GESTÃO DE REDES E SISTEMAS POR SCRIPTS

Paulo Pereira, **Pedro Pessoa, Luís Agostinho**

Instituto Superior Técnico

INESC - R. Alves Redol, 9. 1000-029 LISBOA

Tel.: +351-1-3100345, Fax: +351-1-3145843.

E-mail: prbp@inesc.pt, prp@pobox.com, lmmago@camoes.rnl.ist.utl.pt

Paulo Pinto

Faculdade de Ciências e Tecnologia.

Universidade Nova de Lisboa.

UNINOVA, Quinta da Torre. 2825 Monte da Caparica.

Tel: +351-1-2948545, Fax: +351-1-2948532.

E-mail: pfp@uninova.pt

Sumário

Apresenta-se uma aplicação de gestão baseada em *scripts*, que permite ao gestor organizar, configurar e executar *scripts* de gestão. Novos *scripts* são criados num ambiente de desenvolvimento com operações de elevada semântica tanto para realizar o interface gráfico, como para programar as operações de monitorização, controlo e planeamento.

1. INTRODUÇÃO

As redes de computadores têm crescido em número, dimensão, complexidade e variedade de equipamentos. Para serem geridos, a aproximação standard é a existência nos equipamentos de uma base de dados de informação de gestão (MIB) mantida por um agente de gestão, à qual as aplicações de gestão acedem através de um protocolo de gestão de redes (SNMP na comunidade Internet, e CMIP na comunidade OSI). No entanto, existe uma grande distância entre o tipo de informação existente nas MIBs e o tipo de conceitos que tornariam a tarefa de um gestor eficiente.

Em trabalho anterior [1], descreveu-se uma plataforma de gestão de redes que permite suportar redes de dimensão e complexidade arbitrária, pela utilização de uma arquitectura hierárquica distribuída de gestores intermédios, que usa domínios para agrupar os objectos geridos, e delegação de *scripts* para automatizar e descentralizar as tarefas de gestão. Porém, as operações de gestão tinham de ser programadas na linguagem escolhida, Java, usando a Java Management API. Verificou-se que, se bem que seja uma linguagem com um bom ambiente de desenvolvimento, corresponde a um nível de abstracção muito baixo, onde as interacções do protocolo de gestão têm de ser programadas explicitamente, exigindo grandes conhecimentos da parte do gestor/programador, o que limita seriamente as possibilidades de desenvolvimento de novos *scripts*.

Neste trabalho procurou reduzir-se a diferença semântica entre as operações oferecidas ao gestor, que se pretende que possam estar no grau de abstracção mais elevado possível, e as operações do protocolo de gestão que manipulam directamente a MIB. Para isso, desenvolveu-se uma aplicação de gestão que permite ao gestor seleccionar facilmente os *scripts* a executar, configurá-los e recolher os seus resultados. Além disso, oferece-se ao programador um ambiente de desenvolvimento de *scripts* com operações de gestão de elevada semântica, e ainda operações para

a fácil criação de um interface de execução gráfico, permitindo-lhe trabalhar a um nível de abstracção tão alto quanto desejado.

Na secção 2 analisam-se algumas linguagens de gestão de redes propostas pela comunidade científica, justificando-se aquela que foi tomada como base para este trabalho. Na secção 3 apresenta-se o ambiente de desenvolvimento de *scripts*, e descreve-se a linguagem disponibilizada ao programador. Na secção 4 apresenta-se a aplicação de gestão que permite gerir os *scripts*, oferecendo o interface com o gestor. Na secção 5 apresentam-se alguns exemplos de *scripts* de gestão que permitem automatizar algumas funções de gestão. Finalmente, na secção 6 apresentam-se algumas conclusões e propostas de trabalho futuro.

2. LINGUAGENS DE GESTÃO

As linguagens para programação de aplicações de gestão mais simples são interfaces de programação (API) para o protocolo de gestão. Exemplos são: Java Management API [2] para a linguagem Java; WinSNMP [3] específica para ambientes MS Windows; SNMP++ [4] sob a forma de classes para C++; UCD-SNMP [5] com uma biblioteca para unix e MS Windows; Systems Management Scripting Language (SMSL) [6], norma internacional do ITU-T; SNMP Script Language [7], draft expirado do IETF; Scotty [8] que é uma extensão à linguagem Tool Command Language (Tcl) para aplicações de gestão, cobrindo entre outros protocolos o SNMP e o CMIP.

Outras linguagens são construídas sobre estas, como [9] que propõe uma linguagem DEAL baseada em SQL, que permite manipular tabelas para além dos procedimentos normais de gestão. Esta linguagem é traduzida para um *script* SNMP baseado em Tcl, que por sua vez é interpretado pelo Scotty. [10] apresenta uma linguagem para folhas de cálculo que inclui especificações de recolha de informação e de eventos além dos procedimentos normais de gestão. [11] propõe uma linguagem baseada em CCS (Calculus of Communicating Systems) para descrever operações de monitorização e controlo. [12] propõe uma camada de Regras Sintácticas Básicas por cima de uma infra-estrutura SNMP, que pode ser estendida com módulos especializados para implementar tipos específicos de aplicações de gestão. As extensões podem incluir eventos, acções e definições de procedimentos tais como recolha de informação, definição de limiares de actuação, registo de informação, e geração de alarmes.

Outros métodos de mais alto nível de automatização de operações de gestão são habitualmente baseados em técnicas de inteligência artificial, inteligência artificial distribuída, políticas, ou papeis. Políticas [13] são regras que especificam o que pode (permissão), ou o que deve (obrigação) ser feito. Papeis [14] identificam direitos, deveres, funções, e interacções associadas com uma posição dentro da empresa, resultando na definição de conjuntos de políticas específicos para cada papel. No entanto, políticas e papeis são mais adequados para gestão de segurança, configuração e contabilização, sendo pouco adequadas para gestão de desempenho, falhas e planeamento, às quais foi dada ênfase neste trabalho.

Neste trabalho optou-se por criar um ambiente de desenvolvimento de *scripts* de gestão tanto de redes como de sistemas distribuídos recorrendo à linguagem Tcl/Tk [15] e às extensões proporcionadas pelo Scotty [8]. O Scotty proporciona um ambiente altamente extensível que permite facilmente criar aplicações poderosas com muito poucas linhas de código. O ambiente de desenvolvimento criado, que será descrito na próxima secção, é uma prova disso. Funciona sobre o Scotty, oferecendo um conjunto extra de operações muito simples, e no entanto muito poderosas, para a criação de *scripts* de gestão. Por outro lado, o uso de linguagens interpretadas, como é o caso do Tcl/Tk, não constitui uma perda de desempenho significativa, atendendo às vantagens referidas, principalmente a partir da versão 8.0 que inclui um compilador em tempo de execução que origina transparentemente ganhos de velocidade da ordem de 2 a 20 vezes relativamente à versão interpretada.

3. AMBIENTE DE DESENVOLVIMENTO DE *SCRIPTS*

O ambiente de desenvolvimento de *scripts* construído está representado na figura 1. A aplicação de gestão permite consultar a biblioteca de *scripts* de gestão, as suas descrições, produzir o interface de configuração necessário a cada *script*, lançá-lo e observar os resultados produzidos.

Para programar novos *scripts* é necessário desenvolver os seguintes itens que são descritos em subsecções separadas:

1. o interface de configuração com a descrição do *script* e seus parâmetros.
2. o código do *script*.
3. o interface de execução do *script*, para apresentar resultados da execução.

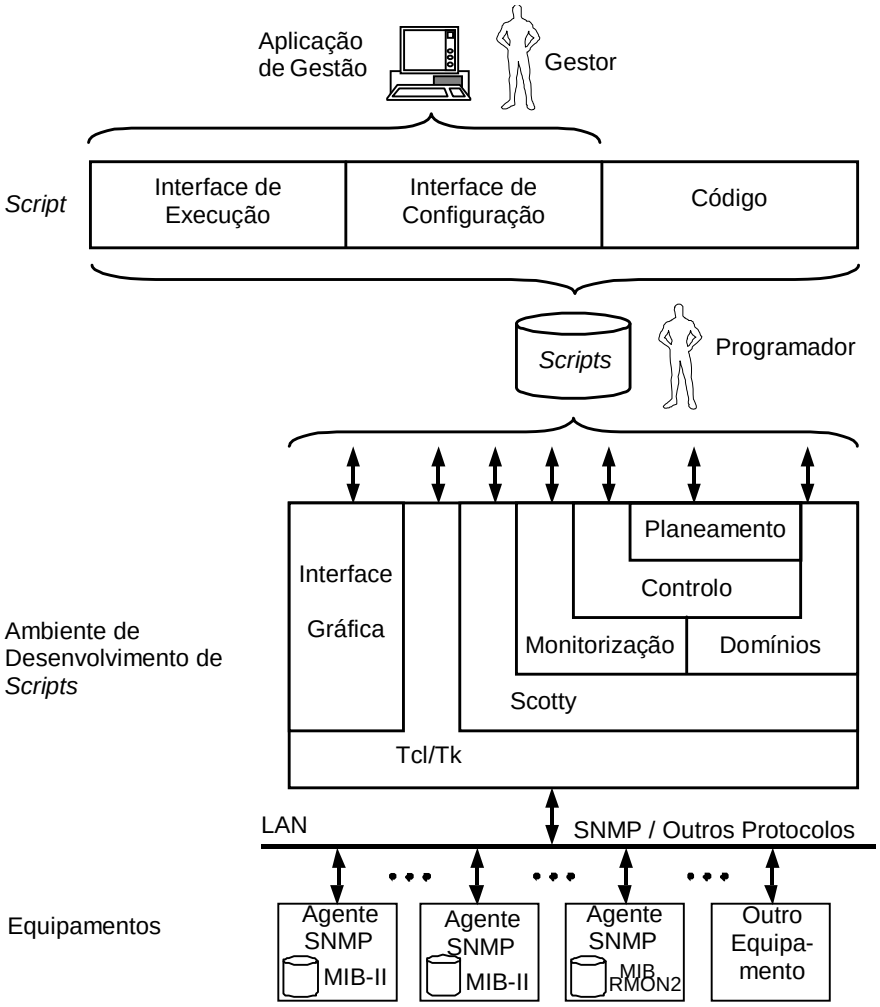

Figura 1. Ambiente de Desenvolvimento de *Scripts*.

O ambiente de desenvolvimento de *scripts* assenta sobre o Tcl/Tk, utilizando as extensões do Scotty e outras extensões desenvolvidas focadas em assuntos mais específicos. Dada a importância na área de gestão do modo como os dados são apresentados ao gestor, e que tipo de informação deve ser mostrada, a parte gráfica foi foco de um desenvolvimento especial. Dentro do próprio ambiente de desenvolvimento de *scripts*, foram desenvolvidos comandos para a fácil criação de um interface gráfico. Outro aspecto coberto é a definição de que equipamentos devem ser geridos de um modo agregado por um único *script*. Existem diferentes aproximações na literatura. A opção tomada é uma variante do conceito de **domínio** [16] que é um agrupamento lógico de equipamentos, ou de outros domínios, para fins de gestão. Finalmente, dada a importância da interacção entre entidades, foram desenvolvidos comandos que permitem (figura 2) a monitorização, isto é, a recolha periódica de informação dos objectos geridos para determinar o seu estado; o controlo, isto é, a execução de

acções com base em eventos obtidos pela interpretação do estado dos objectos geridos; e também operações de planeamento, isto é, a interpretação da história dos estados dos objectos geridos com vista à modificação dos procedimentos de controlo.

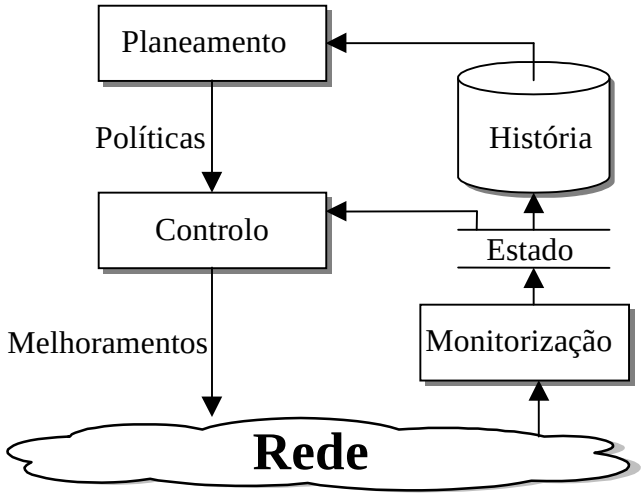

Figura 2. Gestão e Controlo de Redes

3.1 Configuração de *Scripts*

Para construir o interface de configuração dos *scripts* foram desenvolvidos comandos para criar vários tipos de elementos gráficos para receber os parâmetros de configuração. De seguida apresenta-se a descrição de cada um (os parâmetros entre "[...]" são opcionais).

`#$ device label` – cria uma *combobox* com as diversas máquinas do sistema. (uma *combobox* é um menu *pulldown* para seleccionar uma opção de entre várias, como se poderá observar na figura 5 mais à frente)

`#$ domain label` – cria uma *combobox* com os diversos domínios do sistema.

`#$ domordev label` – cria uma *combobox* com as diversas máquinas e os diversos domínios do sistema.

`#$ (device|domain|domordev)list label` – cria uma lista com os elementos indicados (de modo idêntico aos apresentados anteriormente), mas permitindo a escolha simultânea de vários elementos.

`#$ number label default [min max]` – caso não sejam especificados min e max, cria uma *entry box* para introduzir um valor, caso contrário cria uma *scale* para definir um valor.

`#$ option label default [opt ...]` – cria uma *combobox* com os valores fornecidos.

`#$ string label [default]` – cria uma *entry box*, existindo a possibilidade de ser apresentada com um valor já definido.

Todos estes comandos devem ser precedidos de "#$" pois destinam-se a ser interpretados pela aplicação de gestão para produzir o interface de configuração, sendo comentários para o interpretador Tcl.

3.2 Código de *Scripts*

Para escrever o código dos *scripts* podem ser utilizados quaisquer comandos da linguagem Tcl e do Scotty. Foram ainda desenvolvidos vários outros comandos para facilitar operações específicas de gestão de sessões do Scotty, operações de monitorização e controlo, sendo os principais descritos nesta subsecção.

Foi definida uma base de dados com informação sobre os agentes SNMP para gerir as sessões do Scotty. A base de dados contém os nomes de comunidade, valores de *timeouts* e *retries* para cada agente. A base de dados está estruturada em três níveis, um com definições específicas por máquina, outro definindo por subrede, e o terceiro mantendo os valores por omissão. Assim, quando se pretende obter a informação SNMP de uma máquina, procura-se inicialmente na informação específica, tentando encontrar a máquina em causa, e subindo de nível em caso de falha. Esta base de dados é mantida com os comandos `updatePollingInfo`, `getPollingInfo`, `writePollingInfo`, sendo as sessões SNMP criadas e destruídas com os comandos `getsession` e `destroysession`.

O principal comando relativo aos aspectos de monitorização e controlo descritos acima é o comando `pollingLoop` que recolhe um conjunto de variáveis de cada elemento de uma lista de máquinas ou domínios, executando um *script* `call_back` quando a informação correspondente a cada uma das variáveis é recebida. O *script* `call_back` recebe como argumentos o nome da máquina e uma lista com a variável e o respectivo valor como é usual no Scotty. Deve ser construído um *script* `call_back` para processamento dos dados recolhidos adequado às funcionalidades pretendidas. Exemplos são: guardar os dados em ficheiro, calcular expressões sobre as várias variáveis, disparar alarmes quando certos limiares são ultrapassados, etc.

```
pollingLoop lista_maquinas lista_variaveis call_back
proc call_back { nome_maquina varbindlist }  { ... }
```

As sondas RMON [17] e RMON2 [18] contêm informação já processada de acordo com certos critérios. O acesso a campos específicos foi optimizado pela criação de novos comandos. Por exemplo, o seguinte comando cria um relatório na tabela hostTopNControl de uma sonda RMON, espera pelo seu resultado e devolve o endereço MAC do equipamento com maior contribuição para o Top.

```
CriaReport sessaoSNMP variavel duracao num_hosts
```

Um outro comando permite obter as estatísticas de um dado protocolo da tabela alHost de uma sonda RMON2, apresentando-se também um exemplo de utilização deste comando para o caso específico de recolha da informação correspondente aos pacotes do protocolo `http` enviados pela máquina `daryl`:

```
alMon.tcl sonda comunidade_leitura maquina variavel protocol1 [... protocolN]
```

```
alMon.tcl vivian hp_admin daryl outPkts wildcard ip tcp http
```

Outro conjunto de comandos permite, respectivamente, criar eventos em sondas RMON, programar alarmes e os correspondentes eventos devolvendo o índice do alarme criado, e ainda para ficar à espera que um dado alarme dispare:

```
CriaEvento sessaoSNMP descricao_textual tipo IpOwner
```

```
CriaAlarme sessaoSNMP Intervalo Variable SampleType StartupAlarm
RisingThreshold FallingThreshold IPOwner
```

```
Escuta indice comando
```

Tal como explicado anteriormente, as operações de planeamento pressupõem a criação de histórias da evolução de variáveis, para poder detectar tendências de evolução, desvios relativamente ao comportamento normal, e os correspondentes possíveis melhoramentos. Para auxiliar estas tarefas, desenvolveu-se o comando `historico` que recorre ao comando `pollingLoop` anteriormente descrito, para construir bases de dados onde se mantém a informação recolhida periodicamente da rede:

```
historico.tcl lista_maquinas lista_variaveis periodo [ficheiro]
```

3.3 Interface de Execução de *Scripts*

Para além dos outros comandos de Tcl/Tk que podem ser usados para gerar o interface de execução dos *scripts*, pode ser criada uma janela para apresentação do `stdout` com o comando:

```
#$ execfile tkfrontend.tcl script_a_executar
```

Para criar gráficos de barras com os valores de certas variáveis, disponibiliza-se o seguinte comando:

```
tkhist.tcl lista_maquinas lista_variaveis tempo_ciclo num_dados [ficheiro_historico]
```

4. APLICAÇÃO DE GESTÃO

A aplicação de gestão - Another Network Management Tool (ANMT), apresentada na figura 3, permite:

- Entre sessões de trabalho guardar o estado de administração da rede.
- Gerir *scripts* hierarquicamente através de um sistema de pastas.
- Observar os *scripts* disponíveis, configurá-los e ver os resultados da execução.
- Criar uma base de dados onde poderá manter um registo de equipamentos e de respectivas relações.
- Personalizar a língua através de um ficheiro de configuração.

Figura 3. Aplicação de Gestão.

A aplicação de gestão apresenta a lista de *scripts* disponíveis, criando uma janela para apresentar os resultados produzidos por cada *script* em execução. Na lista de *scripts* disponíveis encontram-se duas pastas, cada uma com vários *scripts*. Quando um *script*, ou uma pasta, se encontra seleccionado, aparece no rodapé da janela a descrição correspondente.

Para criar a base de dados com os equipamentos existentes na rede existe uma opção no menu Rede que executa um *script* de autodescoberta da rede. Este pesquisa na rede os equipamentos existentes, determina se têm ou não agentes SNMP e cria uma base de dados com a informação recolhida. Os equipamentos podem então ser classificados em domínios. Os domínios podem ser criados (figura 4) como uma lista de máquinas escolhidas de entre as existentes na rede, ou por uma regra que permita definir domínios em termos de outros domínios já existentes. Ao definir domínios por regra, o novo domínio é definido como um primeiro domínio em união (+), intersecção (X), ou exclusão (-) com outros domínios a especificar. No caso da figura 4, é definido o domínio “Máquinas Unix do Quarto Andar” como o domínio “Máquinas Unix” em intersecção com o domínio “Quarto Andar”.

Figura 4. Definição de domínios por regras.

5. EXEMPLOS

Foram desenvolvidos vários *scripts* de exemplo para demonstrar as funcionalidades do ambiente de desenvolvimento e da aplicação de gestão.

5.1 Broadcast Storm

Um nível anormal de *broadcasts* numa rede é um problema conhecido por tempestade de *broadcasts*, que pode acontecer quando há uma falha num interface de rede de um equipamento. Foi concebido um *script* para detectar e, se possível, corrigir este problema, desactivando ou reinicializando o interface.

O *script* utiliza os serviços de sondas RMON para programar um alarme sobre o número de *broadcasts* ocorridos num dado intervalo de tempo. O interface de configuração apresenta-se na figura 5, sendo criado com os seguintes comandos:

```
#$ number "Nº Máx. Broadcasts:" 1000 0 5000
#$ number "Intervalo:" 10 1 600
#$ number "Tempo Report" 10 1 600
#$ string "Variável:" etherStatsBroadcastPkts.1
#$ device "Sonda:"
#$ string "Expressão:" "(%V - %A) / 2"
```

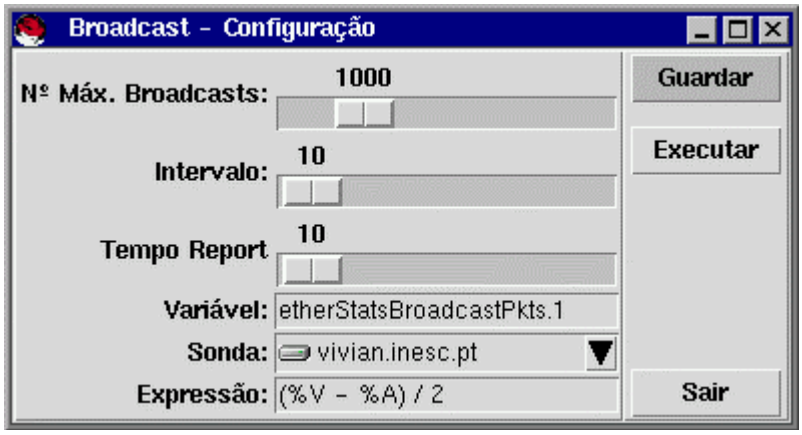

Figura 5. Interface de configuração do *script* Broadcast.

Desta maneira, se o número de *broadcasts* exceder o limiar escolhido é enviado um trap SNMP com o alarme para a aplicação de gestão. Quando este é recebido, o *script* consulta o TopN de *broadcasts* existente na sonda, e caso encontre um equipamento que esteja, de facto, a enviar um número significativo de *broadcasts*, procura na sua MIB qual o interface que causa o problema,

gera um relatório para *log*, e se possível reinicializa esse interface. Para determinar se há um equipamento particular a causar o problema, o valor de *broadcasts* do primeiro equipamento no Top é comparado com a expressão fornecida, onde %V é o valor de *broadcasts* depois do Report, e %A é o número de *broadcasts* antes do Report. No caso da expressão indicada, se o primeiro equipamento do Top for responsável por mais de metade dos *broadcasts*, é considerado em falha.

Este é um exemplo de detecção e correcção de um determinado tipo de falhas que é muito facilmente automatizável através de um *script* de gestão. Pelo facto de se utilizar uma linguagem interpretada, é fácil fornecer expressões, ou comandos através de parâmetros configuráveis, como é o caso da "Expressão" neste exemplo.

5.2 Histórico

O *script* Histórico tem como objectivo manter o histórico de variáveis escolhidas de um conjunto de máquinas ou domínios. Na figura 6 mostra-se o interface de configuração obtido com os comandos:

```
#$ domordevlist "Máquinas:"
#$ string "Variáveis:" etherStatsBroadcastPkts.1
#$ number "Período (seg):" 5 1 30
#$ string "Ficheiro p/ escrita:"
```

onde os domínios são representados por ícones de pastas, e os restantes equipamentos por ícones de terminais.

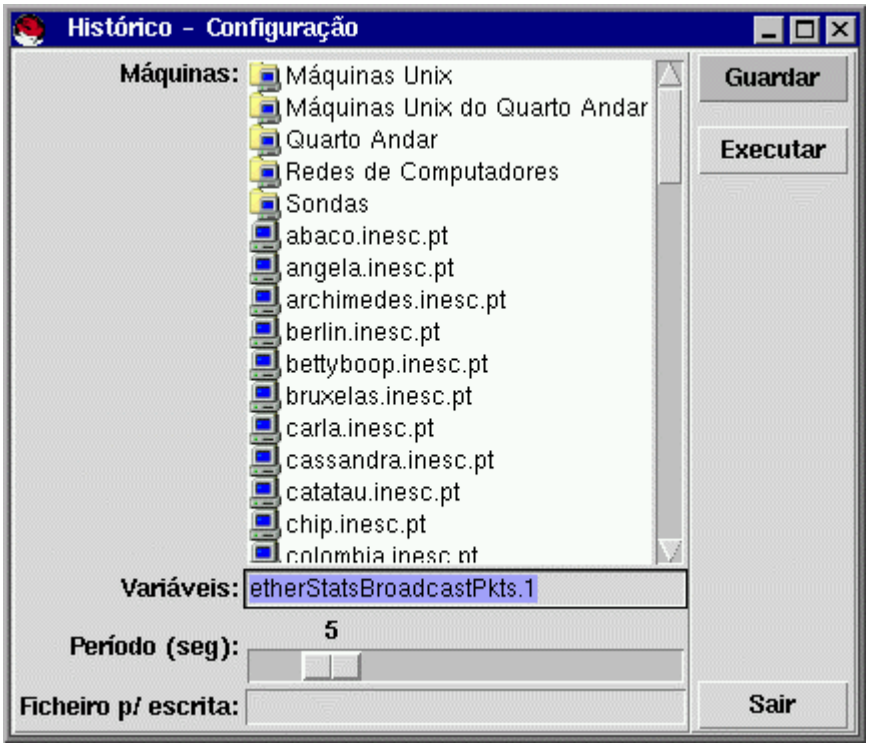

Figura 6. Interface de configuração do *script* Histórico.

A título de exemplo mostra-se parte do código do *script* que recorre ao comando `PollingLoop` com o procedimento de *callback* `MyCB`. Os argumentos `argv` são obtidos do interface de configuração anteriormente descrito, sendo numerados a partir de zero.

```
proc MyCB {addr vblist}  {
  global file
  set date [clock seconds]
# acrescentar "máquina variável data novo-valor" no fim do ficheiro
  puts $file "$addr [mib name [lindex $vblist 0]] $date [lindex $vblist 2]"
  flush $file
}
```

```
proc ciclo {sec}  {
  global argv

  PollingLoop [lindex $argv 0] [lindex $argv 1] MyCB
  after [expr $sec * 1000] "ciclo $sec"
}

# executar o ciclo com o período fornecido
ciclo [lindex $argv 2]
```

Na figura 7 mostra-se graficamente a evolução da variável escolhida, obtida enviando a saída do histórico por um *pipe* para `tkhist.tcl`.

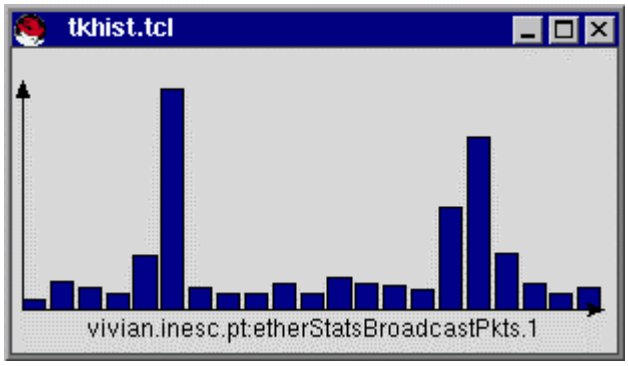

Figura 7. Saída gráfica correspondente a uma variável da MIB.

6. CONCLUSÕES E TRABALHO FUTURO

Construiu-se uma aplicação de gestão de redes e sistemas que permite facilmente ao gestor seleccionar os *scripts* a executar, configurá-los e recolher os seus resultados. Esta aplicação permite classificar as máquinas em domínios, aplicando os *scripts* a domínios dados. Para além disso, oferece-se ao programador um ambiente de desenvolvimento de *scripts* com operações de gestão de elevada semântica, incluindo monitorização, controlo e planeamento. Oferecem-se também operações para a fácil criação de interfaces de configuração e execução gráficos.

Como trabalho futuro, seria útil desenvolver formas de subir mais o grau de abstracção das operações disponibilizadas ao programador, identificando padrões de problemas de gestão a resolver, principalmente no campo de planeamento de operações de gestão onde pouco foi ainda feito. Tal pode ser conseguido com diferentes funções de `call_back` adequadas a cada problema particular a utilizar no ciclo `pollingLoop`. Também seria útil ter domínios inteligentes que fossem calculados em tempo de execução de acordo com determinada regra, por exemplo o domínio das máquinas com agente SNMP que estão ligadas.

AGRADECIMENTO

Este trabalho foi parcialmente suportado pelo PRAXIS XXI, sob o contrato 2/2.1/TIT/1633/95.

7. REFERÊNCIAS

[1] Paulo Pereira, Jorge Sousa, Pedro Teixeira, Paulo Pinto. Plataforma Dinâmica de Gestão de Redes. *Actas da 1ª Conferência sobre Redes de Computadores*, CRC'98, Coimbra, Portugal, 9-10 de Novembro de 1998, pp.31-34.

[2] Javasoft - Java Management API Overview. http://java.sun.com/products/JavaManagement/

[3] ftp://sunsite.unc.edu/pub/micro/pc-stuff/ms-windows/winsnmp/

[4] Kim Banker, Peter E. Mellquist. SNMP++: A Portable Object-Oriented Approach to Network Management Programming. *ConneXions*, **9**(3):35-41, Março 1995. http://rosegarden.external.hp.com/snmp++/

[5] The University of California at Davis SNMP Project. http://ucd-snmp.ucdavis.edu/
[6] Command Sequencer for Systems Management. International Organization for Standardization, Draft International Standard 10164-21, (ITU-T X.753), 1996.
[7] Jeff D. Case, David B. Levi. SNMP Script Language Internet Draft. 9/1/1994. Expired 3/1/95.
[8] Jürgen Schönwälder, H. Langendörfer. Tcl Extensions for Network Management Applications. *3rd Tcl/Tk Workshop*, Toronto, Julho de 1995, pp.279-288.
[9] Simon Znaty, Michel Lion, Jean-Pierre Hubaux. DEAL: A DElegated Agent Language for Developing Network Management Functions. *PAAM'96 The first International Conference and Exhibition on the Pratical Application of Intelligent Agents and Multi-Agent Technology*, Londres, Abril de 1996.
[10] Pramod Kalyanasundaram, Adarshpal S. Sethi, Christopher M. Sherwin, Dong Zhu. A Spreadsheet-Based Scripting Environment for SNMP. *IM'97, Integrated Network Management V*. Chapman & Hall, 1997, pp.752-765. ISBN: 0-412-80960-5.
[11] Nam Hong Cheng. A High-Level Approach in Network Management. *Journal of Network and Systems Management*, Plenum Publishing Corp., **5**(1):73-93, Março de 1997.
[12] Alfredo C. S. C. Brites, Paulo A. F. Simões, Paulo M. C. Leitão, Edmundo H. S. Monteiro, Fernando P. L. Boavida Fernandes. A High-Level Notation for the Specification of Network Management Applications. *Proceedings of the INET'94, the Annual Conference of the Internet Society/JENC5, the 5th Joint European Networking Conference*, Junho 1994, pp.561-1 a 561-8.
[13] M. Sloman, J. Magee, K. Twidle, J. Kramer. An Architecture for Managing Distributed Systems. *Proceedings of the 4th IEEE Workshop on Future Trends of Distributed Computing Systems*, 22-24 de Setembro 1993, Lisboa, Portugal, IEEE Computer Society Press, pp. 40-46.
[14] Emil C. Lupu, Morris Sloman. Towards a Role Based Framework for Distributed Systems Management. *Journal of Network and Systems Management*, vol.5, n. 1, Plenum Press, 1997.
[15] John K. Ousterhout. Tcl and the Tk Toolkit. Addison-Wesley, 1994. ISBN: 0-201-63337-X.
[16] Morris Sloman, Kevin Twindle. Domains: A Framework for Structuring Management Policy. Capítulo 16 de Network and Distributed Systems Management. Addison Wesley. 1994. ISBN: 0-201-62745-0.
[17] S. Waldbusser. Remote Network Monitoring Management Information Base. IETF RFC 1757. Fevereiro de 1995.
[18] S. Waldbusser. Remote Network Monitoring Management Information Base Version 2 using SMIv2. IETF RFC 2021. Janeiro de 1997.