Redes de Computadores

(RCOMP – 2015/2016)

Protocolo HTTP.

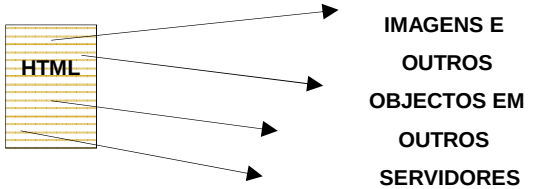
Gestão de redes. Protocolo SNMP.

Transferência de Ficheiros em Rede

Com o surgimento dos documentos de híper texto em rede, o protocolo mais usado nessa altura para transferir ficheiros, o FTP (“File Transfer Protocol”), revelou-se inapropriado para esse tipo de aplicação.

O FTP é demasiado complexo e torna-se lento quando se pretende realizar muitas transferências de ficheiros de dimensão reduzida, envolvendo diversos servidores, situação típica em documentos de híper texto como HTML (“Hypertext Markup Language”).

Para “carregar” completamente um ficheiro HTML, é necessário obter o ficheiro e também um conjunto mais ou menos vasto de outros ficheiros correspondentes a referências existentes.

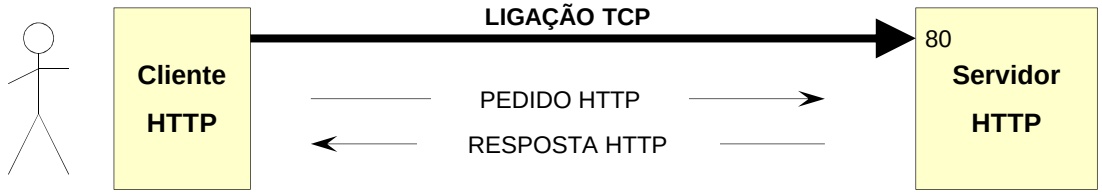


O FTP, com a necessidade de uma ligação de controlo de sessão e autenticação de utilizador não é de todo adequada para este tipo de aplicação. Muitas vezes demora mais tempo a estabelecer a sessão FTP do que a transferir o ficheiro.

HTTP - Hypertext Transfer Protocol

Embora existissem versões anteriores em uso, a primeira versão completamente funcional e compatível com as seguintes surgiu em 1996, o “HTTP 1.0”.
O objetivo do HTTP é proporcionar uma forma expedita de transferir ficheiros, segundo o modelo cliente servidor, com especial predomínio para as transferências no sentido servidor para cliente.

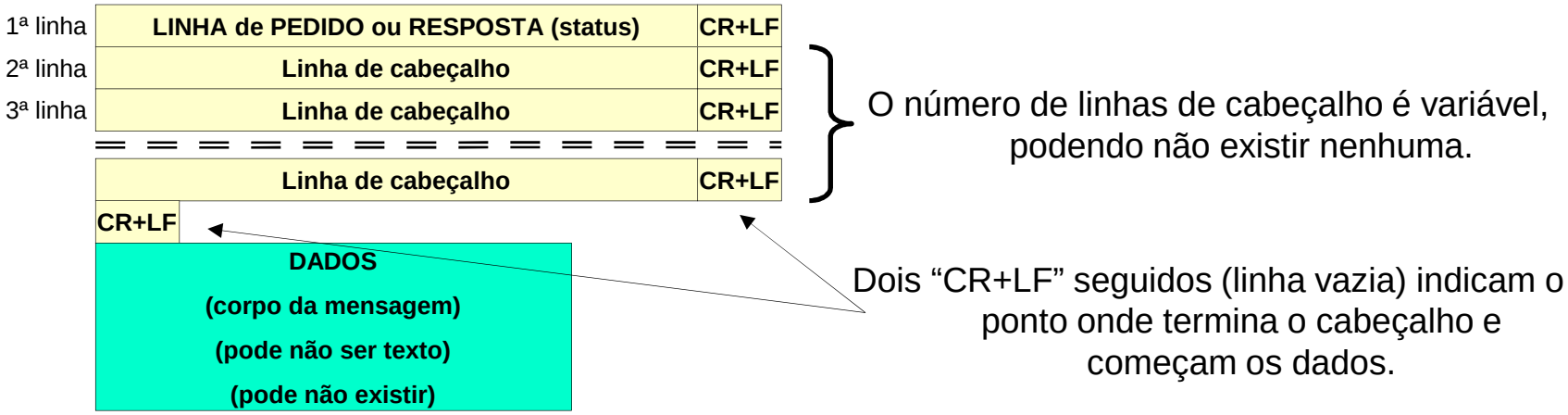
Normalmente o cliente começa por estabelecer uma ligação TCP com o servidor que está à espera no porto número 80, depois de estabelecida a ligação o protocolo HTTP usa-a para comunicação entre as duas entidades. Seguindo o modelo cliente / servidor, o cliente envia um “pedido HTTP” e servidor devolve uma “resposta HTTP”.



O “HTTP 1.1” define vários tipos de pedido: OPTIONS; GET; HEAD; POST; PUT; DELETE; TRACE e CONNECT. Nem todos são suportados pelo “HTTP 1.0”.

Mensagens HTTP

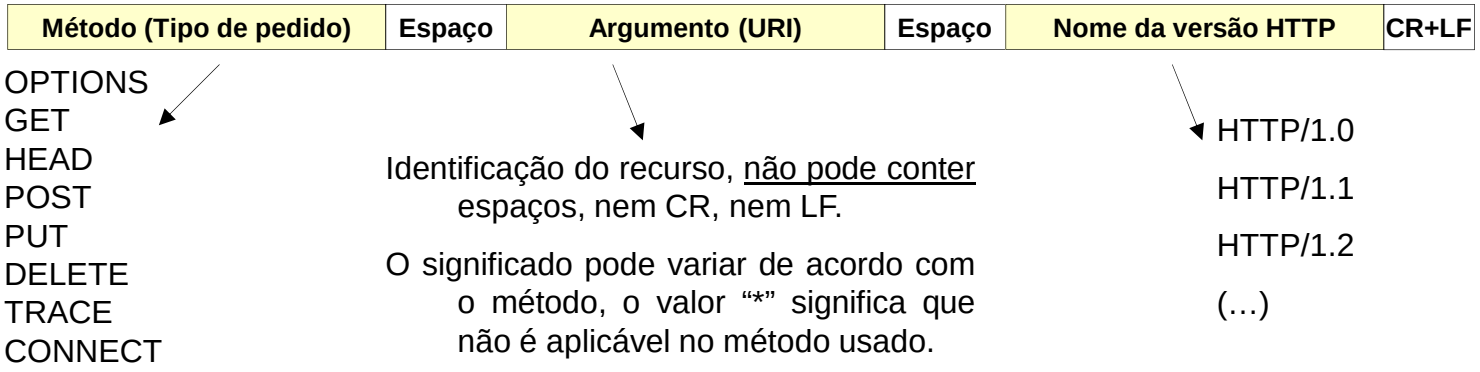
As mensagens HTTP (pedidos e respostas) obedecem a um formato bem definido, organizado em linhas de texto de comprimento variável, terminadas sempre por CR+LF (CR=Carriage Return ; LF=Line Feed):



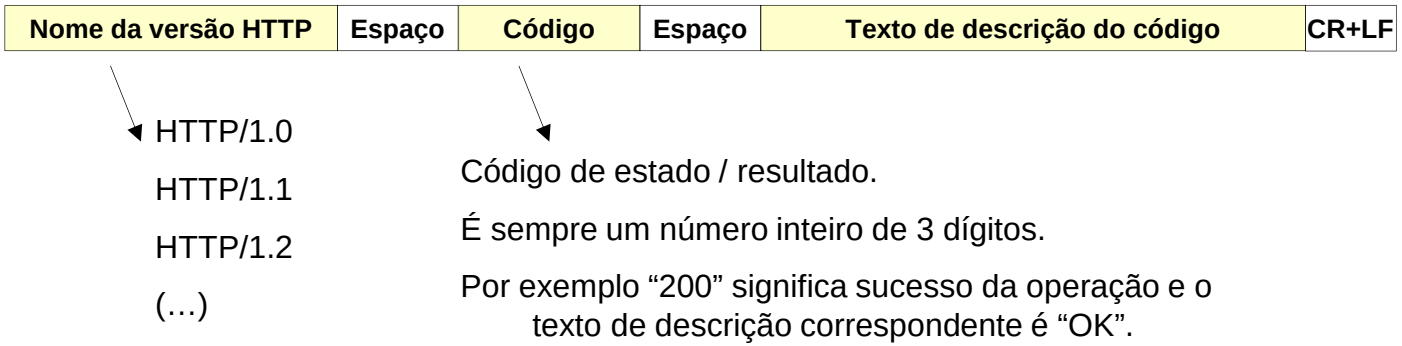

```
CR = 13 (0x0D) = \r
LF = 10 (0x0A) = \n
```

HTTP – Pedidos e respostas

- A linha de pedido (1ª linha do pedido) tem o seguinte formato:



- A linha de resposta / estado (1ª linha da resposta) tem o seguinte formato:



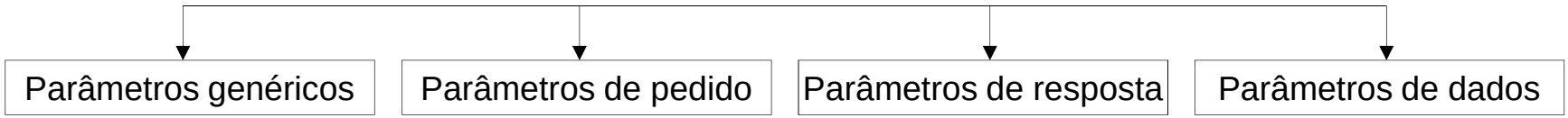
HTTP – Linhas de cabeçalho (parâmetros de cabeçalho)

As linhas de cabeçalho servem para implementar diversas funcionalidades do protocolo HTTP, a sua forma geral é:

Nome do parâmetro	:	Valor do parâmetro	CR+LF

O “nome do parâmetro” é um identificador com significado especial para o protocolo HTTP, a interpretação deste identificador não é sensível a maiúsculas e minúsculas. Segue-se imediatamente o sinal de dois pontos.
O “valor do parâmetro” pode ser precedido de caracteres brancos (ESPAÇO, TAB, etc.) que deverão ser ignorados. O valor de um parâmetro pode ocupar mais do que uma linha (“folding”), sempre que após o “CR+LF” surge um caractere branco, então trata-se de uma continuação da linha anterior e não uma nova linha.

Tanto os pedidos como as respostas podem conter parâmetros de cabeçalho, mas nem todos, os parâmetros de cabeçalho dividem-se em 4 categorias:



(General Header Fields)

Aplicam-se tanto a pedidos como respostas.

(Request Header Fields)

(Response Header Fields)

Uma explicação detalhada e exaustiva dos vários parâmetros pode ser consultada da RFC2616 (HTTP/1.1)

(Entity Header Fields)

Elementos relativos aos dados transportados ou recurso sobre o qual se pediu informações.

HTTP – General Header Fields

Alguns dos parâmetros genéricos de cabeçalho mais usados são:

Cache-Control	Este parâmetro permite controlar os vários atributos do armazenamento da informação ao longo do percurso entre cliente e servidor, alguns valores possíveis são “no-cache”; “no-store”; “max-age”; “public”; “private”.
Connection	O valor “close” indica que a ligação TCP deve ser fechada após a transferência dos dados. No HTTP/1.1 as ligações TCP entre cliente e servidor podem ser mantidas para além da satisfação do primeiro pedido. Este é o comportamento por omissão no HTTP/1.1 e seguintes. Se um pedido ou uma resposta contém a linha de cabeçalho “Connection: close” a ligação será quebrada.
Date	Contém a data/hora (em formato HTTP) a que a mensagem foi produzida.
Pragma	O valor “no-cache” indica que não devem ser usadas cópias em “cache”, é equivalente ao valor “no-cache” no parâmetro “Cache-Control”. O HTTP/1.0 não suporta o parâmetro “Cache-Control”.
Warning	Permite adicionar à mensagem um aviso, entre outros elementos o aviso contém um código numérico de 3 dígitos, a identificação da entidade que o adicionou, uma descrição em texto e a data.

HTTP – Entity Header Fields

Estes parâmetros aplicam-se ao documento transportado ou referido, os mais usados são:

Allow	Contém um conjunto de identificadores de métodos (GET; POST; etc.) que são aceites para o documento em causa.
Content-Encoding	Contém o identificador do método de codificação aplicado ao documento, por exemplo “gzip”.
Content-Language	Contém o identificador da linguagem associada ao documento, por exemplo “pt-PT” ou “pt-BR”.
Content-Length	Contém o tamanho do documento, em octetos (bytes).
Content-MD5	Contém o resultado da aplicação do algoritmo MD5 ao documento, serve para controlo de integridade, mas não garante qualquer tipo de segurança.
Content-Type	Identifica o conteúdo do documento, o identificador é constituído da seguinte forma: “TIPO/SUB-TIPO ; parâmetros”. Exemplo: “Content-Type: text/html; charset=ISO-8859-4”
Expires	Contém a data/hora (formato HTML) em que o documento em “cache” perde a validade.
Last-Modified	Contém a data/hora (formato HTML) em que o documento foi alterado pela última vez. Normalmente o servidor obtém este valor do sistema de ficheiros.

HTTP – Request Header Fields

Parâmetros que podem acompanhar os pedidos HTTP, os mais usados são:

Accept	Contém um conjunto de identificadores de tipo (“Content-Type”) que serão aceitas como resposta ao pedido.
Accept-Charset	Idêntico ao anterior, mas para identificadores de conjuntos de caracteres.
Accept-Encoding	Idêntico ao anterior, mas para tipo de codificação (“Content-Encoding”).
Accept-Language	Idêntico ao anterior, mas para tipo de linguagem (“Content-Language”).
Authorization	Serve para autenticação do utilizador. Contém um “string” de validação, por exemplo contendo o nome de utilizador e respetiva “password”. Torna-se necessário após uma resposta “401 Unauthorized” do servidor.
From	Contém o endereço de correio eletrónico do utilizador.
If-Match	Permite definir uma condição baseada numa propriedade do documento “Entity Header” para que o pedido seja atendido.
Referer	Contém o URI do documento de onde partiu a referencia ao URI pedido.
User-Agent	Contém a identificação da aplicação que emitiu o pedido, normalmente um “BROWSER”.
Cookie	Contém um par “nome=valor” que foi fornecido pelo servidor e serve para este identificar a sessão do cliente.

HTTP – Response Header Fields

Alguns dos parâmetros específicos que podem acompanhar as respostas HTTP são:

Location	Contém o URI absoluto do documento pedido.
Retry-After	Associado a uma resposta “503 Service Unavailable” ou a uma resposta “3xx”, indica que o cliente deve voltar a enviar o pedido mais tarde.
Server	Contém um texto de identificação da aplicação servidora. Exemplo: “Apache/1.3.27 (Unix) (Red-Hat/Linux)”
WWW-Authenticate	Acompanha a resposta “401 Unauthorized“. Contém uma identificação do método de autenticação que o cliente deve usar e parâmetros associados ao processo. Estão previstos dois métodos de autenticação: “Basic” – neste caso o conjunto USERNAME/PASSWORD são enviados em forma legível no pedido através do parâmetro “Authorization”. Apenas é aceitável se usado sobre TLS (HTTPS). “Digest” – forma segura de autenticação em que é enviado o resultado da aplicação do algoritmo MD5 e não a PASSWORD.
Set-Cookie Set-Cookie2	Contém um par “nome=valor” que o cliente deve guardar e fornecer em todos os pedidos subsequentes com este servidor.

HTTP/1.1 – Métodos OPTIONS e GET

OPTIONS	Espaço	Argumento (URI)	Espaço	HTTP/1.1	CR+LF

Serve para obter uma lista de métodos aceites para acesso a um dado URI, ou genericamente pelo servidor, nessa caso o URI deve ser um asterisco.
Na resposta "200 OK" será incluído o parâmetro "Allow:" com a lista de métodos suportados e eventualmente outros parâmetros que sirvam para definir as capacidades do servidor.

GET	Espaço	Argumento (URI)	Espaço	HTTP/1.1	CR+LF

Serve para obter o documento identificado por "URI".
Se o URI identificar uma aplicação (Ex.: ficheiro executável), então o servidor poderá executar essa aplicação e devolve o seu resultado ("output"). Esta técnica é conhecida por CGI (Common Gateway Interface).
Neste contexto dos CGI podem ser fornecidos dados pelo cliente ao servidor (normalmente recolhidos por um formulário) esses dados têm de ser acrescentados ao URI, sendo separados do nome da aplicação por um ponto de interrogação. O que se segue ao ponto de interrogação é conhecido por "query string" e pode ser composto por vários campos com valores atribuídos separados por "&".

Exemplo: "http://www.server1.net/login?username=teste&password=nenhuma&departamento=5"

O método GET não é a forma ideal para fornecer dados a um CGI no servidor. Por um lado os valores dos campos aparecem visíveis no URI o que nem sempre será o mais adequado sob o ponto de vista de privacidade, além disso apenas são suportados valores em formato de texto. Por outro lado alguns clientes e servidores impõem limites ao tamanho do URI. O método POST é mais adequado para este tipo de aplicação.

HTTP/1.1 – Métodos HEAD, POST, PUT e DELETE

HEAD	Espaço	Argumento (URI)	Espaço	HTTP/1.1	CR+LF

Serve para obter uma resposta exatamente igual à que seria obtida com o método GET, mas o documento não é enviado. Todos os parâmetros de cabeçalho devem ser iguais aos que seriam obtidos usando o método GET com o mesmo URI.

POST	Espaço	Argumento (URI)	Espaço	HTTP/1.1	CR+LF

O objetivo geral do POST é enviar dados a um URI. A forma como o método é processado é da responsabilidade do servidor, tipicamente o URI corresponde a uma aplicação CGI, a diferença relativamente ao método GET é que os dados são enviados no corpo da mensagem, dessa forma não há restrições quanto ao volume de dados ou tipo de dados enviados.

A técnica de CGI assume uma importância bastante grande na utilização atual do HTTP, foram desenvolvidas ou adaptadas diversas linguagens de programação especialmente para este efeito. Relativamente a linguagens interpretadas (scripts) destacam-se o PHP, Perl, Python e ASP.

PUT	Espaço	Argumento (URI)	Espaço	HTTP/1.1	CR+LF

Trata-se do método inverso do GET, ou seja serve para colocar um documento no servidor. O nome a dar ao documento é fornecido no URI, o conteúdo do documento é transportado no corpo da mensagem.

DELETE	Espaço	Argumento (URI)	Espaço	HTTP/1.1	CR+LF

Serve para eliminar um documento (URI) do servidor.

HTTP/1.1 – Códigos de resposta

As respostas HTTP podem agrupar-se em 5 categorias:

HTTP/1.1	Espaço	1XX	Espaço	Texto de descrição do código	CR+LF

Os códigos 1XX são de informação, não existiam no “HTTP/1.0”. Exemplos:
“100 Continue” – indica que a primeira parte do pedido foi recebida e que o servidor aguarda algo mais.

HTTP/1.1	Espaço	2XX	Espaço	Texto de descrição do código	CR+LF

Os códigos 2XX indicam sucesso na operação realizada. Exemplos:

“200 OK” – indica sucesso num GET; HEAD ou POST.
“201 Created” – indica que um recurso foi criado, por exemplo com o método PUT.
“202 Accepted” – indica que o pedido foi aceite, mas poderá não ter sido executado de imediato.

HTTP/1.1	Espaço	3XX	Espaço	Texto de descrição do código	CR+LF

Os códigos 3XX indicam uma falha e necessidade de redirecionar a operação. Exemplos:

“300 Multiple Choices” – indica que há várias possibilidades para executar o pedido. Fornece uma lista.
“301 Moved Permanently” – indica que um recurso foi deslocado, nova localização no campo “Location:”.
“307 Temporary Redirect” – situação temporária, nova localização no campo “Location:” do cabeçalho.

HTTP/1.1 – Códigos de resposta (4XX e 5XX)

HTTP/1.1	Espaço	4XX	Espaço	Texto de descrição do código	CR+LF

Os códigos 4XX indicam um erro da responsabilidade do cliente. Exemplos:

“400 Bad Request” – o pedido foi mal formulado e não foi compreendido pelo servidor.
“401 Unauthorized” – o pedido só pode ser satisfeito após autenticação do utilizador.
“402 Payment Required”
“403 Forbidden” – o acesso ao recurso não é permitido.
“404 Not Found” – o pedido foi compreendido, mas o recurso não existe.
“405 Method Not Allowed” – o método usado no pedido não é aceitável para o URI.

HTTP/1.1	Espaço	5XX	Espaço	Texto de descrição do código	CR+LF

Os códigos 5XX indicam um erro da responsabilidade do servidor. Exemplos:

“500 Internal Server Error” – erro grave no servidor que impede o seu funcionamento normal.
“501 Not Implemented” – o pedido necessita de uma funcionalidade não disponível.
“503 Service Unavailable” – o pedido não pode ser satisfeito devido a uma anomalia temporária.
“505 HTTP Version Not Supported” – o servidor não suporta a versão HTTP indicada no pedido.

HTTPS (*Hyper Text Transfer Protocol Secure*)
HTTP sobre TLS (SSL)

O HTTPS não é um protocolo distinto do HTTP, são o mesmo. Enquanto o serviço HTTP é normalmente disponibilizado no número de porto 80, o HTTPS é normalmente disponibilizado no número de porto 443, o *browser* assume esses números de porto em função da parte inicial do URL, respetivamente “http://” ou “https://”, a menos que o número de porto seja indicado explicitamente, por exemplo “http://server.pt:8080”.

A diferença entre o HTTP e o HTTPS reside na camada de transporte que utilizam, enquanto o HTTP utiliza diretamente uma ligação TCP, o HTTPS utiliza indiretamente uma ligação TCP através da camada TLS (*Transport Layer Security*).

A camada TLS proporciona autenticação e privacidade, para efeitos de autenticação são utilizados certificados de chave pública particularmente importantes para garantir a autenticidade dos servidores. Na sequência do processo de autenticação é gerada uma chave secreta que garante a privacidade das transações posteriores entre cliente e servidor.

HTTP(S) como protocolo genérico de comunicação entre aplicações

Dada a sua simplicidade e flexibilidade existe atualmente um forte tendência para a utilização do HTTP(S) como protocolo de uso geral na implementação de comunicações entre aplicações.

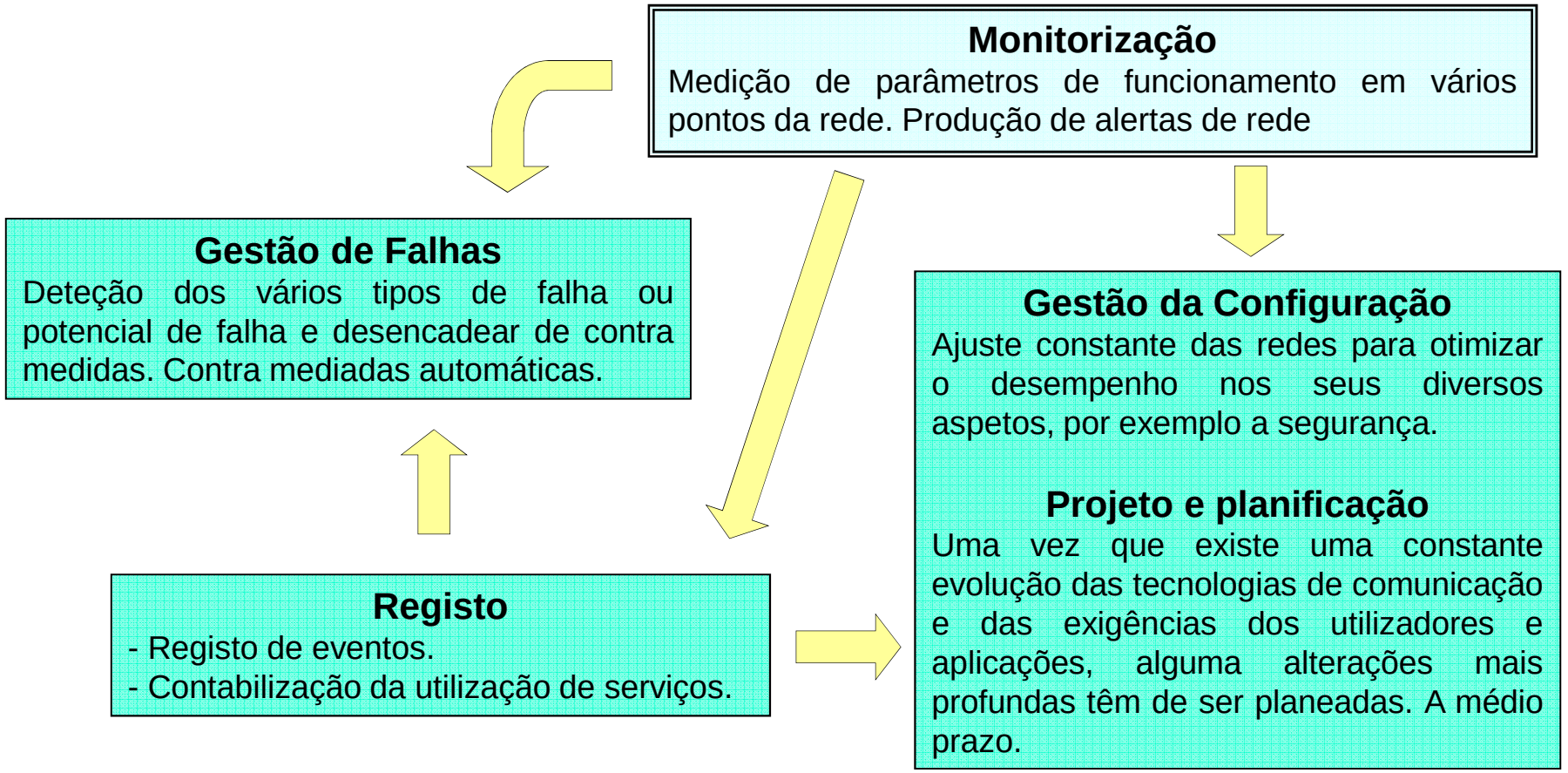
Em lugar de desenvolver um novo protocolo de aplicação sobre a API de *sockets*, simplesmente recorre-se ao HTTP(S), tipicamente com os métodos GET e POST para realizar as transações entre as várias aplicações.

Web Service – é um serviço normalmente baseado em HTTP(S) disponibilizado por uma aplicação para outras aplicações sem qualquer intervenção humana direta. Os dados devem ser transferidos numa representação genérica, tipicamente XML (*eXtensible Markup Language*) ou JSON (*JavaScript Object Notation*), geralmente apoiados no SOAP (*Simple Object Access Protocol*).

Web API – biblioteca para utilização de *Web Services*, atualmente estas APIs tendem a centrar-se nos recurso e não nos serviços segundo a arquitetura REST (*REpresentational State Transfer*) que simplifica o desenvolvimento das aplicações e as torna menos pesadas sem o SOAP.

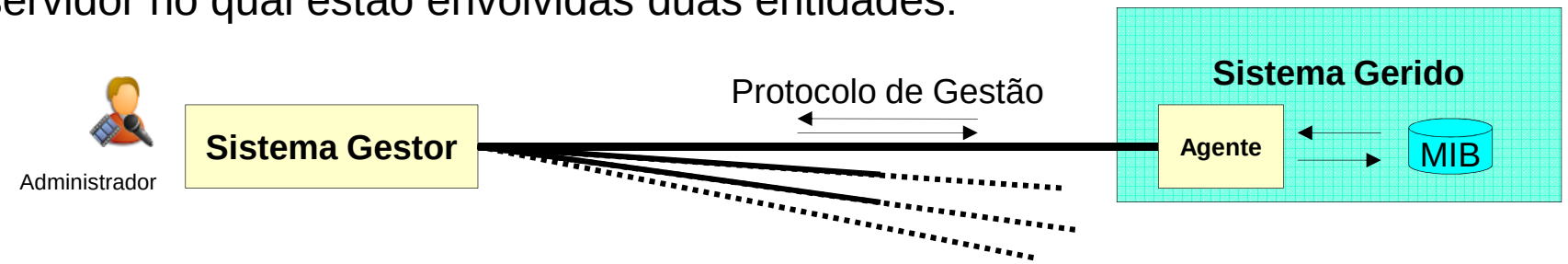
Gestão de redes

Gestão de redes refere-se neste contexto às atividades relacionadas com a **manutenção do bom funcionamento** de um conjunto de redes.



Modelo “Agente – Gestor”

A maioria dos sistemas de gestão de rede adota um modelo do tipo cliente servidor no qual estão envolvidas duas entidades:



O sistema gerido (repetidor, comutador, router, servidor, etc.) utiliza um repositório de informação conhecido por MIB (“Management Information Base”).

O agente é um serviço de rede residente no sistema gerido e que permite o acesso à MIB usando um protocolo de gestão através da rede.

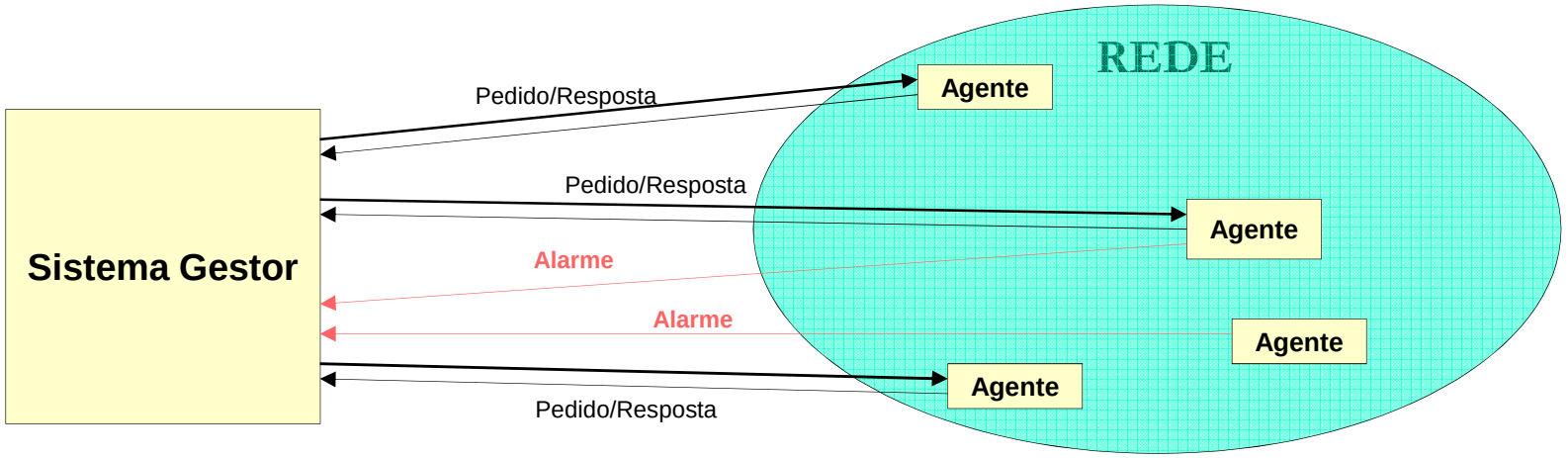
O sistema gestor dialoga com os agentes residentes nos vários dispositivos da rede e constrói uma visão global. A gestão da rede centraliza-se no sistema gestor, que pode ser mais ou menos automatizado.

Protocolo de Gestão

O protocolo de gestão envolve dois tipos de transações:

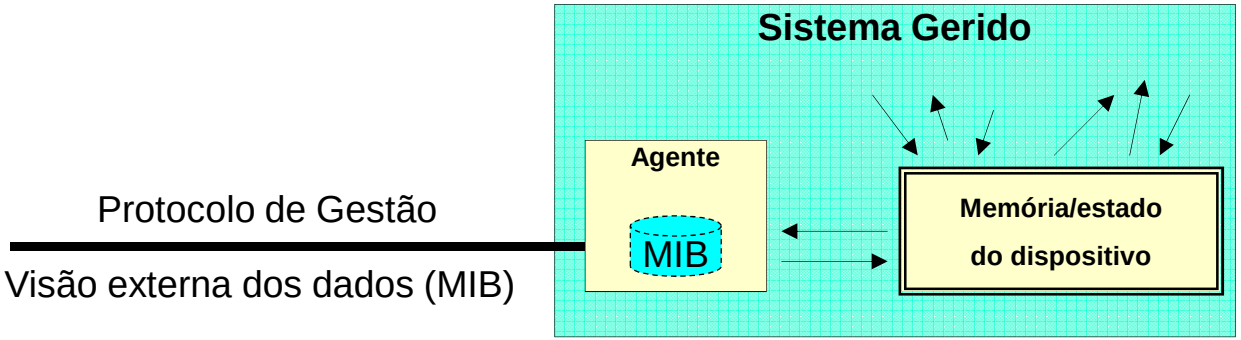
- Pedidos enviados pelo sistema gestor aos agentes, neste caso os agentes assumem o papel de servidores (modelo cliente-servidor). Os pedidos são relativos a operações de gestão, quer de consulta, quer de alteração da configuração.

- Alarmes enviados ao sistema gestor, por iniciativa dos agentes. Normalmente tratam-se de alertas relativos a eventos ocorridos na rede ou até simples registos de atividades.



MIB (“Management Information Base”)

A MIB é o conjunto de dados associado a cada dispositivo de rede com capacidade de gestão.



Na realidade a MIB é uma visão externa (segundo o protocolo de gestão) do conjunto de dados internos do sistema. O “Agente” é responsável por criar essa visão conceptual (base de dados virtual) e interagir externamente segundo ela.
Por ser especialmente adequado para este tipo de aplicação, a definição da MIB é muitas vezes orientada a objetos, sendo o paradigma dos objetos levado mais ou menos longe conforme a implementação em causa.

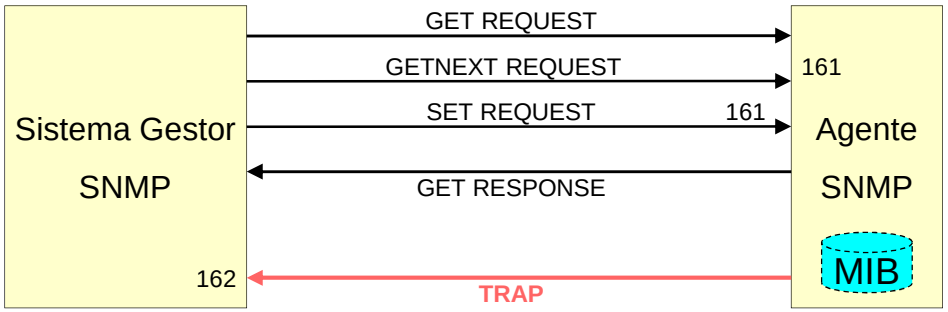
Numa MIB de objetos pura, cada objeto de gestão é uma instanciação de uma classe (definida numa estrutura de classes e subclasses com herança). Cada classe define os métodos apropriados para interagir com o objeto.

SNMP (“Simple Network Management Protocol”)

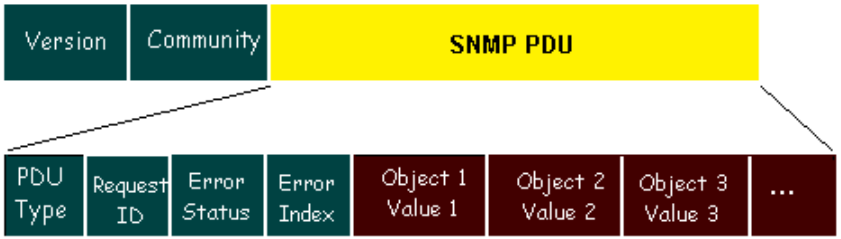
Embora existam outros protocolos de gestão, destacando-se o CMIP (“Common Management Information Protocol “) proposto pelo modelo OSI, a verdade é que associado à pilha de protocolos TCP/IP o protocolo usado é o SNMP. A maioria dos dispositivos de rede atuais suporta o protocolo SNMP.

Estando as redes cada vez mais centradas na pilha TCP/IP e atendendo às várias evoluções que o protocolo SNMP tem sofrido no sentido do seu enriquecimento, não é de esperar a sua substituição no futuro.

A versão 1 (SNMPv1) ainda é muito usada devido à quantidade de dispositivos que não suporta outras versões. No SNMPv1 existem apenas 5 mensagens possíveis, são enviadas sob a forma de datagramas UDP, o Agente recebe pedidos no porto 161, segundo o modelo cliente-servidor e o Gestor recebe alarmes (“Traps”) no porto 162.
No SNMPv1 os objetos da MIB são simples variáveis. As mensagens “get request” permitem obter os valores de objetos da MIB através da mensagem “get response”.
A mensagem “set request” permite alterar o valor dos objetos, sendo confirmada por uma “get response”.
Na MIB os objetos são armazenados em sequência, a mensagem “getnext” permite uma consulta sequencial.



SNMPv1 - Mensagens



Todas as mensagens SNMPv1 seguem um formato geral, o campo “Version” identifica a versão do protocolo, o valor zero identifica o SNMPv1.
O campo “Community” contém uma cadeia de caracteres que pode ser usada para controlo de acesso.

“PDU Type” identifica o tipo de mensagem:

0 - GetRequest
1 - GetNextRequest
2 - GetResponse
3 - SetRequest
4 - Trap

O campo “Request ID” identifica um pedido, o valor é repetido na resposta o que permite relacionar a mesma com um pedido formulado anteriormente. Dado que o SNMP usa uma plataforma de transporte não fiável (UDP) esta característica torna-se importante.

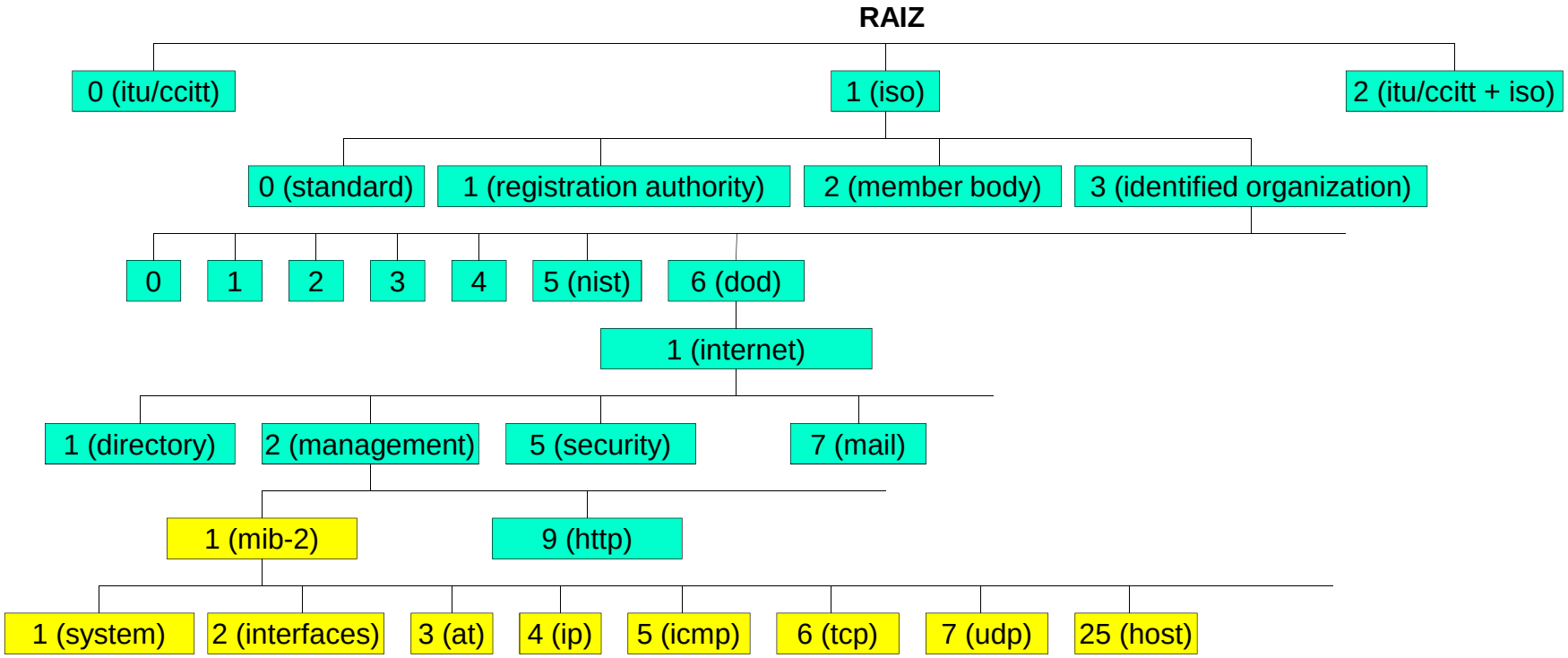
“Error Status” identifica o resultado da operação:

0 - noError
1 - tooBig
2 - noSuchName
3 - badValue
4 - readOnly
5 - genErr

Em caso de erro, “Error Index” pode servir para indicar relativamente a qual dos objetos ocorreu esse erro.
As mensagens tipo 0, 1, e 3 usam sempre o valor zero nos campos de erro. A mensagem “Trap” possui um formato diferente.

OID – Object Identifier

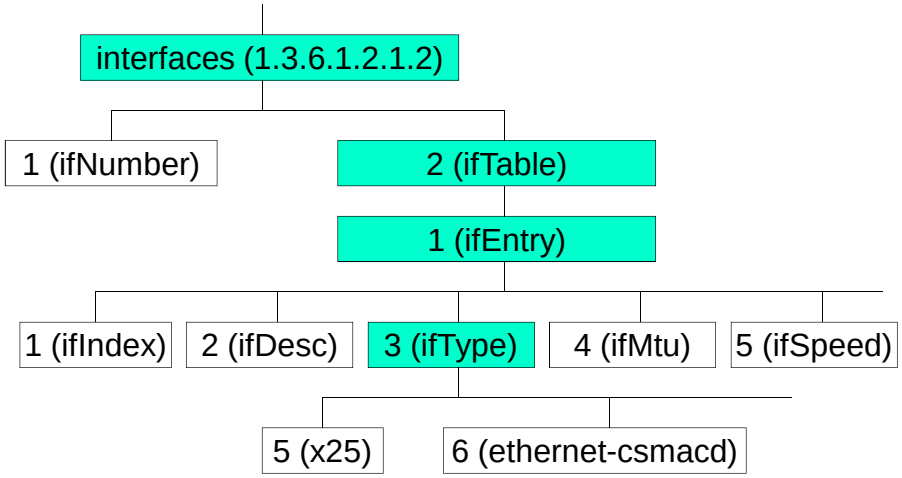
Os objetos (variáveis) residentes na MIB SNMP não são identificados por nomes. A norma ASN.1 (Abstract Syntax Notation One) define um sistema universal de nomeação baseado numa árvore numérica.



Cada tipo de objeto (classe) é identificado pela sequência de números até à raiz, por exemplo o objeto “internet” é identificado por “1.3.6.1”, todos os objetos abaixo começam por esta sequência, os objetos relativos à gestão de rede começam por “1.3.6.1.2.1”.

Exemplo: “interfaces”, OID = 1.3.6.1.2.1.2

No ramo das interfaces (1.3.6.1.2.1.2) está definido o tipo de objeto “ifNumber” que contém o número de interfaces existentes.
O tipo ifTable é uma tabela de objetos do tipo ifEntry (1.3.6.1.2.1.2.2.1), cada objeto do tipo ifEntry contém o objeto ifIndex que contém o número da interface, o seu valor pode ir de 1 a ifNumber e não pode haver números repetidos.



O protocolo SNMP usa os OID, contudo os OID referem-se a classes de objetos e não instancias de objetos como os que existem na MIB do agente. Uma vez que podem existir várias instancias da mesma classe, o SNMP acrescenta mais um número para identificar a instancia, começando por zero para a primeira instancia.
Por exemplo para saber quantas interfaces de rede um dispositivo tem envia-se o pedido “GET 1.3.6.1.2.1.2.1.0”.
Em tabelas o SNMP usa determinados valores dos elementos da tabela para os identificar. No caso do “ifTable” usa o “ifIndex”, portanto ifDesc.8 ou 1.3.6.1.2.1.2.2.1.2.8 representa a descrição da oitava interface que o sistema possui.
Outras classes de objetos tipo tabela são: “atTable”, “ipAddrTable”; “ipRoutingTable”; “tcpConnTable” e “egpNeighTable”. Para todas elas o SNMP define regras especificas para identificar cada elemento da tabela.

Segurança no SNMP

No SNMPv1 e SNMPv2c os mecanismos de segurança são muito elementares, normalmente é possível definir dois tipos de acesso: “leitura apenas” e “leitura e escrita”. Os dois tipos de acesso são associados a “comunidades” distintas. O identificador (nome) da comunidade serve ele próprio como “password”, além disso não existe qualquer tipo de cifragem o nome da comunidade (“password”) circula pela rede sem proteção.
O SNMPv2p e SNMPv2u diferem do SNMPv2c pela existência de mecanismos de segurança mais sofisticados, mas não tiveram grande sucesso.

No SNMPv3 os mecanismos de segurança foram finalmente revistos. O SNMPv3 usa criptografia simétrica para garantir quer a autenticação (e controlo/diferenciação de acesso) quer a privacidade. Na implementação destes mecanismos de segurança está implícita a distribuição previa manual de um segredo (chave secreta) entre os dois extremos da ligação, eventualmente a através de uma “PASSWORD” do administrador.

Apesar da evolução ainda há algumas limitações importantes, a autenticação baseia-se em MD5 ou SHA e a cifragem usa DES.

Tanto a autenticação como a cifragem são opcionais, mas para ter cifragem é obrigatória a autenticação pois a chave de secreta para a cifragem é gerada durante a autenticação.

SNMPv2c, SNMPv3 e RMON

Além de alterações e atualizações na MIB, o SNMPv2 introduz novas mensagens:

- “Getbulk Request” : serve para obter volumes de informação da MIB superiores aos permitidos pelo “Get Request”.

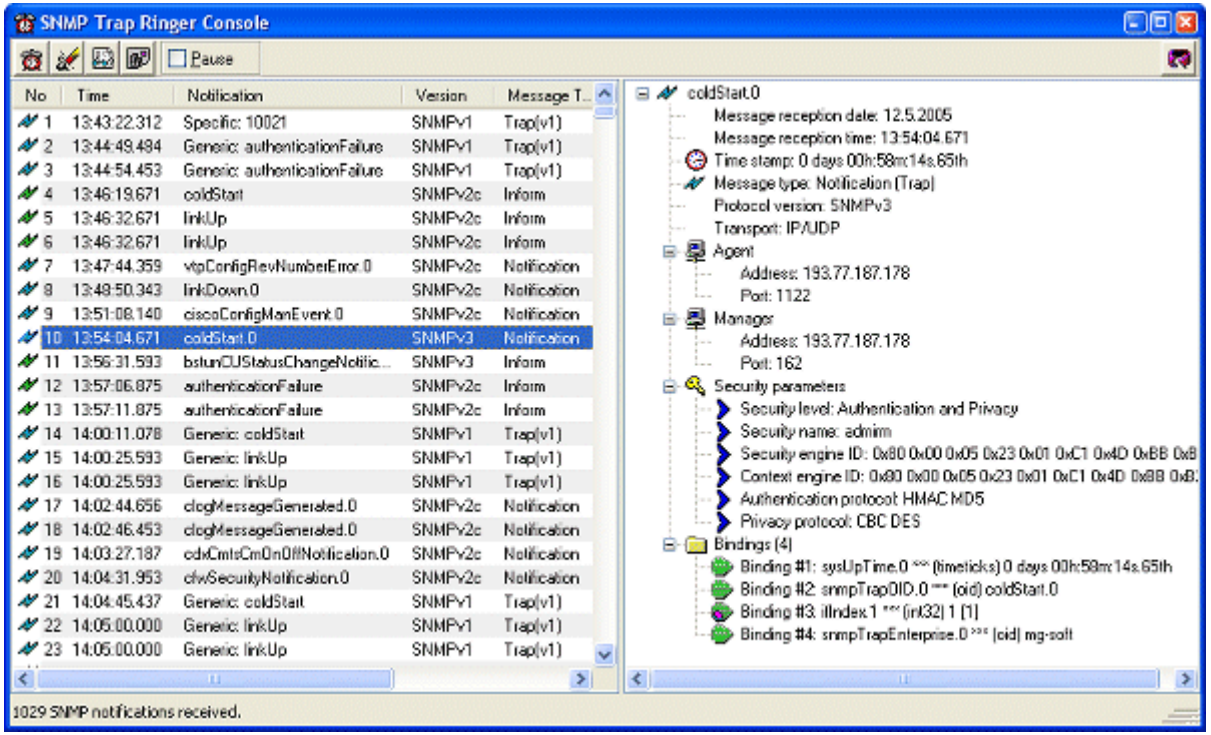
- “Info Request”: equivalente à mensagem “Trap”, mas com confirmação da receção.

As novas versões vieram dar suporte a uma maior variedade de aplicações, nomeadamente em redes de muito grande dimensão onde a recolha de informação tem de ser hierarquizada. Nesse domínio o suporte de ligações agente-agente traz novas possibilidades.

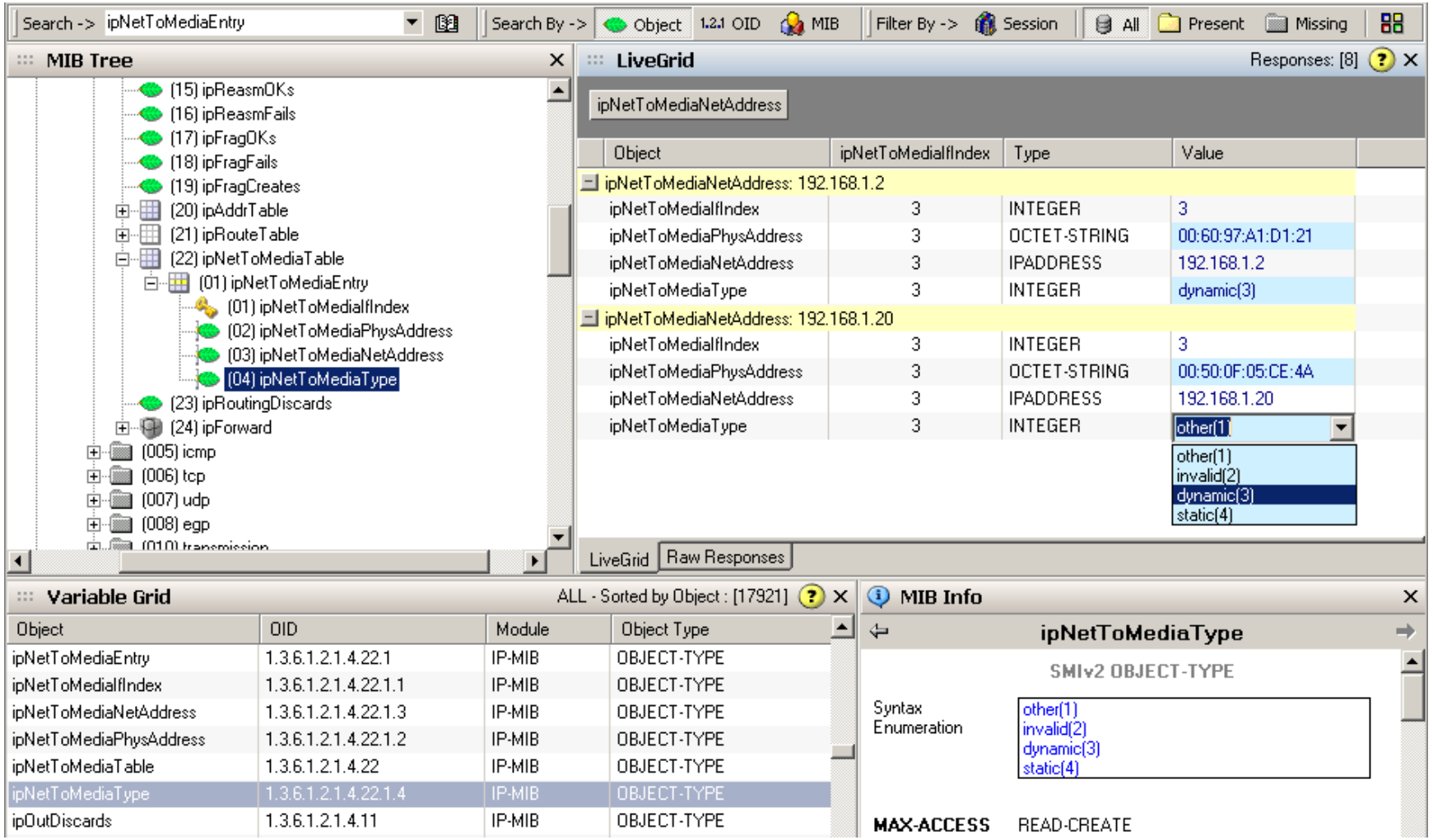
Os dispositivos com capacidade RMON (“Remote monitoring”) funcionam como gestores locais recolhendo informação dos agentes próximos via SNMP. Além disso têm capacidade de monitorizar diretamente a rede (funcionamento como sonda) guardando todas estas informações numa MIB apropriada (MIB RMON).

As MIB RMON podem depois ser consultadas pelo gestor SNMP central.

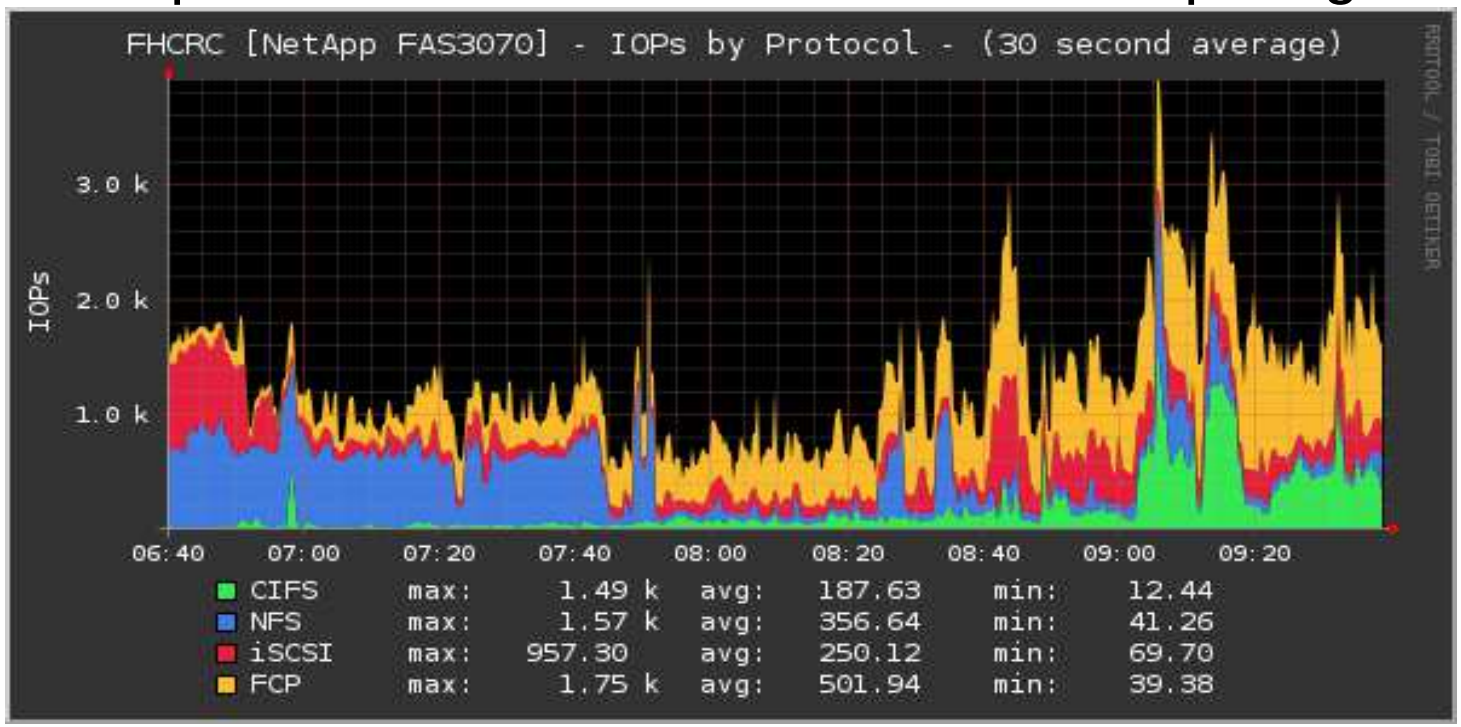
Exemplos SNMP – “Traps”

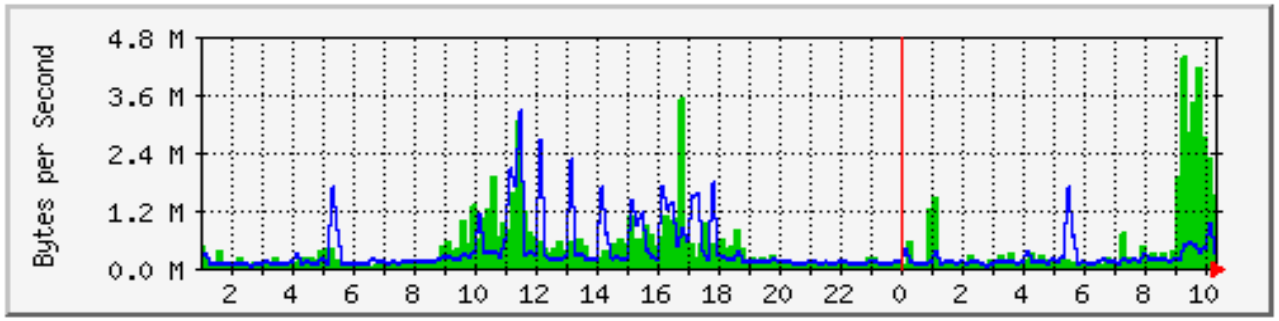


Exemplos SNMP – MIB



Exemplos SNMP – Gráficos desenhados pelo gestor





Max In:4416.5 kB/min (76.7%) Average In:490.1 kB/min (8.5%) Current In:1530.6 kB/min (26.6%)
Max Out:3281.8 kB/min (57.0%) Average Out:352.7 kB/min (6.1%) Current Out: 336.0 kB/min (5.8%)