SUSAN MÖLLER FERREIRA

CONTRIBUIÇÃO À PADRONIZAÇÃO DA COMUNICAÇÃO EM CONTROLE DE TRÁFEGO VEICULAR URBANO

FLORIANÓPOLIS
2007

UNIVERSIDADE FEDERAL DE SANTA CATARINA

PROGRAMA DE PÓS-GRADUAÇÃO
EM ENGENHARIA ELÉTRICA

CONTRIBUIÇÃO À PADRONIZAÇÃO DA COMUNICAÇÃO EM CONTROLE DE TRÁFEGO VEICULAR URBANO

Dissertação submetida à
Universidade Federal de Santa Catarina
como parte dos requisitos para a
obtenção do grau de Mestre em Engenharia Elétrica.

SUSAN MÖLLER FERREIRA

Florianópolis, Maio de 2007.

CONTRIBUIÇÃO À PADRONIZAÇÃO DA COMUNICAÇÃO EM CONTROLE DE TRÁFEGO VEICULAR URBANO

SUSAN MÖLLER FERREIRA

'Esta Dissertação foi julgada adequada para a obtenção do título de Mestre em Engenharia Elétrica, Área de Concentração em *Controle, Automação e Sistemas*, e aprovada em sua forma final pelo Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Santa Catarina.'

Werner Kraus Junior, Ph.D.
Orientador

Kátia Campos de Almeida, Ph.D.
Coordenador do Programa de Pós-Graduação em Engenharia Elétrica

Banca Examinadora:

Werner Kraus Junior, Ph.D.
Presidente

Carlos Barros Montez, Dr.
Co-orientador

Jean Marie Farines, Dr.

Marcelo Ricardo Stemmer, Dr.

Frank Augusto Siqueira, Dr.

Aos meus pais... ...

AGRADECIMENTOS

Agradeço a todos que de alguma forma colaboraram com este trabalho, principalmente aos professores Werner Kraus Junior e Carlos Barros Montez pelas conversas, incentivo, paciência e orientação.

A minha família por todo suporte, carinho e incentivo durante todo caminho até aqui. Especialmente ao meu pai pelo exemplo e sempre acompanhamento, a minha mãe por ensinar a sempre seguir em frente sorrindo e ao meu irmão Celso por mostrar que na vida é necessário ter tempo para se dedicar ao trabalho e ainda aproveitar o por do sol na lagoa e um bom dia de praia.

Aos professores e funcionários dos Departamentos de Automação e Ciência da Computação pelos ensinamentos e contribuição indispensável para meu amadurecimento intelectual. Especialmente ao professor Melgarejo, ao Ricardo e equipe do Edugraf pela ajuda com o XML, e a equipe do Sincmobil pelo auxilio indispensável com o controlador semafórico.

Agradeço também aos amigos do DAS pelo incentivo, auxilio nas dúvidas, pelas horas de estudo e momentos de descontração. Aos amigos do NPD pelo apoio, churrasquinhos e cafezinhos desde a graduação.

E finalmente ao meu namorado e aos amigos que seja em São José dos Campos, Florianópolis ou ainda os mais distantes, agradeço pela força, apoio, energia, paciência, conversas, filosofias, passeios e festas.

Resumo da Dissertação apresentada à UFSC como parte dos requisitos necessários para obtenção do grau de Mestre em Engenharia Elétrica.

CONTRIBUIÇÃO À PADRONIZAÇÃO DA COMUNICAÇÃO EM CONTROLE DE TRÁFEGO VEICULAR URBANO

SUSAN MOLLER FERREIRA

Maio/2007

Orientador: Werner Kraus Junior, Ph.D.
Co-orientador: Carlos Barros Montez, Dr.
Área de Concentração: Controle, Automação e Informática Industrial
Palavras-chave: ITS, NTCIP, Protocolos de Comunicação, Gerência Remota de Dispositivos, Padronização, XML
Número de Páginas: xix + 97

Este trabalho visa contribuir para a definição de um padrão nacional para comunicação de controladores semafóricos. Faz-se um estudo do padrão NTCIP (*National Transportation Communications for ITS Protocol*), com foco nos protocolos de aplicação para comunicação entre centrais e equipamentos de campo. É avaliada a compatibilidade do NTCIP com a prática de engenharia de tráfego brasileira, resultando na implementação e teste de um conjunto de objetos gerenciáveis específicos para controladores semafóricos nacionais.

Visando avaliar alternativas tecnológicas para a metodologia, estudou-se o XML e algumas ferramentas associadas. A partir do estudo, formula-se uma proposta de solução que usa XML e XSLT para a definição dos elementos do protocolo. É feita a implementação da comunicação com um controlador semafórico dotado de protocolo proprietário para ilustrar a capacidade da proposta em estender a funcionalidade de centrais de controle para comunicação com dispositivos legados.

Abstract of Dissertation presented to UFSC as a partial fulfillment of the requirements for the degree of Master in Electrical Engineering.

CONTRIBUTION TO THE STANDARDIZATION OF THE COMMUNICATION IN URBAN TRAFFIC CONTROL

SUSAN MOLLER FERREIRA

Maio/2007

Orientador: Werner Kraus Junior, Ph.D.
Co-orientador: Carlos Barros Montez, Dr.
Area of Concentration: Control, Automation and Industrial Informatics
Key words: ITS, NTCIP, Communication Protocols, Remote Management of Field Equipment, Standardization, XML
Number of Pages: xix + 97

This research aims at contributing for the definition of a Brazilian standard for the communication of semaphore controllers. The NTCIP (National Transportation Communications for ITS Protocol) set of standard protocols is examined focusing on the application protocols for the communication between central control stations and field devices. The compabillity of NTCIP with the Brazilian traffic engineering practice was evaluated, leading to the implementation and testing of a group of management objects specifically designed for national semaphore controllers.

Aiming to evaluate technological alternatives for the methodology, were studied XML and associate tools. From this study, a solution proposal was formulated that uses XML and XSLT for the definition of the protocols elements. The solution was implemented for the communication with a semaphore controller with a proprietary protocol, thus illustrating the capacity of the proposed method in extending the functionality of control centers for communication with legacy devices.

Sumário

Lista de Figuras

Lista de Tabelas

LISTA DE ABREVIATURAS

ASN.1 - *Abstract Syntax Notation - One*

BER - *Basic Encoding Rules*

CET-SP - *Companhia de Engenharia de Tráfego de São Paulo*

CORBA - *Common Object Request Broker Architecture*

DATEX - *Data Exchange*

DOD - *Department of Defense*

DRIVE - *Dedicated Road Infraestructure for Vehicle Safety in Europe*

DTD - *Documento Type Definition*

FHWA - *Federal Highway Administration*

FTP - *File Transportation Communication*

HTML - *Hypertext Markup Language*

IAB - *Internet Architecture Board*

IP - *Internet Protocol*

ISO - *International Organization for Standardization*

ITS - *Intelligent Transportation Systems*

MIB - *Management Information Base*

NEMA - *National Electrical Manufactures Association*

NIA - *National ITS Architecture*

NTCIP - *National Transportation Communication for ITS Protocol*

OER - *Octet Encoding Rules*

OID - *Object Identifier*

OSI - *Open System Interconnection*

PDU - *Protocol Data Unit*

PMPP - *Point-to-MultiPoint Protocol*

PPP - *Point-to-Point Protocol*

RFC - *Request for Comments*

SFMP - *Simple Fixed Message Protocol*

SGML - *Standard Generalized Markup Language*

SMI - *Structure of Management Information*

SNMP - *Simple Network Management Protocol*

STMP - *Simple Transportation Management Protocol*

TCP - *Transmission Control Protocol*

TLV - *Tag-Length-Value*

TMML - *Traffic Model Markup Language*

T2 - *Transportation Transport Profile*

UDP - *User Datagram Protocol*

URI - *Uniform Resource Identifier*

URL - *Standard Uniform Resource Locator*

XML - *Extensible Markup Language*

XSLT - *Extensible Stylesheet Language Transformations*

W3C - *World Wide Web Consortium*

Capítulo 1

Introdução

Atualmente, no Brasil, não existe um padrão de comunicação para dispositivos de controle de tráfego veicular [18]. Cada fabricante utiliza uma forma de comunicação para a troca de informações entre sua central de controle e os equipamentos de campo. Assim, por exemplo, os controladores semafóricos de um fabricante "A"só podem se comunicar com controladores e com uma central de controle deste mesmo fabricante.

A falta de um padrão de comunicação para controle de tráfego acarreta dois problemas:

- incapacidade de dispositivos de diferentes tipos trabalharem juntos, formando um sistema único. Por exemplo, um controlador semafórico não pode trocar dados com um sistema de monitoração de excesso de velocidade.
- dispositivos do mesmo tipo (por exemplo controladores semafóricos em rede), em um sistema, não podem ser substituídos por equipamentos de diferentes fabricantes garantindo que seja mantida a interação entre eles.

Em conseqüência disso, se em uma cidade o sistema de transporte for construído com controladores e uma central de controle de trafego de um fabricante "A", para garantir a interação entre dispositivos, o órgão gestor deve sempre comprar novos controladores deste mesmo fabricante. Ou seja, dessa forma, fica impossibilitada a compra de equipamentos mais baratos ou mais eficientes produzidos por outros fabricantes.

Para solucionar este problema, este trabalho tem o intuito de contribuir para definição de um padrão nacional para comunicação de equipamentos de controle. Existem iniciativas de padronização em outros países [1, 37, 40] *apud* [2]; tais propostas são normalmente

agrupadas no campo de estudo denominado de ITS (*Intelligent transportation systems*). ITS engloba tecnologias, que integradas à infra-estrutura de transportes e veículos, visam aprimorar a segurança, mobilidade e produtividade através do uso de tecnologias avançadas de comunicação [38].

1.1 Objetivos do Trabalho

1.1.1 Objetivo geral

Contribuição para a definição de um padrão nacional para comunicação de controladores semafóricos, garantindo a interoperabilidade e intercambiabilidade dos equipamentos.

1.1.2 Objetivos Específicos

Para atingir o objetivo acima, definem-se os seguintes objetivos específicos:

- pesquisa das iniciativas de padronização de comunicação de controladores semafóricos no mundo;
- compreensão do funcionamento do padrão de comunicação norte-americano NTCIP (*National Transportation Communications for ITS Protocol*);
- estudo dos protocolos de aplicação sugeridos pelo NTCIP (SNMP, STMP e SFMP);
- pesquisa envolvendo a prática de Engenharia de Tráfego Nacional;
- avaliação da compatibilidade entre o padrão NTCIP e a prática de Engenharia de Tráfego nacional e proposição de possível adaptações;
- proposta de outras tecnologias que possam ser utilizadas na solução.

1.2 Organização do Texto

No capítulo 2 é feito um estudo geral do padrão NTCIP, já no capítulo seguinte são estudados os protocolos de aplicação, propostos pelo NTCIP, para comunicação entre central e equipamentos de campo. No capítulo 4 é feito um levantamento sobre a prática de engenharia

de tráfego no Brasil, avaliando sua compatibilidade ou não com o padrão NTCIP. Então no capítulo 5 é apresentado o XML(*Extensible Markup Language*), visando utilizá-lo na proposta e implementação detalhadas no capítulo seguinte. Finalizando, no capítulo 7 são apresentadas conclusões e sugestões para trabalhos futuros.

Capítulo 2

Padronização de Comunicação em Sistemas de Transportes

Os sistemas de transportes atuais são compostos por dezenas de subsistemas, envolvendo uma grande quantidade de componentes e dispositivos interconectados por redes de comunicação. A gerência remota de dispositivos, nesses subsistemas, facilita muito a operação do tráfego em uma área metropolitana. Porém, para a realização dessa gerência, é desejável que equipamentos de diferentes fabricantes possam comunicar-se entre si. Tal integração operacional entre componentes pressupõe que a propriedade de interoperabilidade seja atendida, sendo tambem desejada a intercambiabilidade entre equipamentos.

O termo interoperabilidade refere-se à capacidade de dispositivos de diferentes tipos trabalharem juntos, como um sistema único. Já a intercambiabilidade é a capacidade de substituição de dispositivos do mesmo tipo em um sistema, mantendo a interação entre os dispositivos, mesmo no caso de equipamentos de diferentes fabricantes [13].

Para garantir estas propriedades, faz-se necessária a adoção de protocolos padronizados para o processo de troca de informações entre dispositivos. No âmbito de sistemas de transportes, existem algumas iniciativas de padronização no mundo, mas na maioria dos casos este desenvolvimento ainda está se iniciando e existe muito pouca documentação sobre o assunto. Os esforços de padronização podem ser divididos em três grande grupos: Japonês, Europeu e Norte-americano.

Devido à sua grande densidade populacional e rede viária limitada, o Japão apresenta diversos problemas relacionados ao transporte, assim iniciou bastante cedo seu investimento

em ITS (*Intelligent Transportation Systems*) [40] *apud* [2]. Implementou alguns programas de pesquisa e leis de incentivo que visam à pesquisa e desenvolvimento de ITS, integração de projetos individuais, construção de uma arquitetura de sistemas, pesquisa e desenvolvimento, padronização e cooperação internacional [37] *apud* [2].

Já o programa ITS europeu envolve a interoperabilidade entre vários países da comunidade européia. O programa DRIVE (*Dedicated Road Infraestructure for Vehicle Safety in Europe*) visa desenvolver sistemas capazes de aumentar a segurança no trânsito, maximizar a eficiência do tráfego e reduzir a incidência de impactos ambientais [1] *apud* [2]. Há ainda, uma preocupação com questões relacionadas a usuários como a segurança de pedestres e usuários de transportes coletivos.

A iniciativa que se mostra mais avançada e com documentação mais completa é a proposta norte-americana denominada de *National Transportation Communications for ITS Protocol* (NTCIP)[26]. Este padrão proposto pela *Federal Highway Administration* (FHWA) está bastante avançado e vem sendo implantado em algumas cidades norte americanas. Devido à vasta documentação disponibilizada sobre o NTCIP e à praticidade da solução adotada, utilizaremos este padrão como base para o estudo e contribuição para a proposição de um padrão nacional para equipamentos de tráfego.

2.1 National ITS Architecture

Atualmente os progressos relacionados às pesquisas em ITS nos EUA apresentam uma relevância significativa. Existe uma arquitetura nacional definida (NIA - *National ITS Architecture*), a qual traz diretrizes para o desenvolvimento de ITS em instâncias regionais. Além disso, foram definidos padrões que permitem a interoperabilidade nacional entre os serviços ITS oferecidos [20]. A Figura 2.1 representa a arquitetura física definida pela NIA.

São definidos pela arquitetura física 21 subsistemas, os quais estão concentrados em 4 categorias: subsistemas centrais, subsistemas de viajantes, subsistemas da rede viária e subsistemas de veículos. Esses são componentes elementares da arquitetura física, e definem de maneira geral a estrutura de toda a arquitetura, podendo ser assim definidos:

Subsistemas Centrais: são responsáveis pela administração e gerência do sistema de transporte. Comunicam-se entre si para coordenar suas atividades e com subsistemas de Rede Viária e de Veículos para coletar e fornecer informações de controle. Essa comunicação

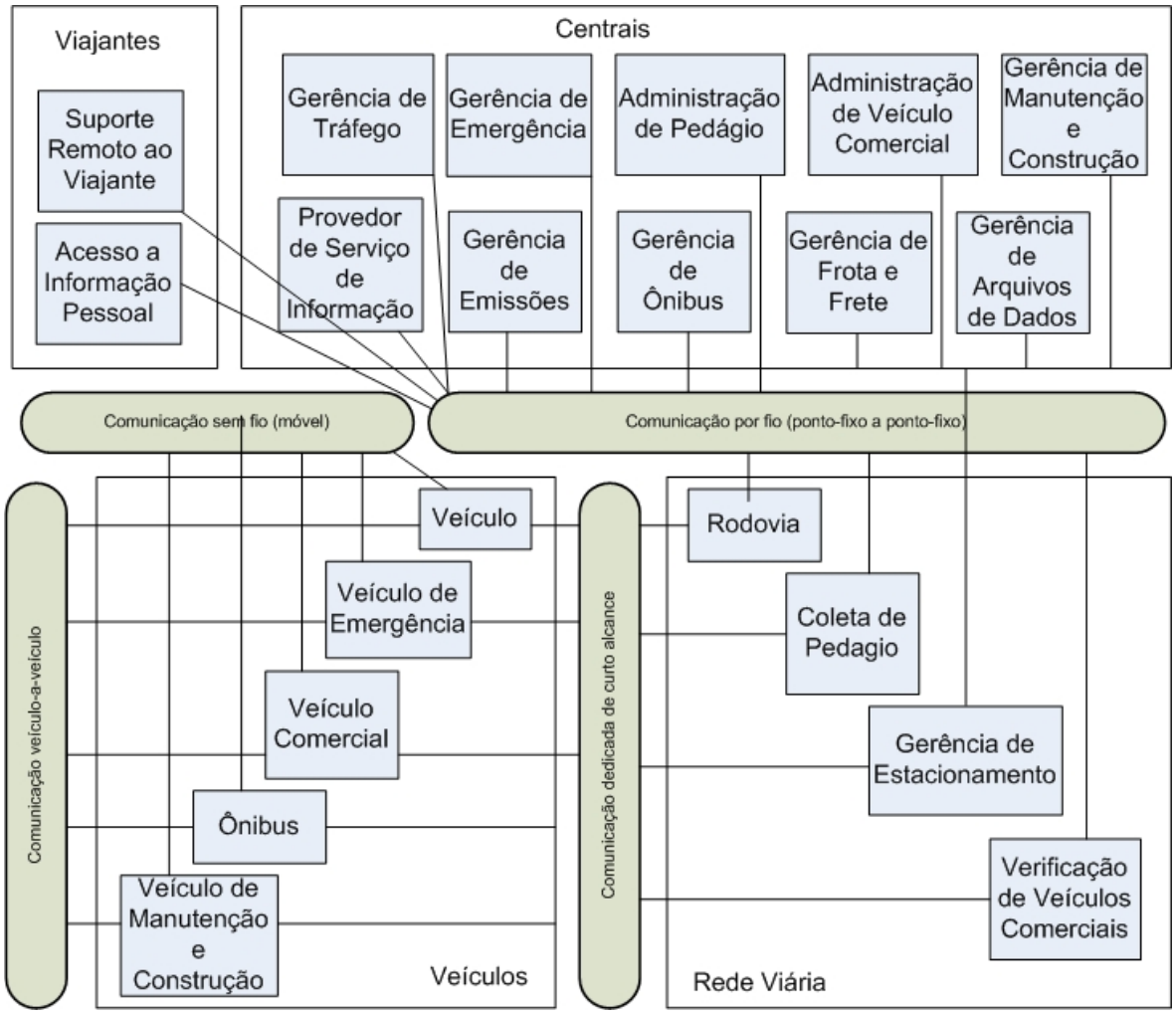

Figura 2.1: *National ITS Architecture [13].*

é representada na Figura 2.1 pelos bastões situados entre as categorias de subsistemas. Os subsistemas desta categoria são: Administração de veículos, Gerenciamento de frota e fretes, Administração de taxas, Gerenciamento de trânsito, Gerenciamento de emergências, Gerenciamento de emissões, Gerenciamento de dados, Gerenciamento de tráfego, Serviço de provimento de informações e Gerenciamento de Manutenção e Construção.

Subsistemas de Viajantes: esta categoria de subsistemas atende aos viajantes, que representam os usuários de ITS, em relação às suas necessidades de mobilidade. Esta categoria contém dois subsistemas: Suporte Remoto a Viajantes e Acesso Particular de Informação.

Subsistemas de Rede Viária: nesta classe de subsistemas são fornecidas interfaces diretas com a rede viária, veículos e viajantes, permite o acesso à observação das situações das rodovias, provisão de informações e à execução de planos de controle. Subsistemas: Rodovia, Coleta de taxas, Gerenciamento de estacionamentos e Conferência de veículos comerciais.

Subsistemas de Veículos: os subsistemas que fazem parte dessa categoria compartilham informações gerais de motoristas, navegação de veículos e funções avançadas de sistemas de segurança. Os subsistemas são: Veículos de Passeio, Veículos de Emergência, Veículos Comerciais, Veículos de Manutenção e Construção e Transporte Público.

2.2 NTCIP

O NTCIP (*National Transportation Communications for ITS Protocol*) é um conjunto de padrões norte-americano que define dados e padrões de comunicação a serem utilizados pelos dispositivos de tráfego. Fazem parte do conjunto NTCIP protocolos de uso geral como TCP, IP, Ethernet, PPP e também padrões desenvolvidos especificamente para ITS. O NTCIP define assim uma família de protocolos de comunicação e dicionários de dados de transporte que suportam a maioria dos sistemas computacionais e equipamentos de campo usados na gerência de transportes. O objetivo dos padrões é garantir a interoperabilidade entre sistemas de transporte e a possibilidade de intercâmbio de equipamentos de um sistema para outro [13].

As aplicações para NTCIP são divididas em duas categorias: C/C (Central/Central) e C/V (Central/Via). A comunicação Central/Via normalmente envolve dispositivos de campo

comunicando-se com um software de gerência em um computador central. Já as aplicações Central/Central geralmente envolvem a comunicação entre computadores, podendo estes estar na mesma sala, em centros da gerência operados por agências adjacentes ou através do país.

Baseado na arquitetura NIA, apresentada anteriormente, alguns exemplos de sistemas que podem se beneficiar com o NTCIP são:

Central/Via (C/V):

- Painéis de mensagens variáveis: indicam, através de texto, o estado atual das vias e sugestões de rotas;
- Semáforos;
- Controladores Semafóricos Mestre;
- Dispositivos de coleta de dados e monitoração, tais como contadores do tráfego, classificador do tráfego e estações de pesagem em movimento;
- Sensores e controladores embarcados em veículos;
- Sensores ambientais;
- Controladores de acesso a rodovias;
- Detectores veiculares;
- Circuito fechado de câmera de televisão (controle da câmera somente);
- Interruptores do vídeo;
- Controlador da iluminação de rodovias.

Central/Central (C/C):

- Gerência de tráfego (rodovia/rua, urbano/rural);
- Gerência de transporte público (ônibus/ trilhos/ outros);
- Gerência de incidentes;
- Gerência de emergências;
- Gerência de estacionamento;
- Informação ao viajante (todas as modalidades);
- Regulamentação de operações de veículos comerciais ;
- Qualquer combinação das anteriores.

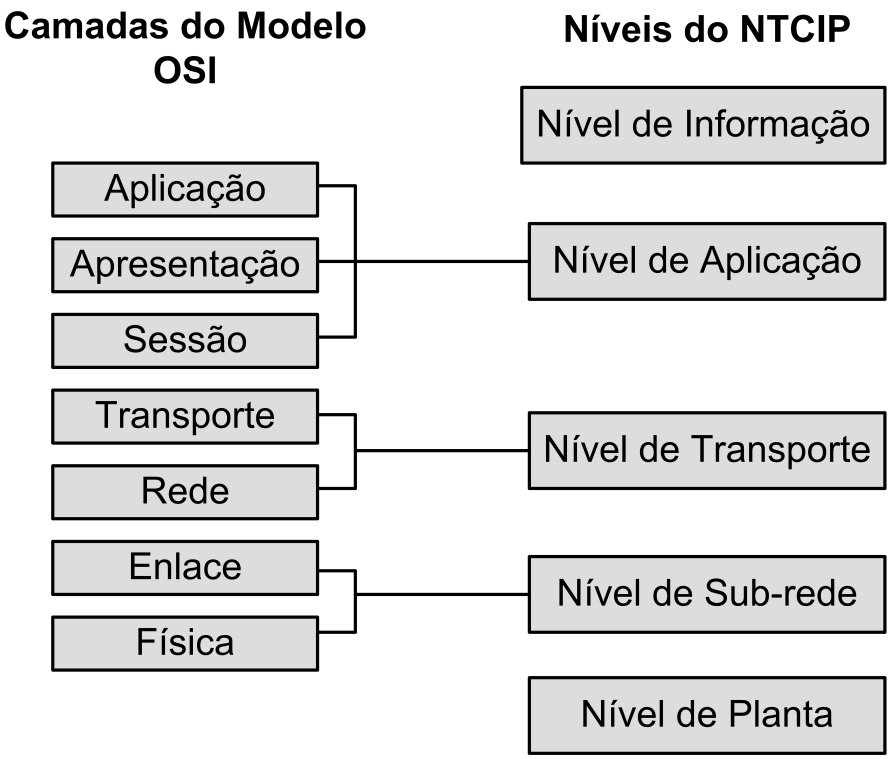

Figura 2.2: *Mapeamento das camadas OSI para os níveis NTCIP [13].*

O NTCIP utiliza uma estrutura em camadas, ou modular, inspirada na estrutura em camadas *Open Systems Interconnection* (OSI), definida pela *International Organization of Standards* (ISO), porém diferente desta, conforme será visto a seguir. O modelo OSI divide o processo de comunicação em sete camadas bem definidas, sendo que cada camada tem sua finalidade e geralmente é independente de camadas adjacentes. Já no modelo definido pelo NTCIP o processo de comunicação é dividido em cinco níveis, conforme Figura 2.2. Esta mostra a relação entre o modelo OSI, definido pela ISO, e o modelo definido pelo NTCIP.

A familia de protocolos NTCIP é baseada no conceito de camadas para manter os protocolos simples e, principalmente, abertos [28]. A seguir, discute-se os níveis NTCIP com base na pilha de protocolos das Figuras 2.2 e 2.3:

Nível de Informação: No nível de informação é feita a definição do significado de dados e mensagens. Isto é, definem padrões (tipos de dados, semântica) de informações de sistemas de transportes inteligentes (ITS). O nível de informação não trata de informações sobre a rede de comunicação, logo não existe uma camada na arquitetura OSI correspondente a ele.

Nível de Aplicação: Padrões de aplicação definem regras e procedimentos para a troca

de dados, estas regras incluem definições da gramática e da sintaxe. Como pode ser visto na figura 2.3, são exemplos de protocolos do nível de aplicação SNMP (*Simple Network Management Protocol*), STMP (*Simple Transportation Management Protocol*) e CORBA (*Common Object Request Broker Architecture*).

Nível de Transporte: Os padrões do transporte definem as regras e os procedimentos para a troca de dados da aplicação entre o ponto "A" e o ponto "X" em uma rede, incluindo roteamentos, montagem e desmontagem da mensagem e funções de gerência de rede necessárias. Por exemplo TCP (*Transmission Control Protocol*) e IP (*Internet Protocol*) são protocolos do nível de transporte do NTCIP.

Nível de Subrede: Padrões de subrede definem regras e procedimentos para a troca de dados entre dois dispositivos adjacentes sobre algum meio de comunicação. Estão no nível de subrede *Ethernet*, PPP (*Point-to-Point Protocol*) e PMPP (*Point-to-MultiPoint Protocol*).

Nível de Planta: O nível de planta é mostrado na estrutura de NTCIP somente como meio de fornecer um ponto da referência àqueles que trabalham com NTCIP. O nível de planta inclui a infra-estrutura sobre a qual os padrões de comunicação NTCIP devem ser utilizados e tem impacto direto na seleção de um nível de subrede apropriado. Na maioria de casos, tem-se uma boa idéia a respeito dos meios de comunicações utilizados na implementação do sistema no inicio da fase de projeto. As infra-estruturas definidas no nível de planta podem ser, por exemplo, par trançado, rede sem fio ou fibra óptica.

A Figura 2.3 mostra os diferentes protocolos que podem ser escolhidos em cada nível, representados como caixas, e a compatibilidade entre eles, que são representadas através das linhas que conectam as caixas. No desenvolvimento de um sistema NTCIP devem ser selecionados quais protocolos serão utilizados em cada nível. Entretanto, nem todas as configurações compatíveis fazem sentido, existindo escolhas mutuamente exclusivas. Por exemplo, o SNMP funcionando sobre o TCP/IP não é tipicamente utilizado nas aplicações de transportes.

Para a transmissão de uma mensagem em particular, deve ser utilizado pelo menos um protocolo de cada nível da estrutura NTCIP, sendo a seqüência dos protocolos utilizados na transmissão da mensagem chamada de pilha de protocolos ("*protocol stack*"). Alguns padrões do NTCIP definem um protocolo por nível, outros definem duas caixas de protocolo em um nível da estrutura.

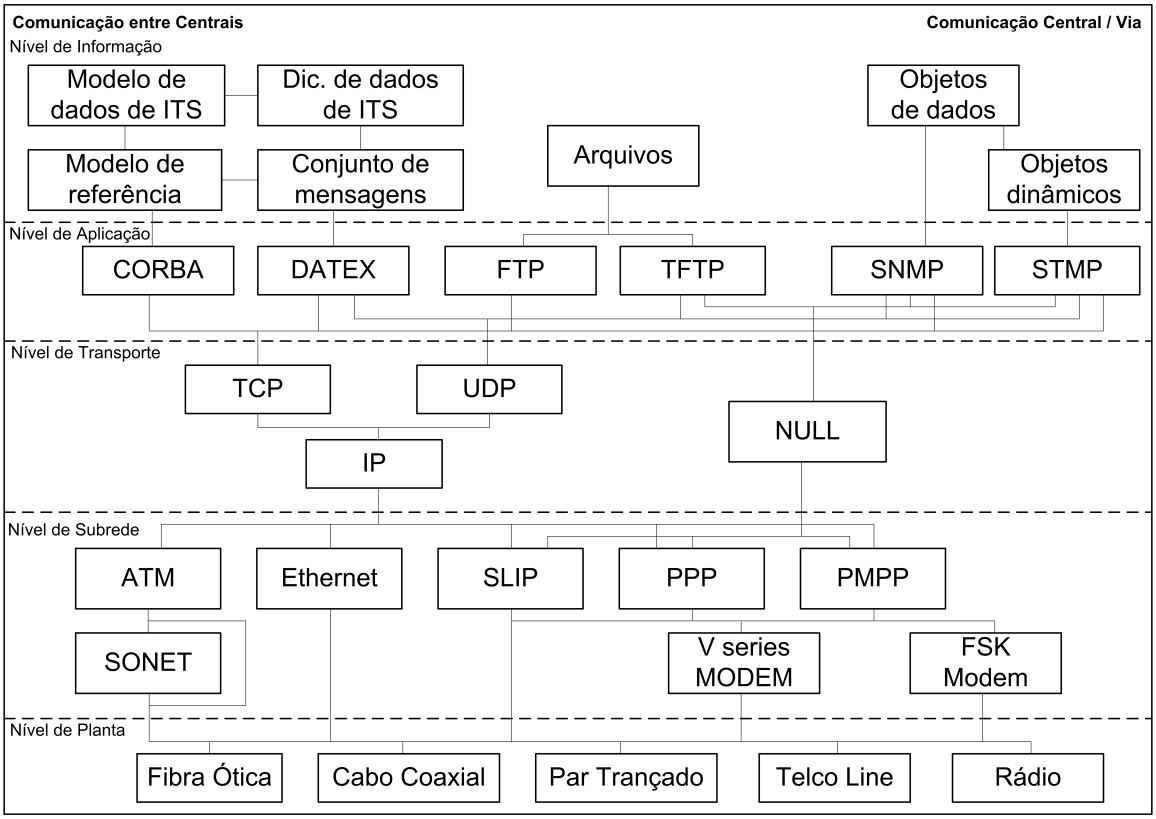

Figura 2.3: *Pilha de protocolos NTCIP [13].*

Um par de dispositivos eletrônicos pode trocar algumas mensagens utilizando uma determinada pilha de protocolos e para outras mensagens utilizar uma pilha diferente. Mas geralmente a diferença entre essas pilhas será somente em um ou dois níveis. Na Figura 2.3, as linhas que conectam padrões em níveis diferentes mostram opções de padrão em cada nível. Se houver uma linha contínua de um padrão a outro, então estes são compatíveis e podem ser utilizados juntos como parte de uma pilha de protocolo.

A maior parte dos protocolos abaixo do nível de aplicação (como por exemplo TCP, IP, PPP) apresentados na estrutura NTCIP não foi desenvolvida especificamente para o NTCIP, pelo contrario, são protocolos padronizados comercialmente e altamente utilizados na indústria de telecomunicações. A maioria dos protocolos desenvolvidos unicamente para Sistemas de Transporte Inteligentes estão nos dois primeiros níveis observados na Figura 2.3 (Nível de Informação e Nível de Aplicação). Cada pilha de protocolos NTCIP envolve uma mistura de padrões, sendo pelo menos um de cada nível.

Os primeiros padrões de NTCIP desenvolvidos eram voltados para aplicações de C/V. Assim foram criados um novo protocolo de nível de aplicação chamado de *Simple Transportation Management Protocol* (STMP), um novo protocolo do nível de transporte conhe-

cido como *Transportation Transport Profile* (T2 ou T2/NULL), e diversos padrões de elementos de dados "*object definitions*" no nível da informação. O desenvolvimento inicial do padrão NTCIP envolveu também referências a três padrões existentes: *Point-to-Point Protocol* (PPP) , *Point-to-MultiPoint Protocol* (PMPP) e *Simple Network Management Protocol* (SNMP).

No nível de aplicação, conforme pode ser observado na Figura 2.3, são definidos os seguintes padrões:

- **Para aplicações C/V** - *Simple Network Management Protocol* (SNMP), *Simple Transportation Management Protocol* (STMP) e *Simple Fixed Message Protocol* (SFMP);
- **Aplicações C/C** - *Data Exchange* (DATEX), *Common Objece Request Broker Architecture* (CORBA) e mais recentemente *Extensible Markup Language* (XML).

Neste trabalho, objetiva-se resolver problemas de padronização para a comunicação C/V. Em particular deseja-se estudar a comunicação entre uma central e um controlador semafórico. Logo, na próxima seção, será aprofundado o estudo dos protocolos de comunicação C/V.

2.3 Protocolos Central/Via

Como o foco deste trabalho está na comunicação entre centrais e controladores semafóricos, deve ser aprofundado o estudo na categoria central/via. Para isso devem ser estudados os protocolos indicados pelo padrão para aplicações desta categoria.

Como já foi visto, o NTCIP fornece três escolhas relacionadas ao protocolo do nível da aplicação para comunicações C/V: o *Simple Network Management Protocol* (SNMP), *Simple Transportation Management Protocol* (STMP) e *Simple Fixed Message Protocol* (SFMP), que está agora em desenvolvimento [16]. Estes dois últimos, utilizam o paradigma get/set utilizado no SNMP e os mesmos elementos de base de dados, definidos em séries de publicações NTCIP 1200. Entretanto, diferem no nível de complexidade para implementação e nos tipos de serviços oferecidos. A Tabela 2.1 e a Figura 2.4 ilustram os serviços oferecidos e as exigências de implementação para os três protocolos citados. Informações mais aprofundadas sobre a estrutura e o funcionamento destes protocolos serão apresentadas posteriormente neste documento.

Tabela 2.1: *Comparação dos protocolos STMP, SNMP e SFMP [16].*

Característica	**SNMP**	**STMP**	**SFMP**
Pode enviar qualquer elemento de dado?	Sim	Sim	
Uso eficiente da largura de banda?	Ruim	Bom	Em
Suporta roteamento e conexão discada?	Opcional	Opcional	desenvolvimento
Suporte ao conjunto de mensagens SNMP?	Suportado	Limitado a 13	
Facilidade de implementação	Fácil	Difícil	

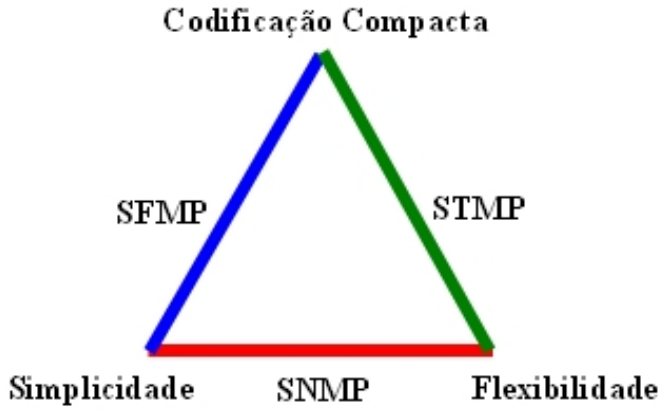

Figura 2.4: *Protocolos de Comunicação Central/Campo [16].*

Como pode ser observado na Tabela 2.1 e na Figura 2.4, o *Simple Transportation Management Protocol* (STMP) é a opção mais eficiente quanto a largura de faixa e suporta a demanda de mensagens infreqüentes do SNMP. O STMP implementa o SNMP como um subconjunto, de modo que todo o sistema de gerência que executar STMP pode também se comunicar com um dispositivo que suporte somente o SNMP. Este protocolo utiliza o SNMP para definição dos objetos dinâmicos e também, ocasionalmente, para a transmissão de mensagens que necessitem maior segurança. A maior vantagem do STMP é seu suporte aos objetos dinâmicos que, quando combinados com um esquema de codificação mais eficiente, reduzem muito o tamanho do pacote. Uma descrição mais detalhada destes objetos será apresentada na seção 3.2. Os objetos dinâmicos permitem aos usuários a definição de mensagens compostas por um único ou um conjunto de elementos de dados individuais. Entretanto, estes elementos de dados terão que ser definidos tanto no computador central quanto nos dispositivos do campo. STMP é a opção mais eficiente quanto à largura de banda e flexibilidade, porém é a que apresenta maior complexidade na implementação [16].

2.4 Conclusão

Para garantir a propriedade de interoperabilidade é necessário a adoção de um padrão para comunicação entre dispositivos de tráfego. Como foi dito nesta seção, a proposta mais avançada é a americana (NTCIP) , que apresenta uma solução interessante para comunicação entre centrais e também entre central e equipamentos de campo, existindo ainda vasta documentação disponível.

Neste trabalho objetiva-se resolver problemas de padronização para a comunicação C/V. Assim o restante deste documento será dedicado a questões relacionadas a C/V. A próxima seção discute em detalhe o protocolo SNMP e seus derivativos para transporte (STMP e SFMP), com o objetivo de avaliá-los para uso na realidade brasileira.

Capítulo 3

A Comunicação com Dispositivos de Campo no NTCIP

O capítulo anterior apresentou o NTCIP e sua estruturação em camadas de protocolos. Neste capítulo serão detalhados os protocolos de nível de aplicação entre central e via sugeridos pelo padrão e a estrutura dos objetos gerenciáveis.

O NTCIP utiliza a mesma filosofia para comunicação com equipamentos de campo que a utilizada na gerência de redes de computadores. Para o monitoramento de redes de computadores são necessárias 4 entidades: um gerente, um agente, uma base de dados e um protocolo. Neste domínio, o interesse está no monitoramento do tráfego de mensagens e no controle de certos dispositivos como *switches*, *gateways* e roteadores [34]. No NTCIP uma das sugestões oferecidas como protocolo de nível de aplicação para comunicação entre central e via é o *Simple Network Management Protocol* (SNMP). Este é o protocolo mais usado para gerência de redes de computadores, sendo um protocolo padrão da Internet para gerenciar e monitorar dispositivos de rede gerenciados. Além do SNMP, neste capítulo também serão descritos o STMP e o SFMP, ambos protocolos de aplicação derivados do SNMP e otimizados para a utilização em transportes.

3.1 SNMP

No SNMP existem dois tipos de entidades: o gerente e o agente [33]. O gerente é um computador que pode lidar com tarefas de monitoramento e controle de dispositivos.

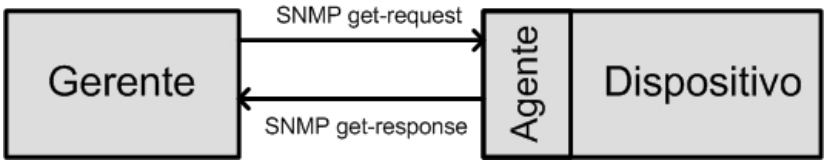

Figura 3.1: *Comunicação SNMP Gerente Agente.*

O agente é uma entidade de software executada nos dispositivos gerenciados da rede, por exemplo, um switch ou um roteador. A Figura 3.1 apresenta um diagrama da comunicação entre um gerente e um agente SNMP.

A comunicação entre gerente e agente é feita através de mensagens SNMP. As mensagens são, basicamente, de dois tipos: SET e GET. O primeiro tipo permite a alteração, pelo gerente, de um valor na base de dados do agente, enquanto que o segundo serve para obter valores da base, sejam eles valores anteriormente modificados pelo gerente ou não. O gerente envia ao agente mensagens de escrita (set) ou leitura (get) de dados, então o agente retorna o valor do objeto especificado, realizando-se assim o monitoramento e controle do dispositivo.

Para garantir a interoperabilidade entre sistemas, é necessário que ambos utilizem um mesmo padrão de base de dados. As informações gerenciadas no SNMP são definidas como objetos gerenciáveis e estão armazenados na forma de tabelas. Estes são definidos através da "*Structure of Management Information*"(SMI), que define como os objetos gerenciados são nomeados e especifica os respectivos tipos de dados associados [33]. As informações gerenciadas são representadas como um conjunto de objetos gerenciáveis, *Management Information Base* (MIB), as quais são definidas utilizando-se a sintaxe *Abstract Sintax Notation One* (ASN.1). Essas estruturas serão descritas sucintamente nas seções a seguir.

3.1.1 Objetos Gerenciáveis

Para que haja padronização no protocolo, é preciso definir objetos gerenciáveis comuns a todas as implementações de um dispositivo do mesmo tipo (*switch, gateway*, controlador semafório, p. ex.). O acesso a estes objetos também deve ser padronizado, isto é, o identificador do objeto deve ser único, independentemente do fabricante do dispositivo particular. Como forma de garantia desta unicidade, os objetos gerenciados são organizados em uma hierarquia em árvore, cuja parte pode ser observada na Figura 3.2. Esta hierarquia em árvore é a base para a atribuição de nomes do SNMP, sendo composta por um *namespace* hierárquico que contém identificadores de objeto.

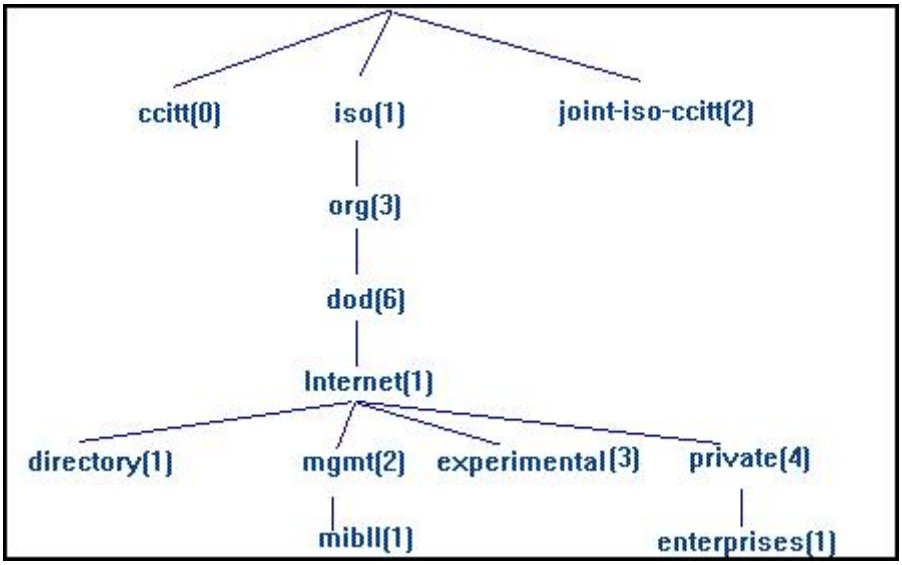

Figura 3.2: *Árvore de objetos da SMI.*

O nome ou identificador do objeto é representado pelo *Object Identifier* (OID). Este é exclusivo para cada objeto e geralmente é exibido em dois formatos: numérico e o "legível pelo ser humano". O OID do objeto é formado por uma seqüência de inteiros (ou nomes), baseada nos nós da árvore e separada por pontos. Por exemplo, o objeto "mgmt"observado na Figura 3.2 faz parte da subarvore iso(1).org(3).dod(6).internet(1) e seu OID pode ser representado como:

iso.org.dod.internet.mgmt ou 1.3.6.1.2

3.1.2 Tipos de dados

Como visto anteriormente, o SNMP utiliza a *Abstract Syntax Notation One* (ASN.1) para a definição dos objetos gerenciados. Vários tipos de dados são definidos através do ASN.1 (um resumo do funcionamento desta sintaxe pode ser encontrada no Apêndice A). Os tipos definem a informação que um objeto gerenciado pode conter, como por exemplo: *integer, octet string e IpAddress*

Por uma questão de simplificação, apenas alguns tipos do ASN.1 são utilizados na definição dos objetos do SNMP. Alguns destes tipos são: *integer, octet string, sequence e sequence of.*

A definição dos objetos incluídos na MIB é feita utilizando-se uma macro chamada OBJECT-TYPE, que é definida nas RFCs 1155 [31] e 1212 [32]. Abaixo é mostrado um

exemplo de um objeto definido utilizando a macro OBJECT-TYPE. Mais informações sobre a definição de macros pode ser encontrada no Apêndice A.

```
tempoVerde OBJECT-TYPE
  SYNTAX OCTET STRING
  ACCESS  read-write
  STATUS  mandatory
  DESCRIPTION
     " tempo de verde do plano "
  ::= { entradaTabelaPlano 2 }
```

No exemplo é definido o objeto `tempoVerde`. São apresentadas algumas características do objeto, entre elas o tipo, no caso `octet string`.

3.1.3 MIB

O agrupamento lógico de objetos gerenciados forma uma *Management Information Base* (MIB). Cada MIB deve ser desenvolvida utilizando-se a linguagem ASN.1. Como exemplo da sintaxe de definição de um objeto gerenciável, é apresentado abaixo um trecho da definição de uma MIB do NTCIP utilizando a sintaxe ASN.1 [15]. Trata-se da representação de um trecho da definição dos objetos da tabela de fases (*phaseTable*), esta é uma das tabelas mais importantes na programação de controladores semafóricos norte-americanos e será discutida na seção 4.1.2.

```
1   phaseEntry OBJECT-TYPE
2   SYNTAX PhaseEntry
3   ACCESS not-accessible
4   STATUS optional
5   DESCRIPTION
       "<Definition> Parameters for specific Actuated Controller Unit phase.
       <DescriptiveName> NTCIP-1202::ASC.phaseEntry
       <DataConceptType> Entity Type
       <Unit> " "
6   INDEX { phaseNumber }
7   ::= { phaseTable 1}
```

```
8
9   PhaseEntry ::= SEQUENCE {
10      phaseNumber INTEGER,
11      phaseWalk INTEGER,
12      phasePedestrianClear INTEGER,
13      phaseMinimumGreen INTEGER,
14      phasePassage INTEGER,
15      phaseMaximum1 INTEGER,
16      phaseMaximum2 INTEGER,
17      phaseYellowChange INTEGER,
18      phaseRedClear INTEGER,
19      ..............
20      phaseOptions INTEGER,
21      phaseRing INTEGER,
22      phaseConcurrency OCTET STRING }
```

No exemplo, as linhas de 1 a 7 definem o objeto *phaseEntry*. Nestes campos, são indicados o tipo do objeto(no caso `PhaseEntry`), o tipo de acesso permitido, o *status* e a descrição do objeto. Da linha 9 a 22 é definida a seqüência de objetos que compõem uma entrada da tabela de fase (`PhaseEntry`), alguns campos foram omitidos por brevidade. Do exemplo, interessa destacar que a sintaxe de descrição é rígida e deve ser obedecida; caso contrário, a implementação não é compatível com o padrão e, por conseguinte, os objetivos de interoperabilidade e intercambialidade não serão atingidos.

A grande vantagem da estruturação dos objetos gerenciados na forma de MIBs é a possibilidade da compilação das MIBs, existindo diversos *softwares* disponíveis capazes de compilar MIBs. Estes lêem e interpretam o arquivo de uma MIB, transmitindo informações a um software de gerência ou mostrando a base de dados a um usuário. A Figura 3.3 mostra objetos da MIB tabela de fases, do exemplo anterior, compilada por um navegador *web*.

A compilação das MIBs é bastante útil, pois garante a extensibilidade do protocolo, permitindo que novas MIBs de uso geral ou criadas especificamente para determinados dispositivos sejam facilmente compreendidas e gerenciadas por um central de controle já em operação. A Figura 3.4 ilustra a possibilidade de uma central de controle compilar tanto uma MIB padrão do NTCIP, quanto MIBs proprietárias desenvolvidas por diferentes fabricantes.

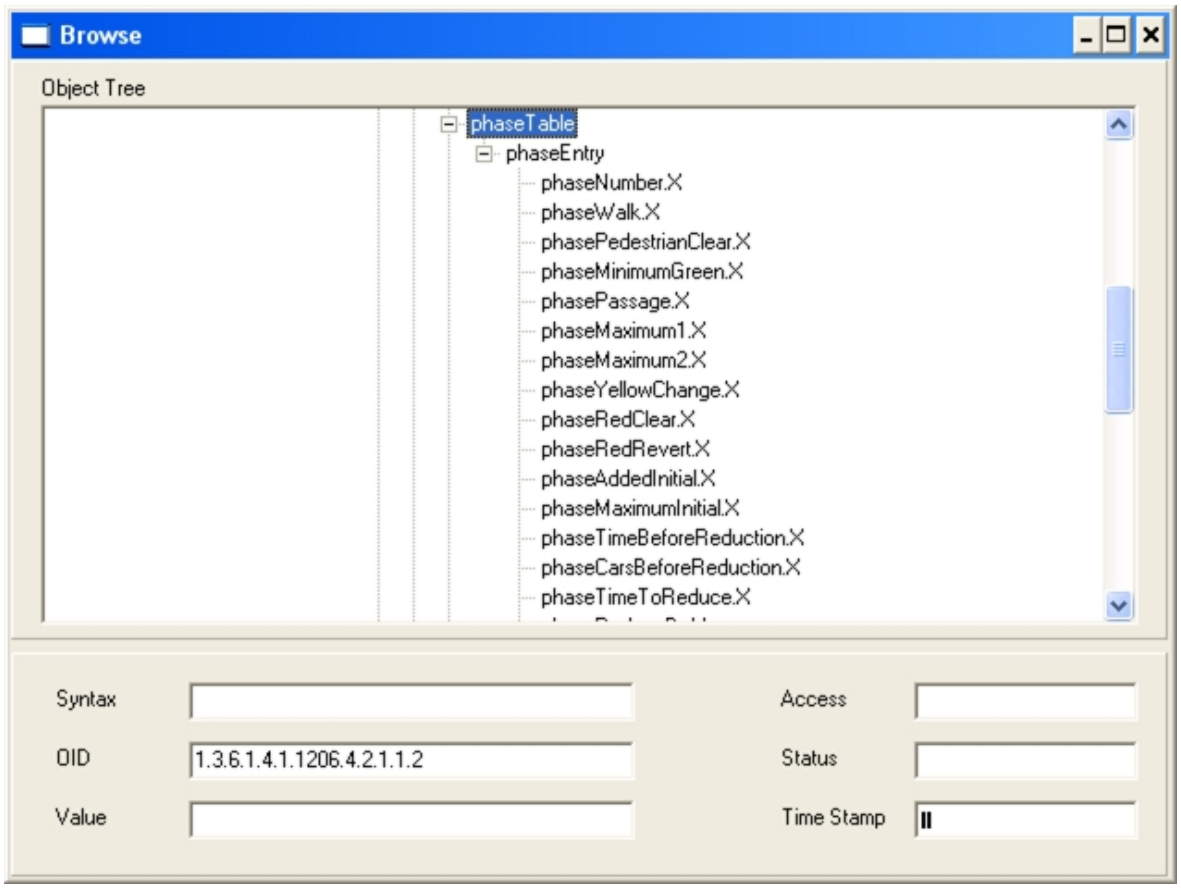

Figura 3.3: *Objetos da MIB disponíveis após compilação.*

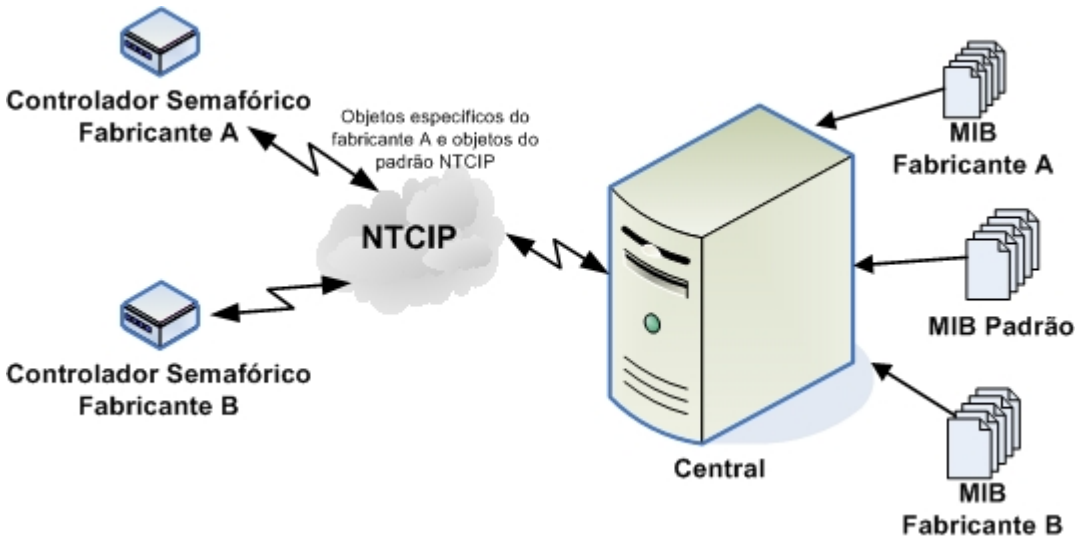

Figura 3.4: *Integração de MIBs [11]*

Já compreendida a estrutura de uma MIB, nas próximas seções serão apresentadas as operações do SNMP e a estrutura de suas mensagens.

3.1.4 Operações SNMP

O SNMPv1 (utilizado no NTCIP) suporta as operações: *get request, get nextrequest, get response, set request e trap.* As Figuras 3.5, 3.6 e 3.7 ilustram exemplos de algumas destas operações. Existem versões mais novas do SNMP, SNMPv2 e SNMPv3, que suportam outras operações como *get-bulk, notification, inform e report.* A seguir são apresentadas com mais detalhes as operações do protocolo SNMPv1 e suas unidades de dados:

Set request: É utilizado pelo gerente para escrever um valor em um objeto da MIB do agente. Para realizar a operação set o gerente deve ter permissão de escrita para esse objeto. Na Figura 3.5 é requisitada a escrita do valor `UFSC` no objeto `sysLocation`.

Get request: Get request é a PDU (*Protocol Data Unit*) mais comum no SNMP. É utilizada pelo gerente para perguntar a um agente SNMP o valor de uma determinada MIB. Na Figura 3.6 o gerente requisita o valor do objeto `nome do sistema`.

Get nextrequest: O gerente utiliza `get nextrequest` para obter dados sobre a MIB subseqüente à última MIB pesquisada. Pode ser usado para ler os dados das MIBs até o fim da tabela.

Get response: O agente responde a todos os comandos através desta PDU. Como pode ser observado nas Figuras 3.5 e 3.6, é a resposta do agente a um `get request`, `get nextrequest` ou `set request`. Nesta mensagem os campos das variáveis são preenchidos com os valores das variáveis requisitadas (para escrita ou leitura).

Trap: Um agente SNMP pode ser programado para enviar um `trap` ao gerente caso uma determinada situação ocorra. Por exemplo, como pode ser observado na Figura 3.7, se a temperatura do sistema ultrapassar um valor determinado o agente notifica o gerente através de um trap.

`Trap` é a única operação iniciada pelo agente. As demais operações do agente são sempre respostas a solicitações (`get`, `set`) do gerente.

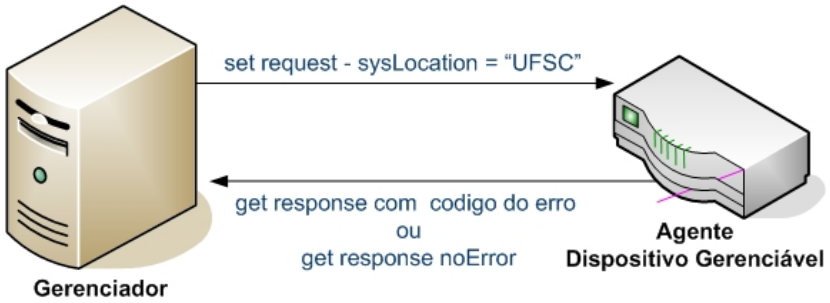

Figura 3.5: *Exemplo SNMP set-request - get-response*

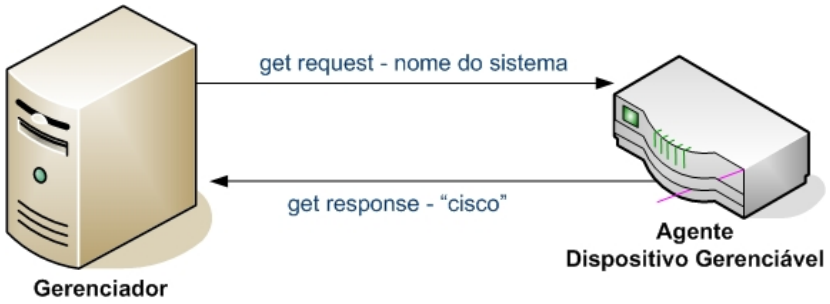

Figura 3.6: *Exemplo SNMP get-request - get-response*

Figura 3.7: *Exemplo SNMP trap*

Figura 3.8: *Formato basico de mensagem SNMPv1*

Tipo de PDU	ID da Requisição	Status do Erro	Índice do Erro	Valor do Objeto 1	Valor do Objeto 2	Valor do Objeto 3

Figura 3.9: *Formato do PDU para mensagens de Get, GetNext, Response e Set*

3.1.5 Formato das mensagens SNMP

As mensagens do SNMP versão 1 contêm duas partes: o cabeçalho da mensagem e a PDU (*Protocol Data Unit*) que é a parte que contém informação sobre a operação a ser realizada e os dados transmitidos. A Figura 3.8 ilustra o formato básico de uma mensagem SNMP versão 1.

O cabeçalho da mensagem tem tamanho igual a treze bytes e contém dois campos: versão e comunidade. No campo versão é especificada a versão do SNMP utilizada, por exemplo versão 1. Já o campo comunidade contém o nome de uma comunidade, que tanto o gerente quanto agente devem conhecer. O nome da comunidade serve para realizar uma autenticação (fraca) dos dispositivos durante a comunicação. Tanto o gerente quanto o agente só aceitam mensagens que contenham um nome de comunidade previamente conhecido, podendo ainda ser configurados diferentes níveis de acesso para diferentes comunidades.

Já a PDU contém a identificação de um comando especifico (`get`, `set`, etc..), identificadores de um objeto e, dependendo da operação, o valor do objeto envolvido. A Figura 3.9 ilustra os campos das mensagens SNMP para as operações `get, getNext`, `response` e `set`.

O campo tipo da PDU especifica o tipo da PDU transmitida, identificando o tipo da operação a ser realizada (`0 - get request`, 1 - `get nextrequest`, 2 - `get response`, 3 - `set request` e 4 - `trap`). O campo Request ID associa as requisições SNMP com as respectivas respostas. Como pode ser observado na Figura 3.9, existem também dois campos de erro. O `Status do Erro` indica um determinado tipo de erro ou zero, apenas as operações de resposta preenchem este campo. O `Índice de Erro` associa um erro a uma instância de objeto em particular, sendo também preenchido apenas nas operações de resposta. Já o campo das variáveis contém cada identificador de objeto associado ao seu valor. No caso

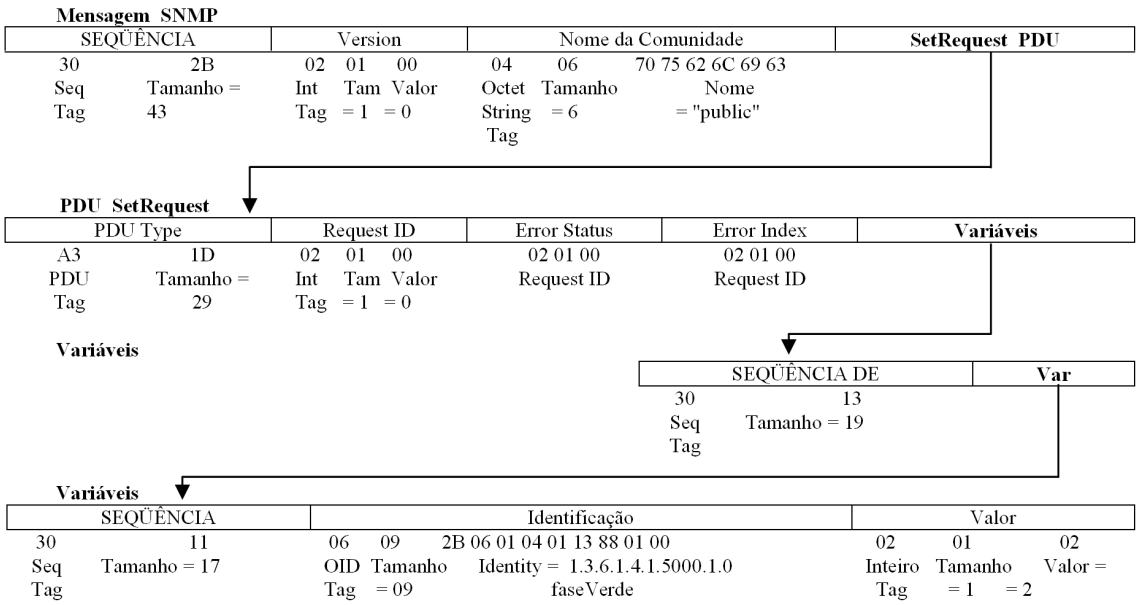

Figura 3.10: *Estrutura de uma mensagem SNMP set-request*

de operações de `get` e `getNext`, é omitido o valor das variáveis, aparecendo então apenas o identificador.

Como exemplo prático da codificação para envio de uma mensagem SNMP, considere a Figura 3.10, a qual mostra a codificação de uma operação `set-request` do SNMP. No exemplo, os dados são codificados segundo as regras de codificação BER (*Basic Encoding Rules*), cujos detalhes podem ser encontrados no Apêndice A.

Conforme pode ser observado, a mensagem SNMP possui cabeçalho e campos, como *status* de erro, que acrescentam alguns bytes ao tamanho do pacote. Outro fator responsável por alguns bytes extras é o tamanho do identificador de cada dado a ser transmitido. No caso da gerência de rede de computadores não há uma preocupação com o tamanho da mensagem SNMP, mas no caso de transporte, onde a capacidade dos meios de transmissão é bastante baixa, isso é um fator de grande relevância. Assim, com o intuito de minimizar o tamanho dos pacotes a serem transmitidos foram criados os protocolos STMP e SFMP, ambos derivados do SNMP. Nas seções seguintes será feita a descrição destes dois protocolos.

3.2 STMP

O *Simple Transportation Management Protocol* (STMP) tem uma filosofia de funcionamento bastante parecida com o SNMP, porém com algumas alterações visando minimizar o tamanho da mensagem enviada.

Uma inovação do STMP é a utilização de objetos dinâmicos. Estes são "blocos de objetos" definidos em tempo de execução [16]. A vantagem do uso de objetos dinâmicos é a redução do número de bytes do pacote a ser enviado, otimizando o uso da largura de banda. No entanto, há um aumento da complexidade do software a ser desenvolvido. A Tabela 3.1 representa a tabela de configuração e a definição dos objetos dinâmicos.

Tabela 3.1: *Tabela de configuração e definição de objetos dinâmicos [16].*

dynObjNumber	**dynObjConfig Owner**	**dynObjConfig Status**	**dynObj Index**	**dynObjVariable**
1	Proprietário Obj. 1	Status Obj. 1	1	OID 1^o dynObj
			2	OID 2^o dynObj
			...	...
			255	OID 255^o dynObj
2	Proprietário Obj. 2	Status Obj. 2	1	OID 1^o dynObj
			2	OID 2^o dynObj
			...	...
			255	OID 255^o dynObj
3	Proprietário Obj. 3	Status Obj. 3	1	OID 1^o dynObj
			2	OID 2^o dynObj
			...	...
			255	OID 255^o dynObj
...	...	...	...	...
13	Proprietário Obj. 13	Status Obj. 13	1	OID 1^o dynObj
			2	OID 2^o dynObj
			...	...
			255	OID 255^o dynObj

Na Tabela 3.1, os campos são assim definidos:

dynObjNumber: o campo `dynObjNumber` indica o índice do objeto, variando de 1 a 13. É permitida a criação de no máximo 13 objetos dinâmicos.

dynObjConfigOwner: indica quem criou o objeto.

dynObjConfigStatus: define o status do objeto. Este pode ser válido, inválido ou em construção. A Tabela 3.2 ilustra as possibilidades de alteração no status objeto.

- válido - informação válida, objeto pode ser utilizado para gerência;
- inválido - invalida e limpa campos relacionados ao objeto;
- em construção - gerente pode modificar os valores do objeto, informação não é considerada válida até que o gerente altere o *status* para válido;

dynObjIndex: posição (índice) da variável no objeto dinâmico. Cada objeto pode conter até 255 variáveis, sendo que cada variável corresponde a um objeto gerenciável de uma MIB já conhecida e compilada tanto pelo gerente quanto pelo agente;

dynObjVariable: OID de cada objeto gerenciável que compõe o objeto dinâmico.

Tabela 3.2: *Tabela de mudança de status do objeto dinâmico.*

Estado atual	**Estado requisitado**		
	inválido	em construção	válido
inválido	inválido(1)	em construção(6)	inválido(3)
em construção	inválido(2)	em construção(3)	válido(4) ou em construção(5)
válido	inválido(2)	válido(3)	válido(1)

(1) nenhuma ação ocorre, resposta noError
(2) estado muda para inválido, entradas do obj. são limpas ou deletas, resposta noError
(3) nenhuma ação ocorre, resposta badValue
(4) se o obj. dinâmico for validado o estado muda para válido, resposta noError
(5) se validação do obj. dinâmico falhar estado continua em construção, resposta genErr
(6) estado muda para em construção, resposta noError

Um objeto dinâmico pode ser configurado para agrupar até 255 objetos gerenciáveis. Assim, depois de configurado, podem ser efetuados comandos de `set e get` neste grupo de objetos utilizando como identificador apenas o índice do objeto dinâmico. Isso minimiza a quantidade de bytes a serem enviados, pois evita a transmissão repetitiva dos OID dos objetos agrupados.

A Figura 3.11 ilustra um exemplo de configuração de um objeto dinâmico, no caso objeto número 3.

Inicialmente é enviado um comando `snmp-set` definindo o *status* do objeto como inválido, assim os campos relacionados ao objeto são limpos e invalidados. Na seqüência, o

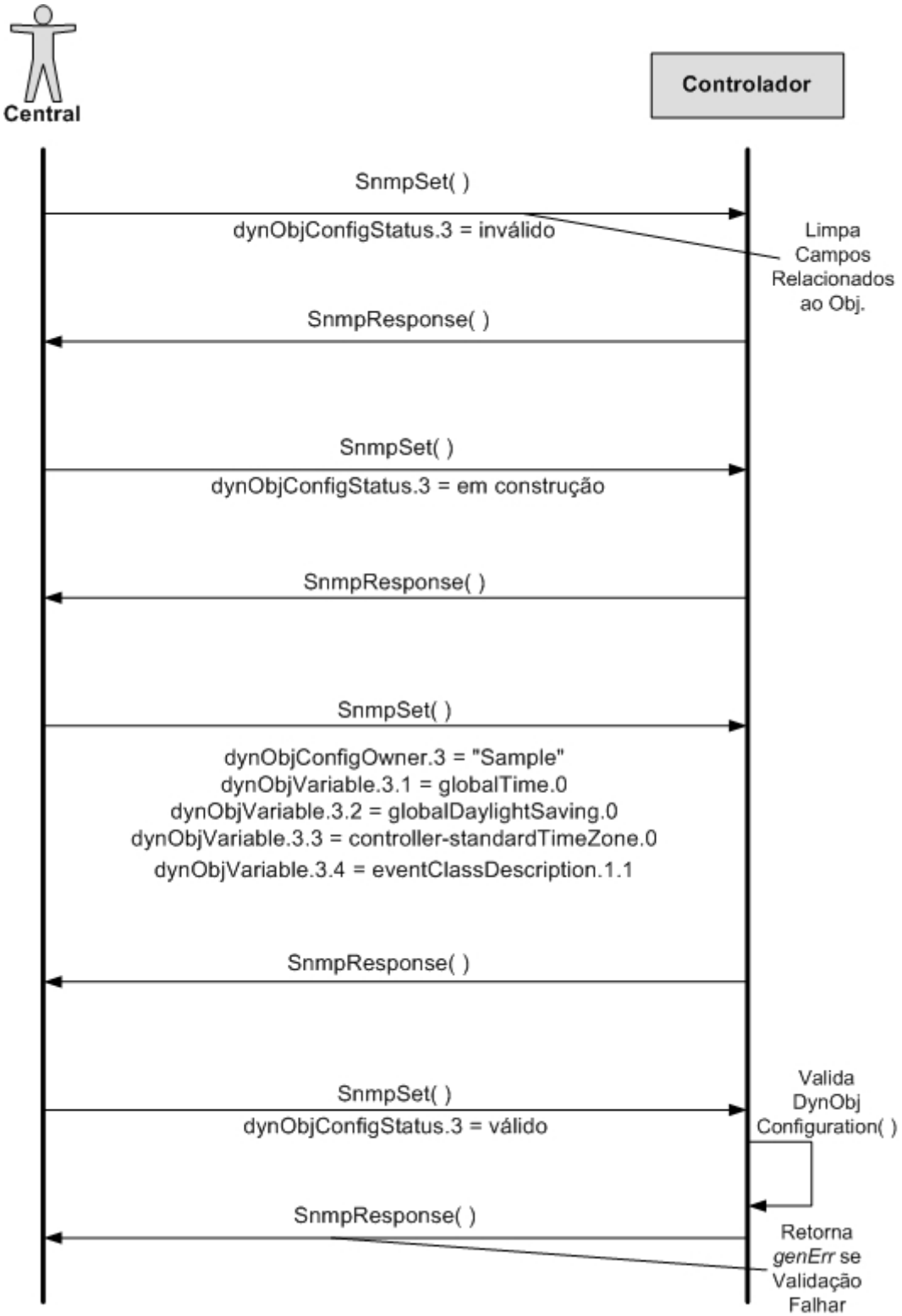

Figura 3.11: *Exemplo de configuração de um objeto dinâmico [16].*

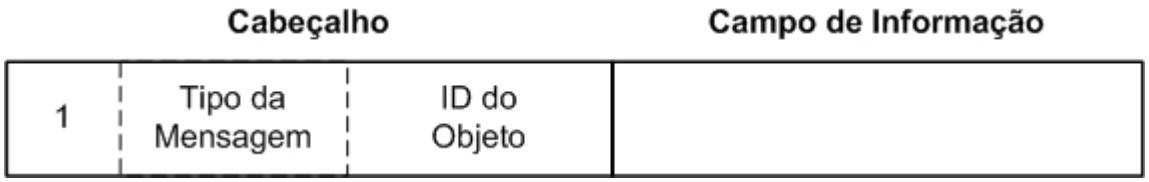

Figura 3.12: *Estrutura dos pacotes STMP [16].*

status é setado como em construção, indicando que os campos do objeto serão alterados. Então os campos `dynObjVariable` com índice de 1 a 4 são preenchidos com os OID dos objetos gerenciáveis `globalTime`, `globalDaylightSaving`, `controller-standardTimeZone` e `eventClassDescription` respectivamente. Finalizando a configuração, o status do objeto é setado como valido e a informação contida nos seus campos passa a ser então considerada válida.

O pacote STMP é composto por um cabeçalho e um campo de informação, conforme pode ser observado na Figura 3.12.

O cabeçalho STMP tem tamanho de 1 byte, onde cada bit tem o seguinte significado:

7 Formato da PDU:

`0 - reservado para o SNMP ou outro uso futuro.`

`1 - Indica que o pacote é STMP ou SFMP.`

6-4 Tipo da Mensagem (descrição do tipo da mensagem se aplica apenas a mensagens STMP, onde o formato da PDU é 0x1 e o Object ID é entre 0x0001 e 0x1101):

`000 - STMP-GetRequest-PDU`

`001 - STMP-SetRequest-PDU`

`010 - STMP-SetRequest-NoReply-PDU`

`011 - STMP-GetNextRequest-PDU`

`100 - STMP-GetResponse-PDU`

`101 - STMP-SetResponse-PDU`

`110 - STMP-ErrorResponse-PDU`

`111 - Reservado para uso futuro.`

3-0 ID do objeto:

`0000 - Reservado para SFMP`

`0001-1101 - ID de objetos dinâmicos do STMP`

```
1110 - Reservado para uso futuro
1111 - Reservado para uso futuro
```

O campo de informação deve ser nulo para mensagens do tipo `STMP-GetRequest`, `STMP-GetNextRequest` e `STMP-SetResponse`. Já para mensagens do tipo `STMP-GetResponse`, `STMP-SetRequest` e `STMPSetRequestNoReply` o campo de informação deve conter uma série de componentes, cada um correspondente a um dos objetos referenciados, seguindo a ordem associada ao `dynObjIndex`. Cada componente deve ser codificado seguindo a regra *Octet Encoding Rules* (OER) que elimina os campos de tipo e tamanho quando conhecido.

Abaixo pode ser observado um exemplo de um comando `get-request` para o objeto dinâmico numero 3 configurado na Figura 3.11.

PDU - GetRequest:

```
Cabeçalho:
83  - stmp-get-request objeto dinamico numero 3
```

PDU - GetResponse:

```
Cabeçalho:
C3 - stmp-get-response objeto dinamico numero 3

Campo de informação:
3A 24 63 20 variável 1 = globalTime.0 = November 29, 2000 at 2:00 am UTC
03 variável 2 = globalDaylightSaving.0 = 3 = enableUSDST
FF FF B9 B0 variável 3 = controller-standardTimeZone.0 = -18000 = EST
06 53 61 6D 70 6C 65 variável 4 = eventClassDescription.1 (6 bytes)
= "Sample"
```

O comando `get-request` contém apenas o campo do cabeçalho com tamanho de um byte, indicando o comando `stmp-get-request` para o objeto dinâmico numero 3 . Como resposta é enviado um comando `get-response`. Este, além do cabeçalho, tem o campo da informação contendo o valor dos objetos na ordem em que foram configurados.

3.3 SFMP

O SFMP (*Simple Fixed Message Protocol*) é uma versão simplificada e mais compacta do SNMP [16]. Traz algumas mudanças que reduzem a complexidade e o tamanho dos pacotes. São elas:

- Identifica seqüências de objetos (objetos compostos) utilizando um OID que identifica todo o grupo, evitando referenciar cada elemento do grupo separadamente.
- Assume que todos os objetos compostos estão abaixo do nó NEMA, diminuindo assim o tamanho do OID o que facilitando a codificação e diminui o tamanho da mensagem.
- Ao invés do BER, utiliza *Octet Encoding Rules* (OER) para codificação dos componentes. As regras de codificação BER Apêndice A;
  - BER - TLV - tamanho - tipo - valor
  - OER - elimina campos de tipo e tamanho quando conhecido.
- Modificação na estrutura dos pacotes, eliminando campos desnecessários. Por exemplo, no pacote get-request não é necessário um campo para os dados, nem campos de erros, porém os campos de OID do objeto e `request-number` são mantidos.

Abaixo é mostrado um exemplo de pacote SFMP representando um comando `SFMP-GerRequest` para o objeto `globalTime`.

```
Bytes                SFMP Get-Request Data-Packet:
80                   CHOICE = [0] (i.e., context specific) = sfmp-get
                     SFMP-GetRequest-PDU = SEQUENCE
14                      Preamble = 0001 0100 =
                            Bit 1 = 0 - extension absent
                            Bit 2 = 0 - default version = version-1
                            Bit 3 = 0 - default community name = "public"
                            Bit 4 = 1 - request-number present
                            Bit 5 = 0 - error-data absent
                            Bit 6 = 1 - message-oid present
                            Bit 7 = 0 - data absent
                            Bit 8 = 0 - reserved
```

```
01                          request number = 1
06 04 02 06 03 01 00        message-oid of 6 bytes at nema.4.2.6.3.1.0
                             = globalTime.0
```

3.4 Conclusão

No padrão dos EUA são indicados, para nível de aplicação, os protocolos SNMP, STMP e SFMP. Todos esses seguem uma filosofia bastante parecida, porém apresentam algumas diferenças de configuração e estrutura de seus pacotes. Baseando-se neste estudo do NTCIP, na próxima seção avaliaremos a adequação do padrão norte americano à prática de engenharia de tráfego brasileira.

Capítulo 4

Avaliação da Compatibilidade do Padrão NTCIP à Realidade Brasileira

No capítulo 2 foi apresentada a estrutura de comunicação proposta pelo padrão NTCIP, seguindo-se no capítulo 3 com um estudo mais detalhado dos protocolos de nível de aplicação sugeridos pelo padrão. Este capítulo tem o intuito de avaliar a adequação do NTCIP, à realidade da engenharia de tráfego brasileira.

Ao se realizar esta análise, foram encontradas dificuldades de nível cultural e de nível tecnológico. Nas próximas seções, ambas serão estudadas detalhadamente, objetivando-se a obtenção de possíveis soluções para os problemas.

4.1 Estado da Prática de Engenharia de Tráfego Urbano

Inicialmente, nesta seção, será feito um estudo da forma usual de programação semafórica do Brasil, contrastando-a, em seguida, com a forma de programação proposta pelo NTCIP, visando obter soluções para as divergências encontradas.

4.1.1 Forma Usual de Programação Semafórica no Brasil

Atualmente no Brasil não existe um padrão para programação de controladores semafóricos. Como indicativo da prática corrente, pode-se verificar como os fabricantes nacionais

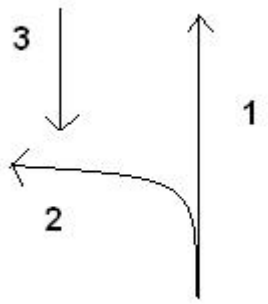

Figura 4.1: *Movimentos do tráfego em um cruzamento.*

do setor definem seus equipamentos. Neste sentido, dos quatro maiores fabricantes, três deles usam conceito de estágio/fases [3] [7] [9] e um emprega o conceito de intervalos [35].

Em acordo com esta constatação, editais de licitação de equipamentos de controle de tráfego elaborados pelas gerências de trânsito também especificam programação por estágios. Por exemplo, a Companhia de Engenharia de Tráfego de São Paulo (CET-SP) publica, em seus editais, informações detalhadas da operação semafórica, as quais são baseadas no conceito de estágio; o uso de outra estratégia deve ser fortemente justificado [5].

Um estágio é um período de tempo dentro de um ciclo onde as indicações luminosas do cruzamento não mudam de aspecto, e uma ou mais correntes de tráfego e/ou pedestres têm o direito de passagem [8]. Para facilitar o entendimento dos parâmetros de configuração de um controlador segundo o conceito de estágio, apresenta-se um exemplo com tabelas de configuração de controladores semafóricos de um fabricante que atende à especificação da CET-SP. Equipamentos fabricados por outras empresas podem apresentar algumas diferenças na forma de programação, mas os conceitos básicos são equivalentes.

Para simplificar o problema, sejam os sentidos de movimentos apresentados na Figura 4.1 e a utilização de um controlador de tempo fixo para resolução dos conflitos. Conforme pode ser observado na Figura 4.2, pode-se programar um plano semafórico para este cruzamento com o uso de dois estágios.

- Estágio 1:
  - Movimentos ativos: 1 e 2;
  - Movimento 1: tempo de verde 20s;
  - Movimento 2: tempo de verde 20s, tempo de amarelo 4s e tempo de vermelho de limpeza 1s;

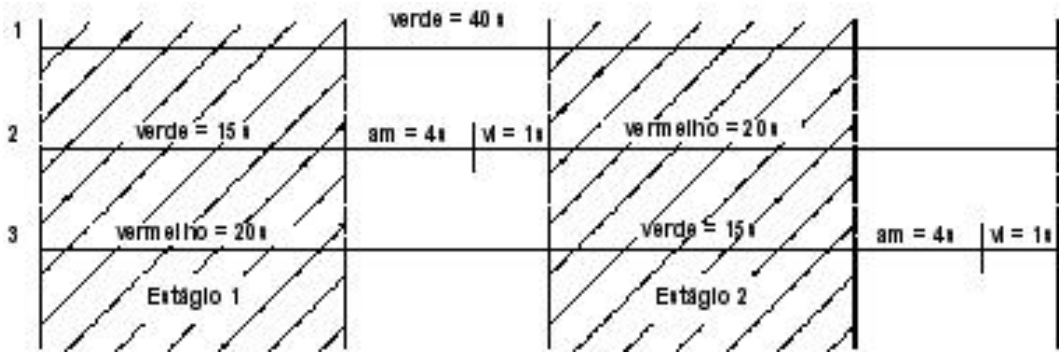

Figura 4.2: *Diagrama de tempos com estágios projetados.*

- Estágio 2:
  - Movimentos ativos: 1 e 3;
  - Movimento 1: tempo de verde 20s;
  - Movimento 3: tempo de verde 20s, tempo de amarelo 4s e tempo de vermelho de limpeza 1s.

Para exemplificar a programação de um plano para o cruzamento da Figura 4.1, utilizam-se os parâmetros de configuração de controladores semafóricos para estágios, planos e verdes conflitantes de um fabricante nacional [3]. Na Tabela 4.1, encontram-se os parâmetros de configuração de estágios, que são:

- `num-estagio:` número inteiro que identifica um estágio;
- `mm-verde:` tempo de verde mínimo para o estágio, dado em s;
- `mm-amarelo:` tempo de amarelo mínimo para o estágio, dado em s;
- `mm-amarelo-ant:` tempo de amarelo antecipado mínimo, no caso do estágio servir para indicação de travessia de pedestres. Indica o mínimo tempo em segundos em que a indicação de vermelho piscante de pedestres deve iniciar antes do início do amarelo do estágio veicular;
- `mm-vermelho-int:` tempo mínimo de vermelho integral, definido como indicação de vermelho em todos os grupos semafóricos do cruzamento;
- `mm-vml:` tempo mínimo de vermelho de limpeza do estágio;
- `gs-ativo:` movimentos permitidos no estágio.

Tabela 4.1: *Tabela de estágios relacionada ao exemplo da Figura 4.1.*

num-estagio	mm-verde	mm-amar	mm-amar-ant
1	12	4	0
1	12	4	0
mm-verm-int	**mm-vml**	**gs-ativo**	
0	1	1, 2	
0	1	1, 3	

Na Tabela 4.2 aparecem os parâmetros que definem os planos semafóricos. Basicamente, tais planos consistem da seqüência de estágios e do tempo de permanência em cada estágio.

Tabela 4.2: *Tabela de planos relacionada ao exemplo da Figura 4.1.*

num-plano	nr-estag	tmp-verde	ext-verde	max-verde	tmp-amarelo
1	1	15	0	15	4
1	2	15	0	15	4
tmp-vm-int	**tmp-vml**	**tipo**	**dvi**	**amare-antec**	**defasagem**
10	1	0	0	0	0
10	1	0	0	0	0

As linhas da tabela definem um passo do plano semafórico. Os parâmetros de programação do plano são:

- `num-plano:` número inteiro que identifica o plano semafórico;
- `nr-estagio:` número do estágio definido na tabela de estágios;
- `tmp-verde:` duração do verde para o estágio;
- `ext-verde:` tempo de extensão do verde no caso de deteção de veículo por laço indutivo; válido apenas para operação atuada pelo tráfego;
- `max-verde:` máximo tempo de verde, no caso de operação atuada pelo tráfego;
- `tmp-amarelo:` duração do amarelo do estágio;
- `tmp-vm-int:` duração do vermelho integral;
- `tmp-vml:` duração do vermelho de limpeza do estágio;
- `dipo:` indica demandas de pedestres ativas;

- `dvi:` indica demandas veiculares ativas;
- `amare-antec:` tempo de amarelo antecipado do passo;
- `defasagem:` momento de início do passo 1 em relação à referência da rede.

Com os parâmetros acima, pode-se definir a configuração das indicações semafóricas para um cruzamento. Para completar a programação, deve ser possível definir a hora em que inicia a vigência do plano, comumente programada em uma tabela de horas. Nesta tabela, associa-se os planos a horas e aos dias da semana (de domingo a sábado) em que deve vigorar o plano no horário pré-especificado. Também, datas especiais devem ser programadas, e isto é feito na tabela de datas. Estas tabelas não serão discutidas em detalhe neste trabalho.

Por último, uma tabela importante para fins de segurança da programação semafórica é a tabela de verdes conflitantes, apresentada na Tabela 4.3. Nesta tabela, são definidos quais movimentos estão em conflito. Durante a operação do controlador, os verdes são continuamente monitorados e na eventualidade de verdes em conflito acenderem simultaneamente, o controlador deve iniciar operação em modo de amarelo intermitente. Este tipo de ocorrência deve-se à má programação ou a um defeito eletrônico no equipamento.

Tabela 4.3: *Tabela de verdes conflitantes relacianada a exemplo da Figura 4.1.*

Movimento	1	2	3
1	-		
2		-	X
3		X	-

Conforme discutido anteriormente, a maioria dos equipamentos de controle semafórico atualmente encontrados em campo no país apresenta forma de programação similar à apresentada neste texto. Uma vez conhecida a prática de engenharia de tráfego no Brasil no que tange à programação semafórico, na próxima sessão estudaremos o padrão proposto pela FHWA visando verificar a sua adequação à prática de engenharia de tráfego nacional.

4.1.2 NTCIP

Controladores semafóricos adquiridos em projetos com recursos federais nos EUA devem seguir os padrões NTCIP 1201 e 1202 ([12] [15]). Diferenciado-se da prática de engenharia

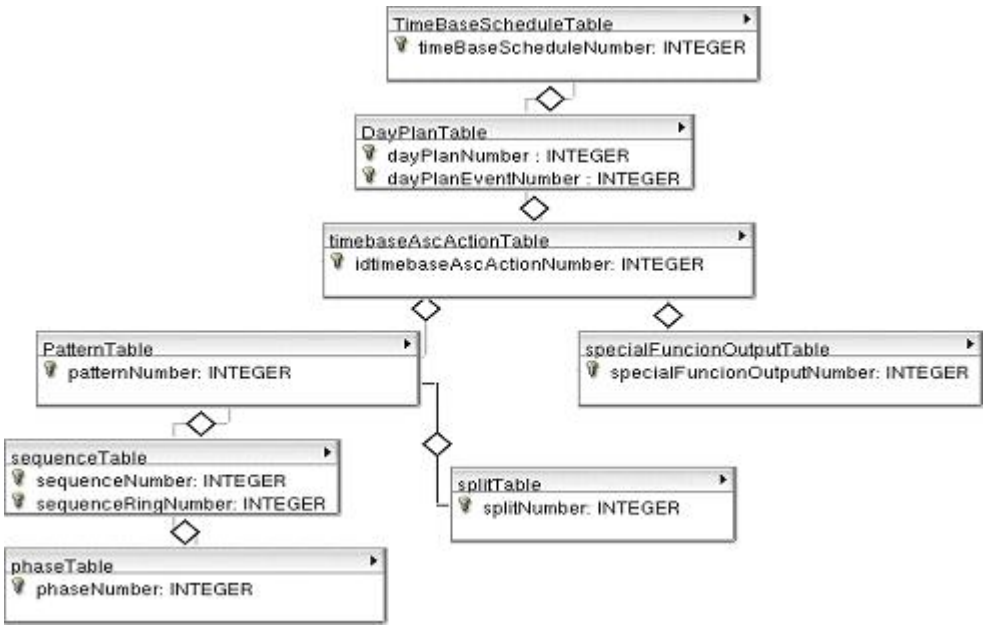

Figura 4.3: *Tabelas NTCIP para programação semafórica.*

de tráfego nacional, que utiliza a estratégia de programação por estágios, no padrão NTCIP a programação do plano é feita utilizando-se o conceito de fase.

Uma fase é compreendida pelos tempos de verde, amarelo e vermelho de limpeza dentro de um ciclo concedidos a uma corrente de tráfego [15]. Além de uma tabela de fases, o padrão prevê diversas outras tabelas de dados que se interconectam, permitindo assim a programação dos controladores. A Figura 4.3 representa algumas tabelas de dados envolvidas na programação semafórica e a conexão entre elas. A hierarquia apresentada permite verificar que, no topo, está a tabela de escalonamento de eventos, a qual usa os dados da tabela de planos diários, que por sua vez usa a tabela de ações a serem tomadas e assim por diante.

Para melhor compreender o modelo de programação NTCIP, será utilizado o mesmo exemplo de cruzamento mostrado na seção anterior. Contudo, a programação do plano será feita utilizando os dados definidos no padrão NTCIP, ou seja, utilizando a estratégia de programação por fases ao invés de estágios.

No cruzamento com três correntes de tráfego mostrado na Figura 4.1, para cada corrente de tráfego há uma fase correspondente, conforme pode ser observado na Figura 4.4.

No caso apresentado, tem-se a fase 1 com tempo de verde 40s, igual à duração do ciclo. Já a fase 2 apresenta tempo de verde igual a 15s, amarelo 4 e vermelho de limpeza também 1s. Na seqüência pode ser observada a fase 3 também com tempo de verde de 15s e amarelo e vermelho de limpeza iguais a 4s e 1s, respectivamente.

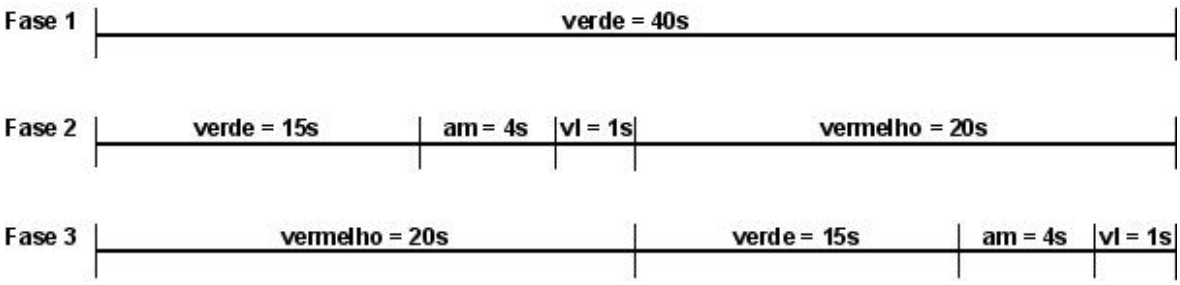

Figura 4.4: *Diagrama de Tempo para as fases do exemplo.*

Parte da programação de um plano para essa interseção, utilizando-se os objetos definidos no NTCIP, pode ser observada nas Tabelas 4.4 e 4.5. Nestas tabelas estão representados os dados das fases e das seqüências das fases, respectivamente. A sequência das fases usa o conceito de anel (*ring*), que consiste de uma seqüência de duas ou mais fases conflitantes organizadas em uma ordem estabelecida [15]

Tabela 4.4: *Tabela de Fases do NTCIP.*

number	**Walk**	**PedesClear**	**MinGre**	**Passage**
1	0	0	40	0
2	0	0	15	0
3	0	0	15	0
Maximum1	**Maximum2**	**YellowChan**	**RedClear**	**RedRevert**
40	40	0	0	0
15	15	4	1	20
15	15	4	1	20
AddInitial	**MaxInitial**	**TimeBefoRed**	**CarsBefRed**	**TimToRed**
X	X	X	X	X
X	X	X	X	X
X	X	X	X	X
ReducBy	**MiniGap**	**Dnamic**	**DinMaxStep**	**Startup**
X	X	X	X	3
X	X	X	X	3
X	X	X	X	3
PhaseOption	**Ring**	**Concurrency**		
1001 0000 ...	1	2,3		
1001 0000 ...	2	1		
1001 0000 ...	2	1		

Conforme pode ser observado no exemplo na Figura 4.1, as fases 2 e 3 são conflitantes, logo elas pertencem ao mesmo anel e devem ser habilitadas em momentos diferentes. No exemplo apresentado, habilita-se primeiro a fase 2 e em seguida a fase 3. Como a fase 1 não

Tabela 4.5: *Tabela de Seqüência do NTCIP.*

Number	RingNumber	Data
1	1	1
1	2	2, 3

está em conflito com as outras fases, ela pertence a um anel diferente e permanece habilitada durante todo o ciclo.

Na tabela de fases estão os dados de configuração de cada fase. Muitos campos da tabela são destinados a controladores atuados pelo tráfego. Para simplificar o problema, trata-se neste trabalho apenas de controladores de tempo fixo, logo o preenchimento destes campos não será estudado no presente exemplo.

Para cada fase, deve ser informado o tempo de verde da fase (*phaseMinimunGreen*), o tempo de amarelo (*phaseYellowChange*) e o tempo de vermelho de limpeza (*phaseRedClear*). Além disso, devem ser informados o estado de inicialização de cada fase (*phaseStartup*), o anel que ela pertence (*phaseRing*) e as fases que não estão em conflito com ela (*phaseConcurrency*). No campo phaseOptions, cada bit 1 corresponde a uma determinada opção estar habilitada, com 0 indicando inibição. No exemplo dado, as opções habilitadas para as fases são: "*Enabled Phase*"e "*Non-Actuated* 1", indicando que as fases estão habilitadas e podem trabalhar em modo não atuado.

A forma de programação de um plano apresentada acima difere da prática nacional. No caso do NTCIP, a programação está baseada em fases, o que difere da forma usualmente praticada no Brasil onde o plano é programado baseando-se em estágios. Outro fator importante a ser observado é que, no caso do NTCIP, os dados sobre grupos semafóricos conflitantes estão anexados à tabela de fases, não estando de acordo com especificações de editais brasileiros que exigem que estes estejam em uma tabela independente [5].

Tendo em vista as particularidades da programação semafórica acima apresentadas, na próxima seção será proposto um padrão baseado na tecnologia do NTCIP, porém com algumas modificações visando uma melhor adaptação à cultura nacional.

4.1.3 Proposta de MIBs Nacionais

Conforme já discutido anteriormente, é importante a adoção de um padrão para comunicação de controladores semafóricos que reflita adequadamente a cultura local de programa-

ção semafórica, visando assim minimizar o impacto gerado pela sua adoção. Em particular, propõe-se que um padrão nacional permita que se mantenha a prática de programação por estágios.

Ao estudar o modelo NTCIP de comunicação, é possivel perceber diversas dificuldades em adotá-lo como padrão nacional. Além da forma diferente de programação, defronta-se com o problema de compreensão da nomenclatura, necessitando a introdução de novos conceitos como padronagem (*patterns*) e anel (*ring*), além dos usuais fase e intervalo.

A solução encontrada para elaboração de uma proposta de possível padrão nacional de comunicação de controladores semafóricos é utilizar a tecnologia NTCIP adaptando-a para melhor adequá-la à cultura local. Ou seja, aproveitando a idéia de gerência remota de dispositivos, utilizando SNMP, porém definindo outros conjuntos de objetos gerenciáveis para controladores semafóricos.

Abaixo pode ser observado um trecho da definição das novas MIBs, representado uma entrada das tabela de estágio, plano e verdes conflitantes. Conforme estabelecido no protocolo SNMP, as MIBs estão escritas segundo a sintaxe ASN.1. O codigo fonte completo das MIBs implementadas pode ser encontrado no Apêndice B.

Entrada Tabela de Estágio:

```
EntradaTabelaEstagio::= SEQUENCE {
          numeroEstagio INTEGER,
          minimoVerde INTEGER,
          minimoAmarelo   INTEGER,
          minimoAmareloAntecipado  INTEGER,
          minimoVermelhoIntegral   INTEGER,
          minimoVermelhoLimpeza  INTEGER,
          gruposSemaAtivos    INTEGER }
```

Entrada Tabela Plano:

```
EntradaTabelaPlano::= SEQUENCE {
          numeroPlano INTEGER,
          tempoVerde INTEGER,
          tempoExtensãoVerde   INTEGER,
          tempoMaximoVerde  INTEGER,
          tempoAmarelo   INTEGER,
```

```
            tempoVermelhoIntegral  INTEGER,
            tempoVermelhoLimpeza  INTEGER,
            tipo  INTEGER,
            dvi  INTEGER,
            tempoAmareloAntecipado INTEGER }
```

Entrada Tabela Verdes Conflitantes:

```
EntradaTabelaVerdesConflt::= SEQUENCE {
            numeroGrupo INTEGER,
            grupoConflito   INTEGER }
```

Na definição do conjunto de objetos gerenciáveis, foi dada prioridade em se manter a prática nacional de estratégia de programação por estágios. Assim, seguindo as especificações de um fabricante nacional que as adota, e em concordância com critérios de gerências de tráfego [5], são definidas três tabelas: tabela de estágio, tabela de planos e tabela de verde conflitantes. As duas primeiras são bastante similares às tabelas de estágio e plano já apresentadas na Seção 4.1.2.

A tabela de verdes conflitantes é, nesta proposta, caracterizada por um objeto independente dos anteriores, diferentemente do padrão norte-americano. A tabela de verdes conflitantes apresenta duas colunas, uma indicando o número do grupo e outra os grupos conflitantes a este. Os grupos em conflito são indicados na forma binária; um bit igual a 1 indica que há conflito entre os grupos e um bit 0 indica que não há conflito.

A proposta do NTCIP se baseia no já consolidado protocolo de gerência de redes SNMP para troca de mensagens e utiliza seu modelo gerente - agente. Ao se definir MIBs mais compatíveis com a realidade nacional, evita-se a necessidade, no caso de adoção do padrão, de uma alteração brusca na prática de engenharia de tráfego do país.

4.1.4 Implementação

Como forma de avaliar a corretude das MIB´s nacionais desenvolvidas, estas foram compiladas utilizando o software NTCIP *Exerciser*. O *Exerciser* é um software desenvolvido pela FHWA visando prover uma ferramenta para que agências e indústrias possam testar a interoperabilidade de equipamentos de campo NTCIP [36]. É uma ferramenta de testes e desenvolvimento voltada para desenvolvedores e avaliadores do sistema NTCIP [27]. Permite

a compilação de MIB´s escritas em ASN.1, e a comunicação entre dois equipamentos. No caso, um simulando uma central e outro um equipamento de campo, havendo a possibilidade de comunicação entre eles via *serial*, *modem* ou *dial-up*.

Quanto aos protocolos de aplicação, segundo um dos proponentes do NTCIP, Robert De Roche [30], o *Exerciser* apresenta alguns fragmentos de código STMP, mas o suporte não foi completamente implementado. A implementação do STMP e dos objetos dinâmicos foi baseada em um rascunho de 1997, não estando de acordo com o padrão atual [27]. Já a comunicação SNMP funciona corretamente.

Na Figura 4.5 podem ser observadas as MIB's nacionais já compiladas pelo *Exerciser*. O nó *tpBr* representa objetos relacionados a controladores de tempo fixo brasileiros e está alocado abaixo do nó nema. A alocação das MIB's nacionais abaixo do nó *devices* é uma forma de representar a facilidade de incorporar as MIB's nacionais aos objetos já definidos pelo padrão NTCIP, sendo que o endereço final das MIB's na árvore SMI é uma questão que deve posteriormente ser discutida e avaliada.

Após as MIB's brasileiras serem compiladas, tanto no lado do agente como no gerente, é possível, utilizando operações SNMP, ler e alterar valores dos objetos gerenciáveis do equipamento de campo. A seguir, será apresentado um exemplo onde é alterado o valor do objeto `minimoVerde`.

Como pode ser visto na Figura 4.6, o objeto `minimoVerde` é referente a uma entrada da tabela de estágio e inicialmente contém valor 15. Após ser indicado um novo valor, no caso 23, é enviada uma mensagem de escrita para o objeto.

A Figura 4.7 mostra a codificação, lado do gerente, da mensagem de escrita enviada. Os dados são codificados conforme a regra de codificação BER (Apêndice A) e correspondem a uma operação `set-request`, para o objeto `minimoVerde`, indicando valor 23.

Então na Figura 4.8 é mostrado o agente decodificando a mensagem de `set-request` recebida. Após a decodificação, o novo valor (23) é atribuído ao objeto e o agente envia ao gerente um `get-response`, confirmando o recebimento da mensagem.

Assim, através do Exerciser, foi possível compilar as MIBs nacionais e realizar operações de `set-request` e `get-request` utilizando os novos objetos definidos.

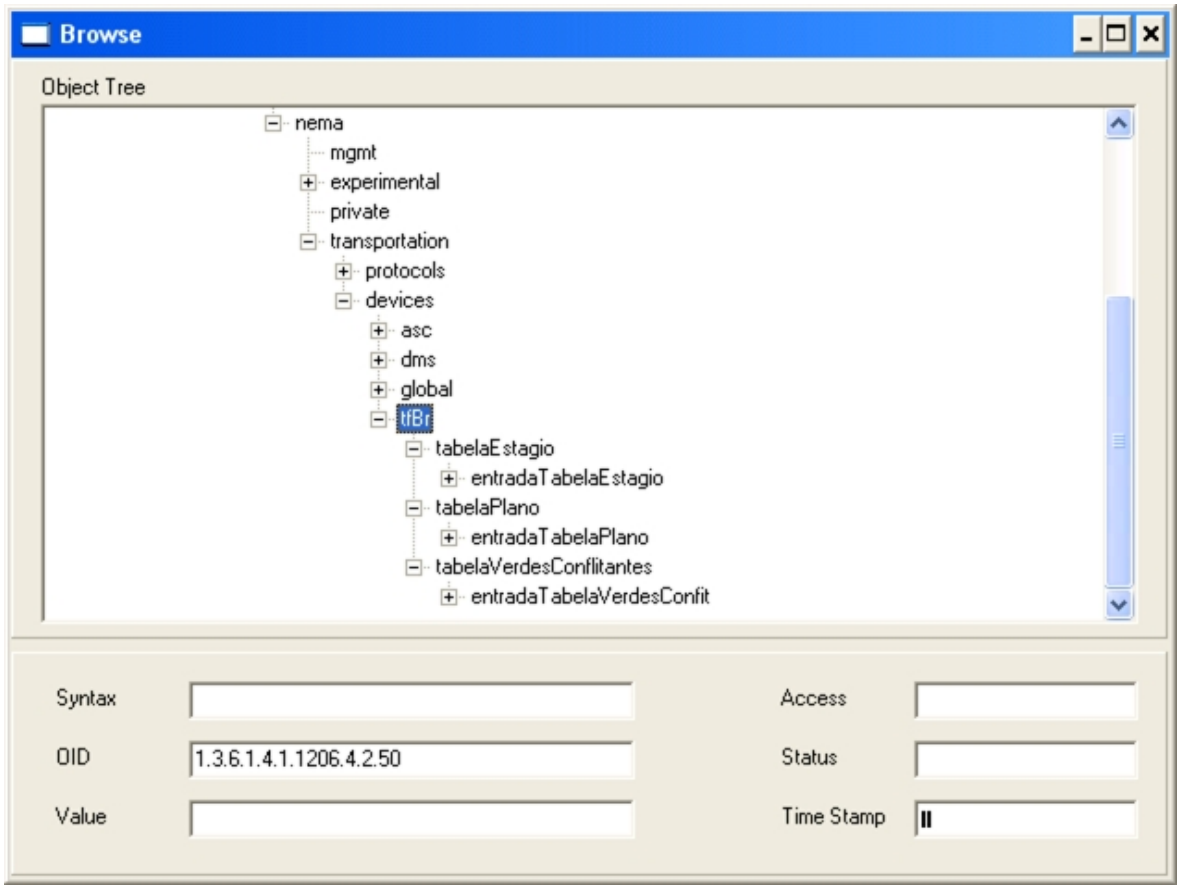

Figura 4.5: *MIB's nacionais compiladas no software NTCIP Exerciser.*

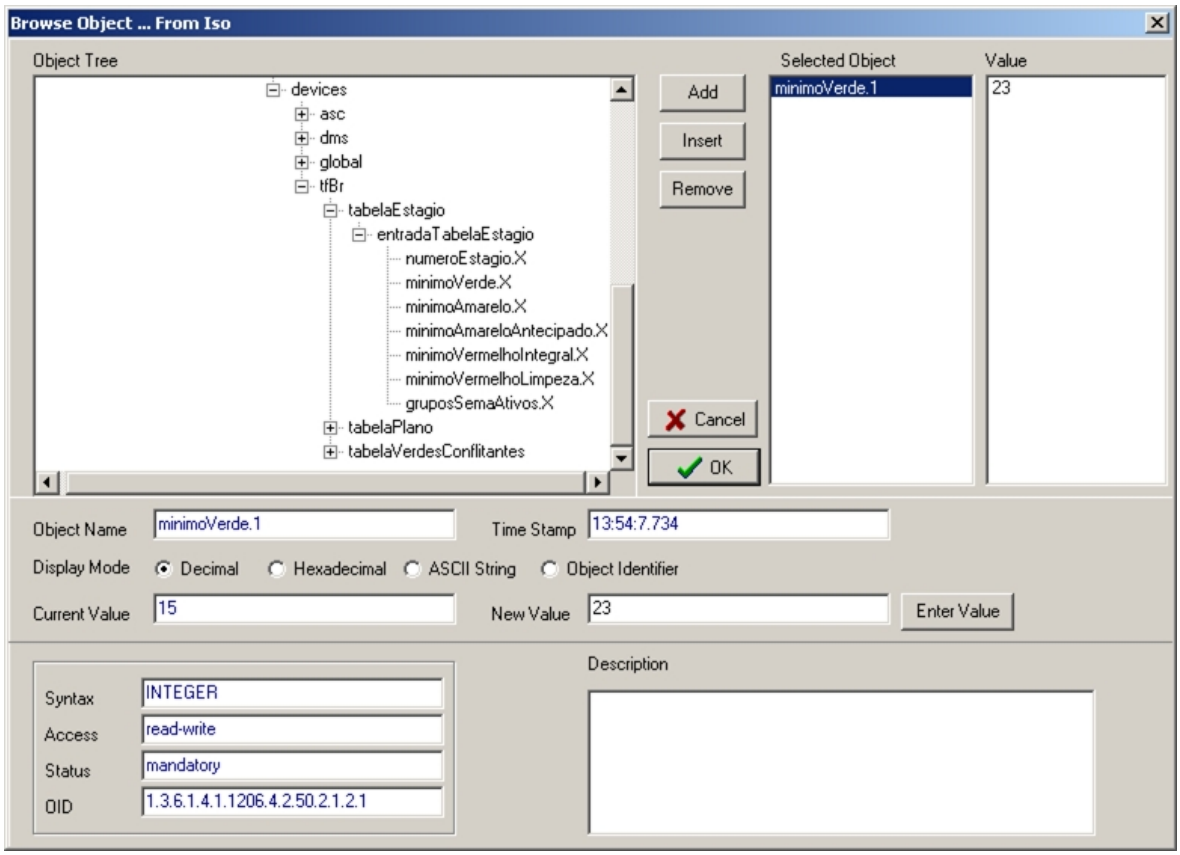

Figura 4.6: *Interface gerente, detalhes do objeto minimoVerde.*

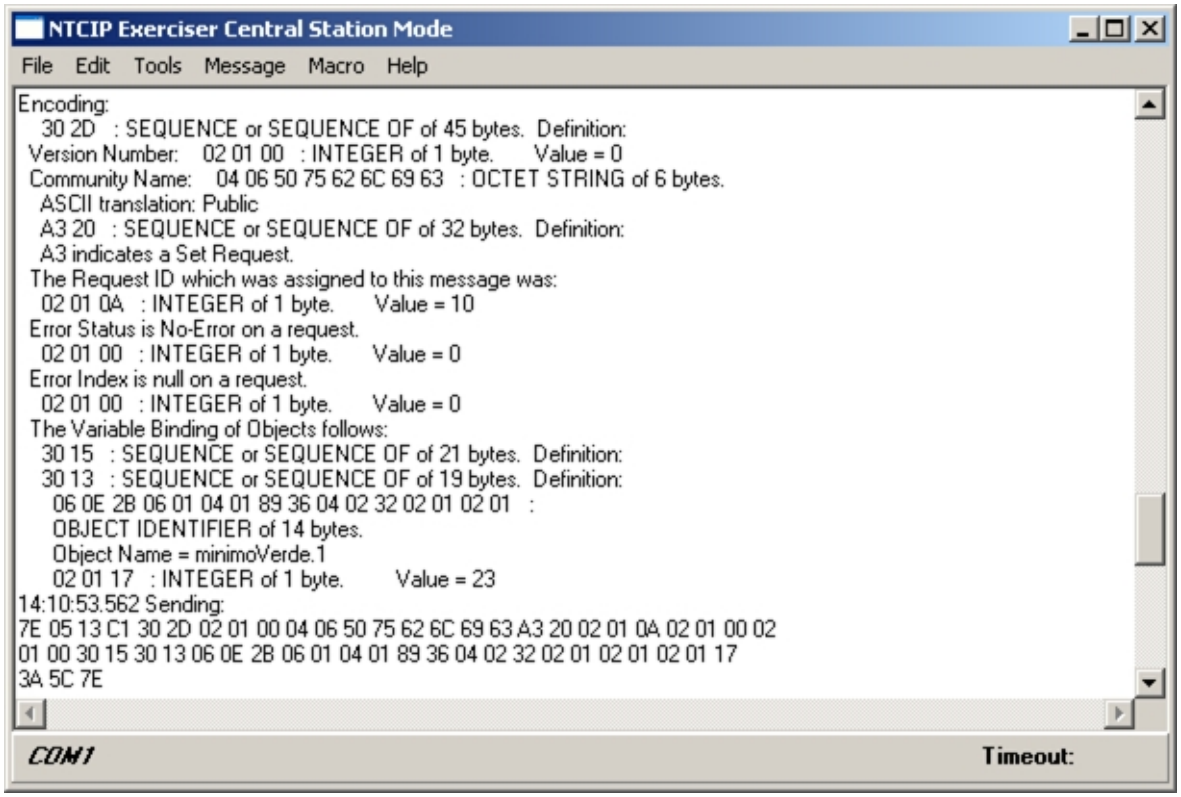

Figura 4.7: *Envio, pela Central, de mensagem para escrever valor 23 em minimoVerde.*

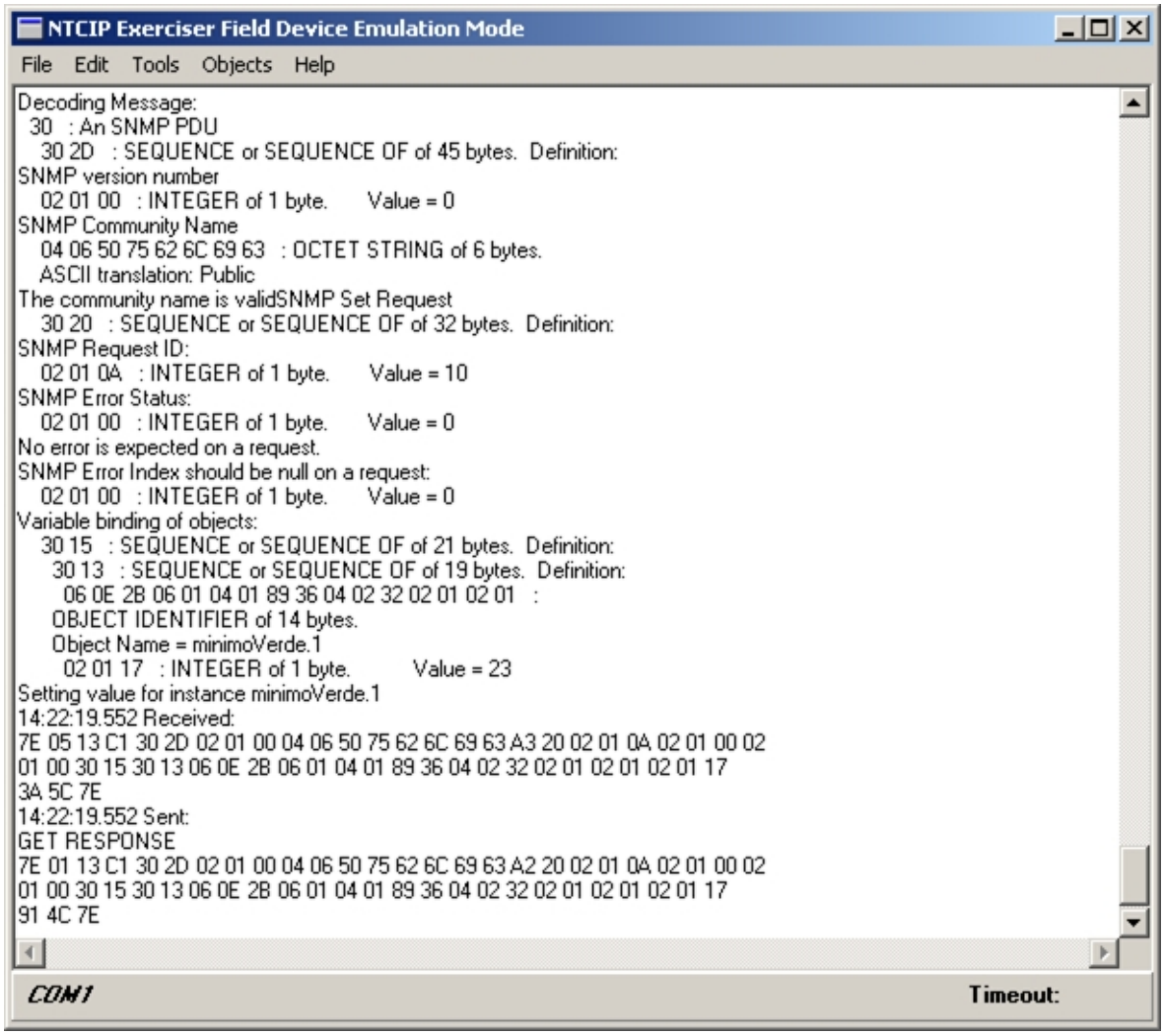

Figura 4.8: *Lado do agente, decodificando mensagem da central (gerente).*

4.2 Compatibilidade dos Protocolos de Aplicação

Apesar da facilidade de definição de MIBs, foram encontrados diversos obstáculos para a implementação do padrão NTCIP no Brasil. Para camada de aplicação é sugerido pela norma os protocolos SNMP, STMP e SFMP. Conforme [13], o SFMP ainda está em fase de desenvolvimento, assim não será incluído nesta análise. Segundo [11], o SNMP é o protocolo padrão recomendado para um dispositivo estar de acordo com a norma NTCIP, sendo a implementação de STMP opcional.

O uso do SNMP apresenta algumas vantagens, pois possui bibliotecas de código aberto, por exemplo o net-snmp, disponíveis para várias linguagens de programação. Como é bastante utilizado para a gerência de redes de computadores, o SNMP é também bastante popular, com grande número de conhecedores a nível de usuário, porém poucos desenvolvedores.

Porém, a idéia de utilizar-se unicamente o SNMP não é a melhor solução. Em sistemas com muitos dispositivos e grande tráfego de dados sobre o canal de comunicação, o protocolo tende a consumir muitos recursos da rede [4]. Como pode ser visto na seção 3.1.5, a dificuldade encontrada com o SNMP é que este apresenta uma mensagem bastante grande. O pacote SNMP apresenta vários bytes extras no cabeçalho e OID da mensagem, sendo este um empecilho para sua implementação, já que nos sistemas de transportes brasileiros o meio físico utilizado possui pouca largura de banda.

Conforme pode ser observado na seção 3.2, o pacote STMP tem tamanho reduzido em relação ao SNMP, amenizando assim a questão da largura de banda. Porém não existem bibliotecas STMP disponíveis, havendo trechos de código no *software Exerciser* mas não a implementação completa do protocolo [30].

A etapa de codificação/decodificação em STMP poderia ser facilmente implementada a partir da "disponibilidade"(existência) de PDUs SNMP, da seguinte forma: em vez do envio da PDU, o "transformador" STMP retiraria os campos "*Tag*" e "*Length*" do campo "*variableBindings*", pois a configuração antes realizada estabeleceu estes campos nos dois lados da comunicação (agente e gerente). Por outro lado, criar processos de registro das configurações não é tarefa tão simples e, infelizmente, não há bibliotecas disponíveis para tal.

Logo o uso do STMP seria uma solução para o tamanho elevado das mensagens SNMP, mas a falta de bibliotecas disponíveis dificulta bastante sua utilização, já que a implementação destas é uma tarefa bastante complexa.

4.3 Conclusão

Ao analisar a compatibilidade do padrão NTCIP com a realidade brasileira, inicialmente tivemos problemas com relação à forma de programação dos controladores proposta pelo padrão e a prática nacional. Isto, porém, pode ser facilmente resolvido através da criação de MIB's nacionais compatíveis com a realidade brasileira.

Outro problema encontrado foi em relação aos protocolos de aplicação. Quanto ao SNMP, recomendado pela norma, o elevado tamanho da mensagem se mostrou incompatível com a capacidade dos meios de comunicação utilizados na engenharia de tráfego brasileira. Já em relação às suas variações, SFMP e STMP, foram encontradas dificuldades devido ao SFMP ainda estar em fase de desenvolvimento e o STMP ser de implementação bastante complexa, não existindo bibliotecas disponíveis.

Assim, considerando as dificuldades observadas com relação aos protocolos de aplicação sugeridos pelo NTCIP, na próxima seção iremos estudar o XML (*Extensible Markup Language*) como uma possível forma de proporcionar a intercambiabilidade entre controladores semafóricos brasileiros.

Capítulo 5

XML e Suas Aplicações em Sistemas Inteligentes de Transporte

Conforme visto na seção 4.2, foram encontrados problemas tecnológicos na adaptação do NTCIP à realidade de engenharia de tráfego brasileira. Entre outras dificuldades, o SNMP apresenta um grande número de bytes por mensagem enviada; já o STMP não possui bibliotecas disponiveis, sendo de implementação bastante complexa.

Uma possível solução seria a utilização do XML (*Extensible Markup Language*) no desenvolvimento da proposta do padrão de comunicação. Esta escolha do XML se deve às seguintes características:

- É um padrão de fato, com ampla base de desenvolvedores, ferramentas e bibliotecas disponíveis;
- Disponibilidade de ferramentas de transformação que permitem diminuir o *overhead* de texto nos documentos XML;
- Devido à adequação de XML para a abordagem orientada a serviços (*Web Services*), existem propostas para XML em comunicação entre centrais. Assim, torna-se mais fácil automatizar trocas de mensagens entre centrais, e destas com equipamentos de campo, se o ambiente é homogêneo em termos de tecnologias utilizadas, facilitando dessa forma a interoperabilidade e a integração entre sistemas.

Assim, neste capítulo, serão apresentados um pouco do XML, algumas de suas características e exemplos de sua utilização em Sistemas Inteligentes de Transportes.

5.1 XML

O XML (*Extensible Markup Language*) é uma linguagem de marcação bastante popular, sendo definida e recomendada pela W3C (*World Wide Web Consortium*). É de crescente importância na troca de dados na *Web*. Linguagens de marcação descrevem a forma de um documento, indicando como ele deve ser interpretado, utilizam *tags*, elementos que marcam o documento, que indicam como deve ser feita a interpretação de seu conteúdo [19].

O XML é um subconjunto do SGML (*Standard Generalized Markup Language*) [41], uma linguagem de marcação bastante geral e com grande capacidade. Devido à grande generalidade e abrangência do SGML, ela se tornou uma linguagem de difícil aprendizado, já o XML tem a proposta de ser uma linguagem de marcação flexível como o SGML e de simples utilização [19]. Outra linguagem de marcação bastante popular baseada no SGML é o HTML (*Hypertext Markup Language*). O HTML, apesar de bastante simples, perde em flexibilidade, pois possui um grupo de *tags* pré-definidos, tornando-o uma linguagem não expansível.

Um documento XML poderia ser por exemplo [19]:

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <DOCUMENTO>
3        <SAUDACAO>
4              Alô do XML.
5        </SAUDACAO>
6        <MENSAGEM>
7              Bem-vindo ao mundo do XML.
8        </MENSAGEM>
9  </DOCUMENTO>
```

No exemplo acima, a linha de número 1 indica a versão do XML, no caso 1.0, e o tipo de condificação que está sendo utilizada (*UTF-8 character enconding*). Logo a seguir, na linha 2, é criada uma nova *tag* chamada `DOCUMENTO`; este é o elemento raiz e se estende até a linha 9 onde pode ser observada a *tag* de fechamento do elemento. Então na linha 3 é incluido o elemento `SAUDACAO`, o qual contém um texto (linha 4) e é fechado na linha 5. Seguindo nas linhas 6 a 8 tem-se o elemento `MENSAGEM` que também contém um texto. Assim foi definido o elemento raiz `DOCUMENTO` que contém dois elementos, `SAUDACAO` e `MENSAGEM`, ambos contendo textos.

Como o XML permite que o usuário crie suas próprias *tags* (no exemplo acima, os três elementos são definidos por *tags* novas) é necessario especificar como estas *tags* devem ser processadas. Isto pode ser feito de duas formas:

- através de um *style sheet*, que indica a um navegador *web* como o conteúdo do documento deve ser formatado;
- através de alguma linguagem de programação, por exemplo Java ou JavaScript, que carregue e processe o documento XML dentro de códigos de programação.

Devido a sua popularidade, existem diversos programas em Java e outras linguagens que processam documentos XML. No final desta seção será visto com mais detalhe o XML *style sheet*.

As principais vantagens do XML são [19]:

facilidade para troca de dados: atualmente, com a grande quantidade de formatos de dados proprietários, são necessários programas ou módulos de conversão para transferência de dados entre aplicações. Os dados e marcações XML são armazenados em modo texto e não apresentam codificação patenteada ou com licença, assim são mais acessíveis, podendo ser facilmente configurados,

linguagens de marcação padronizadas: quando há um acordo no uso de uma determinada linguagem de marcação, esta pode ser considerada um padrão, podendo também ser criados *browsers* e aplicações específicas para esta linguagem;

dados auto descritivos: documentos XML são auto descritivos; dados são apresentados em modo texto e o nome dos elementos dá uma boa indicação da sua funcionalidade;

dados estruturados e integrados: em XML, podem ser especificados não apenas dados, mas a estrutura dos dados e como os elementos integram-se entre si, o que é bastante útil para a manipulação de dados complexos.

A seguir serão mostradas mais características do XML, como DTD (*Documento Type Definition*), XML namespaces e XSLT (*Extensible Stylesheet Language Transformations*).

5.2 Verificação de sintaxe XML

Para fins de verificação da sintaxe de um documento XML, pode-se associá-lo a um DTD (*Document Type Definition*) ou a um XML Schema. A verificação de sintaxe utilizando um DTD é suficiente para os propósitos deste trabalho, assim utilizaremos DTD para obter a verificação dos documentos XML.

Um DTD especifica a estrutura e a sintaxe de um documento XML, assim, se um documento XML estiver de acordo com o DTD do domínio a que se refere, então diz-se que o documento é "válido". Documentos XML que foram testados com sucesso são chamados documentos válidos [19]. Abaixo, pode ser observado um DTD que especifica a estrutura do documento XML mostrado na seção anterior:

```
1  <!DOCTYPE DOCUMENTO [
2       <!ELEMENT DOCUMENTO (SAUDACAO, MENSAGEM)>
3       <!ELEMENT SAUDACAO (#PCDATA)>
4       <!ELEMENT MENSAGEM (#PCDATA)>
5  ]>
```

O formato básico de um DTD é `<!DOCTYPE nome_do_elemento_raiz [ ]>`. Dentro dos colchetes pode estar o próprio DTD, como no exemplo, ou uma referência para um arquivo em anexo. No exemplo acima, além do elemento raiz (`DOCUMENTO`) estão definidos os elementos `SAUDACAO` e `MENSAGEM`, ambos capazes de armazenar texto. Assim, conforme este DTD, o documento XML apresentado na seção anterior é valido. Abaixo é mostrado um outro exemplo de DTD com os elementos `lista_de_pessoas` (raiz), pessoa, nome e aniversário:

```
1  <!DOCTYPE lista_de_pessoas [
2       <!ELEMENT lista_de_pessoas (pessoa*)>
3       <!ELEMENT pessoa (nome, aniversario?)>
4       <!ELEMENT nome (#PCDATA)>
5       <!ELEMENT aniversario (#PCDATA)>
6  ]>
```

De acordo com a simbologia de linguagem regular mostrada na Tabela 5.1 e o exemplo acima, uma `lista_ de_pessoas` pode conter zero ou mais pessoas. Já uma pessoa tem

que obrigatoriamente possuir um nome e pode ter uma ou nenhuma data de aniversário preenchida. Tanto o elemento nome quanto aniversário devem conter texto. Assim, este DTD pode validar diferentes documentos XML que estejam de acordo com esta estrutura.

Tabela 5.1: *Tabela com Simbologia utilizada no DTD.*

Simbolo	Significado
a+	indica uma ou mais ocorrências de a
a*	zero ou mais ocorrências de a
a?	zero ou uma ocorrência de a
a, b	a seguido de b
a | b	a ou b

5.3 XML Namespace

Um documento XML pode conter elementos e atributos que são definidos e utilizados por diferentes softwares. Estes documentos devem ser contruídos de forma que evitem conflitos, utilizando para isso XML namespaces [43].

XML namespace é uma recomendação oficial da W3C que permite evitar conflitos entre *tags*. Pode ser utilizada para diferenciar *tags*, inclusive *tags* com o mesmo nome [19].

Namespaces são muito utilizados para indentificar *tags* especificas em aplicativos XML. Como por exemplo o SQLX que integra XML e SQL [10], e o XSLT (*Extensible Stylesheet Language Transformations*) [42] que será explicado na próxima seção.

Abaixo é mostrado um exemplo de um documento XML onde é definido um *namespace* [19].

```
< livro:biblioteca
    xmlns:livro=http://www.amazingterrificbooks.com/spec>
    <livro:livro>
        < livro:titulo>
              Terremotos para o almoço.
        </ livro:titulo>
    </ livro:livro>
</ livro:biblioteca>
```

Para definir um *namespace* é utilizado o atributo `xmlns:prefixo`; no caso do exemplo dado o prefixo é `livro`. No exemplo, o prefixo `livro` antecede o nome das *tags* do documento, o que é opcional. Um *namespace* pode também ser definido como *default*, não possuindo nenhum prefixo. Assim, assume-se que todas as *tags* do documento pertencem ao *namespace* definido, como no exemplo abaixo [19]:

```
< biblioteca
    xmlns: =http://www.amazingterrificbooks.com/spec>
    <livro>
        < titulo>
            Terremotos para o almoço.
        </titulo>
    </livro>
</biblioteca>
```

Um *namespace* XML é identificado através da referência a uma URI (*Uniform Resource Identifier*) [43]. Nos exemplos mostrados acima a URI indicada é:

```
"http://www.amazingterrificbooks.com/spec".
```

A especificação XML expande a idéia do URL (*Standard Uniform Resource Locator*), que é um padrão uniforme para localizar fontes de informação, para URI (*Uniform Resource Identifier*), padrão universal para identificar documentos [19].

URI é utilizado para encontrar documentos na Internet, focando a busca no documento e não sua localização atual. Idealmente, na teoria, um URI deveria encontrar o *mirror* mais próximo para um documento, ou ainda, buscar um documento que foi movido para outra localização [19]. Na prática, URIs ainda estão em desenvolvimento, a maioria dos *softwares* apenas suportam URLs.

5.4 XSLT

O XSLT (*Extensible Stylesheet Language Transformations*) é uma linguagem de transformação recomendada pela W3C desde 1999 [42] . Linguagens de transformação permitem manipulação e transformação de documentos de diversas formas. Esta transformação pode

ser realizada em um programa servidor, em um programa cliente ou ainda, separadamente em um programa específico [19].

Através do XSLT (*Extensible Stylesheet Language Transformations*) é possivel processar e transformar documentos XML válidos obtendo diversas possibilidades de resultado, como documentos XML em diferentes formatos, documentos HTML ou uma *string*.

Para realizar uma transformação XSLT é necessário um documento para ser transformado e um *style sheet* que deve especificar como a transfomação deve ser feita. Ambos os documentos devem ser documentos XML válidos. Uma das transformações XSLT mais comuns é a de documentos XML em documentos HTML, conforme exemplo mostrado abaixo [19]:

- Documento XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<PLANETAS>
       <PLANETA>
              <NOME> Mercurio </NOME>
       </ PLANETA >
       <PLANETA>
              <NOME> Marte</NOME>
       </ PLANETA >
</ PLANETAS >
```

- XSL Style Sheet:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl=http://www.w3.org/1999/XSL/Transform>
       <xsl:template match="PLANETAS">
              <HTML>
                     <xsl:apply-templates/>
              </HTML>
       </ xsl:template >
       <xsl:template match="PLANETA">
              <P>
```

```
                    <xsl: value-of select="NOME"/>
            </P>
        </ xsl:template >
</ xsl:stylesheet >
```

- Documento HTML:

```
<HTML>
        <P>Mercurio</P>
        <P>Marte</P>
</HTML>
```

No exemplo, pode ser observado o documento XML a ser transformado, o *Style Sheet* utilizado na transformação e por fim, o documento HTML resultante. As *tags* `PLANETAS` e `PLANETA` são substituidas por *tags* HTML e P respectivamente. Além disso, o conteúdo do elemento `NOME` no documento XML é adicionado entre os elementos `P` no documento HTML resultante.

O exemplo acima mostra uma transformação de um documento XML em HTML. Outras transformações podem ser feitas, como por exemplo, a de um documento XML em outro documento XML mas em diferente formato, ou ainda em um arquivo de *strings* apenas com os conteúdos dos elementos do documento original.

5.5 Aplicações de XML em ITS

A crescente adoção do XML como forma de representação e troca de dados estimulou o desenvolvimento de diversos aplicativos e bibliotecas XML de código aberto. Para a comunidade de transportes, a adoção do XML é considerada vantajosa, não apenas pela grande quantidade de ferramentas de código aberto disponíveis, mas também pela facilidade de comunicação interna e externa [25].

No padrão de comunicação NTCIP, o XML é sugerido como uma das formas de comunicação entre centrais, havendo no padrão dois documentos dedicados à utilização do XML para a comunicação C/C [14, 17]. Segundo [14], mesmo não havendo uma padronização da forma de utilização do XML, diversos estados americanos utilizam XML para comunicação

C/C. Nos documentos é encontrada uma análise dos benefícios de sua utilização e um estudo da combinação de diversas formas de codificar e transportar documentos XML para aplicações C/C [14, 17].

Centrais de Engenharia de Tráfego utilizam muitos programas diferentes. Estes não se comunicam entre si, pois têm diferentes formatos de armazenamento. O XML tem sido utilizado para a troca de dados que estão em diferentes formatos, existindo linguagens XML desenvolvidas especificamente para diversas áreas. Com esta justificativa é proposto o TMML (*Traffic Model Markup Language*) que é uma linguagem XML para armazenar e transportar dados de transportes [6]. Também existe a proposta do uso do XML como uma forma de representação de dados em simulação de tráfego, TrafficXML [25].

Os casos apresentados acima se referem aos EUA, onde há influência do padrão NTCIP. Contudo, foi encontrado também na Irlanda, onde o NTCIP não é utilizado, um estudo para a utilização de XML na comunicação entre agências [23]. No caso da Inglaterra, o SNMP é indicado para comunicação C/V, seguindo o padrão NTCIP, sendo o XML recomendado para comunicação entre centrais e sugerido que no futuro seja também utilizado para comunicação C/V [39].

No estado da Virginia (EUA), que segue o padrão NTCIP, o XML é utilizado como uma solução para a representação de MIBs para Sinais de Mensagens Dinâmicas [29]. Sua utilização é justificada devido ao formato de armazenamento das MIBs escritas em ASN.1 dificultar a implementação de *softwares* orientados a objeto. Uma solução para este problema, seria representação das MIBs no interior do código, como classes, mas devido ao grande número de MIBs já existentes e à possibilidade de criação de novas, haveria problema com códigos muito longos e pouco flexíveis. Dessa forma, o XML é indicado como uma solução para o problema de representação, manipulação e armazenamento de MIBs em Sinais de Mensagens compatíveis com o NTCIP.

O XML é uma linguagem bastante popular, logo existe grande número de profissionais com conhecimento no assunto e muitos aplicativos desenvolvidos na área. Existem diversas iniciativas para utilização do XML em centrais de controle de tráfego, assim o uso do XML também para comunicação C/V torna o sistema homogêneo facilitando a troca de informações. Na próxima seção será mostrada a proposta de metodologia para a padronização da comunicação de controladores semafóricos do Brasil. Como será visto, devido às vantagens e dificuldades já apresentadas, o XML e algumas tecnologias a ele relacionadas, como XML *namespace* e XSLT, serão de grande importância para a solução adotada.

Capítulo 6

Proposta e Implementação

Em seções anteriores foi estudado o padrão NTCIP visando utilizá-lo para a proposta de um padrão de comunicação para controladores semafóricos. Como visto no Capítulo 4, faltam ferramentas para desenvolvimento de soluções baseadas em STMP, o qual seria candidato a uso no Brasil dada a pouca largura de banda requerida pelas mensagens. Assim, nesta seção será feita uma proposta baseada em algumas características do NTCIP e na linguagem XML, visando adequar o arcabouço conceitual da arquitetura gerente-agente, extensível via MIBs, à disponibilidade de ferramentas e à ampla base de desenvolvedores associadas ao XML.

O XML é uma linguagem bastante popular, possuindo diversos aplicativos e bibliotecas disponíveis. Através do XSLT, é proposta uma solução para permitir a comunicação entre uma central de gerenciamento de tráfego e sistemas legados.

Inicialmente será apresentada a proposta para comunicação de controladores semafóricos. Em seguida, será feita uma descrição e avaliação da implementação de parte desta proposta.

6.1 Proposta

Como visto anteriormente, apesar de ser uma solução bastante interessante, há dificuldades em adequar a tecnologia do NTCIP à realidade brasileira. Assim, o protocolo aqui proposto se baseia na utilização de algumas estruturas sugeridas pelo padrão NTCIP, porém com algumas alterações, mais especificamente utilizando-se a linguagem XML.

Devido às características apresentadas no Capítulo 5, o XML mostrou-se como uma tecnologia bastante adequada para compor a solução do problema. Sendo ainda uma linguagem bastante atual, com diversas bibliotecas e ferramentas de código aberto disponíveis.

A solução proposta neste trabalho mantém a filosofia de `set` e `get` proposta pelo padrão NTCIP, assim como a sua extensibilidade. Dessa forma, novos dados a serem gerenciados podem ser facilmente incluídos no sistema. A central de controle deve possuir um compilador de objetos gerenciáveis. Este é bastante similar ao compilador de MIBs do SNMP, porem a descrição dos objetos é feita utilizando-se a linguagem XML.

Existindo também, na central, uma interface com o usuário e um codificador para XML. O codificador para XML recebe os dados do usuário e gera um documento XML padrão que representa a mensagem do protocolo.

Para garantir a possibilidade da central se comunicar com controladores semafóricos legados, de diferentes fabricantes, existe a possibilidade de aplicação de transformações XSLT na mensagem do protocolo. Na transformação são aplicadas regras, especificas de cada fabricante, que geram como saída uma mensagem compatível com o controlador.

Na Figura 6.1 pode ser observado o diagrama de atividades que representa a solução proposta. Inicialmente a central aguarda que o operador inicie alguma atividade, esta pode ser: compilar novos objetos gerenciáveis, instalar regras de transformação XSLT ou configurar algum controlador semafórico do sistema. Ainda na Figura 6.1, cada uma destas atividades é detalhada em um diagrama de atividades especifico. A descrição de cada estrutura que compõe a solução é apresentada a seguir.

Descrição dos Objetos Gerenciáveis (1): Descreve os objetos gerenciáveis, contendo informações como tipo do objeto, descrição e acesso permitido. É bastante similar ao conceito de MIB do SNMP, sendo mantido na proposta como forma de garantir a extensibilidade do padrão. A descrição dos objetos difere das MIBs do SNMP com relação a forma de representação, no SNMP é utilizado o ASN.1 e aqui propomos a utilização do XML. Por exemplo, abaixo é mostrado um trecho da descrição de uma entrada da tabela de estágio, contendo o objeto numeroEstagio.

```
    ...
<mib oid="1.2.1.1"
system="1.0"
```

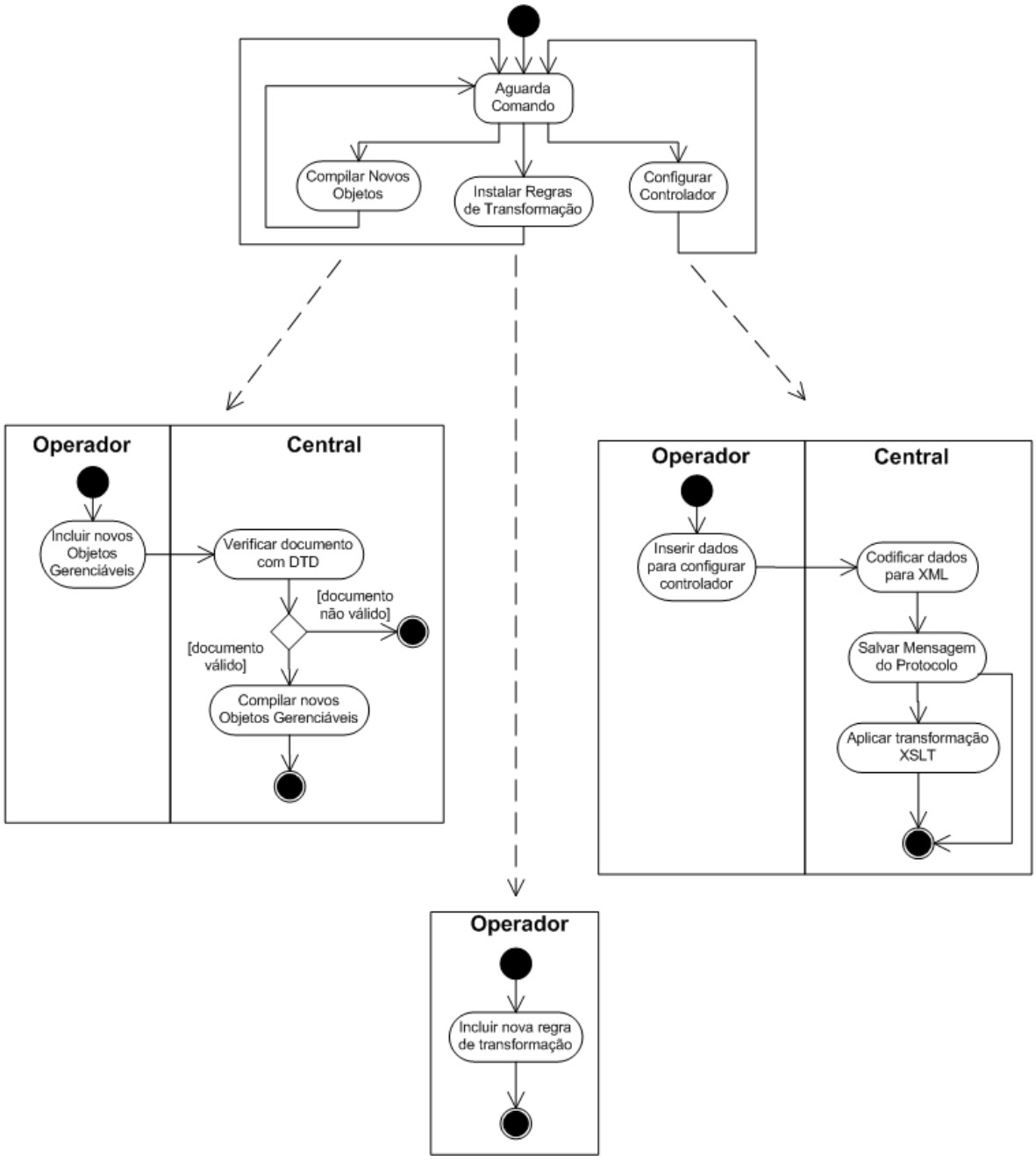

Figura 6.1: *Diagrama de atividades representando solução proposta.*

```
type="numeroEstagio"
syntax="INTEGER"
access="read-only"
status="mandatory">
       <description>
              Este valor representa o numero do estagio.
       </description>
</mib>
...
```

DTD (2): Documento utilizado para verificar se a descrição dos objetos (item 1) é um documento válido. Garantindo também assim, a validade da mensagem do protocolo (item 6) gerada. Abaixo é mostrado, como exemplo, um trecho do DTD que valida o elemento numeroEstagio mostrado acima:

```
...
<!ELEMENT mib (description?) EMPTY>
<!ATTLIST mib
oid CDATA #REQUERIDO
system CDATA #REQUERIDO
type CDATA #REQUERIDO
syntax CDATA #REQUERIDO
size CDATA #OPCIONAL
access CDATA #REQUERIDO
status CDATA #REQUERIDO
>
...
```

Compilador de Objetos Gerenciáveis (3): Lê o arquivo de descrição do objeto e retira as informações necessárias, como tipo de acesso permitido, por exemplo. A possibilidade de compilação de novos objetos gerenciáveis garante a extensibilidade do protocolo. Um fabricante pode criar um documento que descreva novos objetos a serem gerenciados, então o compilador lê e interpreta este documento, passando ao sistema as informações necessárias para a inclusão e monitoramento destes objetos.

Dados do Usuário (4): O usuário informa o endereço do controlador semafórico, qual objeto deseja monitorar e se deseja realizar uma operação de escrita ou leitura. No caso

de uma operação de escrita, deve informar também o valor dos dados a serem enviados ao controlador.

Codificador para XML (5): Já em posse das informações referentes aos objetos gerenciados, obtidas pelo compilador de objetos, obtém as informações preenchidas pelo usuário. Produz então, como saída, um documento XML contendo a mensagem do protocolo.

Mensagem do Protocolo (6): É um documento XML que representa a mensagem do protocolo. Este documento é gerado pelo Codificador para XML e indica os objetos a serem gerenciados, se a operação realizada é de leitura ou escrita e, no caso de uma operação de escrita, contém também o valor dos objetos. Um exemplo da mensagem do protocolo poderia ser:

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<documento>
     <gravar>
          <tabelaDeHora>
               <entradaDaTabelaDeHora>
                    <dia>05</dia>
                    <hora>00</hora>
                    <minuto>00</minuto>
                    <planoProgr>2</planoProgr>
               </entradaDaTabelaDeHora>
          </tabelaDeHora>
     </gravar>
</documento>
```

A definição de uma mensagem padrão para o protocolo é uma tarefa bastante complexa, devendo ser amplamente analisada e discutida. Assim, a mensagem apresentada acima é apenas um modelo utilizado para a implementação e testes mostrados a seguir.

Regras de Transformação (7): Através do XSLT é possível realizar transformações em um documento XML. A Mensagem do Protocolo é transformada seguindo determinadas regras de transformação, gerando assim uma saída no formato desejado. O uso do XSLT proporciona diversas vantagens ao protocolo, documentos com diferentes regras XSLT produzem diferentes saídas.

A possibilidade de fazer transformações na mensagem do protocolo possibilita que equipamentos em sistemas legados possam se comunicar com a central de controle. Para isso, cada fabricante deve fornecer as regras de transformação, a serem aplicadas no documento padrão, a fim de produzir uma saída que possa ser compreendida por seus equipamentos.

Assim a proposta permite a comunicação da central com sistemas legados, recomendando que inicialmente os fabricantes forneçam regras de transformação compatíveis com seus equipamentos, e que no futuro desenvolvam equipamentos compatíveis com a *string* padrão do protocolo.

Foram mantidos, no protocolo, alguns conceitos do padrão norte-americano NTCIP, como por exemplo, a filosofia do `get-set` para leitura e escrita de dados. A extensibilidade do protocolo foi garantida baseando-se no compilador de MIBs utilizado pelo protocolo SNMP, possibilitando que novos dados sejam facilmente adicionados ao sistema.

Para facilitar a compreensão do proposta, é apresentado o exemplo abaixo:

Exemplo: Seja o caso de um fabricante que fornece um novo equipamento a uma central de controle já em operação. As ações que devem ser realizadas, para possibilitar a inclusão do novo controlador ao sistema, são apresentadas na Figura 6.2.

O fabricante deve fornecer a descrição dos objetos gerenciáveis proprietários, caso existam, as regras de transformação XSLT, para gerar uma saída compatível com seu controlador, e bibliotecas com código de retorno. O formato das bibliotecas para retorno da mensagem não é definido neste trabalho, ficando como sugestão para trabalhos futuros.

As ações mostradas na Figura 6.2 são realizadas somente quanto um equipamento não é compatível com o protocolo padrão e necessita ser conectado na rede. Na prática, tal ação é realizada umas poucas vezes por ano. O Diagrama de seqüência, para esta ação, é representado pela Figura 6.3.

O operador deve cadastrar na central os novos objetos gerenciáveis, a central então, utilizando um DTD, verifica a validade do documento recebido e compila os novos objetos incluindo-os no sistema de gerência. A central então confirma, ao operador, a correta compilação dos objetos. Em seguida, o operador deve instalar a transformação XSLT e bibliotecas de retorno, recebendo em ambos os casos uma confirmação da central.

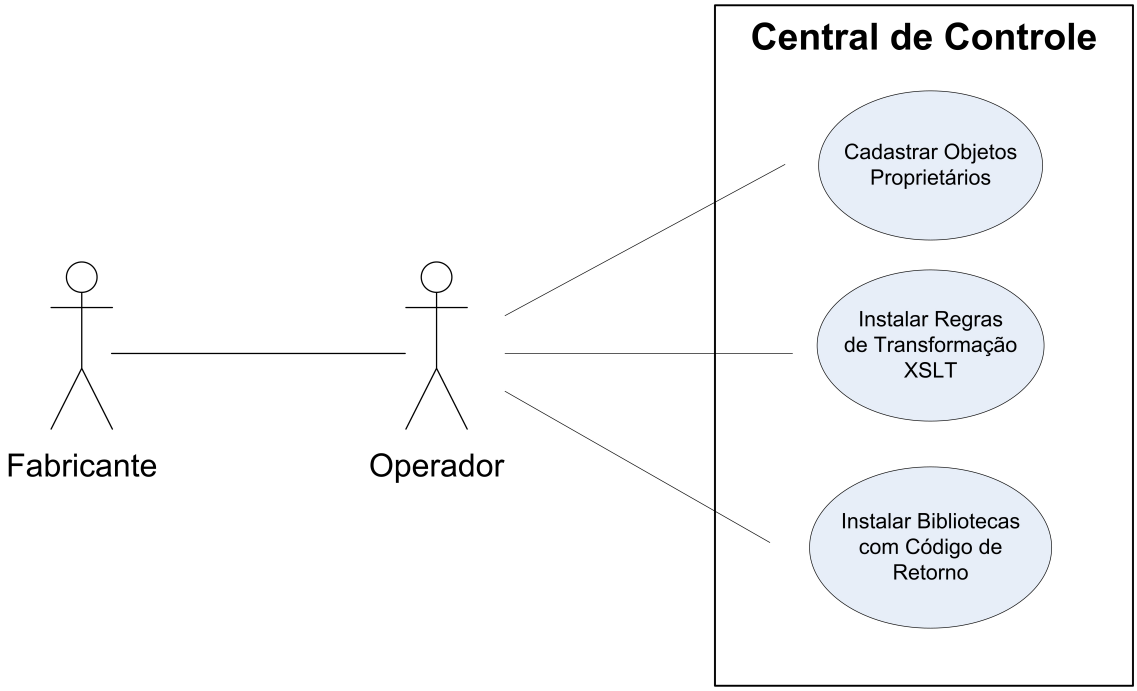

Figura 6.2: *Inclusão de novo controlador no sistema.*

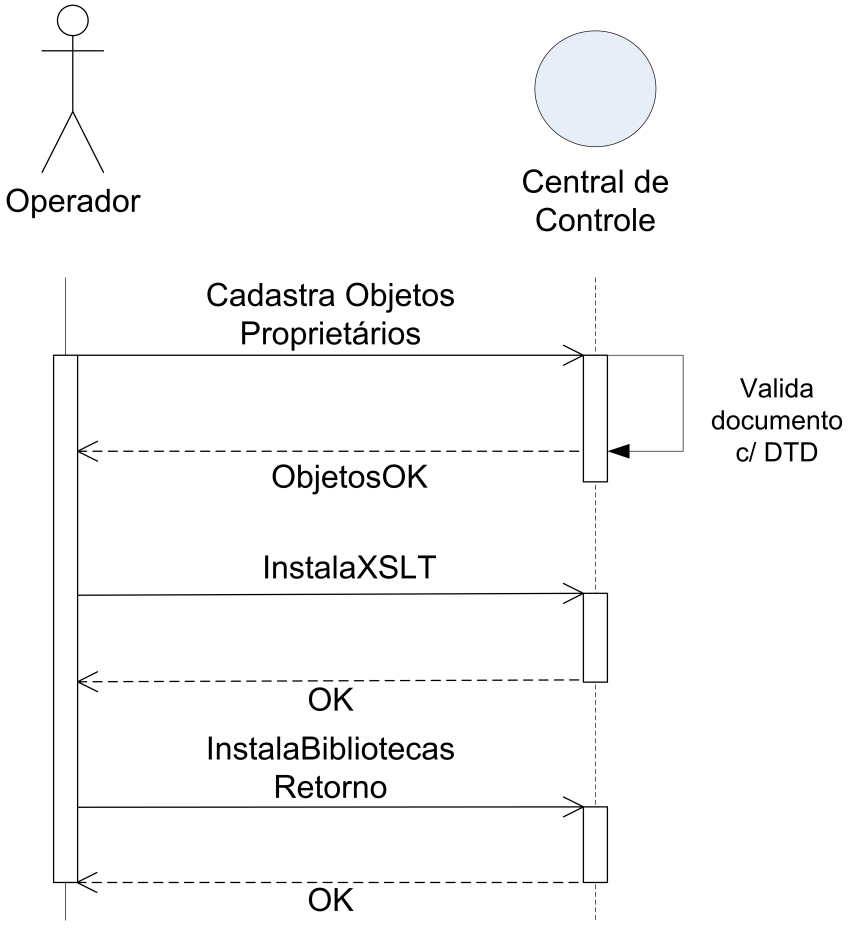

Figura 6.3: *Diagrama de seqüência - Inclusão de novo controlador no sistema.*

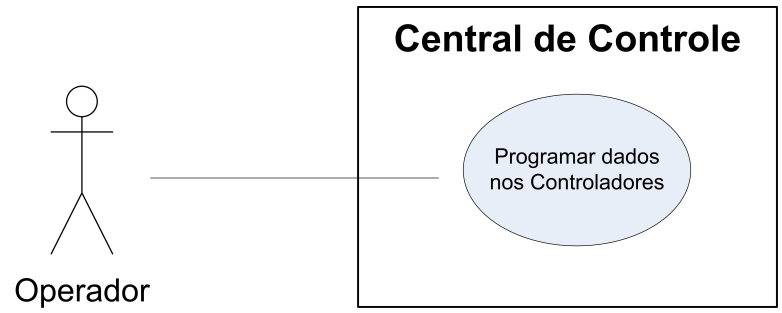

Figura 6.4: *Programação Controlador.*

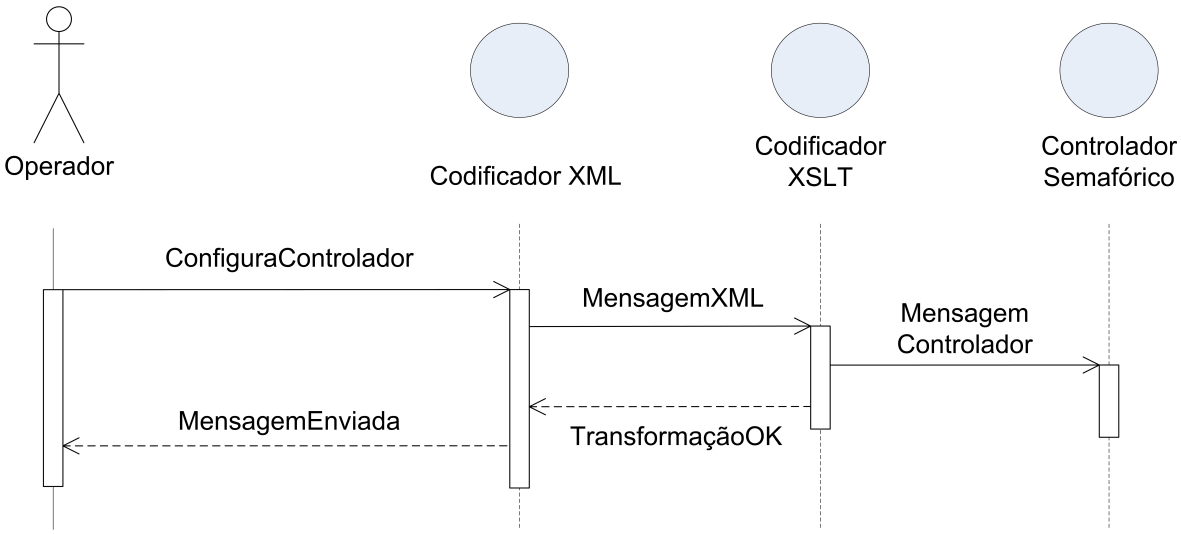

Figura 6.5: *Diagrama de seqüência - Programação Controlador.*

Já compilados os objetos gerenciáveis e instaladas as regras de transformação, específicas do controlador, é necessário a configuração de seus planos, estágios e outras tabelas, como data e hora. Para isto, é preciso uma ação do operador do sistema para programar dados no controlador, esta é representada na Figura 6.4.

A ação da Figura 6.4 ocorre em intervalos curtos durante a instalação ou re-configuração de controladores semafóricos ou, em condições usuais de operação do sistema viário, em intervalos de vários meses. O diagrama de seqüência para a configuração do controlador é representado na Figura 6.5.

O operador informa para central os dados que deseja programar no controlador. O codificador XML recebe então estes dados e gera um documento XML, contendo a mensagem padrão do protocolo com os dados recebidos. O documento XML é então enviado ao codificador XSLT que, utilizando as regras de transformação especificas para o controlador em questão, transforma o documento gerando uma saída compatível com o controlador.

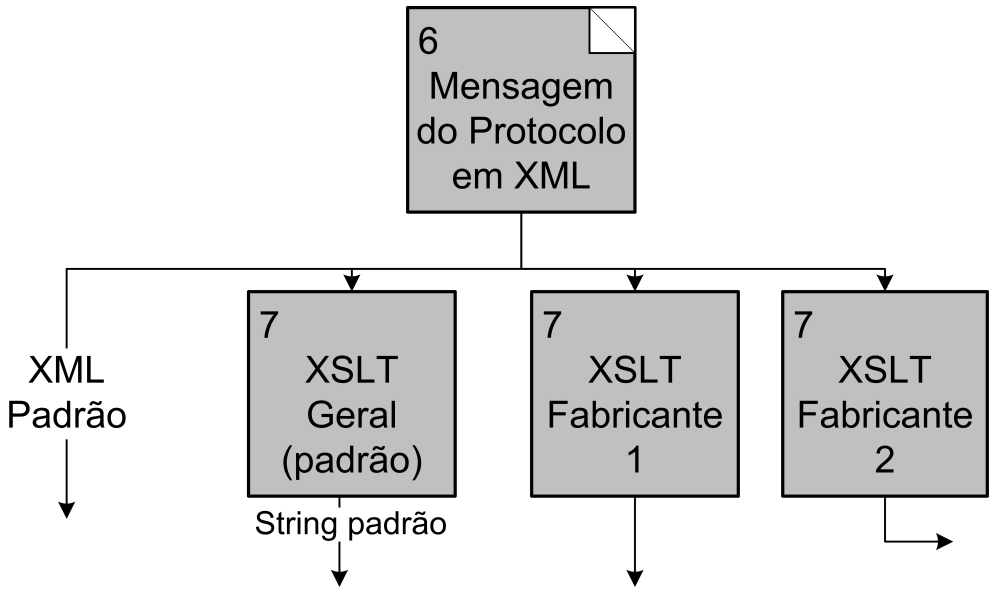

Figura 6.6: *Possíveis transformações XSLT disponíveis em central de controle.*

Finalizando, esta saída é enviada ao controlador gerando uma válida programação deste.

Como se observa, na Figura 6.5, não há retorno do controlador para o codificador XSLT. As respostas do controlador são enviadas para outro processo na central que trata do retorno das mensagens. Como dito anteriormente, o processo de retorno das mensagens não foi implementado neste trabalho, ficando como sugestão para trabalhos futuros.

Em uma implementação , várias transformações XSLT podem estar disponíveis em uma mesma central, conforme representado na Figura 6.6.

Na Figura 6.6, coexistem:

- XML padrão - para comunicação com equipamentos com grande capacidade de comunicação;
- XSLT geral - que produz mensagens compactadas, codificadas de forma padronizadas. A definição do XSLT geral deve ser analisada e discutida pela comunidade de transportes, ficando como sugestão para trabalhos futuros;
- XSLTs particulares de cada fabricante.

Na seção seguinte será apresentada a implementação de parte desta proposta. Nesta, será testada a possibilidade de comunicação entre uma central e um controlador semafórico legado, utilizando um documento XML e regras de transformação XSLT.

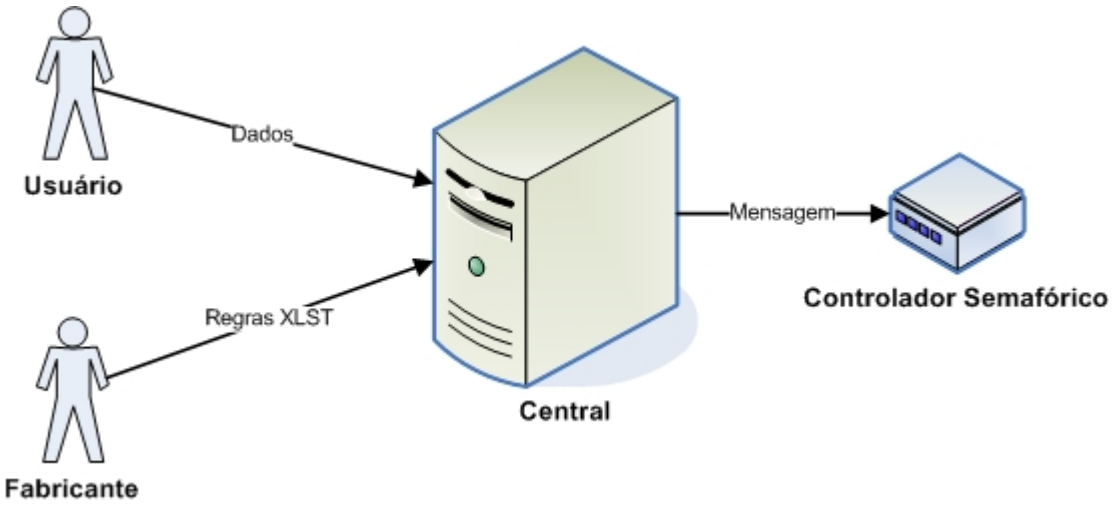

Figura 6.7: *Exemplo implementação.*

6.2 Implementação

Baseando-se na proposta apresentada na seção anterior, nesta seção será apresentada a implementação de parte desta. Objetiva-se a comunicação de uma central, que utiliza mensagens em formato XML, com um sistema legado de controle de tráfego. Para possibilitar esta comunicação será necessário o uso de regras XSLT que transformem o documento XML da central em uma saída em um formato que possa ser compreendido pelo controlador.

Não foram incluídas na implementação a compilação de novos objetos gerenciáveis e o retorno das mensagens, deixando para um estudo futuro o desenvolvimento destes processos. A Figura 6.7 representa o sistema implementado.

Na Figura 6.7 pode ser observado que a central recebe informações do usuário e regras XSLT do fabricante, gerando então, um documento XML com os dados recebidos do usuário. A este, são aplicadas as regras XSLT específicas do fabricante, transformando o documento em um comando válido para o controlador avaliado.

A implementação foi feita utilizando a linguagem Java. A utilização desta se justifica não apenas pelo fato de Java ser uma linguagem orientada a objeto multiplataforma, mas também por possuir diversas APIs para XML, sendo inclusive a linguagem mais usual para a implementação de transformações XSLT [19]. A seguir será apresentada, com mais detalhes, cada etapa desta implementação.

Dados do Usuário: Ao usuário é apresentada uma interface gráfica que possibilita a escrita de entradas de planos semafóricos, de estágios de um plano e da tabela de hora. A

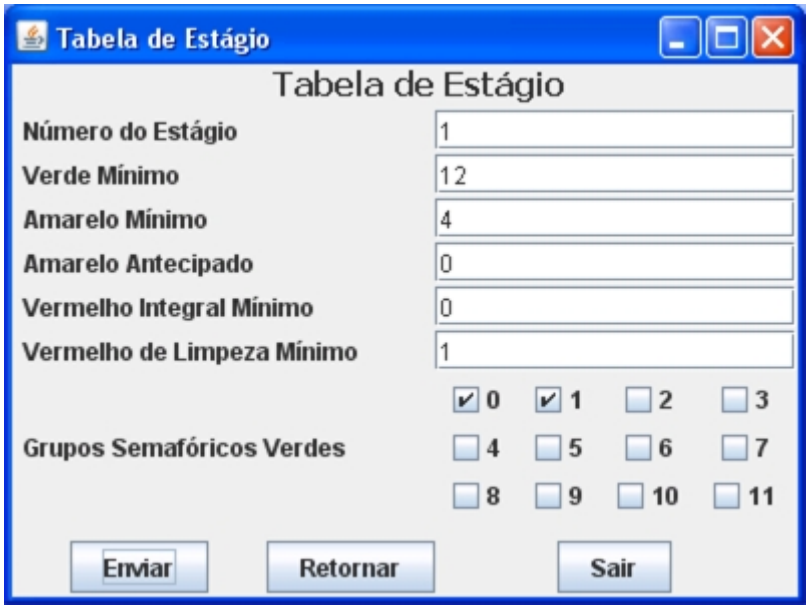

Figura 6.8: *Interface Tabela de Estágio.*

escolha destes dados foi baseada na proposta de MIBs nacionais da seção 4.1.3. Como pode ser observado na Figura 6.8, a interface da tabela de estágio apresentada ao usuário é bastante similar à MIB para tabela de estágio sugerida na seção 4.1.3. O mesmo acontece com as tabelas de plano e hora.

Os valores preenchidos na Figura 6.8 são baseados nas correntes de tráfego mostradas na Figura 4.1, assim como nos dados de programação para estas correntes observados nas tabelas da seção 4.1.1. Além de informar qual tabela deseja alterar e o valor dos dados a serem escritos, o usuário, através de uma interface gráfica, informa os endereços da central e do controlador semafórico.

Codificador para XML: É de responsabilidade do codificador para XML obter os dados informados pelo usuário e construir um documento XML representando a mensagem do protocolo. Este documento deve informar o tipo do comando a ser realizado (leitura ou escrita), o objeto e os valores informados pelo usuário.

A criação dos documentos XML foi realizada utilizando-se bibliotecas baseadas no XML DOM (Document Object Model). Este é um padrão da W3C para a manipulação de documentos XML [19].

Mensagem do Protocolo em XML: Como já foi visto nesta seção, a mensagem do protocolo é gerada utilizando-se o DOM. Abaixo é mostrado o documento gerado, para os dados inseridos na interface da Figura 6.8.

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<documento>
     <gravar>
          <tabelaDeEstagios>
               <entradaDaTabelaDeEstagio>
                    <numeroDoEstagio>1</numeroDoEstagio>
                    <verdeMinimo>12</verdeMinimo>
                    <amareloMinimo>4</amareloMinimo>
                    <amareloAntecipadoMinimo>0</amareloAntecipadoMinimo>
                    <vermelhoIntegralMinimo>0</vermelhoIntegralMinimo>
                    <vermelhoDeLimpezaMinimo>1</vermelhoDeLimpezaMinimo>
                    <movimentosPermitidos>
                         <movimento>0</movimento>
                         <movimento>1</movimento>
                    </movimentosPermitidos>
               </entradaDaTabelaDeEstagio>
          </tabelaDeEstagios>
     </gravar>
</documento>
```

Este documento corresponde à operação de gravar uma entrada da tabela de estágios e contém os dados informados pelo usuário. Documentos similares podem ser gerados para a tabela de plano e hora.

XSLT Fabricante: Com o intuito de permitir a comunicação da central com um controlador semafórico legado, devem ser aplicadas regras de transformação a mensagem do protocolo. Para possibilitar a correta transformação, primeiramente devemos compreender o formato usual das mensagens recebidas pelo controlador. Abaixo é mostrado o formato da mensagem esperada pelo controlador.

```
1o Byte = 5e h
2o Byte = número de Bytes contados a partir do 2o Byte
3o Byte = endereço de RX
4o Byte = endereço de TX
5o Byte = comando a ser executado
```

```
6o Byte = dado
     "
     "
N  Byte = CRC(montado com XOR dos bytes a partir do 2°Byte até n-1 Byte)
```

Considerando o modelo de referência ISO/OSI, concluímos que estamos tratando de um protocolo da camada de aplicação. Logo, neste nível, informações como endereço e CRC não devem ser tratados, sendo de responsabilidade de uma camada inferior. Assim, neste ponto, estaremos preocupados apenas com os dados e comandos a serem enviados.

Como pode ser visto no formato do protocolo mostrado acima, o comando deve ser representado no 5° byte seguido dos dados a serem transmitidos. O comando a ser executado é definido pelo fabricante conforme alguns exemplos abaixo:

```
LE_PLANO           43h Lê plano do escravo ou do mestre
GRAVA_PLANO        03h Grava plano no mestre o no escravo
GRAVA_TABHORA      57h Grava tabela de hora no controlador
LE_VRDCNF          15h Lê tabela de Verdes Conflitantes
GRAVA_VRDCNF       27h Grava tabela de Verdes Conflitantes
LE_ESTAG           5bh Lê tabela de Estágios
GRAVA_ESTAG        5ch Grava tabela de Estágios
LE_DATAS           45h Lê tabela de Datas
GRAVA_DATA         47h Grava tabela de Datas
LE_TABHORA         5eh Lê tabela de Horas
```

Já com relação aos dados a serem enviados, existem alguns detalhes na implementação do controlador que fazem com que algumas alterações e ajustes sejam necessários. Como exemplo, abaixo é mostrado o formato esperado da saída resultante das transformações XSLT aplicadas à mensagem do protocolo mostrada acima.

```
5° byte = 5a;
6° byte = numeroEstagio;
7° byte = 00;
8° byte = amareloMinimo;
9° byte = verdeMinimo;
```

```
10° byte = vermelhoIntegralMinimo;
11° byte = amareloAntecipadoMinimo;
12° byte = vermelhoDeLimpezaMinimo;
13° byte ao 24° byte = considerar posições de 0 a 11.  Preencher com 1 as
posições indicadas em movimentosPermitidos e com 0 o restante.
    Ex.: movimentosPermitidos = 1, 2 -> preencher com "1" 14° e 15° bytes,
     o restante preencher com "0".
```

Saída: O resultado das transformções XSLT, a aplicadas à mensagem do protocolo, deve ser do seguinte formato:

```
5e01000c040000010100010000000000000000000
```

O *byte* inicial 5a representa a operação de escrita da tabela de estágio. Já os bytes seguintes representam os dados referentes à tabela de estágio organizados de forma que possam ser lidos pelo controlador.

Antes de ser enviado ao controlador, devem ser anexados à mensagem dados como endereço do controlador, tamanho da mensagem e CRC. Após estes dados serem anexados à mensagem, esta é enviada através da porta serial ao controlador. Abaixo é mostrada a mensagem pronta a ser enviada ao controlador semafórico.

```
2e1a77775e01000c040000010100010000000000000000000
```

6.3 Resultados

Devido às dificuldades encontradas para adaptação do padrão NTCIP à realidade nacional, foi definida uma proposta que utiliza alguns conceitos do padrão NTCIP em conjunto com a linguagem XML. Assim foi desenvolvida uma proposta de um padrão que mantém a filosofia de escrita e leitura de dados do SNMP e a possibilidade de fácil inclusão de novos objetos gerenciáveis no sistema.

É gerado um documento em XML, mensagem do protocolo, que indica se a mensagem é de leitura ou escrita, o objeto gerenciado e, no caso de escrita, o valor dos dados. Este documento pode ser enviado diretamente a um equipamento de campo, compatível com a mensagem padrão, ou sofrer algumas alterações. Estas transformações visam adaptar a

mensagem, tornando a saída compatível com controladores semafóricos já em operação, ou minimizar o tamanho da mensagem, utilizando uma transformação XSLT padrão.

O tamanho final da mensagem enviada, ao equipamento de campo, varia de acordo com a transformação aplicada a mensagem do protocolo. Podendo assim, ser definida uma regra de transformação padrão que gere uma saída com tamanho reduzido, próxima as mensagens do STMP e SFMP. Na Tabela 6.1 é feita uma comparação do tamanho de uma mensagem de leitura, do objeto globalTime (definido pelo NTCIP), utilizando os protocolos de aplicação sugerido pelo NTCIP (SNMP, STMP, SFMP) e a proposta deste trabalho, utilizando XML.

Padrão	Tamanho da Mensagem
SNMP	43 bytes
STMP	1 byte
SFMP	9 bytes
Proposta com XML	Tamanho variado *
* depende da transformação XSLT aplicada a mensagem	

Tabela 6.1: *Comparação do tamanho de mensagens de leitura do objeto GlobalTime.*

Utilizando as linguagens XML, XSLT, Java, e ainda, bibliotecas Java que implementam DOM e XSLT, foi desenvolvida uma implementação para validar parte da proposta. Esta implementação tem como intuito comprovar a possibilidade de uma central, que produza a mensagem do protocolo em XML, se comunicar com um controlador semafórico legado. Esta comunicação é possível através da aplicação, ao documento XML, de regras de transformação XSLT específicas.

Através da implementação foi verificado que a comunicação entre a central, desenvolvida para os testes, e o controlador semafórico legado funcionou corretamente. Podendo ser observado que a saída gerada pelas transformações XSLT aplicadas à mensagem em XML foi compreendida pelo controlador, gerando assim uma valida programação deste.

No próximo capitulo serão apresentadas as conclusões deste trabalho e ainda sugestões para futuros estudos dando continuidade ao trabalho realizado.

Capítulo 7

Conclusão e Trabalhos Futuros

Para atingir o objetivo deste trabalho foi pesquisado o padrão de comunicação NTCIP, avaliando sua compatibilidade com a prática de engenharia de tráfego nacional. Foram encontradas algumas dificuldades na adoção do padrão norte-americano, entre elas algumas diferenças de nomenclatura e na forma de programação semafórica. Para solucionar este problema, foram desenvolvidos novos conjuntos de objetos gerenciáveis compatíveis com a prática nacional.

Estes objetos foram desenvolvidos de acordo com as especificações da norma NTCIP, assim os novos objetos puderam ser compilados pelo software NTCIP *Exerciser*. Foi possível simular, utilizando o *Exerciser*, o processo de escrita e leitura destes novos dados entre uma central e um equipamento de campo.

Apesar do sucesso na definição de novos objetos gerenciáveis, a proposta do NTCIP continuou não se mostrando adequada, principalmente pela falta de bibliotecas STMP disponíveis para desenvolvimento de programas para controle. Assim, houve a necessidade da busca de uma nova tecnologia para a solução do problema.

Devido à grande quantidade de bibliotecas disponíveis e da existência de sistemas que utilizam XML para a comunicação entre centrais, o que facilita a integração entre sistemas [25], o XML se mostrou adequado para a solução do problema. Assim, baseando-se em tecnologias indicadas na norma NTCIP e na linguagem XML, foi desenvolvida uma proposta de um padrão extensível e que permite a fácil integração de equipamentos de campo legados ao sistema.

Parte da proposta apresentada pôde ser implementada e avaliada. Mais precisamente,

foi implementado um protótipo de central de controle que produz um documento XML padrão e permite que transformações sejam aplicadas a este documento, gerando uma saída compatível com um controlador semafórico legado.

7.1 Conclusão

Devido ao pouco suporte de bibliotecas para desenvolvimento de soluções para ITS, foi concluído que o padrão NTCIP não é a melhor solução para a definição de um padrão brasileiro para a comunicação semafórica. Entretanto, no caso de se optar por este padrão, objetos gerenciáveis nacionais foram criados e integrados ao sistema graças ao mecanismo de compilação de MIBs do SNMP. Estes objetos foram desenvolvidos visando uma melhor adequação do padrão NTCIP à prática de engenharia de tráfego nacional.

A proposta de comunicação desenvolvida se mostra bastante interessante. Mantém a propriedade de extensibilidade, encontrada no padrão NTCIP, e ainda possibilita a fácil integração de equipamentos de campo legados ao sistema. Esta integração é obtida através da aplicação de regras de transformação XSLT na mensagem padrão do protocolo.

A definição da mensagem padrão do protocolo é um assunto que ainda deve ser intensamente estudado e discutido visando obter uma solução eficiente e que minimize o tamanho da mensagem enviada.

Conclui-se então que a proposta apresentada contribui para a definição de um padrão nacional para comunicação entre uma central e controladores semafóricos. Esta proposta visa atender as propriedades de intercambiabilidade e interoperabilidade.

7.2 Trabalhos Futuros

Com o intuito de dar continuidade a esta pesquisa, ficam como sugestão para trabalhos futuros os seguintes itens:

- implementação do compilador de objetos gerenciáveis, estes escritos em XML, apresentado na seção 6.1;
- definição da mensagem padrão do protocolo;

- proposição da uma técnica para a correta transmissão e compreensão dos dados enviados pelo controlador semafórico para a central de controle.
- substituição da SMI em árvore por estrutura baseada em *namespaces*.

Apêndice A

Abstract Syntax Notation One (ASN.1)

No processo de transmissão de dados, entre sistemas finais, são definidas sintaxes de representação dos dados. Estas determinam a forma de representação das informações em diferentes componentes do sistema. A **sintaxe abstrata** é a forma de representação da informação no componente de aplicação. Esta sintaxe lida com tipo e valores de dados e é utilizada para a troca de informação entre componentes de aplicação de diferentes sistemas. Já a **sintaxe de transferência** é a representação dos dados na forma binária interagindo com o componente de transferência de dados [33]. Este formato é utilizado para a troca de dados entre componentes de transferência de dados.

Deve ser utilizada alguma forma de codificação, para que, dados representados na sintaxe abstrata possam ser representados na forma de sintaxe de transferência. O *Abstract Syntax Notation One* (ASN.1) é uma forma de representar sintaxes abstratas para os dados da aplicação. O ASN.1 é uma linguagem formal desenvolvida e padronizada pela CCITT (X.208) e ISO (ISO 8824). Como será mostrada na próxima sessão, esta é uma linguagem que define estrutura de dados na forma de módulos.

A.1 Conceitos ASN.1

No ASN.1 as estruturas de dados são representadas na forma de modulo [33]. Nesta sessão será mostrado o formato dos módulos e alguns tipos de dados serão apresentados.

Módulos

O formato básico dos módulos é conforme mostrado abaixo:

```
<nome> DEFINITIONS ::=
BEGIN
EXPORTS
IMPORTS
AssignmentList
End
```

O campo `EXPORTS` indica as definições do modulo que podem ser importadas por outros módulos, o campo `IMPORTS` indica as definições que o modulo deve importar de outros módulos. E finalmente o campo AssignmenteList consiste na definição de tipos e valores. Todo tipo no ASN.1 tem uma classe e um numero associado a ele. Existindo quatro classes [33]:

Universal - tipo para uso geral, é definido em um padrão.

Application-wide - relevante a uma aplicação em particular.

Context-specific - também é relevante a uma aplicação em particular, mas num contexto limitado.

Private - tipos definidos pelo usuário, não representados em um padrão.

Alem de divisão em classes os tipos podes ser classificados em 4 categorias: simples, estruturado, tagged e other. O tipo estruturado é formado pela composição de outros tipos ASN.1, os tipos estruturados possíveis são:

Sequence of - define uma lista ordenada de valores do mesmo tipo.

Sequence - define uma lista ordenada de valores de um ou mais tipos.

Set e set of - são similares ao sequence e sequence of, respectivamente, mas neste caso a ordem dos componentes não é significante.

A Tabela A.1 ilustra tipos ASN.1, definidos na classe universal, e seus identificadores.

Tabela A.1: *Tipos do ASN.1 e seus identificadores. [21]*

Tag	Tipo
UNIVERSAL 0	Reserved for use by the encoding rules
UNIVERSAL 1	Boolean type
UNIVERSAL 2	Integer type
UNIVERSAL 3	Bitstring type
UNIVERSAL 4	Octetstring type
UNIVERSAL 5	Null type
UNIVERSAL 6	Object identifier type
UNIVERSAL 7	Object descriptor type
UNIVERSAL 8	External type and Instance-of type
UNIVERSAL 9	Real type
UNIVERSAL 10	Enumerated type
UNIVERSAL 11	Embedded-pdv type
UNIVERSAL 12	UTF8String type
UNIVERSAL 13	Relative object identifier type
UNIVERSAL 14-15	Reserved for future editions of this Recommendation International Standard
UNIVERSAL 16	Sequence and Sequence-of types
UNIVERSAL 17	Set and Set-of types
UNIVERSAL 18-22, 25-30	Character string types
UNIVERSAL 23-24	Time types
UNIVERSAL 31-...	Reserved for addenda to this Recommendation International Standard

A.2 ASN.1 Macro

Na especificação do ASN.1 está incluída a notação de ASN.1 macro. Esta notação permite ao usuário estender a sintaxe do ASN.1 definindo novos tipos e seus valores [33]. Na macro existem três níveis:

Macro notation - notação utilizada para definir macros.

Macro definition - é expressa através de uma notação macro e é utilizada para definir um conjunto de instancias macro.

Macro instance - gerada a partir da substituição de valores em variáveis de uma definição macro.

Uma definição macro pode ser vista como um template que é utilizado para gerar um conjunto de tipos relacionados. Uma definição macro tem a forma mostrada a seguir.

```
<macroname> MACRO ::=
BEGIN
TYPE NOTATION ::= <new-type-syntax>
VALUE NOTATION ::= <net-value-syntax>
<supporting-productions>
END
```

O nome da macro deve ser escrito em letras maiúsculas. O campo `new-type-syntax` descreve o novo tipo e o campo `new-value-syntax` o valor do novo tipo. Já o `supporting-productions` prove regras gramaticais adicionais tanto para a sintaxe do tipo quanto do valor. Quando valores específicos são substituídos nas variáveis e argumentos da definição da macro é formada uma instancia macro. Esta representa um tipo ASN.1 básico chamado *returned type*. e um valor ASN.1 genérico chamado *returned value*.

A.3 BER

Regras de codificação permitem que as informações de gerência sejam transmitidas pela rede. O BER (*Basic Encoding Rules*) define uma sintaxe de transferência que pode ser aplicada aos tipos definidos utilizando uma notação ASN.1 [22].

As sintaxes de transferência trabalham com a comunicação entre equipamentos a nível de bits. A estrutura de codificação do BER utiliza uma representação na forma *Type-Length-Value* [24]. O campo *type* (tipo) contêm um identificador da estrutura codificada, corresponde ao identificador do tipo ASN.1. O campo *length* (tamanho) indica o numero de octetos que o campo valor irá conter. Já o campo *value* (valor) contêm o valor do dado.

Seguindo a estrutura de TLV, existem 3 métodos de codificação [33]:

***Primitive, define-length encoding*:** Utilizado para tipos simples e *tag*. Tem tamanho conhecido e possui 3 campos:

- *Identifier* »> ID: _ _ (2 bits classe) + 0 (tipo primitivo) + _ _ _ _ _ (numero da tag ou 11111 e outro octeto);
- *Length* »> 0 + 7 bits (tamanho do conteudo) ou 1 + 7 bits (numero de octetos extras) + outros octetos;

- *Contents* »> *string* de octetos;

***Contructed, define-length encoding*:** Tipos estruturados, *simple-strings* e *tag*. Tem tamanho conhecido e possui 3 campos:

- *Identifier* »> ID: _ _ (2 bits classe) + 1 (tipo construido) + _ _ _ _ _ (numero da tag ou 11111 e outro octeto);
- *Length* »> 0 + 7 bits (tamanho do conteudo) ou 1 + 7 bits (numero de octetos extras) + outros octetos;
- *Contents* »> concatenação das TLVs dos componentes.

***Contructed, indefine-length encoding*:** Tipos estruturados, *simple-strings* e tag. Com tamanho indefinido, possui 4 campos:

- *Identifier* »> ID: _ _ (2 bits classe) + 1 (tipo construido) + _ _ _ _ _ (numero da tag ou 11111 e outro octeto);
- Length »> 80 hex;
- *Contents* »> concatenação das TLVs dos componentes;
- *End-of-contents* »> dois octetos 0000 hex;

Apêndice B

Definição de MIBs para Controladores Semafóricos Nacionais

```
--2.1     MIB HEADER

tempofixoBr DEFINITIONS ::= BEGIN

IMPORTS
    OBJECT-TYPE
    FROM RFC-1212
    devices
    FROM TMIB;

-- tfBr NODE

tfBr OBJECT IDENTIFIER ::= { devices 50 }

-- Groups in tfBr

--tConfig OBJECT IDENTIFIER ::= { tfBr 1 }
--tabelaEstagio OBJECT IDENTIFIER ::= { tfBr 2 }
--tabelaPlano OBJECT IDENTIFIER ::= { tfBr 3 }
--tabelaVerdesConflitantes OBJECT IDENTIFIER ::= { tfBr 4 }
--tabelaGrupoAtivo OBJECT IDENTIFIER ::= { tfBr 5 }
--tabelaHora OBJECT IDENTIFIER ::= { tfBr 6}
--tabelaData OBJECT IDENTIFIER ::= { tfBr 7}

tabelaEstagio   OBJECT-TYPE
SYNTAX    SEQUENCE OF EntradaTabelaEstagio
ACCESS    not-accessible
```

```
STATUS    optional
DESCRIPTION
     " Este objeto representa uma tabela de dados relacionados a um estagio."
::= { tfBr 2 }

entradaTabelaEstagio   OBJECT-TYPE
SYNTAX    EntradaTabelaEstagio
ACCESS    not-accessible
STATUS    mandatory
DESCRIPTION
     "Este objeto representa uma linha na tabela de estagio."
INDEX   { numeroEstagio }
::= { tabelaEstagio 1 }

EntradaTabelaEstagio::= SEQUENCE {
          numeroEstagio INTEGER,
          minimoVerde INTEGER,
          minimoAmarelo   INTEGER,
          minimoAmareloAntecipado  INTEGER,
          minimoVermelhoIntegral   INTEGER,
          minimoVermelhoLimpeza  INTEGER,
          gruposSemaAtivos    INTEGER }

-- Tabela tEstagio

numeroEstagio OBJECT-TYPE
  SYNTAX INTEGER
  ACCESS  read-write
  STATUS  mandatory
  DESCRIPTION
     "área  lógica que o controlador pertence"
  ::= { entradaTabelaEstagio 1 }

minimoVerde OBJECT-TYPE
  SYNTAX INTEGER
  ACCESS  read-write
  STATUS  mandatory
  DESCRIPTION
     " endereço do controlador "
  ::= { entradaTabelaEstagio 2 }

minimoAmarelo OBJECT-TYPE
  SYNTAX INTEGER
  ACCESS  read-write
  STATUS  mandatory
  DESCRIPTION
     " tempo de amarelo "
  ::= { entradaTabelaEstagio 3 }

minimoAmareloAntecipado OBJECT-TYPE
```

```
  SYNTAX INTEGER
  ACCESS  read-write
  STATUS  mandatory
  DESCRIPTION
     " tempo de amarelo "
  ::= { entradaTabelaEstagio 4 }

minimoVermelhoIntegral OBJECT-TYPE
  SYNTAX INTEGER
  ACCESS  read-write
  STATUS  mandatory
  DESCRIPTION
     " tempo de vermelho "
  ::= { entradaTabelaEstagio 5 }

minimoVermelhoLimpeza OBJECT-TYPE
  SYNTAX INTEGER
  ACCESS  read-write
  STATUS  mandatory
  DESCRIPTION
     " tempo de vermelho "
  ::= { entradaTabelaEstagio 6 }

gruposSemaAtivos OBJECT-TYPE
  SYNTAX INTEGER
  ACCESS  read-write
  STATUS  mandatory
  DESCRIPTION
     " op-default "
  ::= { entradaTabelaEstagio 7 }


tabelaPlano   OBJECT-TYPE
SYNTAX    SEQUENCE OF EntradaTabelaPlano
ACCESS    not-accessible
STATUS    optional
DESCRIPTION
     " Este objeto representa uma tabela de dados relacionados a um plano."
::= { tfBr 3 }

entradaTabelaPlano   OBJECT-TYPE
SYNTAX    EntradaTabelaPlano
ACCESS    not-accessible
STATUS    mandatory
DESCRIPTION
     " Este objeto representa uma linha na tabela de plano."
INDEX   { numeroPlano }
::= { tabelaPlano 1 }

EntradaTabelaPlano::= SEQUENCE {
```

```
          numeroPlano INTEGER,
          tempoVerde INTEGER,
          tempoExtensaoVerde   INTEGER,
          tempoMaximoVerde  INTEGER,
          tempoAmarelo   INTEGER,
          tempoVermelhoIntegral  INTEGER,
          tempoVermelhoLimpeza  INTEGER,
          tipo  INTEGER,
          dvi  INTEGER,
          tempoAmareloAntecipado  INTEGER }

-- Tabela Plano

numeroPlano OBJECT-TYPE
  SYNTAX INTEGER
  ACCESS  read-write
  STATUS  mandatory
  DESCRIPTION
     "área  lógica que o controlador pertence"
  ::= { entradaTabelaPlano 1 }

tempoVerde OBJECT-TYPE
  SYNTAX INTEGER
  ACCESS  read-write
  STATUS  mandatory
  DESCRIPTION
     " endereço do controlador "
  ::= { entradaTabelaPlano 2 }

tempoExtensaoVerde OBJECT-TYPE
  SYNTAX INTEGER
  ACCESS  read-write
  STATUS  mandatory
  DESCRIPTION
     " tempo de amarelo "
  ::= { entradaTabelaPlano 3 }

tempoMaximoVerde OBJECT-TYPE
  SYNTAX INTEGER
  ACCESS  read-write
  STATUS  mandatory
  DESCRIPTION
     " tempo de amarelo "
  ::= { entradaTabelaPlano 4 }

tempoAmarelo OBJECT-TYPE
  SYNTAX INTEGER
  ACCESS  read-write
  STATUS  mandatory
  DESCRIPTION
```

```
     " tempo de vermelho "
  ::= { entradaTabelaPlano 5 }

tempoVermelhoIntegral OBJECT-TYPE
  SYNTAX INTEGER
  ACCESS  read-write
  STATUS  mandatory
  DESCRIPTION
     " tempo de vermelho "
  ::= { entradaTabelaPlano 6 }

tempoVermelhoLimpeza OBJECT-TYPE
  SYNTAX INTEGER
  ACCESS  read-write
  STATUS  mandatory
  DESCRIPTION
     " tempo de vermelho de limpeza "
  ::= { entradaTabelaPlano 7 }

tipo OBJECT-TYPE
  SYNTAX INTEGER
  ACCESS  read-write
  STATUS  mandatory
  DESCRIPTION
     " mudança de plano soft ou abrupt "
  ::= { entradaTabelaPlano 8 }

dvi OBJECT-TYPE
  SYNTAX INTEGER
  ACCESS  read-write
  STATUS  mandatory
  DESCRIPTION
     " tempo Máximo de permanência num estágio "
  ::= { entradaTabelaPlano 9 }

tempoAmareloAntecipado OBJECT-TYPE
  SYNTAX INTEGER
  ACCESS  read-write
  STATUS  mandatory
  DESCRIPTION
     " número de reinicializações em caso de falha "
  ::= { entradaTabelaPlano 10 }


tabelaVerdesConflitantes   OBJECT-TYPE
SYNTAX    SEQUENCE OF EntradaTabelaVerdesConfit
ACCESS    not-accessible
STATUS    optional
DESCRIPTION
     " Este objeto representa uma tabela onde estão marcados os grupos
```

```
      semafóricos conflitantes."
::= { tfBr 4 }

entradaTabelaVerdesConfit   OBJECT-TYPE
SYNTAX    EntradaTabelaVerdesConfit
ACCESS    not-accessible
STATUS    mandatory
DESCRIPTION
     "Este objeto representa uma linha na tabela de grupos semafóricos
      conflitantes."
INDEX   { numeroGrupo }
::= { tabelaVerdesConflitantes 1 }

EntradaTabelaVerdesConfit::= SEQUENCE {
          numeroGrupo INTEGER,
          grupoConflito   INTEGER }

numeroGrupo  OBJECT-TYPE
  SYNTAX INTEGER
  ACCESS  read-write
  STATUS  mandatory
  DESCRIPTION
     " valor inteiro indicando numero do grupo semaforico"
  ::= { entradaTabelaVerdesConfit 1}

grupoConflito OBJECT-TYPE
  SYNTAX INTEGER
  ACCESS  read-write
  STATUS  mandatory
  DESCRIPTION
     " grupos em conflito são indicados na forma binária; um bit igual
      a 1 indica que há conflito entre os grupos e um bit 0 indica que
       não há conflito."
  ::= { entradaTabelaVerdesConfit 2}

END
```

Referências Bibliográficas

[1] Andeli, J. (1989). Advanced vehicle/highway systems and urban traffic problems. OTA Publications. Disponível em: http://govinfo.library.unt.edu/ota/Ota2/DATA/1989/8902.pdf Acesso em: Maio 2007.

[2] Araujo, A. P. M. (2004). Desenvolvimento de sistemas inteligentes de transportes baseados na arquitetura de referência nia. Dissertação (Mestrado em Engenharia Elétrica), Centro Tecnológico, Universidade Federal de Santa Catarina, Florianópolis SC.

[3] Brascontrol (1999). *Manual de Instalação, Programação e Operação.* Brascontrol Indústria e Comércio LTDA., Barueri, SP, Brasil.

[4] Cambruzzi, E. (2003). Avaliação experimental da infraestrutura computacional para sistemas inteligentes de transporte. Dissertação (Mestrado em Engenharia Elétrica), Centro Tecnológico, Universidade Federal de Santa Catarina, Florianópolis SC.

[5] CET-SP (2005). *Especificação Técnica de Controladores Semafóricos Eletrônicos de Médio Porte.* Companhia de Engenharia de Tráfego, São Paulo., São Paulo, SP.

[6] Courage, K. G., Washburn, S. S., e Kim, J.-T. (2002). Development of an xml-based specification for traffic model data exchange. *Transportation Research Record, Transportation Data and Information Technology Research.*, Vol. 1804pp. 144–150.

[7] Dataprom (2005). Controlador semafórico dp-40. Dataprom Equipamentos e Serviços de Informática Industrial Ltda. Curitiba, PR. Disponível em: http://www.dataprom.com/ Acesso em: Maio 2007.

[8] DENATRAN (1979). *Serviços de Engenharia: Manual de Semáforos.* Ministério da Justiça. Conselho Nacional de Trânsito. Departamento Nacional de Trânsito., Brasília, DF.

[9] Digicon (2007). Controlador linha cd-200. Gravataí, RS. Disponível em: http://www.digicon.com.br/produtos/controladores/cd200/index.htm Acesso em: Maio 2007.

[10] Eisenberg, A. e Melton, J. (2001). Sql/xml and the sqlx informal group of companies. YES. Disponível em: http://cies.hhu.edu.cn/pweb/ zhuoming/teaching/MOD/N4/Readings/2.3-E5.pdf Acesso em: Maio 2007.

[11] Faber, O. (1998). Suitability of ntcip applications messaging for UK UTMC users. REPORT 1. Department for Transport, Inglaterra. Disponível em: http://www.utmc.gov.uk/utmc09/pdf/utmc09.pdf Acesso em: Maio 2007.

[12] Federal Highway Administration (2001). National transportation communications for its protocol - global object definition. Disponível em: http://www.ntcip.org/library/documents/pdf/1201v0110b.pdf Acesso em: Maio 2007.

[13] Federal Highway Administration (2002). The ntcip guide/national transportation communications for its -protocol - joint committee on the ntcip. Disponível em: http://www.ntcip.org/library/documents/pdf/9001v0302b.pdf Acesso em: Maio 2007.

[14] Federal Highway Administration (2003). National transportation communications for its protocol - xml in its center-to-center communications. Disponível em: http://www.ntcip.org/library/documents/pdf/9010v0107XMLinC2C.pdf Acesso em: Maio 2007.

[15] Federal Highway Administration (2005a). National transportation communications for its protocol - object definitions of actuated traffic signal controller (asc) units. Disponível em: http://www.ntcip.org/library/documents/pdf/1202v0107d.pdf Acesso em: Maio 2007.

[16] Federal Highway Administration (2005b). National transportation communications for its protocol - transportation management protocols. Disponível em: http://www.ntcip.org/library/documents/pdf/1103v01-26a.pdf Acesso em: Maio 2007.

[17] Federal Highway Administration (2006). National transportation communications for its protocol - application profile for xml message encoding and transport in its center-to-center communications (ntcip - c2c xml). Disponível em: http://www.ntcip.org/library/documents/pdf/2306v0168b.pdf Acesso em: Maio 2007.

[18] Ferreira, S. M., Kraus, W. J., e Montez, C. B. (2006). Estudo para definição de padrão para comunicação de controladores semafóricos no brasil. Proc. *XX Congresso de Pesquisa e Ensino em Transportes, Brasília, DF*.

[19] Holzner, S. (2001). *Inside XML*. New Riders Publishing, Indianapolis, Indiana, 1 edição.

[20] ITS America (2003). Its america home page. Disponível em: http://www.itsa.org Acesso em: Maio 2007.

[21] ITU (2002a). Information technology. abstract syntax notation one (asn.1). specification of basic notation. International Telecommunication Union. Disponível em: http://www.itu.int/ITU-T/studygroups/com17/languages/X.680-0207.pdf Acesso em: Maio 2007.

[22] ITU (2002b). Information technology. asn.1 encoding rules - specification of basic encoding rules (ber), canonical encoding rules (cer) and distinguished encoding rules (der). International Telecommunication Union. Disponível em: http://www.itu.int/ITU-T/studygroups/com17/languages/X.690-0207.pdf Acesso em: Maio 2007.

[23] Kirwan, K. (2004). Xml for distributing real-time traffic information - a practical application. Disponível em: http://www.iasi.rm.cnr.it/ewgt/13conference/80kirwan.pdf Acesso em: Maio 2007.

[24] Miller, M. A. (1999). *Managing Internetworks with SNMP, Third Edition*. MeT Books, Foster City, CA 94404, 3 edição.

[25] Ni, D. e John, L. D. (2004). Development of trafficxml: Prototype xml for traffic simulation. *Transportation research record (Transp. res. rec.) ISSN 0361-1981 CODEN TRREDM*, Vol. 1879pp. 30–40.

[26] NTCIP (2007a). National transportation communications for ITS protocol library home page. Disponível em: http://www.ntcip.org/library/ Acesso em: Maio 2007.

[27] NTCIP (2007b). NTCIP Exerciser. Disponível em: http://www.ntcip.org/library/software/abstract.asp?AbstractID= Acesso em: Maio 2007.

[28] Patel, R. K. e Seymour, E. J. (1997). National transportation communication for its protocol (ntcip) for transportation interoperability. *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC 1997. IEEE, Piscataway, NJ, USA,97TH8331.CODEN: 002845*, Vol.pp. 543–548.

[29] Riley, C. (2001). Ntcip mib's xml storage mechanism. Edwards and Kelcey Technology. Leesburg, Virginia. Disponível em: http://www.chart.state.md.us/downloads/readingroom/CHARTIIDocuments/r1b4 /NTCIP-DriverDesign/NTCIPDriverMibsXMLHighLevelDesign.pdf Acesso em: Maio 2007.

[30] Roche, R. D. (2006). Correspondêcia pessoal.

[31] Rose, M. e McCloghrie, K. (1990). Structure and identification of management information for tcp/ip-based internets. rfc 1155.

[32] Rose, M. e McCloghrie, K. (1991). Concise mib definitions. rfc 1212.

[33] Stallings, W. (1999). *SNMP, SNMPv2, and RMON 1 and 2.* Addison Wesley Longman, Reading, MA, 3 edição.

[34] Tanenbaum, A. S. (1999). *Redes de Computadores.* Editora Campus, Rio de Janeiro, RJ, 4 edição.

[35] Tesc (2006). Controladores eletrônicos de trânsito. Tesc Indústria e Comércio Ltda., São Paulo, S.P. Disponível em http://www.tesc.com.br/index/ezsite.asp?id=467. Acesso em Maio 2006.

[36] Toomey, C., Patel, M., e Allan, T. (1998). Evaluation of ntcip - suitability of ntcip based communications for UK UTMC users. Carl Bro IBI. Disponível em: http://www.utmc.gov.uk/utmc08/pdf/utmc08eval.pdf Acesso em: Maio 2007.

[37] U.S. Department of Transportation (1996). Intelligent transportation systems in japan. Disponível em: http://www.fhwa.dot.gov Acesso em: Maio 2007.

[38] U.S. Department of Transportation (2007). What is its? Disponível em: http://www.its.dot.gov Acesso em: Maio 2007.

[39] UTMC Development Group (2005). UTMC technical specification 003 (ts003:2005). Department for Transport, Inglaterra. Disponível em: http://www.utmc.uk.com/technical/pdf/ts003.pdf Acesso em: Maio 2007.

[40] Vianna, M. M. B., Portugal, L. S., e Balassiano, R. (1999). A telemática aplicada ao gerenciamento de estaconamentos. *In XIII ANPET Congresso de Pesquisa e Ensino em Transportes - Engenharia e Segurança de Tráfego. Associação Nacional de Pesquisa e Ensino em Transportes.*, Vol. 2, No. 12, pp. 12–15.

[41] W3C (1998). Extensible markup language (xml). Disponível em: http://www.w3.org/TR/1998/REC-xml-19980210 Acesso em: Maio 2007.

[42] W3C (1999). Xsl transformations (xslt) version 1.0. Disponível em: http://www.w3.org/TR/xslt Acesso em: Maio 2007.

[43] W3C (2006). Namespaces in xml 1.0 (second edition). Disponível em: http://www.w3.org/TR/REC-xml-names/ Acesso em: Maio 2007.