Estruturação da Base de Conhecimento de um Sistema de Apoio à Gerência de Redes Locais

Fernando Luís Dotti
Dotti@Vortex.UFRGS.BR
Liane Margarida Rockenbach Tarouco
Liane@Vortex.UFRGS.BR

Pós-Graduação em Ciências de Computação
Instituto de Informática
Universidade Federal do Rio Grande do Sul
CP 15064, CEP 91501 - Porto Alegre - RS - Brasil

Sumário

O Sistema de Apoio à Gerência de Redes Locais *desenvolvido na UFRGS tem como objetivo fornecer ao Gerente da Rede uma forma inteligente de apoio. O processo é alimentado pelo tráfego da rede e realiza um trabalho de agregação de conhecimento sobre estes dados que culmina no reconhecimento dos problemas que atingem a rede e sugestão de soluções para estes problemas. Para que isto aconteça, o sistema deve contar com uma base de conhecimento apoiando as diversas fases do processo de agregação. Este artigo demonstra a estrutura desta base de conhecimento e como age o proceso de agregação de conhecimento ora referido.*

Abstract

The main goal of the Local Area Network Management Assistant System *developed at UFRGS is to provide an intelligent assistance to the Network Manager. All network traffic is gathered to a process that executes a knowledge aggregation task. This task results in network problems recognition, giving some advice to solve the problems. To function as expected, the system must have a knowledge base assisting the various phases of the aggregation task. This article brings the structure of this knowledge base and describes the way the knowledge aggregation task acts.*

1 Introdução

O fluxo de informações na sociedade moderna é crescente e mais veloz a cada dia. Isto é inegável e inevitável. Cada vez mais as ferramentas de tratamento da informação, ou seja os computadores, necessitam interconexão de forma a proporcionar acesso a

dados importantes e serviços confortáveis. Neste contexto, as redes de computadores agem como catalizadoras do processo de disseminação e tratamento da informação. A demanda por serviços distribuídos cresce a cada dia, fazendo com que as redes cresçam também. Muitas vezes este crescimento é rápido e desordenado, a entropia do sistema aumenta e seu controle torna-se uma tarefa árdua.

Para que se possa exercer maior controle sobre as redes de computadores são necessárias poderosas ferramentas de auxílio. O Gerente da Rede deve se "armar" com estas ferramentas (e métodos) de forma que possa manter a prestação de serviços à que a rede se propõe. Várias ferramentas de auxílio têm surgido, diferindo em características como:

- Escopo de atuação (equipamentos, tipos de redes, protocolos);
- Forma de gerência (distribuída ou centralizada);
- Grau de complexidade do serviço prestado (reconhecimento e diagnóstico do problema ou apenas filtro de eventos para o operador);
- Capacidade de correção automática de problemas (sistema alerta o gerente ou manda procedimento de correção para o equipamento com problemas);
- Área de gerenciamento em que atua (Falhas, Configuração, Contabilização, Performance, Segurança);

Como o assunto "Gerência de Redes" pode ser considerado novo, muitos aspectos ainda estão indefinidos fazendo com que surjam várias formas de solução. O presente trabalho traz uma contribuição às pesquisas atuais apresentando o modêlo de estruturas utilizadas numa base de conhecimento e os mecanismos de agregação de conhecimento de *Um Sistema de Apoio à Gerencia de Redes Locais*, a partir de um protótipo construído na UFRGS.

2 O Sistema

2.1 Características Principais

O principal objetivo do Sistema é acompanhar periodicamente uma rede desprovida de funções distribuídas de gerência, apontando eventuais problemas e suas soluções. Isto deve ser realizado sem alterações nas estruturas de *software* e *hardware* dos demais equipamentos da rede.

Para cumprir com estes objetivos, a forma apontada para acompanhar ou "saber" o que está acontecendo na rede foi a monitoração do tráfego, pois este retrata o conjunto de ações envolvendo distribuição de recursos.

Assim sendo, uma estação da rede é "eleita" para este trabalho. Esta estação captura todo tráfego da rede e o submete continuamente a um processo de agregação de conhecimento. A adoção desta arquitetura leva a algumas características importantes:

- Os demais equipamentos permanecem inalterados, mas controlados;

- O Sistema não depende da rede para exercer suas funções pois não há necessidade de troca de mensagens de gerência;
- O Sistema não age diretamente sobre a rede pois os demais equipamentos não possuem funções de gerência. O Sistema se restringe a detectar problemas, achar soluções e alertar o Gerente;
- Somente algumas características da rede poderão ser gerenciadas pois o sistema só toma conhecimento das ações envolvendo distribuição de recursos (notadas através do tráfego). Os objetos exclusivos de estações remotas não podem ser gerenciados.

2.2 Arquitetura do Sistema

Para cumprir com os abjetivos propostos foi elaborada uma arquitetura para o sistema, descrita nesta secção. O Sistema possui alguns blocos paralelos, que podem ser especificados da seguinte forma em CSP [7]:

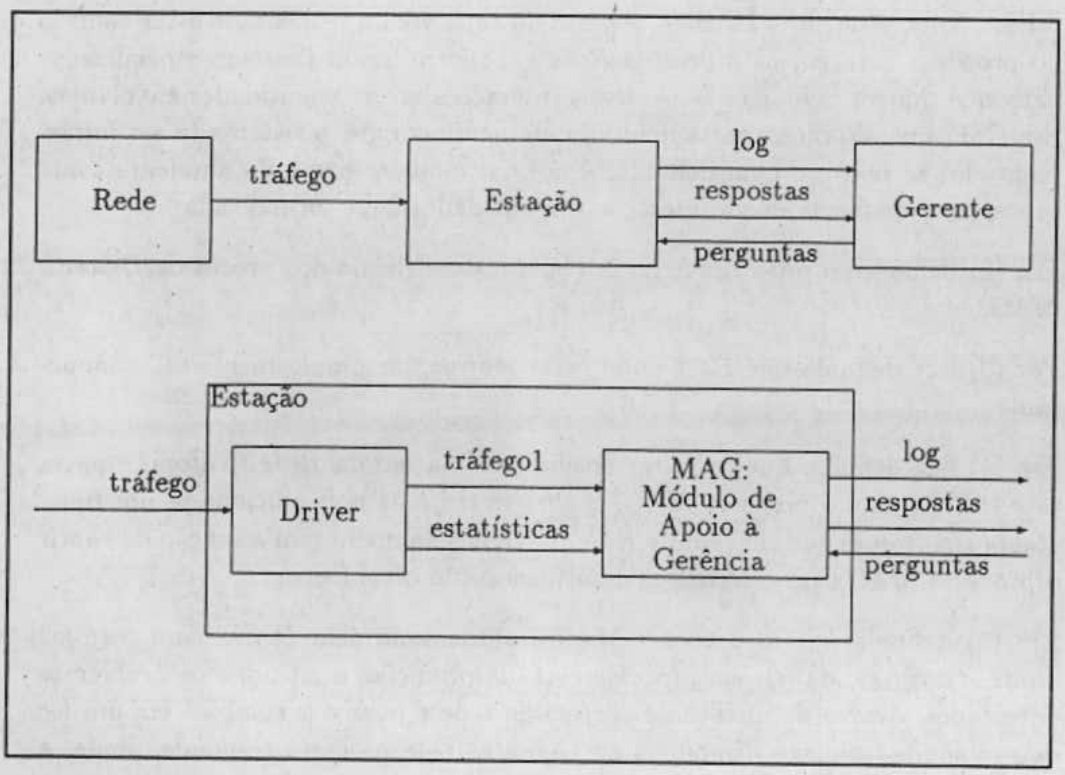

Figura 1: Entidades Concorrentes do Sistema

$$
\begin{array}{lrcl}
(1) & Sistema & = & (Rede \parallel Estacão \parallel Gerente) \\
(2) & Estacão & = & (Driver \parallel MAG) \\
(3) & Rede & = & tráfego \rightarrow Rede \\
(4) & Driver & = & tráfego \rightarrow tráfego1 \rightarrow (Driver \mid \\
 & & & estatísticas \rightarrow Driver) \\
(5) & MAG & = & tráfego1 \rightarrow (MAG \mid \\
 & & & estatísticas \rightarrow MAG \mid \\
 & & & log \rightarrow (MAG \mid \\
 & & & estatísticas \rightarrow MAG) \mid \\
 & & & perguntas \rightarrow respostas \rightarrow MAG) \\
(6) & Gerente & = & (log \rightarrow Gerente \mid \\
 & & & perguntas \rightarrow respostas \rightarrow Gerente) \\
(7) & Restricões & = & R1 + R2 + R3 + R4 \\
(8) & R1 & = & (tr \downarrow tráfego1) \leq (tr \downarrow tráfego) \\
(9) & R2 & = & (tr \downarrow respostas) = (tr \downarrow perguntas) \\
(10) & R3 & = & (tr \downarrow log) \leq (tr \downarrow tráfego1) \\
(11) & R4 & = & (tr \downarrow estatísticas) < (tr \downarrow tráfego1)
\end{array}
$$

A especificação em CSP acima denota o seguinte:

- Em (1) define-se o sistema como a composição de três processos operando em paralelo: *Rede, Estação e Usuário.* Apesar do *software* implementado estar contido no processo Estação, os processos *Rede e Usuário* foram também especificados para que fiquem definidas as possíveis interações entre as entidades envolvidas. Em CSP, quando o comportamento do ambiente em que o sistema (a ser implementado) se insere é bem definido, é normal modelar parte do ambiente como processo(s) de forma que a interação das entidades fique formalizada.
- Em (2) define-se o processo *Estação* como o paralelismo dos processos *Driver e MAG.*
- Por (3) fica definido que *Rede* pode gerar *tráfego*, ou simplesmente estar inoperante.
- Em (4) fica definido que o *Driver* recebe informações da *Rede (tráfego)* e passa este *tráfego* para o processo *MAG*, chama-se *tráfego1* pois adiciona-se um *time-stamp* aos *frames* que chegam de *Rede.* O *Driver* também tem a função de suprir o processo *MAG* com *estatísticas* de utilização do nível físico.
- Em (5) define-se que o processo *MAG* é alimentado pelo *Driver* com *tráfego1 (time-stamp) e estatísticas*, processa estas informações e adiciona os problemas detectados através da análise do *tráfego* de rede e possíveis soluções em um *log* de problemas/soluções disponível ao *Gerente.* Este processo responde, ainda, a perguntas do *Gerente.*

- Por (6) fica definido que o *Gerente* recebe periodicamente um *log* com os problemas detectados pelo sistema e as possíveis soluções, e elabora consultas ao sistema, que retorna as respostas.

- Em (7) foram definidas algumas restrições de consistência para o sistema que podem ser observadas sobre os *traces*.

Os processos *Rede* e *Gerente* ficam, assim, bem definidos. O processo *Estação* (onde está o software construído) é composto pelo paralelismo de dois subprocessos: *Driver* e *MAG*.

O processo *Driver* tem a função de capturar continuamente o tráfego gerado na rede e submeter ao processo *MAG*. O processo *Driver* depende da instalação em que o Sistema roda e não será apresentado neste trabalho.

Parte-se, assim, da situação em que os *frames* que trafegam na rede bem como os dados de estatísticas, adicionados de um *time-stamp*, estão disponíveis em um *buffer*. Este *buffer* alimenta o processo *MAG - Módulo de Apoio à Gerência* que contém a base de conhecimento e realiza a agregação do conhecimento, objeto principal deste artigo.

2.3 O Processo de Agregação de Conhecimento

Esta tarefa é realizada pelo processo *MAG* que agrega conhecimento aos dados de tráfego até que se possa apontar os problemas da rede e suas correções

Para isso, este processo está dividido em quatro fases principais:

- Interpretação ou desencapsulamento das unidades de dado;

- Detecção de situações anormais encontradas nos dados de tráfego;

- Diagnose, partindo das situações anormais detectadas e reconhecendo ou identificando o problema causador;

- Correção, partindo do problema diagnosticado e aconselhando procedimentos de solução ao Gerente.

Cada uma destas fases será abordada a seguir, a fase de diagnose será apresentada em maior detalhe.

A figura 2 na página a seguir ilustra o processo de agregação do conhecimento.

Como o processo *MAG* é um bloco sequencial, a especificação de suas estruturas e processos foi feita em VDM (Vienna Development Method) [1,9]. VDM proporciona poderosas operações sobre conjuntos facilitando o tratamento da base de conhecimento, além disso, os procedimentos ficam em uma forma algorítmica, utilizando estruturas próximas de linguagens comuns de programação.

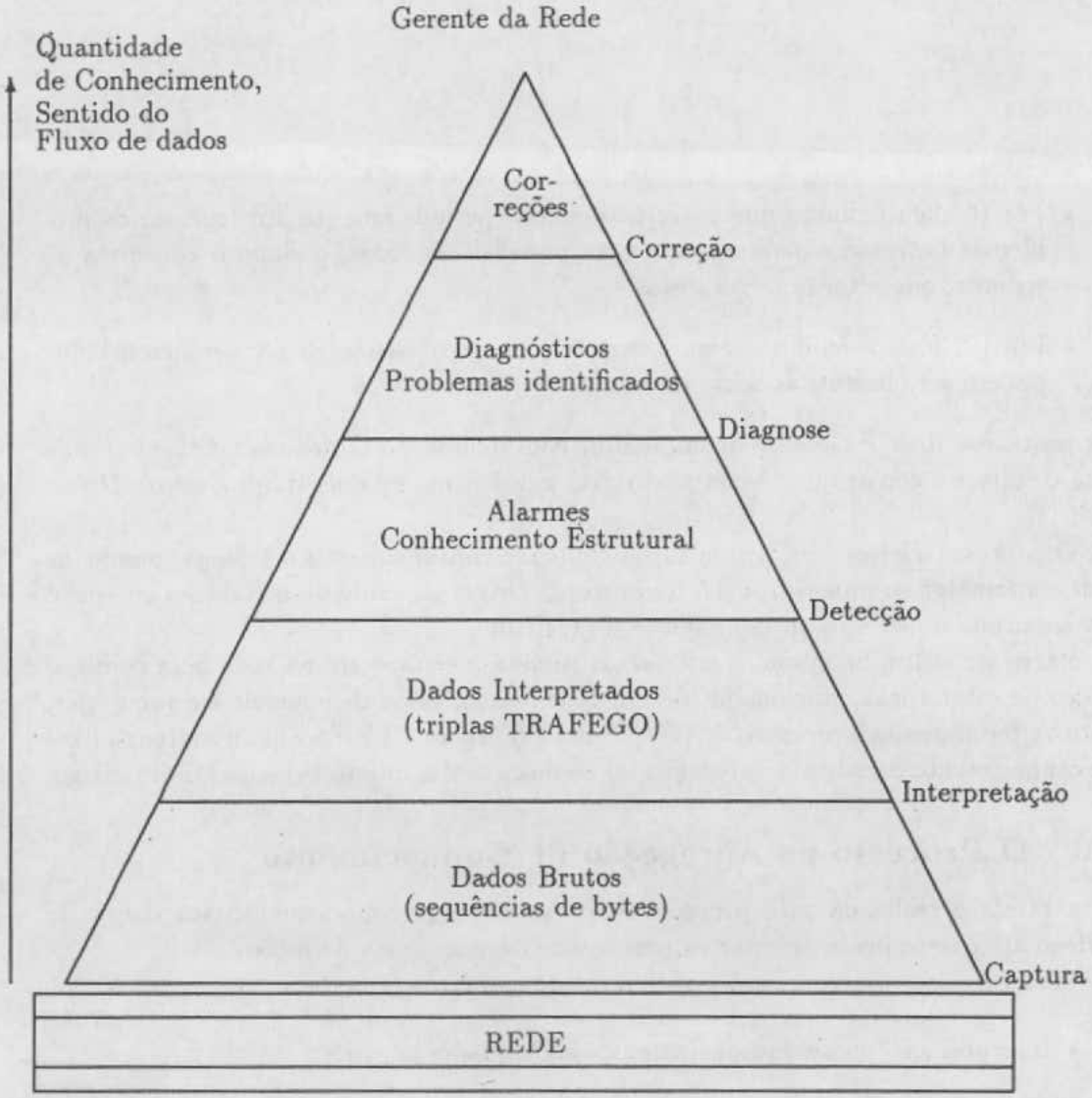

Figura 2: Agregação de Conhecimento

2.3.1 Fase de Interpretação

A Fase de Interpretação age sobre os *frames* capturados separando campos e valores para cada unidade de dado de cada nível de protocolo utilizado na montagem dos *frames*. Os pacotes de estatísticas também devem ser separados em sequencias de campos e valores.

Como os *frames* que trafegam na rede podem conter diferentes protocolos, a interpretação das unidades de dados acontece como o reconhecimento de gramáticas. Existe, então, um conjunto de procedimentos capaz de reconhecer as unidades de dados de cada protocolo de cada nível empregado, gerando uma representação interna que alimenta o Submódulo de Detecção. Os pacotes de estatísticas são também interpretados, gerando

uma representação interna.

Na implementação feita, o reconhecimento das unidades de dados é facilitado devido ao uso de PROLOG com seu mecanismo de *backtracking* embutido.

A principal função da fase de Interpretação:

(12) *Interpreta* : $GRAMATICAS \times BUFFERS \rightarrow TRAFEGO\text{-set}$

O reconhecimento ou interpretação dos dados contidos em *BUFFERS* através das *GRAMATICAS* gera um conjunto de triplas *TRAFEGO*. Cada tripla denota uma unidade de dado capturada, sendo formada pelo nível de protocolo, o instante de captura, e o conjunto de campos e valores do protocolo utilizado.

Abaixo, exemplos da representação interna em PROLOG. Os dados antes da interpretação são sequências de *bytes*, após a interpretação as triplas geradas para os níveis 3 (IP) e 4 (TCP) a partir de um *frame* podem ser:

```
trafego(69723.68131868, n_3_1,
        [(versio , [4]),(hdleng , [5]),(tpserv , [0]),
         (totlen , [0,76]),(identi , [216,191]),(dntfrg , 0),
         (morfrg , 0),(fragof , [0,0]),(tmlive , [60]),
         (protoc, [6]),(hdchck , [133,121]),
         (srcadd , [143,54,1,2]),(dstadd , [143,54,1,5]),
         (option , [])]).

trafego(69723.68131868, n_4_1,
        [(srcprt , [0,21]),(dstprt , [47,144]),
         (seqnum , [39,23,0,105]), (pigack , [6,121,0,26]),
         (hdleng , [5]),(urgent , [0]),(acknow , [1]),
         (eofmsg , [1]),(reset , [0]),(syn , [0]),(fin , [0]),
         (window , [15,231]),(chksum , [153,31]),
         (urgptr , [0,0]),(option , [])]).
```

Estas triplas alimentam a Fase de Detecção de problemas que parte das unidades de dados desmembradas em seus campos. Desta forma fica agregado o conhecimento estrutural aos dados brutos, facilitando a fase seguinte.

2.3.2 Fase de Detecção

A Fase de Detecção recebe as triplas acima mencionadas e submete-as ao teste de uma série de predicados sobre seu conteúdo. A aplicação destes predicados gera eventos que são contabilizados em estruturas separadas. Durante esta contabilização são verificados limites de ocorrências para estes eventos em função do tempo, gerando ou não alarmes. Os alarmes são mantidos em históricos denotando o comportamento da rede,

e colocados em listas de alarmes (separados por nível) juntamente com o instante em que foi disparado, de forma a alimentar a Fase de Diagnose.

Estruturas de mais alto nível da Fase de Detecção:

$$
\begin{array}{lrcl}
(13) & DETECTA & = & AREA\_GER \xrightarrow{m} DET\_AREA \\
(14) & DET\_AREA & = & NIVEL \xrightarrow{m} DET\_N \\
(15) & DET\_N & :: & PRED\_OC\_TRAF\text{-set} \times \\
 & & & \times PRED\_AUS\_TRAF\text{-set} \times \\
 & & & \times CONT \times HIST \times ESTR\_ALARM\text{-set}
\end{array}
$$

As construções denotam:

- Por (13) sabe-se que a detecção de problemas se dará de forma separada por cada área de gerenciamento OSI.
- Em (14) detalha-se que para cada área de gerenciamento, a deteção de problemas será separada conforme os níveis de protocolo em utilização.
- Em (15) detalham-se as estruturas necessárias para a detecção de problemas a cada nível: os predicados *PROD_OC_TRAF* e *PRED_AUS_TRAF* que as unidades de dados devem satisfazer, as estruturas de *CONT*abilização, os *HIST*óricos, e os alarmes gerados naquele nível.

A principal função desta fase:

$$
\begin{array}{lrcl}
(16) & Deteccao & : & AREA\_GER\text{-set} \times NIVEL\text{-set} \times \\
 & & & \times TRAFEGO \rightarrow DETECTA \rightarrow DETECTA
\end{array}
$$

São fornecidas as áreas de gerenciamento (*AREA_GER*) e os níveis de protocolos (*NIVEL*) sobre os quais se quer contabilizar, a estrutura (*TRAFEGO*) cujo conteúdo deverá ser testado sobre os predicados, e as estruturas de detecção (13) tal como definidas. Esta função desencadeia o processo de detecção modificando as diversas estruturas especificadas. Para a fase de diagnose são importantes as listas de alarmes detectados por nível, *ESTR_ALARM* em (15).

Abaixo, exemplos destas estruturas para a teoria *falhas*:

```
pred_oc_traf( falhas,n_3_2,Cvs,
              ocorr(falhas,n_3_2,msg_arp) ).
contab( ocorr(falhas,n_3_2,msg_arp),
                []),
                200, 100, 80, 150,
```

```
                (falhas,n_3_1,arps_excessivos,5)    ).
hist( (falhas,n_3_1,arps_excessivos,TTL),
      [], 20    ).

pred_oc_traf( falhas,n_3_3,Cvs,
              ocorr(falhas,n_3_3,sem_recursos(Da) )) :-
              esta_em((tipo,4),Cvs),
              esta_em((dstadd_msg,Da),Cvs).
default( contab( ocorr(falhas,n_3_3,sem_recursos(Da)),
                 [],
                 10, 100, 60, 5,
                 (falhas,n_3_1,sem_recursos(Da),10)
      )          ).
default( hist( (falhas,n_3_1,sem_recursos(Da),TTL),
               [], 6
      )     ).
```

Desta forma, o conhecimento codificado nos predicados, históricos e alarmes é agregado aos dados de entrada (triplas) da fase de Detecção, gerando os alarmes que alimentam a Fase de Diagnose.

2.3.3 Fase de Diagnose

Os alarmes capatados durante a Detecção são submetidos a este submódulo com o objetivo de diagnosticar possíveis problemas na instalação em análise. Após a diagnose, o resultado é passado à Fase de Correção que fornece uma recomendação para solucionar o problema em questão.

A Fase de Diagnose usa, para isso, o conhecimento de um especialista em forma de regras, alarmes, bem como outros subsídios.

Estruturas da Fase de Diagnose:

(17)	*BANCO_DE_CONHEC*	::	*CJ_TEORIAS* × *CONHEC_ESTRUT*
(18)	*CONHEC_ESTRUT*	=	*FATO_SOBRE_REDE*-set
(19)	*FATO_SOBRE_REDE*	::	*OBJETO*×
			×*ATRIBUTO* × *VALOR*
(20)	*CJ_TEORIAS*	=	*AREA_GER* $\xrightarrow{m}$ *TEORIA*
(21)	*TEORIA*	::	*NIVEL* $\xrightarrow{m}$ *SUBTEORIA*×
			×*INTERRELACOES*
(22)	*SUBTEORIA*	=	*REGRA*-set

(23)	*INTERRELACOES*	=	*REGRA*-set
(24)	*REGRA*	::	*NRO_REGRA* × *CONDICAO*×
			×*ASSERCAO* × *FATOR_DE_CERTEZA*
(25)	*CONDICAO*	=	*COND_COMPOSTA* |
			COND_ELEMENTAR
(26)	*COND_COMPOSTA*	::	*CONDICAO* × *OPBOOL* × *CONDICAO*
(27)	*COND_ELEMENTAR*	=	*ALARME* |
			FATO_SOBRE_REDE | *NO*(*CONDICAO*)
(28)	*ASSERCAO*	=	*FATO_SOBRE_REDE* | *DIAGNOSE*
(29)	*DIAGNOSE*	=	*PROBL* × *TRACE*×
			×*FATOR_DE_CERTEZA*

- Em (17) define-se o *Banco de Conhecimento* dividido em *conhecimento estrutural e em teorias.* O *Conhecimento Estrutural* corresponde à estrutura lógica e física da rede. As *Teorias* contém conhecimento especialista para tratamento de problemas em diferentes áreas.

- O *Conhecimento Estrutural* (18) é um conjunto de fatos verdadeiros sobre a rede, seguindo a estrutura OAV. Através desta estruturação pretende-se criar uma forma bastante permissível de representação de situações lógicas e físicas sobre a instalação (criando uma rede semântica) [5]. Utiliza-se o mesmo *banco de conhecimento estrutural* para a análise em qualquer *teoria.*

- Um *Fato Sobre a Rede* (19) é uma tripla do tipo OAV (Objeto/Atributo/Valor) através da qual pode-se fazer qualquer afirmação sobre a rede. Exemplos:

```
oav(rede,possui,segmento1).
oav(segmento1,tipo,dez_base_dois).
oav(estacao([0,0,180,16,12,107]),equipamento,microvax).
oav(estacao([0,0,180,16,12,107]),conectada_a,segmento1).
oav(estacao([0,0,180,16,12,107]),localizacao,cpd_suporte).
oav(estacao([0,0,180,16,12,107]),funcao_na_rede,gateway_externo).
oav(estacao([170,0,4,0,195,185]),equipamento,pc1).
oav(estacao([170,0,4,0,195,185]),conectada_a,segmento1).
```

  Assim como pode-se representar a estrutura fisica e as funcoes dos equipamentos, pode-se também, no decorrer da análise, criar triplas deste tipo relacionando endereços de rede com enderecos IP, enderecos IP com "ports" do TCP, entidades associadas em uma conexão, etc.

- O *Conjunto de Teorias* (20) divide o conhecimento especialista em várias teorias conforme as áreas de gerenciamento OSI. Assim cada teoria é alimentada por alarmes distintos, gerando diagnósticos distintos.

- Cada *Teoria* (21) para cada área de gerenciamento OSI é, por sua vez, dividida em subteorias seguindo os níveis de protocolos em utilização, e uma subteoria à parte para realizar o interrelacionamento das diagnoses das subteorias dos diversos níveis. Esta estruturação faz com que o conhecimento de cada área de gerenciameto fique dividido para cada nível de protocolo, auxiliando no domínio da complexidade

- Cada *Subteoria* (22) para cada nível é composta por um conjunto de regras, resultado da codificação do conhecimento de um especialista por engenharia de conhecimento. Estas regras associam alarmes e conhecimento estrutural, podendo se valer de históricos também. Exemplos:

```
teoria falhas
subteoria n_3_1
regra  29 : if alarme(sem_recursos(A))      and
               oav(E,endereco_ip,A)         and
               n(oav(E,une,_))
            then goal(
                 diag_parcial(estacao_sem_recursos(A))
                    )
            with 0.9.

teoria falhas
subteoria n_4_1
regra  28 : if nao_aceita_conexao(Sa,Sp,Da,Dp) and
               oav(E,endereco_ip,Sa)           and
               oav(E,conectada_a,Seg)          and
               conta_nro_conexoes(Sa,N)        and
               n( oav(E,n_max_conexoes,N_max) and
               exec(N_max > N)  )
            then goal(diag_parcial(
                      estacao_nao_aceita_conexao(Sa,Sp,Da,Dp)
                    )           )
            with  0.8.
```

- As *subteorias de Interrelacionamento* (23) são também conjuntos de regras. As regras de interrelacionamento associam *diagnoses* das subteorias dos níveis (ou diagnoses parciais) validando, combinando ou cancelando diagnoses. O *banco de conhecimento estrutural* também é importante neste processo. Exemplo:

```
teoria falhas
subteoria interrelacionamento
regra 38 : if diag_parcial(
             estacao_nao_aceita_conexao(Sa,Sp,Da,Dp)
                        )                          and
```

```
    diag_parcial(estacao_sem_recursos(Sa)) and
    oav(E,endereco_ip,Sa)  and
    oav(E,funcao_na_rede,servidor(X))
then
goal(diag_final(
     nao_aceita_por_gargalo_servidor(Sa,Da)
   )          )
with 0.8.
```

- Uma *Regra* (24) é composta por um número que a identifica univocamente dentro de sua *Subteoria*, uma condição que deve ser satisfeita para que a regra seja aplicada, a ação resultante da aplicação da regra, e um fator de certeza associado à regra que auxilia na computação do fator de certeza global para as diagnoses.
- Uma *Asserção* (28) significa a afirmação de algo sobre a rede, no caso pode-se chegar a um diagnóstico, ou pode-se afirmar sobre a rede (utilizando 19) criando subsídios para posterior análise.
- Uma *Diagnose* (29) é composta por um problema ou situação causadora, a árvore de dedução ou *trace* deste problema utilizando-se o corpo de regras da subteoria devida, e o *fator de certeza* para a dedução do problema.

A principal função desta fase:

(30) *Diagnostico* $I \times BANCO\_DE\_CONHEC \times$
$\times AREA\_GER\text{-set} \times NIVEL\text{-set} \rightarrow$
$\rightarrow DIAGNOSE\text{-set}$

Submete-se à esta função o instante até então analisado pelo sistema, o banco de conhecimento conforme definido (17), e as áreas de gerenciamento e níveis de protocolo sobre as quais se quer realizar inferência, gerando diagnósticos.

De posse dos diagnósticos finais, parte-se para a Fase de Correção.

2.3.4 Fase de Correção

A Fase de Correção recebe diagnósticos de problemas da Fase de Diagnose e retorna procedimentos de correção para estes problemas. Estas correções são, então, adicionadas a um *log* de problemas e correções da instalação que ficará a disposição do Gerente da Rede.

Esta Fase não é complexa pois o Sistema é passivo, isto é, não gera mensagens de gerenciamento e sim "aconselha" ou "adverte" o gerente da rede no caso de situações anormais, conforme as áreas de gerenciamento definidas no contexto OSI de gerenciamento de redes.

Estruturas utilizadas nesta fase.

(31) $CORRECOES = PROBL \xrightarrow{m} CORRECAO\text{-set}$
(32) $CORRECAO :: PROCEDIMENTO \times FATOR\_DE\_CERTEZA$

As estruturas acima significam:

- Por (31) fica especificado que para cada *problema* encontrado durante a diagnose podem existir várias formas de correção;
- Cada *Correção* (32) é composta por um procedimento que a implementa e por um *Fator de Certeza* para o procedimento;

As regras anteriormente citadas fazem parte do diagnóstico para o problema de um servidor que não aceita mais conexões por estar congestionado. O procedimento de correção apontado pode ser:

```
correcao(nao_aceita_por_gargalo_servidor(Sa,Da),
        ['
         O servidor ',Sa,' nao esta aceitando
         conexoes de transporte por  falta de recursos
         para o tratamento  de novas conexoes.  Nao ha',','
         equipamento alternativo  no mesmo segmento de
         rede para prestar o mesmo tipo de servico.   ',','
         Verificar os parametros  de instalacao do ro-
         teador.
         Localizacao do servidor: ',L,'
         Equipamento: ',Eq,'
        '],
        0.85) :- oav(estacao(Sa),localizacao,L),
                 oav(estacao(Sa),equipamento,Eq).
```

Note que a fase de correção também pode acessar o conhecimento estrutural. Os Fatores de Certeza, atualmente, não possuem forma alguma de correção dinâmica. Futuramente, pode-se implantar um esquema de ajuste dinâmico destes fatores através de um mecanismo de realimentação ("Feedback Processing" em [13] por exemplo), possivelmente interno à Fase de Correção. Com isto o sistema pode se adapatar a algumas características particulares da instalação sendo gerenciada.

Função principal para a Correção:

(33) $Corrige : CORRECOES \times DIAGNOSE\text{-set} \rightarrow$
$\rightarrow ELEM\_LOG\text{-set} \rightarrow ELEM\_LOG\text{-set}$

Nesta fase, parte-se do pressuposto que as várias formas de correção estão já armazenadas e é necessário apenas um trabalho de recuperação destas informações. Submete-se um conjunto de Diagnoses e retorna-se os diversos conjuntos de Correções para estas Diagnoses. A próxima etapa é repassar estas informações de forma adequada ao usuário (Gerente da Rede).

3 Considerações Finais

Foram apresentadas as etapas principais no tratamento da informação de *Um Sistema de Apoio à Gerência de Redes Locais.*

Pela figura 2 pode-se notar que a quantidade de conhecimento por unidade de dado armazenada aumenta no sentido do fluxo dos dados no sistema, passando pelas diversas fases detalhadas. Este é o modelo natural para sistemas que visam proporcionar um serviço de valor adicionado capaz de realizar tarefas relativamente complexas (Decision Support Systems) [6].

No sistema em questão, o trabalho de modelar o conhecimento do especialista abrange principalmente as etagas de Diagnose e Correção. Durante a fase de Diagnose podem ser criados Eventos e Alarmes (próprios da fase de Detecção) através de primitivas (análogas às primitivas *Get* e *Set* de manipulação de uma MIB). Com isso, concentra-se mais conhecimento na fase de Diagnose.

O sistema foi implementado em PROLOG como forma de validar a proposta. Na experiência de implementação e testes realizados, notou-se a necessidade de uma metodologia mais poderosa na fase de Detecção de problemas para que se possa representar melhor os recursos sendo acompanhados. A utilização de orientação a objetos pode ser uma boa alternativa para sanar esta deficiência.

Enfim, pode-se dizer que a arquitetura apresentada é uma boa opção para iniciar a gerência de uma instalação pois as modificações e os custos acarretados de sua utilização não são um impecilho. Com a evolução dos modelos de Gerência de Redes este sistema pode se integrar no Domínio Gerenciado como um agente procurador. Para isso deve-se adicionar ao sistema um processo agente através do qual seriam reportadas ao gerente central as situações que o conhecimento local não é capaz de tratar. As situações que podem ser gerenciadas com o conhecimento local continuam a ser tratadas da mesma forma.

Referências

[1] D. BJØRNER and C. B. JONES, editors. *The Vienna Development Method: the meta-language.* Volume 61 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlim, 1978.

[2] Ivan BRATKO. *Prolog Programming for Artificial Intelligence.* Addison-Wesley, Great-Britain, 1986.

[3] J.P. CLAUDÉ, M.P. CLAUDÉ, M.P. GERVAIS, and M. PENNA. Management of

open networks in heterogeneous context. In *Proc. of the IFIP TC6 WG6.4a International Simposium on Local Communications Systems Management*, pages 81–100, University of Kent at Canterbury, U.K., Sept. 1990.

[4] F. L. DOTTI and L. M. R. TAROUCO. Um sistema de apoio à gerência de redes locais. In *9 Simpósio Brasileiro de Redes de Computadores*, pages 400–416, Florianópolis, SC, Mai. 1991.

[5] R. GOODMAN, J. MILLER, and P. SMYTH. A platform for heterogeneous interconnection network management. In *Proc. of the IFIP TC6 WG6.4a International Simposium on Local Communications Systems Management*, pages 13–26, Boston, MA, U.S.A., May. 1989.

[6] I GRAHAM and P. JONES. *Expert Systems: Knowledge, Uncertainly and Decision.* Chapman and Hall, Great Bretain, 1988.

[7] C. A. R. HOARE. *Communicating Sequential Processes.* Prentice-Hall International, London, 1985.

[8] ISO. *ISO DP 7498-4 : Information Processing Systems - OSI Reference Model Part 4: Management Framework.* OSI, out. 1986.

[9] C. B. JONES. *Systematic Software Development Using VDM.* Prentice-Hall International, London, 1986.

[10] L. KERSCHBERG, R. BAUM, A. WAISANEN, I. HUANG, and J. YOON. A survey of automated fault management systems for telecommunications networks. In *Departament of Information Systems and Systems Engineering - Geroge Mason University*, pages 1–14, Virginia, U.S.A., 1989.

[11] Yoshikazu KOBAYASHI. Standardization issues in integrated network management. In *Proc. of the IFIP TC6 WG6.4a International Simposium on Local Communications Systems Management*, pages 79–92, Boston, MA, U.S.A., May. 1989.

[12] J. LIEBOWITZ. *Expert Systems Applications to Telecommunications.* John Wiley & Sons, U.S.A., 1988.

[13] T. E. MARQUES. A symptom-driven expert system for isolating and correcting network faults. *IEEE Communications Magazine*, 26(3), Mar. 1988.

[14] L. M. R. TAROUCO and F. L. DOTTI. Mefisto- mechanism efficient to foster the implementation of software totally osi. In *Proc. of the IFIP TC6 WG6.4a International Simposium on Local Communications Systems Management*, pages 221–228, University of Kent at Canterbury, U.K., Set. 1990.

[15] P. WINSTON. *Artificial Intelligence.* Addison-Wesley, E.U.A., 1984. 2a edição.