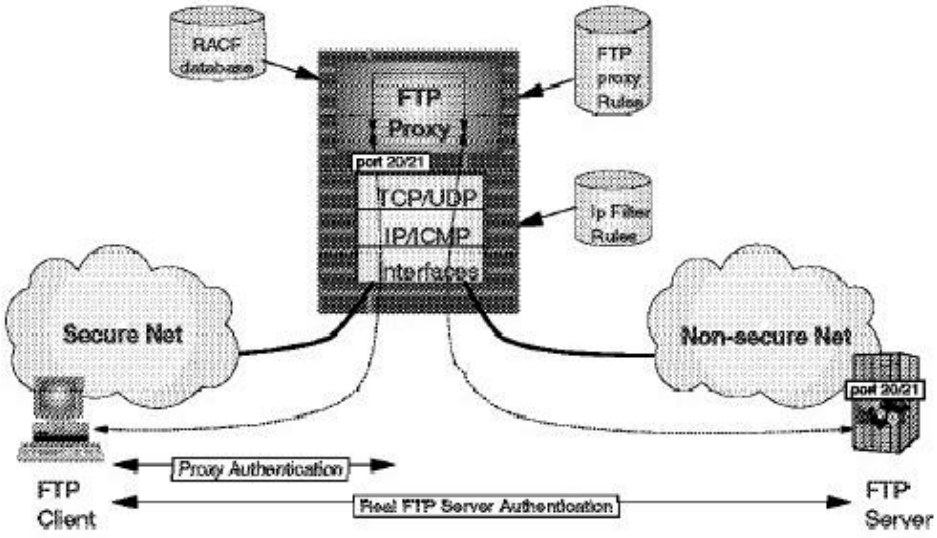


Figura 2.37 – Servidor de Proxy FTP

- **Comparativamente** com a **filtragem IP**, os proxy's fornecem **registos de actividade** muito mais compreensivos, baseados nos dados da aplicação
- Por exemplo, um **proxy HTTP** pode efectuar o **registo dos URL's** visitados pelos utilizadores

3.6.3. Circuit Level Gateway's

- Um **Circuit Level Gateway filtra/reencaminha** conexões TCP e UDP, **sem fornecer** qualquer **processamento** ou **filtragem extra**
- Pode ser visto como um **tipo especial** de Application Level Gateway's

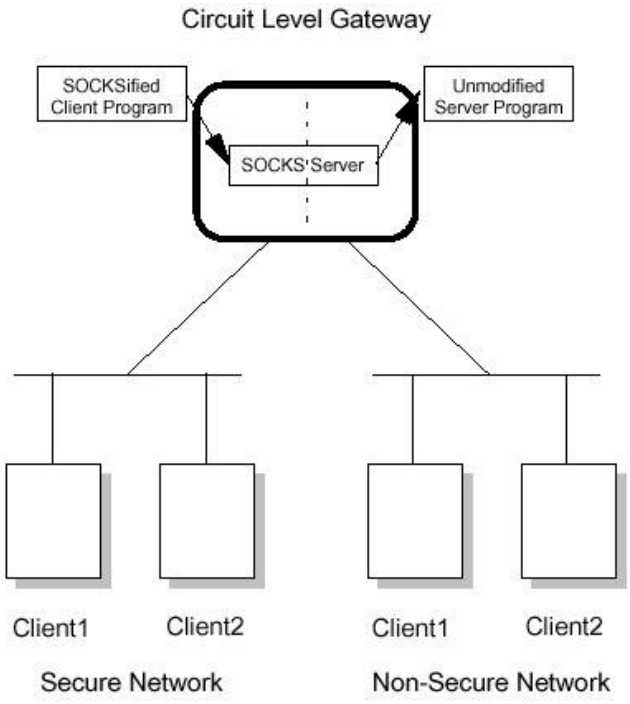


Figura 2.38 – Circuit Level Gateway

?? As principais **diferenças** entre os dois tipos são:

- Os Circuit-Level Gateway's podem **suportar** várias aplicações TCP/IP **sem necessidade de modificações** adicionais no lado cliente de cada aplicação
- Os Circuit-Level Gateway's **não fornecem processamento ou filtragem** de pacotes

  Por isso são geralmente designados como **gateway's transparentes**
- Os Application Level Gateway's normalmente têm **falta de suporte para o UDP**
- Os Circuit-Level Gateway's são frequentemente usados em **conexões para o exterior**, enquanto os proxy's são usados em **conexões para os dois sentidos**

?? O **exemplo** mais conhecido de um Circuit-Level Gateway é o **SOCKS**, actualmente na versão 5 (SOCKSv5)

?? Trata-se de um *proposed standard* do IETF, descrito no RFC 1928

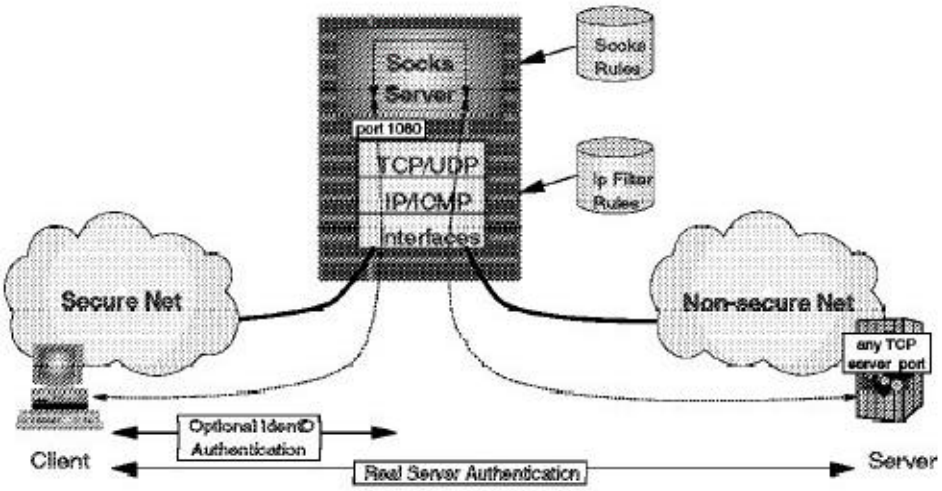


Figura 2.39 – Servidor SOCKS

?? Informação adicional: **http://www.socks.nec.com**

3.6.4. Exemplos de Firewall's

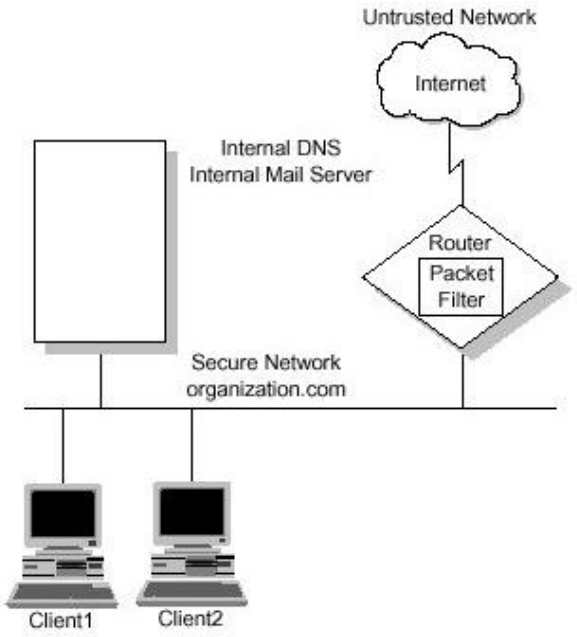


Figura 2.40 – Firewall de Filtragem de Pacotes

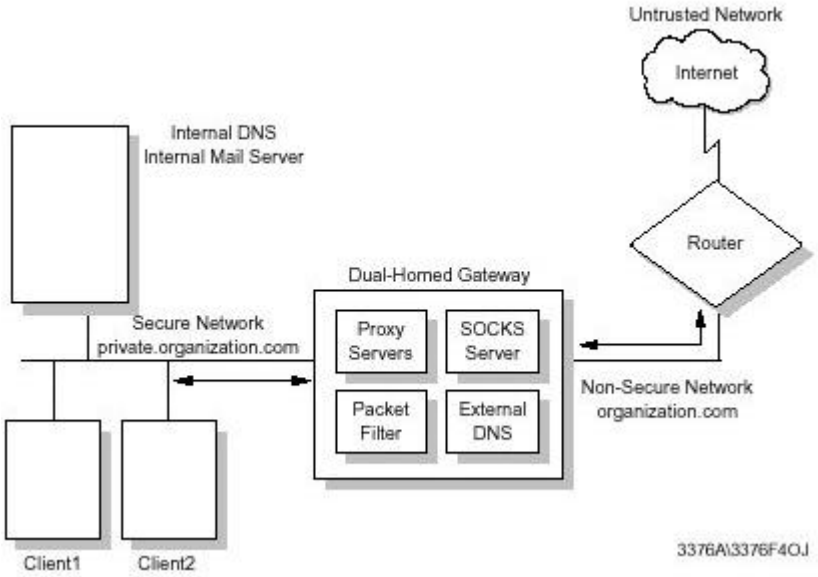


Figura 2.41 – *Dual-Homed Firewall*

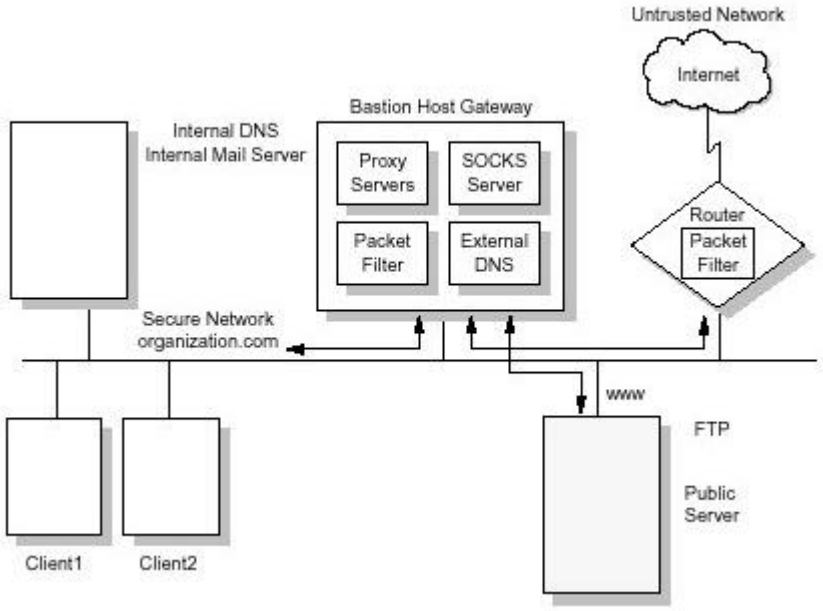


Figura 2.42 – *Screened Host Firewall*

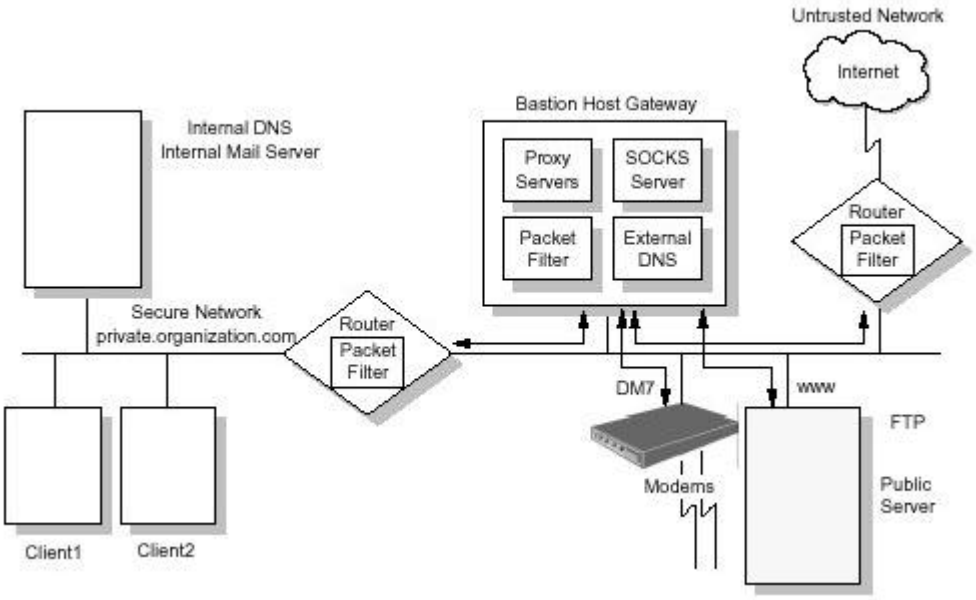


Figura 2.43 – *Screened Subnet Firewall*

3.7. Network Address Translation (NAT)

?? O **NAT** (RFC 1631) surgiu como uma **solução de recurso** para o **problema da exaustão do espaço de endereçamento**, verificado à alguns anos atrás

?? Faz a **tradução de endereços privados para um endereço válido**, permitindo assim o acesso à Internet

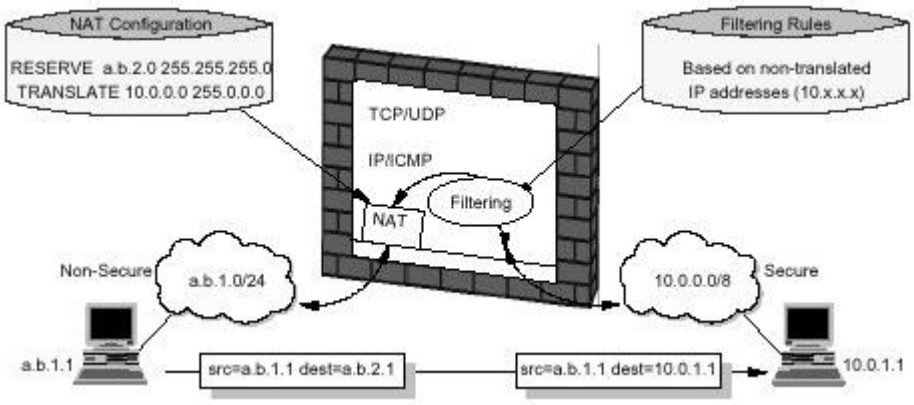


Figura 2.44 – *Network Address Translation* (NAT)

?? Quando **não são necessárias as funcionalidades adicionais** de um proxy ou do SOCKS, ou se algum destes não está disponível, por qualquer razão, o **NAT** pode ser **usado** para **gerir o tráfego** entre uma rede interna e uma rede externa

?? Este **mecanismo** é completamente **transparente** para os hosts e para as aplicações

?? Do ponto de vista de **dois hosts** que trocam pacotes através de um **servidor de NAT**, este **funciona** como um **normal encaminhador**

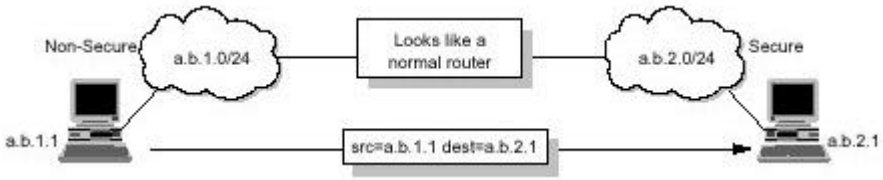


Figura 2.45 – NAT do ponto de vista de Rede Insegura

?? **Mecanismo de Tradução**:

- Por cada **pacote de saída**, é **verificado** o **endereço de origem**, pelas regras de configuração do NAT
- Se o endereço se **enquadra** em alguma regra, este **é traduzido para um endereço oficial**, a partir de uma **pool de endereços** pré-definida

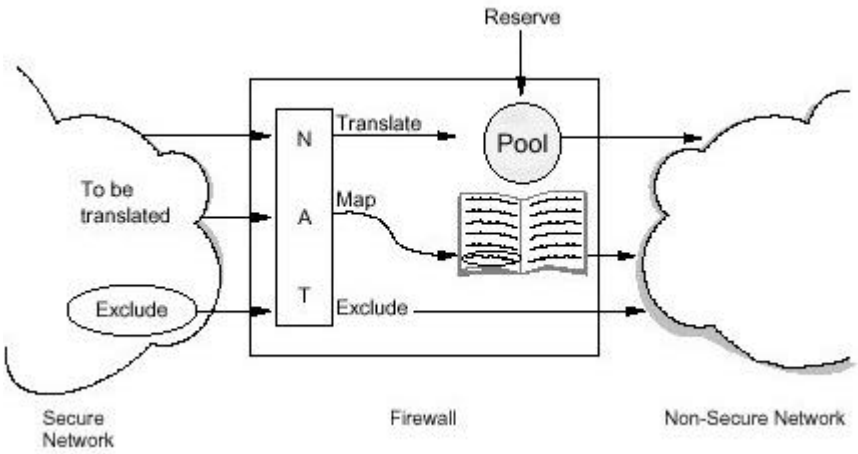


Figura 2.46 – Configuração do NAT

- Por cada **pacote de entrada**, é **verificado** pelo NAT se o **endereço de destino** é usado por este
- Se for, o endereço é **traduzido** para o **endereço privado** original

?? Quando o NAT **traduz** endereços de **pacotes TCP**, **ajusta** ao mesmo tempo o *checksum* do pacote

?? Apenas **pacotes TCP** ou **UDP** são **traduzidos** pelo NAT, **não suportando** o protocolo **ICMP**

3.8. A Arquitectura IPSec

3.8.1. Conceitos

?? *Framework* que tem por **objectivo** introduzir **mecanismos de segurança**, ao nível da **camada de rede** da arquitectura TCP/IP

?? A **Arquitectura IPSec** é **constituída** por **três componentes** principais:

- Authenticaiton Header (AH)
- Encapsulating Security Payload (ESP)
- Internet Key Exchange (IKE)

3.8.2. Authentication Header (AH)

?? O **cabeçalho AH** é **usado** para fornecer **verificação de integridade** e **autenticação** a datagramas IP

?? Existem alguns **campos** do datagrama IP que podem **sofrer alterações** ao **longo do percurso** até ao destino (**campos mutáveis**), o que faz com que seja **impossível** serem **protegidos pelo AH**:

- *Type of Service (TOS)*
- *Flags*
- *Fragment Offset*
- *Time to Live (TTL)*
- *Header Checksum*

?? Quando é **requerida** a **protecção** destes campos, usa-se o mecanismo de ***tunneling*** para esse datagrama

?? O *Payload* de um datagrama IP é considerado **imutável** e é **sempre protegido** pelo AH

?? O AH é **identificado** pelo número **51**, no campo *Protocol* do cabeçalho IPv4 e no *Next Header* de um cabeçalho IPv6

?? Pacotes que **falham** a autenticação fornecida pelo AH são **descartados**

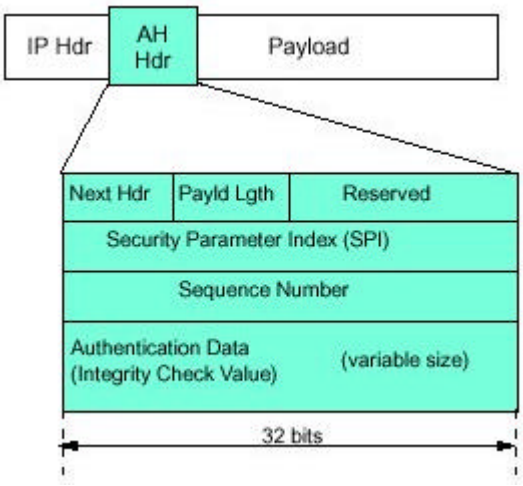


Figura 2.47 – Formato do cabeçalho do AH

?? *Security Parameter Index* (SPI): valor de 32 bits que **identifica** diferentes *Security Associations* (SA)[1] com os **mesmos endereço de destino** e **protocolo de segurança**

[1] Uma SA é um triplo <*Security Parameter Index,* Endereço IP de destino, Protocolo de Segurança>

?? *Authentication Data*: este valor é **calculado** a partir do **algoritmo** seleccionado na inicialização da SA

É **usado** pelo receptor para **verificar a integridade** do pacote recebido

?? O AH é usado de **duas formas**:

- **Modo de Transporte**:
  - o datagrama IP original é examinado, sendo depois inserido o cabeçalho AH a seguir ao cabeçalho IP
  - é usado apenas pelos hosts, não pelos gateways
  - **vantagem**: introduz pouco overhead
  - **desvantagem**: os campos mutáveis não são autenticados

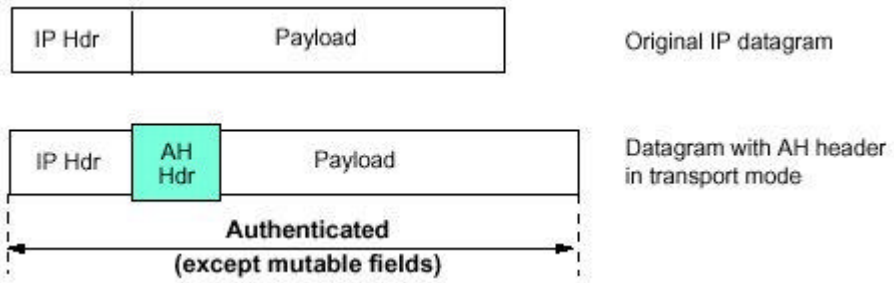


Figura 2.48 – AH em Modo de Transporte

- **Modo de Túnel**:
  - é construído um novo datagrama, que encapsula o original, e ao qual é aplicado de seguida o **Modo de Transporte**
  - é usado sempre que uma parte da *Security Association* (SA) é um gateway
  - entre duas firewalls é sempre usado este modo

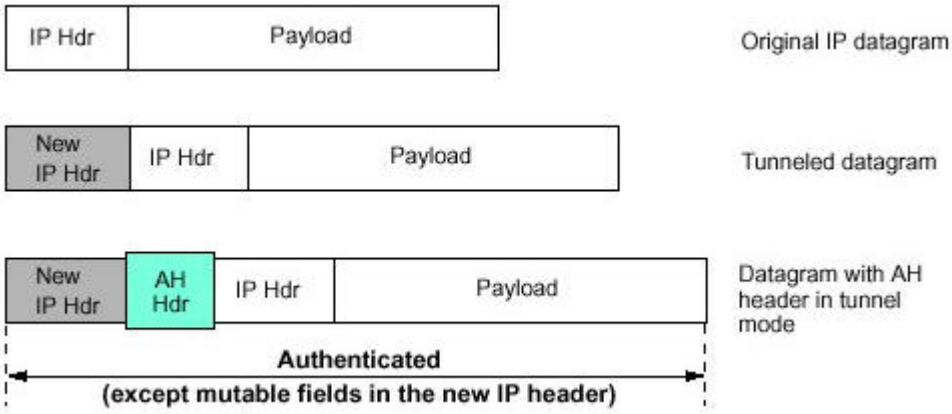


Figura 2.49 – AH em Modo de Túnel

?? O AH é **parte integrante** do IPv6, através de um **cabeçalho de extensão** para autenticação

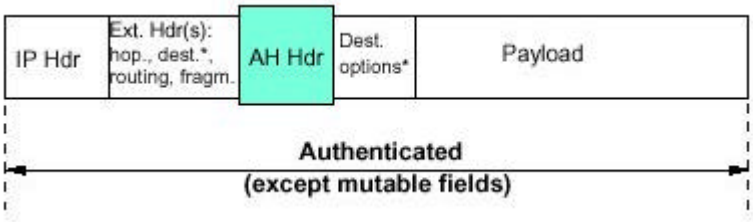


Figura 2.50 – AH em Modo de Transporte com IPv6

3.8.3. Encapsulating Security Payload (ESP)

?? O **ESP** é **usado** para fornecer **verificação de integridade**, **autenticação** e **cifragem** aos datagramas IP, além de **protecção opcional nas respostas**

?? Apresenta algumas **restrições**:

- o Verificação da integridade e autenticação funcionam em conjunto
- o Protecção da resposta é seleccionável apenas com verificação de integridade e autenticação
- o Protecção da resposta pode ser apenas seleccionada pelo receptor

?? A **cifragem** pode ser seleccionada de forma **independente** dos outros serviços

?? É altamente **recomendável** que, se a **cifragem** for **activada**, então a **verificação de integridade** e **autenticação** sejam **também activadas**

?? O ESP é **identificado** pelo número **50**, no campo *Protocol* dos datagramas IPv4 e no campo *Next Header* dos datagramas IPv6

?? É aplicado apenas a **pacotes** IP **não fragmentados**

?? No entanto, um **pacote** com o **ESP aplicado**, pode ser **fragmentado** pelos encaminhadores intermédios

?? Se é **seleccionada** a **cifragem** e **autenticação** com **verificação de integridade**, o receptor autentica primeiro o pacote, e só se este passo for bem sucedido é que se procede à decifragem

?? O **pacote ESP** é constituído por **três partes**:

- ESP Header
- ESP Trailer
- ESP Authentication Data

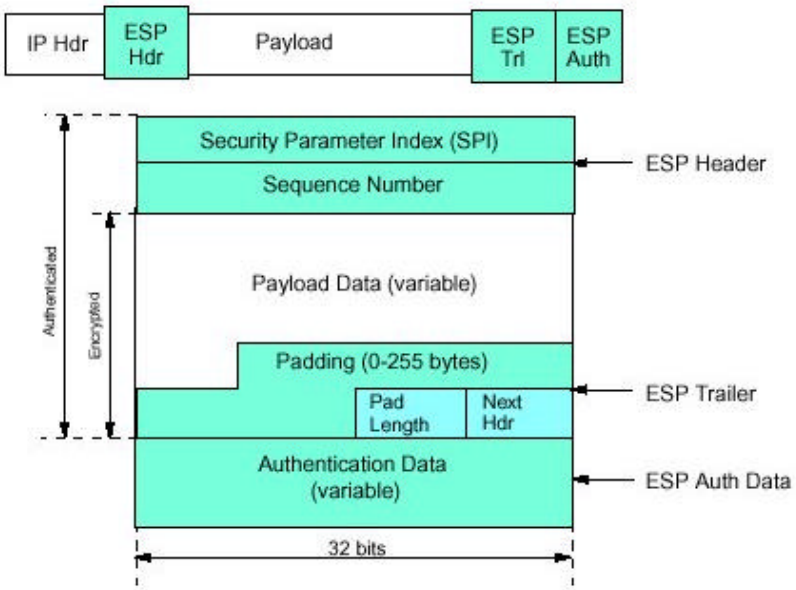


Figura 2.51 – ESP *Header* e *Trailer*

?? A **especificação ESP** requer a **suporte** para o **algoritmo DES**, apesar de outros algoritmos de cifragem poderem ser utilizados

?? Tal como o AH, o ESP pode ser **usado** de **duas formas**:

- **Modo de Transporte**:

  - O cabeçalho ESP é inserido a seguir ao cabeçalho IP

  - Se o datagrama já usar cabeçalhos IPSec, então o cabeçalho ESP é inserido antes destes

  - O ESP *Trailer* e os Dados de Autenticação opcionais são adicionados no final do *Payload*

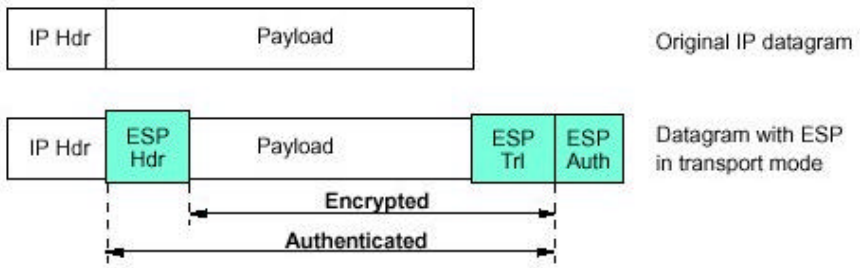


Figura 2.52 – ESP em Modo de Transporte

  - Neste modo não é fornecida autenticação nem cifragem para o cabeçalho IP

  - Introduz pouco *overhead*

- **Modo de Túnel**:

  - é construído um novo datagrama, que encapsula o original, e ao qual é aplicado de seguida o **Modo de Transporte**

  - Como o datagrama original se transforma no *Payload* do novo pacote ESP, a protecção do primeiro é total, se forem seleccionadas a cifragem e a autenticação

- No entanto, o cabeçalho do novo datagrama não é protegido
- É usado sempre que uma extremidade da *Security Association* (SA) é um gateway

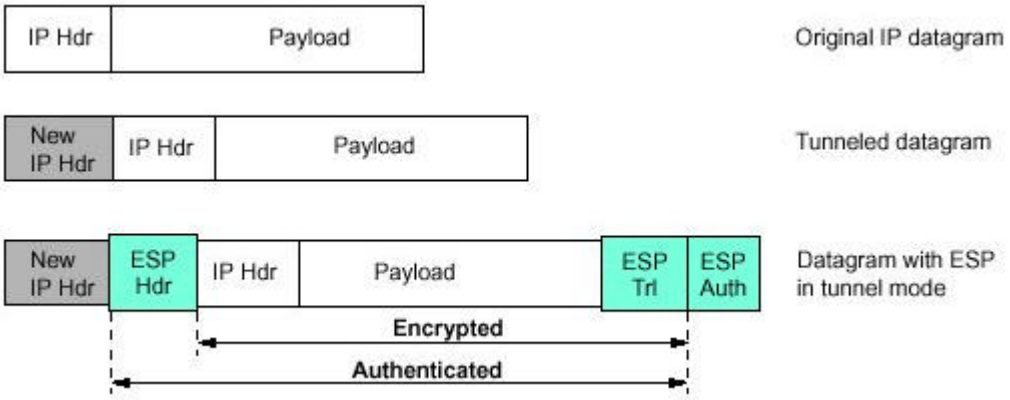


Figura 2.53 – ESP em Modo de Túnel

?? Tal como o AH, o ESP também é parte integrante da especificação IPv6, sendo implementado por um cabeçalho de extensão

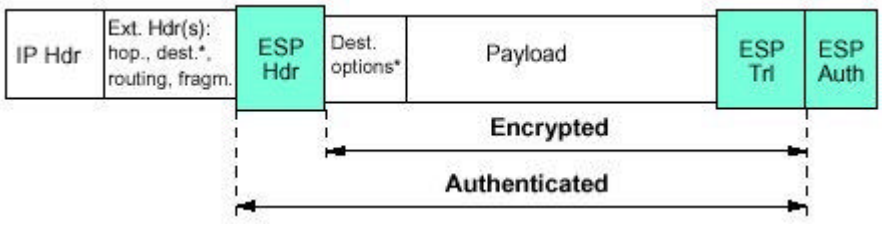


Figura 2.54 – ESP em Modo de Transporte, em IPv6

3.8.4. Internet Key Exchange Protocol (IKE)

- A *framework* **IKE** (anteriormente referenciada como **ISAKMP/Oakley**) suporta **negociação automática** de *Security Associations* (SA's) e **geração automática** e **refrescamento** de **chaves criptográficas**

- Trata-se de um dos elementos fundamentais da utilização do **IPSec** em **larga escala**

- **Elementos fundamentais**:

  - *Internet Security Association and Key Management Protocol* (**ISAKMP**):

    - *Framework* que **define a gestão** das ***Security Associations*** (negociação, alteração e eliminação) e das **chaves**

    - Define ainda o *Payload* para troca de **informação de geração de chaves** e de **dados de autenticação**

    - **Não define** qualquer **protocolo** de troca de chaves

  - ***Oakley***:

    - **Protocolo de troca de chaves**, que pode ser usado com a *framework* ISAKMP

  - **Domínio de Interpretação** (**DOI**):

    - Definição do **conjunto de protocolos** que podem ser **usados** com a *framework* ISAKMP, para um **ambiente particular**

    - Conjunto de **definições comuns**, partilhadas com os protocolos referidos (sintaxe dos atributos das SA, conteúdo dos *payload*, espaço de nomes das transformadas criptográficas, etc)

    - Relativamente ao IPSec, o **DOI** instancia o **ISAKMP** para **utilização com o IP**

- **Protocolo *Internet Key Exchange*** (**IKE**):
  - Usa partes do ISAKMP e partes dos protocolos de troca de chaves Oakley e SKEME
  - Fornece **gestão de chaves** e associações de segurança para os protocolos IPSec AH e ESP

- O **IKE** usa os seguintes **métodos de autenticação**:
  - Chaves pré-partilhadas
  - Assinaturas Digitais (DSS e RSA)
  - Cifragem de Chave Pública (RSA e RSA revisto)

3.9. Secure Sockets Layer (SSL)

- **Protocolo de segurança** desenvolvido pela Netscape Communications Corporation, conjuntamente com a RSA Data Security, Inc
- **Objectivo** base:
  - fornecer um **canal privado** de comunicação **entre aplicações**, que assegure **privacidade dos dados**, **autenticação** das partes e **integridade**
- Fornece uma **alternativa** à **API de sockets TCP/IP**, com possibilidade de implementação de **mecanismos de segurança**
- É **usado** principalmente em conexões **HTTP**
- É composto por **dois níveis**:
  - No **nível mais baixo** é usado um **protocolo** para a **transferência dos dados** usando uma variedade de **cifras pré-definidas** e de combinações de autenticação – ***SSL Record Protocol***

- No **nível mais alto** é usado um **protocolo** para a **autenticação inicial** e transferência das chaves de cifragem – ***SSL Handshake Protocol***

?? Uma **sessão SSL** é inicializada da seguinte forma:

- No **cliente** (browser), o utilizador solicita um documento a um servidor que suporta SSL (utilizando no URL **https**, em vez de http)
- É reconhecido um pedido SSL, pelo cliente, estabelecendo de seguida uma conexão com o servidor, através do **porto TCP 443** deste
- O cliente inicia então a fase de **SSL Handshake**, usando o **SSL Record Protocol** para o transporte da informação

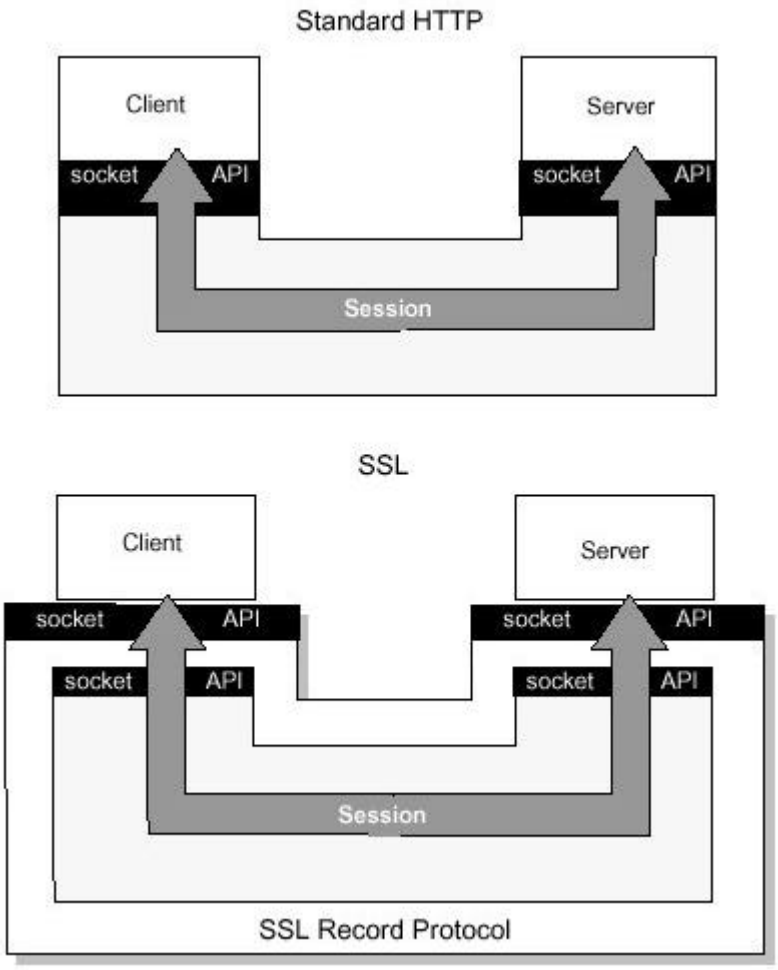


Figura 2.55 – Comparação entre uma Sessões Standard e SSL

- O **SSL** responde às seguintes **questões de segurança**:
  - Privacidade
  - Integridade
  - Autenticação
- O standard actual é o **SSL 3.0**, que mantém, no entanto, compatibilidade com o SSL 2.0
- O Protocolo SSL localiza-se no **topo** da **Camada de Transporte**
- Recebe os dados da **Camada de Aplicação**, **reformata-os** e entrega-os à **Camada de Transporte**
- **Change Cipher Spec Protocol**: protocolo responsável pela **troca de mensagens** relacionadas com os **parâmetros criptográficos**

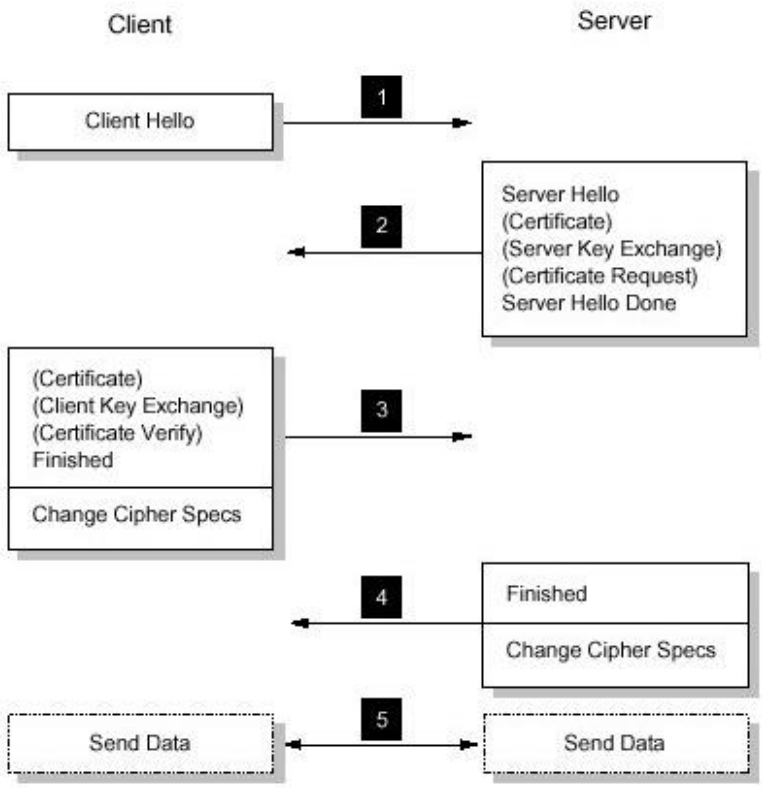


Figura 2.56 – Processo de *SSL Handshake*

- O **SSL Handshake Protocol** permite ao cliente e servidor **determinar os parâmetros requeridos**, para uma **conexão SSL**:
  - Versão do protocolo
  - Algoritmos criptográficos
  - Autenticação opcional do cliente ou do servidor
  - Métodos de cifragem de chave pública, para geração das chaves secretas partilhadas
- Comparativamente com uma **conexão HTTP normal**, a inicialização de uma **sessão SSL** introduz ***overhead*** adicional bastante significativo
- O **protocolo** ainda **evita** algum deste ***overhead***, através da **retenção de informação de sessão**, por parte do cliente e do servidor

3.10. O Sistema de Autenticação e Autorização Kerberos

- O **Kerberos** é um **sistema de segurança**, baseado em **criptografia**
- **Fornece autenticação mútua** entre **utilizadores** e **servidores** num ambiente de rede
- Principais **objectivos**:
  - **Autenticação** para **prevenir** pedidos/respostas fraudulentos, entre utilizadores e servidores, que têm de se manter confidenciais
  - **Autorização** pode ser **implementada** de forma **independente** do sistema de autenticação de cada serviço

O sistema de Autorização pode assumir que a autenticação de um utilizador/cliente é de confiança

- Permite a **implementação** de um **sistema de contabilização integrado**, **seguro**, de **confiança** e **modular**

- **Usado** em primeiro lugar para **autenticação**, mas fornece **flexibilidade** suficiente para poder ser adicionada **informação de autorização**

- ***Principal Identifier***: nome que **identifica** de forma única um **cliente** ou um **serviço**, no sistema Kerberos

- Na **versão 5**, este **identificador** é **constituído** por **duas partes**: ***realm*** e ***remainder***, definidos de acordo com as regras **ASN.1** (Abstract Syntax Notation One, standard ISO 8824)

3.10.1. Processo de Autenticação Kerberos

- Num **sistema Kerberos**, um cliente que pretende contactar um servidor de determinado serviço, primeiro tem de "perguntar" por um **bilhete** (***ticket***), a uma **terceira parte de confiança mutua** – o **Kerberos Authentication Server** (**KAS**)

- O ***Ticket*** é obtido na forma de **função**, onde um dos componentes é a **chave privada**, **conhecida** apenas pelo **serviço respectivo** e pelo **Servidor de Autenticação Kerberos**

- O **processo de autenticação** consiste na **troca de cinco mensagens**

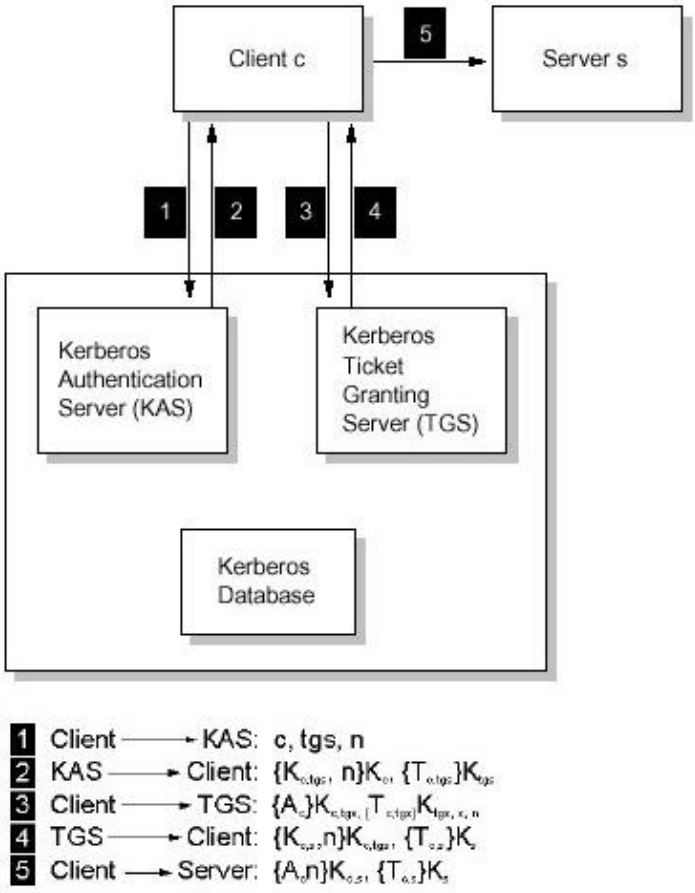


Figura 2.57 – Esquema de Autenticação Kerberos

3.10.2. Administração da Base de Dados Kerberos

?? O Kerberos precisa de um **registo** para cada utilizador e serviço, que armazena a **seguinte informação**:

- *Principal Identifier*
- Chave Privada para este identificador
- Data de Expiração desta entidade
- Data da última modificação neste registo
- Identidade de que modificou pela última vez este registo
- Tempo de vida máximo dos tickets que chegam a esta entidade

- Atributos
- Dados de implementação

?? A **chave privada** é **cifrada** usando uma **chave mestra**, independente destes registos

?? **Kerberos Database Manager** (**KDBM**): entidade **responsável** pela **gestão da base de dados** Kerberos

?? Pode existir ainda um ou mais ***Kerberos Key Distribution Servers*** (**KKDS**), que **contém** uma **cópia** da Base de Dados Kerberos

?? Os **KKDS** têm apenas **acesso de leitura** à Base de Dados, ficando as **operações de actualização** da responsabilidade do **KDBM**

3.11. Secure Electronic Transations (SET)

?? O **SET** é o **resultado** de um **acordo** entre a *MasterCard International* e a *Visa International* para a criação de um **sistema de cartão de crédito electrónico** simples

?? A **especificação SET** descreve um conjunto de **fluxos de transacção** para **carregamento**, **autenticação**, **pagamento**, etc

?? **Fases de uma Transacção**:

- o Inicialização
- o Ordem de Compra
- o Autorização da compra
- o Inquirição
- o Captura

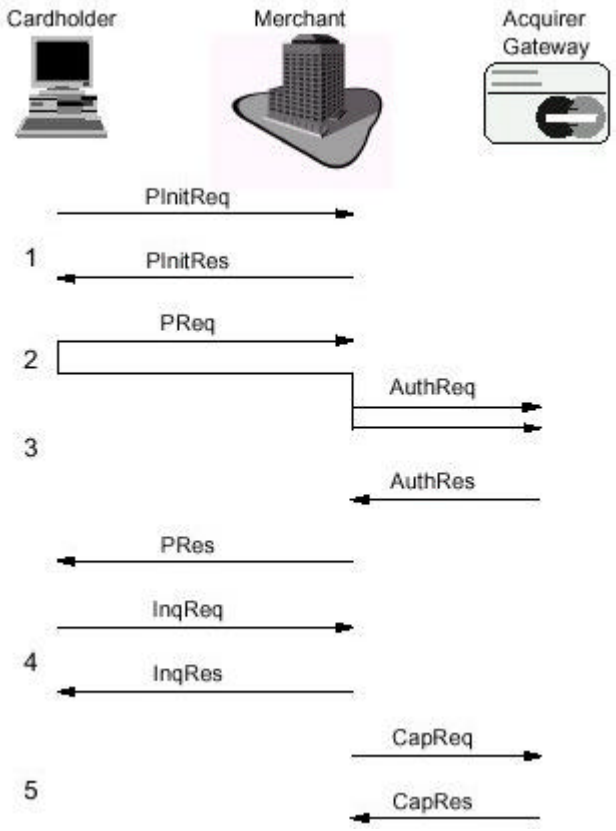


Figura 2.58 – Sequência de uma Transacção SET típica

?? A **especificação SET destina-se** potencialmente a toda a **comunidade Internet**

?? Cada **potencial utilizador** deverá possuir um **certificado de Chave Pública**

?? Todo este **processo de gestão de chaves** em larga escala requer uma **estrutura de certificação leve** e **simples** (ver figura seguinte)

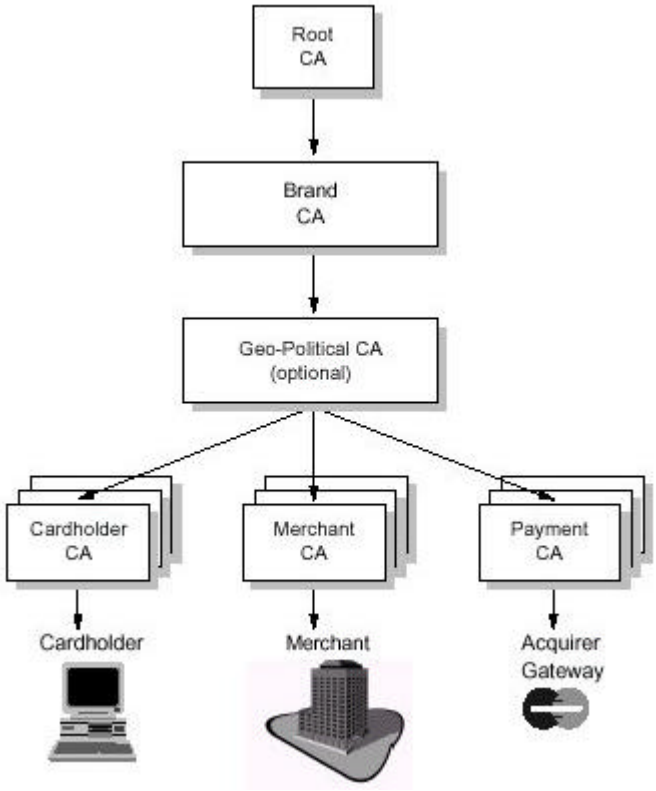


Figura 2.59 – Autoridades de Certificação SET

4. Qualidade de Serviço em redes IP

4.1. O porquê da Qualidade de Serviço

- À medida que a **Internet** vai **crescendo**, em número de **utilizadores**, ***hosts*** ligados e **conteúdos** disponibilizados, com mais intensidade se vão **verificando** algumas das suas **debilidades**

- Uma das que ultimamente mais têm ocupado os investigadores prende-se com a **Qualidade de Serviço - QoS**

- O surgimento de **novos serviços e aplicações**, associados a **diferentes tipos de tráfego**, com **necessidades distintas** (vídeo, áudio, real-time, etc), é um dos motores dos **desenvolvimentos** recentes nesta área

- O **funcionamento da Internet** caracteriza-se, desde o seu surgimento, pelo **paradigma** do ***best effort*** associado ao protocolo IP

- Significa isto que **cada nodo** da rede vai realizar o **melhor esforço** possível para tratar os pacotes IP, dando-lhes no entanto o **mesmo tratamento**, em **função** das **condições de rede** existentes em cada momento

- Muitas das **novas aplicações** têm **diferentes graus de exigência** no tráfego que geram, nomeadamente ao nível do **atraso** do transporte dos dados, **perdas** e **largura de banda** disponível

- São estes **factores** que **não são compatíveis** com o paradigma tradicional do ***best effort,*** e que **motivam** então a **procura** de **novas soluções** que garantam uma **Qualidade de Serviço** diferenciada

- Existem **várias formas** de **caracterizar** Qualidade de Serviço (QoS)

- De uma forma genérica, **QoS** é a **capacidade** dos elementos da rede (aplicações, hosts, encaminhadores) fornecerem algum **nível de garantia** para uma **transferência de dados consistente**

- Existem dois **formas** básicas de estabelecimento de **medidas de QoS**:

  - **Reserva de Recursos**: os **recursos da rede** são **repartidos** de acordo com um **pedido de QoS** de uma aplicação, e em conformidade com a política de gestão da largura de banda

  - **Prioritização**: o **tráfego da rede** é **classificado** e os recursos da rede são repartidos de acordo com critérios da política de gestão da largura de banda

- Estes **tipos de QoS** podem ser aplicados a **fluxos individuais** ou a **fluxos agregados**

- Esta **classificação** dá origem a outras duas formas de **caracterização de QoS**:

  - **Por Fluxo**: um fluxo é definido como uma **sequência de dados individual** e **uni-direccional** entre duas aplicações (cliente e servidor), **identificada** univocamente pelo **tuplo** <protocolo de transporte, endereço de origem, porto de origem, endereço de destino, e porto de destino>

  - **Por Agregação**: um agregado é simplesmente **dois ou mais fluxos**

    Tipicamente, os fluxos têm algo em comum (um ou mais dos parâmetros do tuplo, prioridade, label, etc)

- Ente os **desenvolvimentos** mais recentes nesta área, destacam-se **dois modelos** que têm vindo a ser desenvolvidos no âmbito do IETF:

  - *Integrated Services* – **IntServ** (RFC 2215, RFC 2216), baseado em **Reserva de Recursos**

- *Differentiated Services* – **DiffServ** (RFC 2474, RFC 2475), baseado em **Prioritização**

?? O primeiro define um conjunto de **métodos** de especificação de qualidade de serviço e que **permitem reservar recursos**, que são **alocados** para **fluxos de tráfego** individuais

?? Este modelo tem apresentado no entanto algumas **limitações** que têm vindo a colocar em causa a sua aplicabilidade em escala alargada

?? Como tentativa de ultrapassar estas limitações, surgiu o modelo **DiffServ**

?? Este **classifica** o tráfego em **diferentes classes de serviço** (CoS), com base num conjunto de bits específicos nos cabeçalhos dos pacotes IP (sejam eles pacotes IPv4, sejam pacotes IPv6)

?? O **objectivo** é fornecer um **tratamento particular** a diferentes classes de tráfego, identificadas em cada pacote IPv4 ou IPv6 pelo **campo DS** (RFC 2474), por parte dos sistemas de comunicação, com base na classe de serviço definida para esse mesmo pacote.

4.2. O modelo de Serviços Integrados (*IntServ*)

?? O modelo **IntServ** foi desenvolvido para **optimização da rede** e da utilização de recursos por novas aplicações (multimédia, de tempo real), que requerem garantias de **QoS**

?? Devido aos **atrasos** no encaminhamento e às **perdas** por congestão, as **aplicações de tempo real** não funcionam bem com o tráfego best-effort

?? Estas **aplicações precisam** assim de **largura de banda garantida**

- O **Modelo IntServ** torna isso possível, dividindo o tráfego em **best-effort**, **para** os **serviços tradicionais** e em **fluxos de dados de aplicações** com **QoS garantida**

- Um **encaminhador** que **suporte** o modelo **IntServ** tem de ser capaz de **fornecer** uma **QoS** apropriada para **cada fluxo**, de acordo com o **modelo do serviço**

- O fornecimento de **diferentes qualidades de serviço** é efectuado, no encaminhador, por uma função de **controlo de tráfego**

- Esta **função** é constituída pelos **componentes** seguintes:

  - **Escalonador de Pacotes** (*Packet Scheduler*):

    - Faz a **gestão do envio** das diferentes sequências de dados, nos hosts e nos encaminhadores, de acordo com a sua classe de serviço

    - Usa **gestão de filas** e vários algoritmos de escalonamento

    - Tem de assegurar que o envio dos pacotes corresponde ao parâmetro de QoS de cada fluxo

    - É implementado no ponto onde os pacotes são "enfileirados" (*queued*)

  - **Classificador de Pacotes**:

    - **Identifica pacotes** de um fluxo IP em hosts e encaminhadores, que **recebem determinado nível de serviço**

    - Cada pacote que chega ao classificador é **mapeado** para uma **classe específica**

    - Todos os **pacotes** que são **classificados** com a **mesma classe** recebem o **mesmo tratamento** do Escalonador de Pacotes

    - A **escolha da classe** é baseada nos endereços e portos de origem e destino do cabeçalho do pacote ou num número de classificação

adicional, que pode ser adicionado a cada pacote

- **Controlo de Admissão**:
  - contém o **algoritmo** que o encaminhador usa para determinar se existem recursos de encaminhamento suficientes para aceitar a QoS requisitada para determinado fluxo
  - se esses **recursos não existirem**, esse fluxo tem de ser **rejeitado**
  - se o fluxo é aceite, a instância de reserva no encaminhador activa o Classificador e o Escalonador de Pacotes, para reservar a QoS requisitada
  - O controlo de admissão é invocado em cada encaminhador, ao longo do caminho de reserva

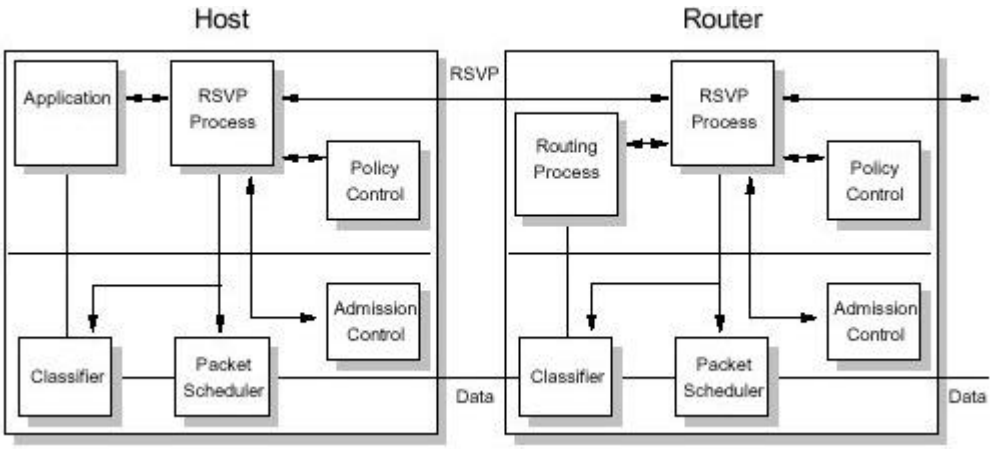


Figura 2.60 – Modelo de Serviços Integrados

4.2.1. Classes de Serviço

- O Modelo IntServ utiliza diferentes **classes de serviço**, definidas pelo *working group IntServ* do IETF

- Especificamente para este modelo foram definidas as **classes**:
  - ***Guaranteed Service*** (RFC 2212):
    - Assemelha-se à emulação de um circuito dedicado virtual
    - Fornece fronteiras rígidas em atrasos na filas fim-a-fim, através da combinação de parâmetros de vários elementos da rede ao longo do caminho
    - Assegura ainda disponibilidade de largura de banda, de acordo com parâmetros TSpec
  - ***Controlled Load Service*** (RFC 2211):
    - Equivalente ao serviço best-effort em condições controladas
    - No entanto, trata-se de um serviço melhor que o best-effort, mas sem o controlo rígido do *Guaranteed Service*

4.2.2. O Protocolo RSVP

- O modelo IntServ usa o ***Reservation Protocol*** (**RSVP**) para a **sinalização** das mensagens de reserva
- O **RSVP** é provavelmente a **mais complexa**, das principais tecnologias de fornecimento de QoS para aplicações (hosts) e elementos de rede (encaminhadores e comutadores)
- As **instâncias IntServ comunicam** através do **RSVP** para criar e manter **estados** de **fluxos específicos**, nos hosts finais e nos encaminhadores ao longo do caminho de um fluxo

?? Uma **aplicação** que pretende **enviar pacotes** de dados num fluxo reservado **comunica** com a **instância de reserva RSVP**

?? O **protocolo RSVP** tenta **activar a reserva** de um fluxo, com a **QoS requisitada**

?? O RSVP solicita ao Classificador e ao Escalonador de Pacotes em cada *nodo* o processamento adequado de cada fluxo

?? O principio simplificado de funcionamento está representado na figura seguinte:

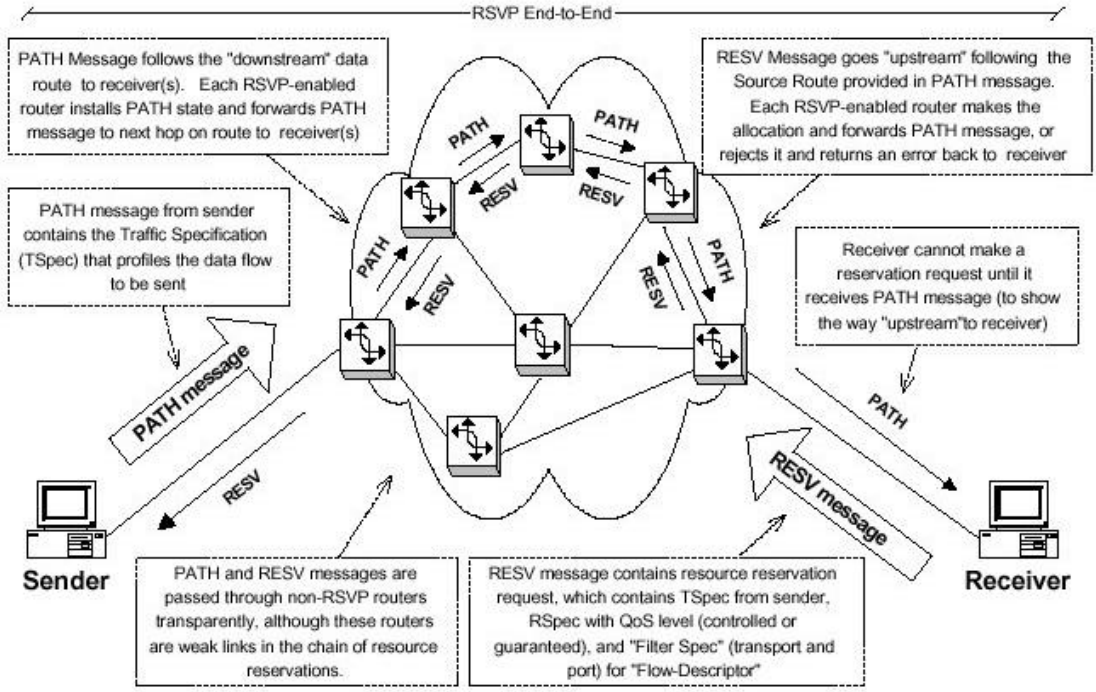


Figura 2.61 – Principio de funcionamento do RSVP

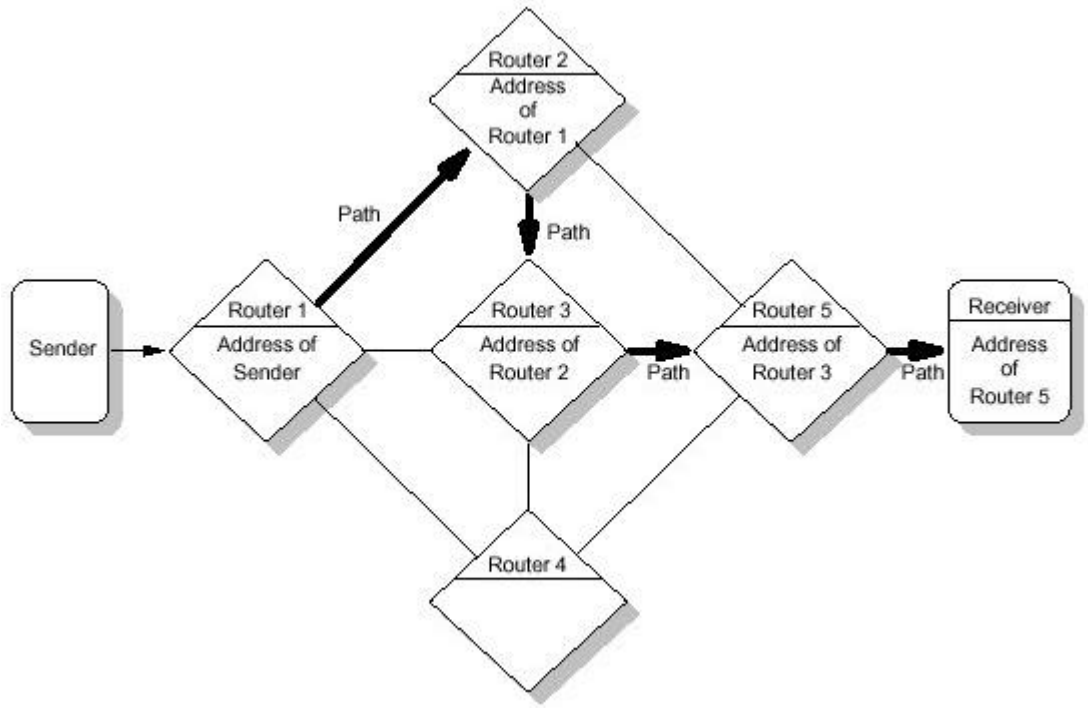


Figura 2.61 – Processo de definição do Caminho RSVP

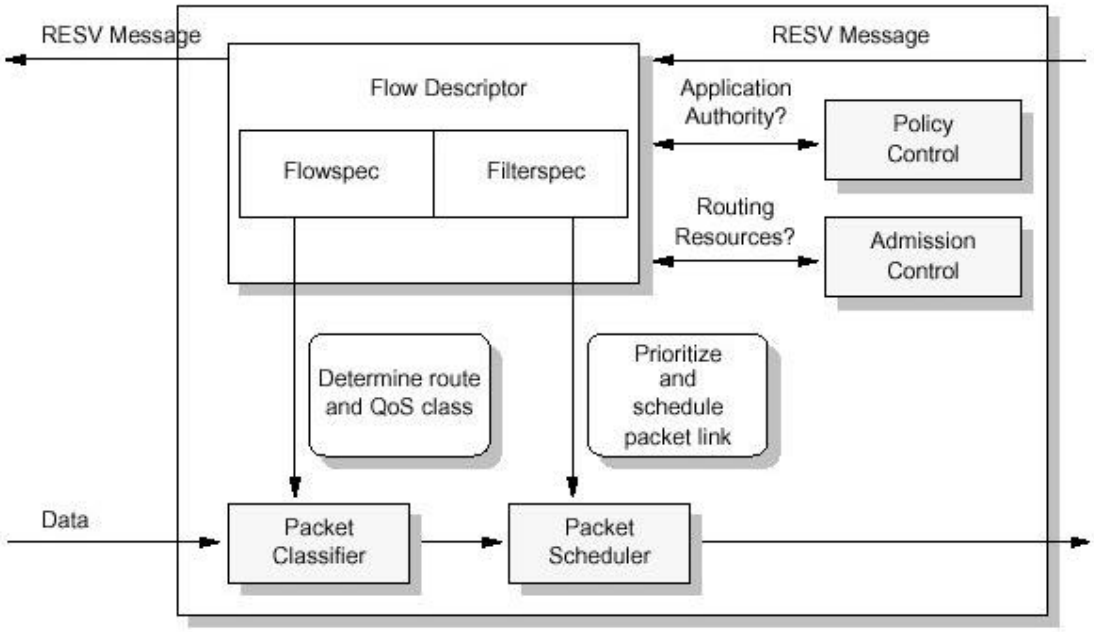


Figura 2.62 – Processo de Reserva RSVP

?? Como referido atrás, o **RSVP** fornece o **nível mais elevado** de **disponibilidade** de **QoS IP**

- Permite a uma aplicação requisitar QoS com um **elevado nível de granularidade** e com as **melhores garantias** possíveis de serviço de transmissão
- No entanto, **nem tudo são boas notícias**, já que apesar destes benefícios existem também **algumas limitações** importantes:
  - **complexidade de implementação** (que se traduz também em carga computacional)
  - **overhead** adicional introduzido na rede, que **limita a escalabilidade** para grandes redes
  - actualmente **poucos hosts** têm **capacidade** de sinalização RSVP
  - Muitas **aplicações** requerem uma forma de QoS, mas **não são capazes** de expressar esses requisitos de acordo com o modelo IntServ

4.3. O modelo de Serviços Diferenciados (*DiffServ*)

- O **Modelo de Serviços Diferenciados** - **DiffServ** fornece um **método simples** de classificação de serviços
- RFC's base:
  - **RFC 2474** - Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers
  - **RFC 2475** - An Architecture for Differentiated Services
- Ao contrário dos Serviços Integrados, as **garantias de QoS** nos **Serviços Diferenciados** são **estáticas**
- Significa que as **aplicações** que usam DiffServ **não precisam** de efectuar **reservas** de QoS para pacotes de dados específicos
- Todo o **tráfego** que passa por uma **rede** "*DS-capable*" pode receber uma **QoS específica**

?? Os **pacotes** são **marcados** com um determinado valor (*Per Hop Behavior* - **PHB**), no **campo DS** (*Differenciated Services Field*), definido no RFC 2474

?? Este campo resultou de uma **redefinição** do campo *Type of Service* (TOS) do cabeçalho do pacote IPv4 e do campo *Traffic Class* do cabeçalho do pacote IPv6

?? Os **seis primeiros bits** do campo DS formam o sub-campo *Differentiated Services CodePoint* (**DSCP**), para seleccionar a classe de tráfego do pacote IP

?? Os últimos **dois bits** formam o sub-campo *Currently Unused* (**CU**), estando reservado para uso futuro

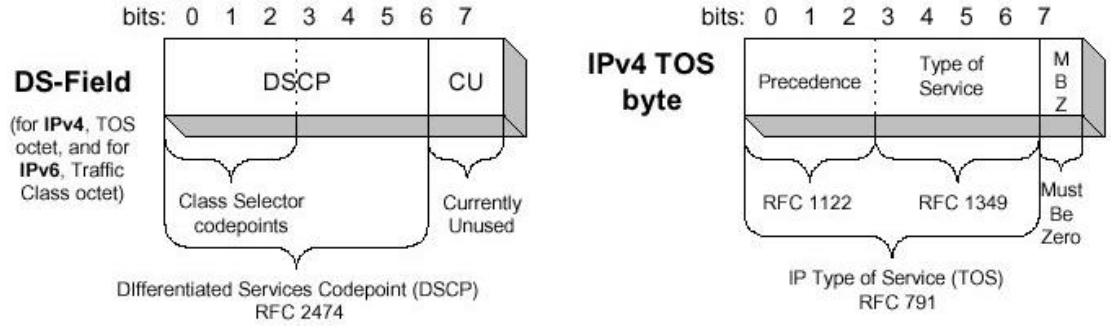


Figura 2.63 – Campo DS

?? O RFC 2475 define a Arquitectura para uma rede "*DiffServ capable*"

?? Aqui são definidas um conjunto de **funções** específicas de **condicionamento do tráfego**, realizadas pelos **encaminhadores DS** (figura seguinte)

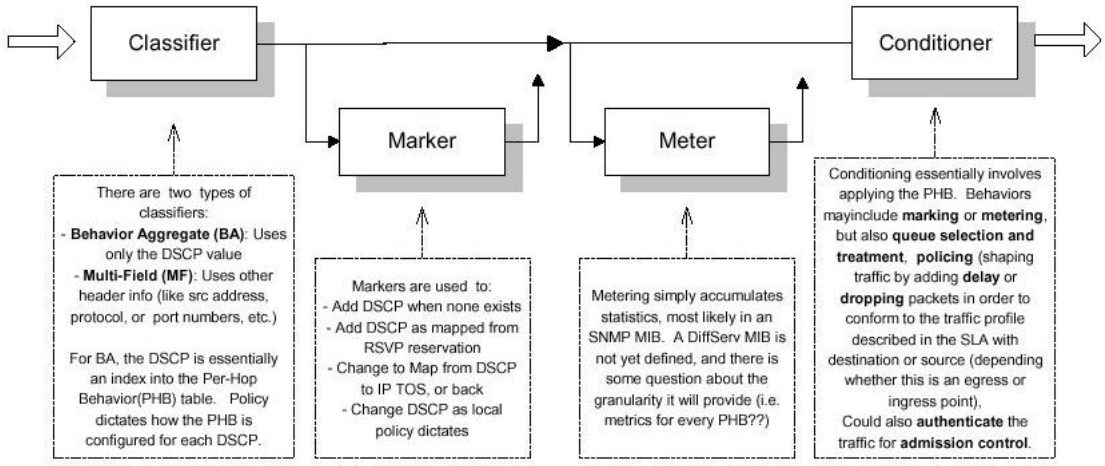


Figura 2.64 – Condicionamento de tráfego num encaminhador DS

?? Apesar de poderem ser usados outros, actualmente existem **dois PHB's** standard definidos:

- **Expedited Forwarding – EF** (RFC 2598):
  - ✍Definido através de um único *codepoint*
  - ✍O valor recomendado para o DSCP é 101110
  - ✍Pretende definir uma classe de tráfego que minimiza o atraso e fornece o nível mais elevado de QoS agregada
  - ✍Todo o tráfego deste PHB que excede o *profile* (definido pela política local) é descartado
- **Assured Forwarding – AF** (RFC 2587):
  - ✍Define quatro classes, com três valores de precedência em cada uma, o que dá um total de 12 *codepoints*

	Class 1	Class 2	Class 3	Class 4
Low Drop Prec	001010	010010	011010	100010
Medium Drop Prec	001100	010100	011100	100100
High Drop Prec	001110	010110	011110	100110

Figura – 2.65 – *Codepoint's* do PHP AF

- O tráfego deste PHB que excede o *profile* definido não é transmitido com a mesma alta probabilidade que o tráfego "*within profile*"

?? As **activação de QoS** garantida com o DiffServ **não** é feita **fim-a-fim**, mas dentro de **Domínios DS**

?? Um **Domínio DS** é uma **porção contínua** de uma **internetwork**, onde são **implementadas políticas** consistentes de Serviços Diferenciados, de uma forma coordenada

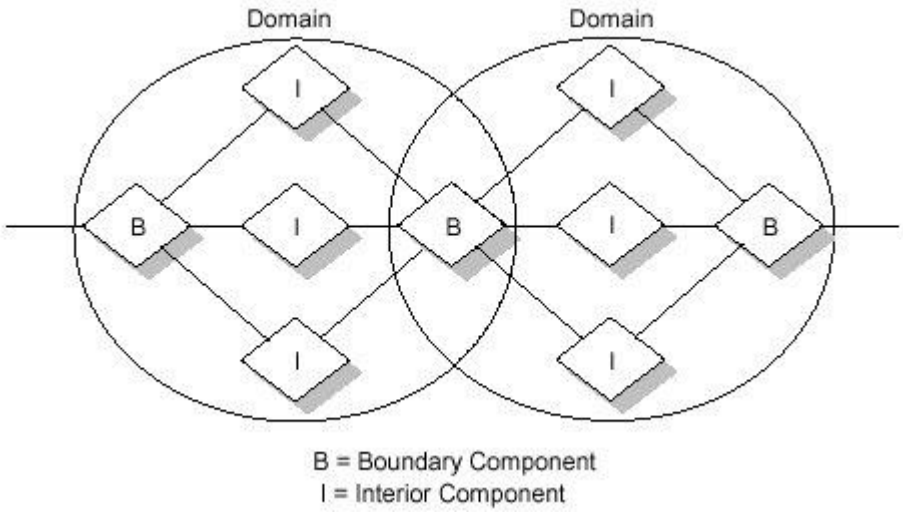


Figura 2.65 - Domínios DiffServ

?? A figura seguinte apresenta, de forma resumida, o **princípio de funcionamento do DiffServ**

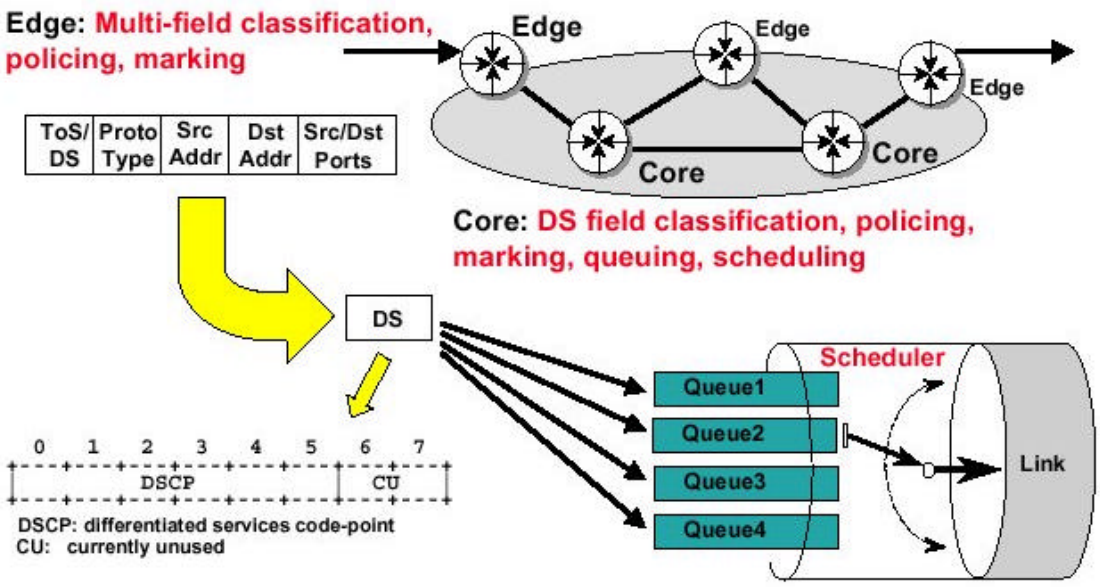


Figura 2.66 – Princípio de funcionamento do DiffServ

?? Como vimos anteriormente, o modelo de **Serviços Integrados** (e mais precisamente o RSVP) apresenta **problemas de implementação em larga escala**, que **limitam** a sua utilização da Internet

?? Assim, a **tendência actual** vai para a utilização do **IntServ** ao nível das **Intranets** e do **DiffServ** ao nível dos **Backbones** da Internet

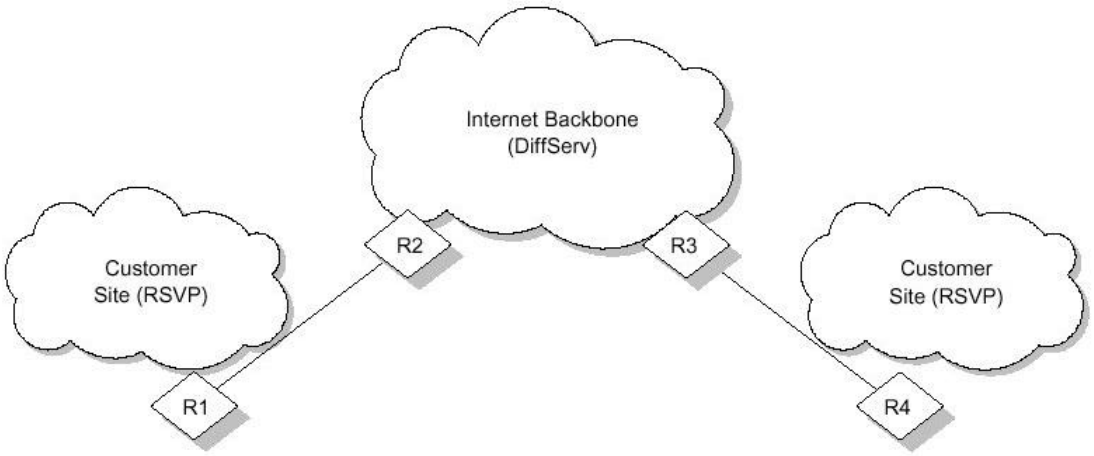


Figura 2.67 – Utilização de RSVP com Serviços Diferenciados

Bibliografia

Halsall, F. **Data Communications, Computer Networks and Open Systems**, Addison-Hesley, 4th edition, 1996

Stallings, W. ***Data and Computer Communications***, Prentice-Hall International, 5th edition, 1997

Murhammer, M; Atakan,O.; Bretz, S.; Pugh, L.; Suzuki, K.; Wood, D.; ***TCP/IP Tutorial and Techical Overview***, IBM, 1998

Wrigt, G.; Stevens, W.; ***TCP/IP illustrated, Vol 1: Protocols***, Addison-Wesley, 1996

Wright, G.; Stevens, W.; ***TCP/IP Illustrated Vol 2: The Implementation***, Addison-Wesley, 1996

Parker, T. ***Teach yourself TCP/IP in 14 days***, Sams Publishing, 1996

Kaeo, M. ***Designing network security***, Cisco Press, 1999

Hunt, C. ***TCP/IP : network administration***, O'Reilly & Associates, 1992

Williamson, B. ***Developing IP multicast networks***, Cisco Press, 2000

Slattery, T.; Burton, W. ***Advanced IP routing with Cisco networks***, McGraw-Hill, 1999

Black, U. ***Voice Over IP***, Prentice Hall, 2000

Chappell, L. **Advanced Cisco router configuration**, Cisco Press, 1999

Valença, J., Cunha, M. **Técnicas Criptográficas**, DI, UM, 1999