Rodrigo Marinho Passos

Modelagem e implementação de protocolos para comunicação com e sem fio de um sistema integrado Hardware/Software em tempo real

Monografia de Graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências da disciplina Projeto Orientado para obtenção do título de Bacharel em Ciência da Computação.

Orientador
Prof. Guilherme Bastos Alvarenga

Lavras
Minas Gerais - Brasil
2002

Rodrigo Marinho Passos

Modelagem e implementação de protocolos para comunicação com e sem fio de um sistema integrado Hardware/Software em tempo real

Monografia de Graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências da disciplina Projeto Orientado para obtenção do título de Bacharel em Ciência da Computação.

Aprovada em *27 de Março de 2002*

Prof. Jones Oliveira de Albuquerque

Prof. Luís Henrique Andrade Correia

Prof. Guilherme Bastos Alvarenga
(Orientador)

Lavras
Minas Gerais - Brasil

Dedico este trabalho a minha mãe por não medir esforços para a felicidade de seus filhos, a minha madrinha Rosa por ter demonstrado ser minha segunda mãe em todos os momentos, a minha avó Rosita pela eterna dedicação, a minha namorada Juliana pelo constante companherismo, e a todas as pessoas que acreditam ser capazes de realizar seus sonhos.

Agradecimentos

Agradeço a Deus pela constante presença.
Agradeço ao Prof. Guilherme Bastos Alvarenga pela oportunidade e pela confiança em mim depositada.
Agradeço à equipe de desenvolvimento da empresa Devex Tecnologia e Sistema pela ajuda e participação.
Agradeço a minha família, amigos e minha namorada Juliana pela paciência.
Agradeço a minha amiga Vanessa pela sinceridade, amizade e preocupação.
Agradeço também a todas as pessoas que diretamente ou indiretamente participaram na realização deste trabalho.

Resumo

Este trabalho descreve o projeto e implementação da comunicação entre os módulos de um sistema integrado de hardware e software em tempo real, que se encontra em desenvolvimento na empresa Devex Tecnologia e Sistemas. Entre outras questões, relaciona-se a integração hardware/software (Codesign) com e sem fio, a estrutura da rede fixa e móvel, a tecnologia de transmissão, o hardware e o software de rede, alvos dos principais esforços deste trabalho. Desta maneira, evidencia-se as hierarquias de protocolo, serviços e referências aos modelos básicos das camadas de rede, bem como as decisões e dificuldades de modelagem e implementação dos conjuntos de regras definidas para a comunicação com e sem fio entre os módulos do sistema.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

A comunicação, uma das atividades mais básicas dos seres humanos, tem sido alvo de estudos em diversas áreas da ciência, muito antes da existência dos computadores. Dessa maneira, é de longa data o problema de modelar regras de comunicação eficientes e não ambíguas [Hol91].

A era da informação tem motivado o desenvolvimento e criação de tecnologias que sejam capazes de suprir a necessidade de comunicação e obtenção de informação, de forma cada vez mais rápida e eficiente. Neste contexto, as redes de computadores têm um papel fundamental na transmissão, confiabilidade e compartilhamento da informação [Tan97].

No projeto das primeiras redes de computadores, o hardware foi colocado como prioridade e o software, em segundo plano. Essa estratégia foi deixada para trás, pois o software de rede está altamente estruturado [Tan97]. Assim, o desenvolvimento do software nas redes de computadores tem contribuído batante para o aumento de desempenho, confiabilidade, adaptabilidade, entre outras questões, dos sistemas baseados em redes.

Com o crescimento da importância do software de rede e a consolidação do hardware de rede, o desenvolvimento de aplicações baseadas em redes de computadores necessita, cada vez mais, de uma análise da especificação do sistema e das tecnologias existentes. Esta análise visa realizar a integração do hardware e software de rede, de modo a atender as restrições e requisitos dos sistemas eficientemente.

As técnicas e modelos de codesign existentes têm servido como um guia na análise e modelagem de sistemas integrados Hardware/Software, de modo a realizar a divisão sobre o que deve ser implementado em software e o que deve ser

implementado em hardware. Na tentativa de reduzir a complexidade da integração do software e o hardware de rede, as redes foram organizadas como uma série de camadas que conversam entre si, através de regras e convenções denominadas protocolos.

A alta complexidade dos sistemas de computação baseados em redes de computadores e computação móvel trazem grandes problemas nas fases de projeto e implementação, que se refletem principalmente em maior tempo e custo de desenvolvimento. Assim, sistemas integrados hardware/software (HW/SW) exigem cada vez mais, que os projetistas considerem múltiplas tecnologias durante a modelagem e implementação, com o objetivo de reduzir os custos e/ou cumprir os prazos [Alb01].

Um destes sistemas complexos encontra–se em desenvolvimento na empresa Devex Tecnologia e Sistemas. Este sistema, nomeado SmartMine, possui cinco módulos principais descritos mais detalhadamente durante este texto, que se comunicam, evidenciando as características de uma das principais etapas dos modelos de Codesign, a integração HW/SW. O SmartMine é um sistema especificado para funcionar 24 horas por dia os 7 dias da semana, recebendo informações dos módulos de hardware a todo instante. Essas e outras características descritas ao longo do texto evidenciam a característica “tempo real” do SmartMine

Sistemas críticos de tempo real, como o SmartMine, que utilizam diferentes meios de transmissão, com e sem fio, necessitam de uma análise da estrutura física e lógica do fluxo das informações, bem como uma modelagem da integração Hardware/Software (HW/SW) para se estabelecer regras de comunicação confiáveis, evitando que o funcionamento do sistema seja interrompido.

Considerando as características e funcionalidades do SmartMine, os modelos de codesign, as técnicas de modelagem de protocolos e as tecnologias existentes, este trabalho descreve a modelagem e implementação da comunicação entre os módulos do sistema SmartMine, evidenciando as dificuldades e decisões de modelagem. Entre outras questões, descreve–se fluxo de dados do sistema, o software de rede e a hierarquia dos protocolos de acordo com o modelo de camadas OSI [DZ83]. Assim, os principais esforços deste trabalho concentram–se no software de rede do sistema SmartMine, definindo–se o protocolo de comunicação entre o firmware (software embarcado do hardware) e o software da aplicação.

Este trabalho está dividido em seis capítulos. O Capítulo 2 aborda as técnicas atuais de modelagem de protocolos e de codesign. O Capítulo 3 descreve o SmartMine, relacionando–se todos os módulos do sistema e enfatizando aqueles de principal importância neste trabalho. O Capítulo 4 aborda a modelagem

e implementação do protocolo de comunicação HW/SW, mais especificamente, a comunicação entre o firmware (software embarcado do hardware) e o software (aplicação). O Capítulo 5 aborda a modelagem e implementação do protocolo de comunicação entre o módulo principal do sistema e a aplicação Web. Finalmente, o Capítulo 6 apresenta as conclusões e futuros trabalhos.

Capítulo 2

Metodologia

Neste capítulo, apresenta–se uma visão geral dos modelos mais utilizados em *Hardware/Software codesign* e algumas metodologias e técnicas utilizadas na modelagem e implementação de protocolos de comunicação, que fazem parte da etapa de integração HW/SW do processo de codesign, principal enfoque deste trabalho.

2.1 Codesign

A busca cada vez mais constante de soluções tecnológicas tem gerado sistemas de alta complexidade, que envolvem uma série de questões tais como: informação em tempo real, integração Hardware/Software, tecnologia de transmissão, armazenamento de grandes massas de dados, entre outros. Conseqüentemente, o projeto destes sistemas tornou–se bem mais complexo, exigindo novas abordagens e metodologias capazes de promover uma melhoria nas fases de projeto e implementação, visando reduzir o custo e tempo de projeto.

Para suportar o projeto de forma efetiva, uma metodologia de projeto denominada *Hardware/Software codesign* vem sendo desenvolvida. *Hardware/Software codesign* é um paradigma de projeto, cujo principal objetivo consiste em projetar sistemas que satisfaçam às restrições de projetos [BC98].

Dentre as principais etapas da metodologia de HW/SW destacam–se:

- Análise de restrições e requisitos;
- Especificação do sistema;
- Particionamento em hardware e software;

- Co–síntese;
- Integração do hardware e do software;
- Validação e verificação do projeto;

A Figura 2.1 mostra a relação entre as principais etapas.

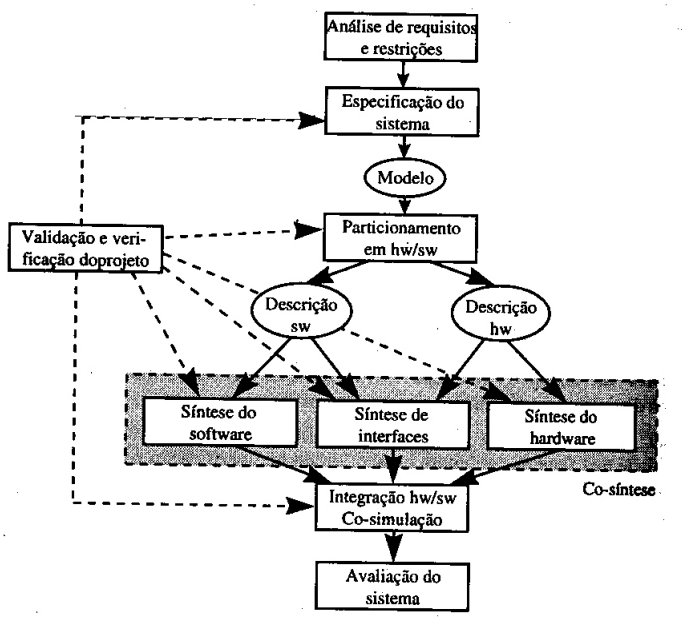

Figura 2.1: Metodologia de Hardware/Software codesign [BC98]

Segundo [BC98], na etapa de análise de requisitos são definidas as características do sistema com base nas especificações do cliente, na qual são capturados requerimentos de tempo real do sistema, tecnologia de realização, custo de desenvolvimento, entre outros.

Na etapa de especificação do sistema, representa–se a análise de requisitos através de um modelo de especificação. Os principais modelos de especificação segundo [BC98] são:

- Modelos orientados a estados;
- Modelos orientados a atividades;

- Modelos orientados a estrutura;
- Modelos heterogêneos.

A decisão sobre quais partes do sistema serão implementadas em hardware e quais partes em software, é realizada na etapa de particionamento. O problema de particionamento é um problema NP–Completo, segundo [GJ79], e existem vários algoritmos citados na literatura para resolução deste problema. Alguns destes algoritmos são listados abaixo:

- Clustering hierárquico [Joh67], [Lag89];
- Min Cut [kL70], [kri84];
- Simulated annealing [SKV83];
- Programação linear inteira [GJ79].

A etapa de co–síntese visa a síntese do hardware, do software e da interface entre ambos [BC98]. A síntese de hardware envolve o mapeamento tecnológico da descrição do hardware nas unidades físicas funcionais da arquitetura. Por sua vez, a síntese de software inclui a compilação dos módulos de software para a arquitetura alvo. A síntese de interface visa a geração de hardware e software necessários para a sincronização dos módulos de hardware e software.

A integração hardware e software é uma das etapas mais complicadas do processo de codesign. Esta etapa deve gerar, como resultado, um protótipo executável do sistema construído fisicamente ou simulado. As pesquisas nessa área ainda se encontram num estágio inicial [BC98].

Finalmente, a etapa de validação e verificação tem por finalidade garantir que o sistema resultante e a especificação sejam semanticamente equivalentes [BC98]. A validação pode se dar através de simulação ou verificação formal de algumas propriedades do sistema.

Foge ao escopo deste trabalho, uma descrição mais detalhada de cada uma destas etapas. No entanto, este trabalho aborda a etapa de integração hardware e software, principalmente no que diz respeito à comunicação entre o software embarcado do hardware e o software em sí. Assim, o objetivo principal é descrever as regras de comunicação “Firmware/Software”, presentes na etapa de integração dos modelos básicos de codesign. Uma descrição mais completa da modelagem das regras de comunicação é assunto da próxima seção.

2.2 Modelagem de Protocolos

2.2.1 Introdução

A modelagem de protocolos é um problema tão antigo quanto a comunicação. Modelar um protocolo não é uma tarefa fácil nos tempos atuais. Os avanços tecnológicos têm exigido mais funcionabilidade e confiabilidade, aumentando os protocolos em tamanho e principalmente em complexidade.

Um protocolo é uma espécie de "acordo" na troca de informação entre as partes envolvidas de um sistema distribuído [Hol91]. Dessa maneira, um protocolo pode ser entendido como um conjunto de regras de transmissão. Uma definição formal de um protocolo se parece mais com a definição de um linguagem:

- Define–se um formato preciso para mensagens válidas;
- Define–se as regras para trocas de dados (uma gramática);
- Define–se um vocabulário de mensagens válidas que possam ser trocadas, com seus vocabulários (semânticas).

A gramática de um protocolo deve ser logicamente consistente e completa. Sob todas as possíveis circunstâncias as regras devem prescrever, em termos não ambíguos, o que é permitido e o que é proibido. Entre outras questões, é importante para a definição de um protocolo, que não existam apenas regras para troca de informação, mas sim um acordo entre o transmissor e o receptor sobre estas regras.

A definição de um protocolo envolve questões como a escolha do meio de transmissão, como a transmissão é iniciada e finalizada, transferência de dados, controle de fluxo, detecção e correção de erros, entre outros. No entanto, o principal desafio da modelagem de protocolos consiste em evitar dois erros: a definição incompleta de um conjunto de regras e a definição de regras que são contraditórias. Essas e as demais questões serão abordadas ao longo do texto.

2.2.2 Estrutura de um protocolo

Para garantir que o conjunto de regras seja completo e consistente, se faz necessário especificar todas as partes importantes da estrutura de um protocolo.

Algumas características principais da estrutura de um protocolo podem ser encontradas ao se descrever a comunicação de duas máquinas que compartilham informações. Inicialmente, para que elas possam se comunicar, o meio físico deve

ser o mesmo, assumindo–se que não exista nenhum tipo de *gateway* ou ponte entre as máquinas. Isto significa que as máquinas devem estar compartilhando o mesmo tipo de rede, seja ela uma rede *ethernet* 10 ou 100 Mbps, ou uma rede *wireless* com comunicação via rádio–modem. As máquinas também devem ter uma codificação de caracteres compatíveis, mesma velocidade de transmissão e recepção, entre outras padronizações.

Ao garantir o cumprimento das configurações iniciais, a próxima preocupação no desenvolvimento de um protocolo consiste em como descobrir se uma das maúquinas está ou não disponível. Isto implica que o transmissor deve parar de transmitir quando o receptor não está disponível e que o transmissor deve voltar a transmitir assim que o receptor estiver disponível novamente.

A comunicação entre duas maúquinas pode ser realizada de duas maneiras. Na primeira e mais comum, qualquer uma das máquians pode se tornar transmissor e a outra conseqüentemente receptor. A segunda maneira, o receptor e o transmissor já são conhecidos e fixos. Em ambos os casos, apesar do fluxo de dados ser a cada instante em um direção (primeira maneira) ou sempre em uma direção (segunda maneira), é necessário um canal de comunicação de dois caminhos para troca de informações de controle. Isto significa que o transmissor e o receptor devem trocar comandos para alcançar um acordo antes de começar a transmitir dados, e realizado o acordo, ambos precisam de artifiícios capazes de garantir que os dados foram entregues intactos, ou seja, sem erros. Um dos principais artifícios utilizados para garantir a transferência de dados, são informações de controle trocadas entre o transmissor e o receptor.

Todos os procedimentos, regras e formatos citados definem um protocolo. Assim, o protocolo serve para garantir o bom uso do canal de comunicação. Segundo [Hol91], um protocolo pode conter regras e métodos para:

- Inicializar e finalizar a troca de dados;
- Sincronização entre transmissores e receptores;
- Detecção e correção de erros de transmissão;
- Formatação e codificação de dados.

Os erros encontrados em transmissão de dados, dependem do meio de transmissão escolhido. Este meio pode inserir, apagar, distorcer, duplicar e reordenar mensagens. Assim, é necessário desenvolver uma estratégia de controle de erros no protocolo.

Em [Hol91] é proposto um modelo de especificação para protocolos composto de cinco partes disintas:

- O serviço a ser fornecido pelo protocolo;
- As características do ambiente no qual o protocolo será executado;
- O vocabulário de mensagens usado para implementar o protocolo;
- O formato de codificação utilizado para cada mensagem do protocolo;
- As regras de procedimento que garantem a consistência da troca de mensagens.

Os modelos de comunicação, seja entre máquinas ou módulos de um sistema, sugerem um modelo hierárquico de modelagem e implementação, onde o processo de comunicação exige, não apenas um, mas vários protocolos para especificar as regras de comunicação entre as partes (camadas) deste modelo. O modelo mas citado na literatura, é o modelo de sete camada OSI (Open Systems Interconection) [DZ83] e sua variações, como o modelo TCP/IP [Bla95], amplamente utilizado durante este trabalho. Uma comparação entre o Modelo OSI e o Modelo TCP/IP pode ser encontrada em [Tan97].

Capítulo 3

Descrição do Sistema: *SmartMine*

3.1 Introdução

O SmartMine é uma solução integrada de hardware/sofware para melhoramento da produtividade e qualidade em minas de céu aberto, e encontra–se em desenvolvimento na empresa Devex Tecnologia e Sistemas, localizada em Belo Horizonte–MG. Este sistema armazena e trata cada viagem realizada pelos caminhões, a fim de controlar toda a movimentação de material, seja este minério ou estéril (material fora da especificação física ou química aceitável na usina de tratamento de minério). Ou seja, deseja–se saber entre outras questões, a origem e o destino do material, qual caminhão realizou a viagem, quanto tempo gastou, qual a massa produzida, etc. Esses dados são extremamente importantes para gerar resultados como a produtividade de cada equipamento, a qualidade de cada depósito, tempo ocioso de operação, entre outros. Assim, o SmartMine tem como finalidade, otimizar as operações das minas de céu aberto, através da análise de variáveis concorrentes como a qualidade do minério e a quantidade a ser produzida, evitando–se fila nos carregamentos e basculamentos, diminuindo o tempo ocioso dos equipamentos.

Os módulos do SmartMine foram projetados para receber informações a todo instante dos equipamentos da mina. Esses equipamentos podem ser caminhões tradicionais, caminhões fora–de–estrada, escavadeiras, carregadeiras e até mesmo equipamentos terceirizados. O SmartMine interage com esses equipamentos, através de um computador de bordo instalado em cada equipamento. Através deste computador, o operador do equipamento recebe qual a sua próxima ação, ou seja, qual o novo ponto de basculamento ou carregamento. O operador pode receber também mensagens de texto. É função deste computador de bordo também, in-

formar a posição de cada equipamento, em um intervalo de tempo configurável, através de um módulo GPS (Global Position System). Adicionalmente neste computador de bordo, indica–se cada estado do equipamento, a fim de registrar o tempo de cada atividade e a produção de cada equipamento. Dessa maneira, o operador deve indicar qual a atividade está sendo executada no momento, ou caso esteja parado, indicar qual o motivo de parada. Alguns estados são obtidos automaticamente pelo computador de bordo como o estado basculando, obtido do sensor de báscula, ou o estado movimentando cheio, obtido pelo deslocamento posterior ao carregamento, entre outros. Os demais estados devem ser indicados pelo o operador como fila, atolamento, refeição, troca de turno, entre outros diversos estados possíveis. Este computador de bordo recebeu o nome de *Tracker SMT 100*.

Com o objetivo de garantir o tratamento das informações obtidas o SmartMine foi projetado em cinco módulos:

- Módulo de Planejamento;
- Módulo de Despacho;
- Módulo GPS;
- Módulo de Consulta;
- Módulo de Otimização.

O módulo de planejamento é a interface entre os departamentos de planejamento e operação das minas. Neste módulo os usuários podem incluir caminhões, escavadeiras, carregadeiras, britadores, pilhas de estéril, estoques, importar áreas de escavação do plano de curto prazo da mina, definir o mapa da mina, etc. Todas as informações são armazenadas em um servidor de banco de dados *Oracle*.

O módulo de despacho representa o mundo real, onde todos os eventos de fato ocorrem. Ele é responsável pela captura da estimulação do mundo real, obtidas pelo *Tracker SMT 100* que transmite todos os eventos ocorridos no equipamento, bem como sua posição no globo através de um sistema *GPS* instalado no equipamento. Todas as decisões de operação da mina são enviadas para os equipamentos através do módulo de Despacho.

O módulo GPS, é composto pelo *Access Point SMT 100*, que funciona como um gateway de comunicação entre o módulo de despacho e os *Trackers SMT 100* dos equipamentos. A descrição desta integração é o foco principal deste trabalho.

O módulo de otimização recebe a todo instante os eventos da mina do módulo de despacho. Assim este módulo pode realizar a simulação de operação da mina

nas próximas horas, resultando em uma sugestão da próxima ação do operador do módulo de despacho.

O módulo de consulta é uma replica da interface do módulo de controle. Assim o usuário pode visualizar tudo o que está ocorrendo na mina, mas não possui as funções de realizar comandos na operação. Este módulo pode ser acessado pelos usuários autorizados através da *WEB*, onde podem ser realizadas diversas consultas no sistema. Este módulo interage com o módulo de despacho através da tecnologia *COM/DCOM*.

A comunicação entre os módulos ocorre com a utilização de três tecnologias diferentes:

- Comunicação sem fio (Rede Wireless) [Nem95], [kF95];
- Comunicação com fio
  - via rede ethernet 10/100 Mbps utilizando o protocolo TCP/IP [Bla95];
  - via tecnologia COM/DCOM [Mic96].

Além da interação hardware/software evidenciada na comunicação entre os módulos, outro ponto crítico deste sistema é uma grande massa de dados a ser tratada e armazenada, resultando em um fluxo contínuo de informações a cada segundo, durante as 24 horas do dia. Este fato exigiu bastante atenção durante o projeto do sistema, pois o SmartMine não pode "esperar" pelo processamento do banco de dados enquanto novas informações chegam a todo instante.

Esta característica assíncrona do SmartMine, evidencia uma das principais características de sistemas em tempo real, de acordo com [Rip89]. Segundo [Rip89]:

> *Uma propriedade fundamental de um programa em tempo real é que algumas, ou todas as suas entradas, chegam do mundo exterior de forma assíncrona, em relação a qualquer trabalho que o programa já está fazendo. O programa deve ser capaz de interromper sua atividade corrente imediatamente e então executar algum código predefinido para capturar ou responder a essa entrada, que é freqüentemente um sinal volátil, temporário.*

Dessa maneira, o SmartMine pode ser considerado um sistema em tempo real, pois seus módulos se interagem de forma assíncrona para satisfazer, em tempo real, as funcionalidades desta solução. Um exemplo da importância do fator tempo real para este sistema é a capacidade do SmartMine de mostrar a posição real de

cada equipamento que esteja na área de cobertura de rádio e que possua um dispositivo GPS (Global Position System) instalado. Esta posição é transmitida em intervalo de tempo de no máximo 4 segundos, dependendo do número de equipamentos transmitindo e da quantidade de "eventos" a serem transmitidos por cada um. Assim, o usuário pode visualizar em sua tela a posição real dos equipamentos e tomar decisões a partir desta informação.

3.2 Topologia Física

A topologia física do sistema é composta por uma rede *Wireless* que utiliza a tecnologia *spread-spectrum* do tipo *frequency hoping* [kF95] conectada à rede *ethernet*. Dessa maneira, todos os módulos do *SmartMine* estão conectados através de uma rede *ethernet* 10/100 Mbps. Os módulos instalados nos equipamentos comunicam com o *SmartMine* através da rede *wireless*. Nesta rede, sinais de rádio são transmitidos para as estações base (*base station*), que podem ser de longo ou curto alcance, que por sua vez, estão conectados aos *Access Points* que funcionam como um *gateway* entre as estações de rádio, ou outros equipamentos, e o *SmartMine*. O *Access Point* está conectado à rede *ethernet* e comunica via *TCP/IP* com o *SmartMine*.

O *SmartMine* pode receber informações de quantos *Access Points* forem necessários, e estes podem ser colocados em qualquer lugar da mina que possua uma conexão *ethernet*. As estações base podem ser utilizadas *stand-alone* (sem um *Access Point*) como uma repetidora para aumentar a cobertura da rede *wireless*.

A Figura 3.1 exemplifica a topologia física do SmartMine.

3.3 Topologia Lógica

O fluxo das informação do sistema, visualizado na Figura 3.2, pode ser descrito da seguinte maneira:

Fluxo A: representa um caminhão equipado com o *Tracker SMT 100* com rádio–modem de longo alcance se comunicando com a estação de rádio ligada à rede. Os tipos de informações encontrados neste fluxo fazem parte da próxima seção.

Fluxo B: representa uma escavadeira equipada com o *Loader SMT 100* com rádio–modem de longo alcance se comunicando com a estação de rádio ligada à rede. Não somente dados dos próprios equipamentos de carga, mas dos caminhões que possuem rádio de curto alcance são mandados para a rede através deste fluxo.

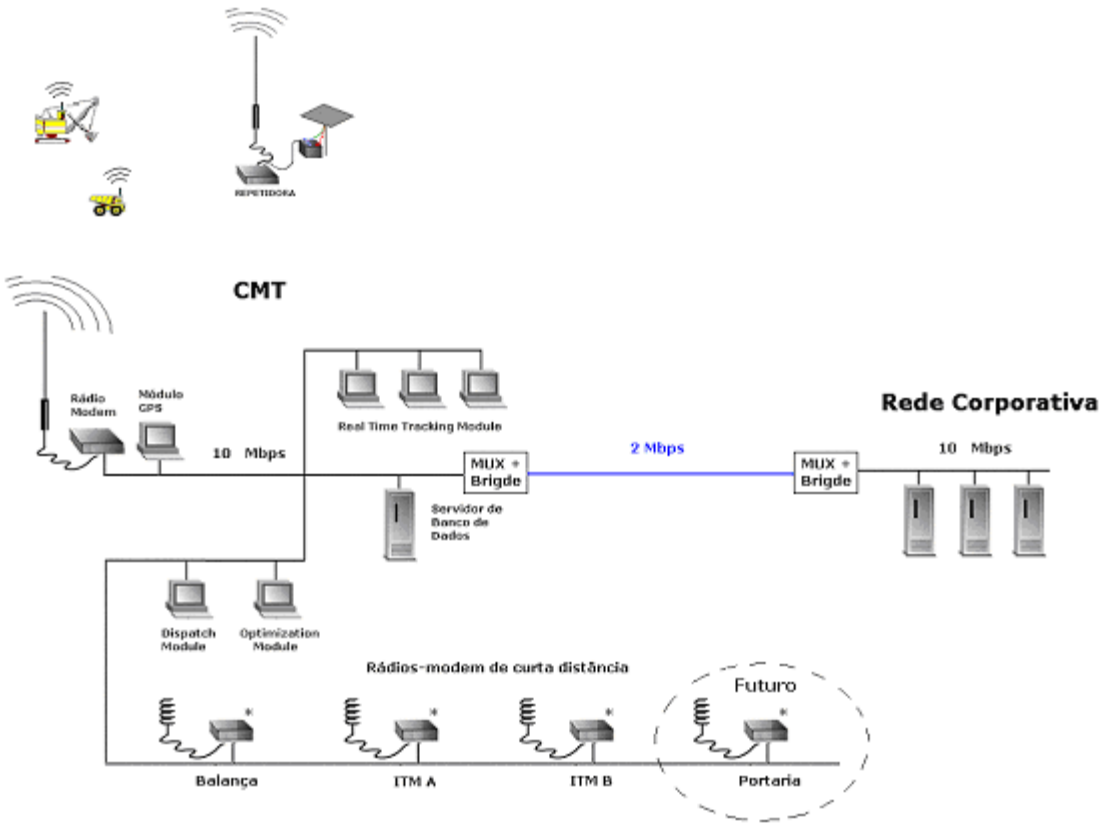

Figura 3.1: Diagrama da rede física utilizada na mina de Capitão do Mato - MBR

Os tipos de infomações encontrados neste fluxo, também fazem parte da próxima seção.

Fluxo C: representa um caminhão com rádio de curto alcance transferindo os dados do *buffer* de seu equipamento para o equipamento de carga, que por sua vez, irá tranmití–los para o Módulo de Despacho através da estação de rádio.

Fluxo D: representa um caminhão com rádio de curto alcance transferindo os dados do *buffer* de seu equipamento para a rede, através de uma base de curta distância.

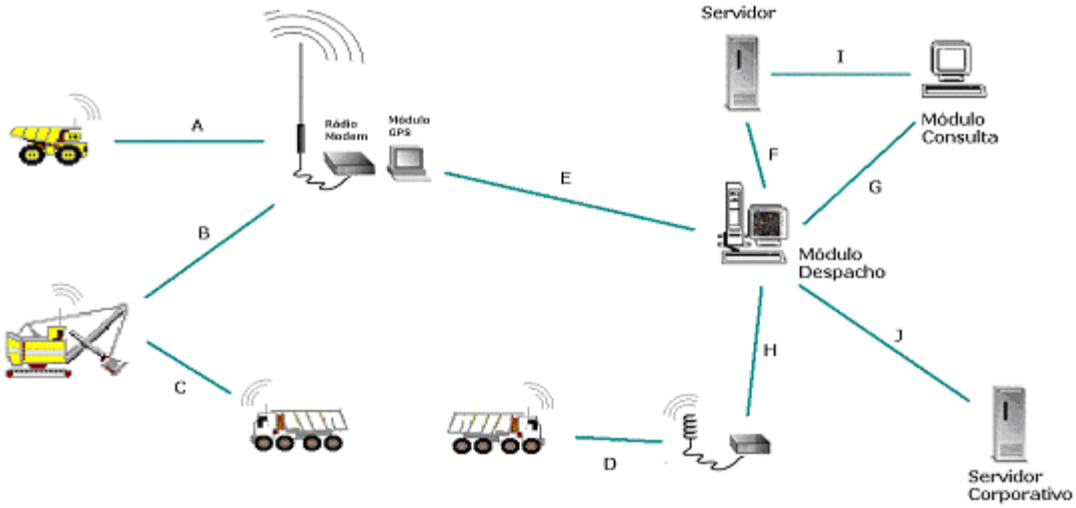

Figura 3.2: Diagrama da rede lógica utilizada na mina de Capitão do Mato

Fluxo E: representa a troca de dados de todos os pacotes de dados (posição dos equipamentos, mudança de estados, etc, entre o Módulo de GPS e o Módulo de Despacho. O Módulo de GPS funciona como regulador do fluxo de informações entre os diversos equipamentos e o Módulo de Despacho.

Fluxo F: atualiza o Banco de Dados Oracle específico do SmartMine.

Fluxo G: atualiza na estação de um usuário, através do Módulo de Consulta e em tempo real, informações como posição dos equipamentos, produtividade, utilização, produção, entre outras.

Fluxo H: representa o recebimento de informações colhidas dos caminhões equipados com rádio de curto alcance e envio de mensagens, como ordens de carregamento/basculamento.

Fluxo I: consultas realizadas na base de dados do servidor Oracle.

Fluxo J: interface com os sistemas corporativos da empresa cliente.

Capítulo 4

Protocolo de Comunicação Hardware/Software

4.1 Introdução

Este capítulo concentra–se na integração entre o Módulo de Despacho e o Módulo GPS do sistema em tempo real SmartMine, descrito na seção anterior. De acordo com os modelos básicos de Codesign, esta seção define a etapa de integração HW/SW, mais especificamente, a comunicação entre o firmware (software embarcado dos *Trackers SMT 100*) e o software (Módulo de Despacho). Dessa maneira, define–se um conjunto de regras para a comunicação entre o Módulo de Despacho e os *Access Points*.

Como descrito anteriormente, o objetivo da comunicação entre esses módulos é a troca de informações entre os equipamentos da mina, ou seja, caminhões e escavadeiras e o módulo de Despacho, que recebe as informações dos equipamentos e também envia comandos para estes. No entanto, existe um caminho intermediário devido aos diferentes meios de transmissão envolvidos nesta comunicação: os *Access Points*. Este harware funciona como um *gateway* entre os equipamentos e o Módulo de Despacho.

Para uma melhor visualização do processo de comunicação, a Figura 4.1 evidencia quais camadas do Modelo de Referência OSI [DZ83] estão envolvidas na comunicação entre o Módulo de Despacho e os *Access Points* e entre os *Access Points* e os equipamentos.

De acordo com a Figura 4.1, a comunicação B utiliza as camadas de transporte, rede, enlace de dados e física, enquanto a comunicação A se concentra nas

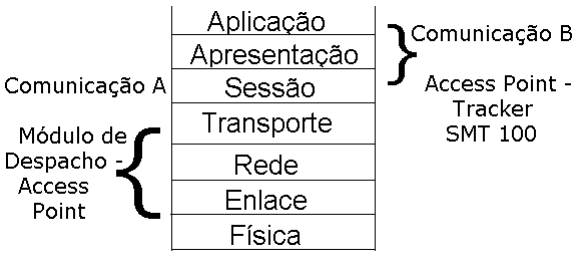

Figura 4.1: Integração do SmartMine baseado no modelo de Referência OSI

camadas de sessão, apresentação e aplicação. No entanto, o protocolo utilizado na comunicação entre os módulos é o TCP/IP [Bla95], e como este não possui as camadas de sessão e nem a camada de apresentação, o protocolo de comunicação entre o Módulo de Despacho e os *Access Points* concentra–se na camada de aplicação.

A seguir, define–se o protocolo entre a camada de aplicação e a camada de transporte, segundo o modelo de referência TCP/IP.

4.2 Modelagem

A camada de aplicação possui funções responsáveis por identificar comunicação, determinar disponibilidade de recursos e sincronizar a comunicação. Ao identificar uma possível comunicação, esta camada determina a identidade e disponibilidade do transmissor ou receptor de uma aplicação com dados a transmitir. Ao determinar disponibilidade de recursos, esta camada deve decidir se existem recursos de rede suficientes para que a requisição de comunicação possa existir. Na sincronização da comunicação, todos os componentes entre as aplicações requerem cooperação que é gerenciada pela camada de aplicação. Alguns exemplos de protocolos encontrados neste camada segundo [Tan97] são: TELNET, FTP, SMTP, DNS, entre outros.

Tendo em vista algumas das funcionalidades da camada de aplicação, o protocolo da camada de aplicação do SmartMine define as regras de comunicação entre a camada de transporte, última camada do *Access Point* e a aplicação, ou seja, o Módulo de Despacho.

4.2.1 Ambiente de Execução

A modelagem de um protocolo depende muito do ambiente de execução e da tecnologia utilizada para a transmissão e recepção dos dados. Conhecendo as características do ambiente, as regras de comunicação devem ser consistentes e fornecer métodos para garantir o "bom uso" do canal de transmissão. As mensagens trocadas durante a comunicação podem sofrer distorções, podem ser duplicadas, alteradas, apagadas e podem chegar fora de ordem.

Dessa maneira, dependendo do meio de transmissão, o protocolo deve conter estratégias para identificação e correção de erros, tratamento de redundâncias, controle de fluxo, congestionamento, roteamento, entre outras.

O protocolo proposto neste capítulo, possui dois ambientes de execução distintos, com e sem fio. O problema inicial, era fazer com que o Módulo de Despacho se comunicasse com os *Trackers* dos equipamentos nas minas. Como os *Trackers* e o Módulo de Despacho estão em redes diferentes, a solução foi a criação de um *gateway* entre esses dois módulos, capaz de reconhecer os dois ambientes de execução do protocolo de comunicação. Assim, o protocolo de comunicação modelado para o SmartMine é o mesmo, tanto para a comunicação Módulo de Despacho com os *Access Points* e destes com os *Trackers*. O *gateway* apenas repassa as mensagens de uma rede para outra, sem realizar nenhum tipo de processamento com a mensagem.

O fato do protocolo de comunicação ser o mesmo para ambos os ambientes de execução, implica em modelar o protocolo de um maneira tal, que possa ser confiável e robusto pois, redes com e sem fio, possuem características particulares que devem ser consideradas. Ao longo deste capítulo, os problemas encontrados para se modelar o protocolo para diferentes ambientes de execução são evidenciados, bem como a solução adotada.

A comunicação entre o Módulo de Despacho e os *Access Points* é realizada através de uma rede *ethernet* 10/100 Mbps baseada no protocolo TCP/IP. Segundo [Tan97] , dois protocolos fim a fim foram definidos na camada de transporte do modelo de referência TCP/IP. O primeiro deles, o TCP (Transmission Control Protocol), é um protocolo orientado à conexão confiável que permite a entrega sem erros de um fluxo de bytes originado de uma determinada máquina. Esse protocolo fragmenta o fluxo de bytes de entrada em mensagens e passa cada uma delas para a camada inter–rede. No destino, o processo TCP remonta as mensagens recebidas no fluxo de saída. O TCP cuida também do controle de fluxo, impedindo que um transmissor rápido sobrecarregue um receptor lento com um volume de mensagens muito grande. O segundo protocolo é o UDP (User Datagram Protocol), é

um protocolo sem conexão e não confiável para aplicações que não necessitam de controle de fluxo, nem da manutenção da sequência das mensagens enviadas.

Devido às características do sistema SmartMine, o protocolo de camada de transporte adotado para a comunicação entre o Módulo de Despacho e os *Access Points* foi o TCP. A necessidade de ordenação das mensagens é fundamental para o funcionamento do SmartMine, pois os equipamentos executam tarefas em seqüência. Por exemplo, os caminhões carregam, movimentam cheios, basculam e movimentam vazios em um ciclo normal de trabalho. A garantia da ordenação das mensagens facilita o trabalho da aplicação, que recebe as informações e apenas as insere no banco de dados pela ordem de chegada.

O controle de fluxo do protocolo TCP também é fundamental, devido à possibilidade de comunicação com vários *Access Points*. Assim, não faz parte da aplicação controlar a velocidade de transmissão dos *Access Points*, resultando em menos processamento pela aplicação. O protocolo TCP unido ao protocolo IP (Internet Protocol), facilita mais um aspecto, que é a identificação dos *Access Points* e a gerência das conexões com o Módulo de Despacho.

A comunicação entre os *Access Points* é realizada por uma rede *Wireless* que utiliza a tecnologia *spread-spectrum* do tipo *frequency hoping* [Nem95]. Segundo [Nem95], os principais problemas a serem considerados em uma comunicação sem fio são: baixa vazão (*troughput*), latência, pacotes perdidos, duplicados ou fora de ordem, e custo. As decisões de projeto, bem como a implementação da rede *wireless*, são temas para um próximo trabalho.

4.2.2 Identificação dos Transmissores e Receptores

A definição das regras de comunicação, dependem do modelo de comunicação, ou seja, quem são os transmissores e quem são os receptores. A comunicação entre o Módulo de Despacho com os *Trackers* deve passar primeiramente, pelo *gateway*, ou seja, pelo *Access Point*. No entanto, de acordo com a topologia física e lógica do SmartMine citada no Capítulo 3, a comunicação entre os *Trackers* e o Módulo de Despacho, pode ser realizada por mais de um *gateway*, o que significa que podem existir vários *Access Points* estrategicamente localizados nas minas.

Tendo em vista que o Módulo de Despacho deve estar preparado para se comunicar com quantos *Access Points* forem necessários, o problema agora é como gerenciar e identificar cada *Access Point*, de modo a garantir a eficiência do protocolo de comunicação. A solução encontrada, baseou–se no paradigma cliente–servidor para gerenciar todos os *gateways* a partir do Módulo de Despacho. A seguir, especifica–se detalhes desta solução.

Na comunicação entre o Módulo de Despacho e os *Access Points*, ambos podem assumir a posição de transmissores ou receptores. Assim, o Módulo de Despacho assume a posição de servidor e pode ter quantos clientes, *Access Points*, forem necessários. A Figura 4.2 exemplifica esse processo.

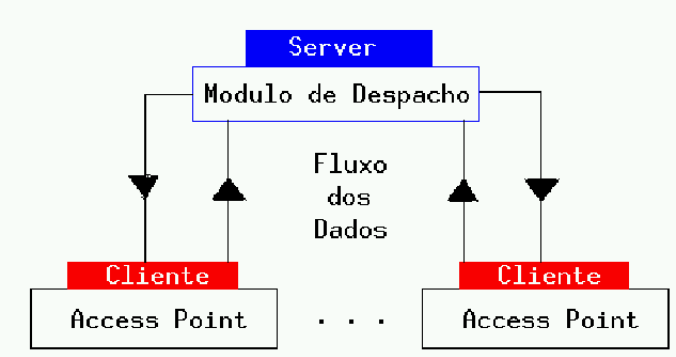

Figura 4.2: Modelo Cliente–Servidor

Assim como no modelo cliente–servidor, o Módulo de Despacho não conhece o endereço (IP) de cada *Access Point* na rede *ethernet* até que este solicite conexão, o que significa que o Módulo de Despacho deve possuir um endereço IP fixo na rede, ou que pelo menos, a máquina em que este módulo é executado esteja registrada no DNS do servidor de rede.

Dessa maneira, parte do *Access Point* a iniciativa de solicitar uma conexão para transmitir seus dados, e cabe ao Módulo de Despacho validar a conexão e gerenciá–la, tomando as devidas decisões na presença de erros. O *Access Point* foi modelado para tentar estabelecer uma conexão com o servidor em um intervalo de tempo configurável. Ao conseguir uma conexão, mesmo que não tenha dados a serem enviados, a conexão permanece até que o servidor encerre a conexão, ou que o *Acess Point* "desconfie" que a conexão está com problemas e tente se reconectar. No melhor dos casos, cada *Access Point* se conecta apenas uma vez e mantém sua conexão.

A identificação de cada *Access Point* pelo Módulo de Despacho é realizada através de um número inteiro, único por *Access Point*, fornecido por este ao conectar–se ao módulo. Assim, cada número é associado a uma conexão. Cada conexão é um objeto do tipo "*socket*" implementado pelos sistemas operacionais, e que guarda informações como o endereço de rede de cada *Access Point*. O Mó-

dulo de Despacho gerencia apenas estes objetos "*socket*", deixando o trabalho de transmissão e recebimento de dados dos *Access Points* certos, para o sistema operacional.

4.2.3 Modelando *TIMEOUTS*

O fato da solicitação de conexão partir do *Access Point*, implica em desenvolver procedimentos capazes de identificar que a conexão está com problemas e tomar a decisão de terminar a conexão ou solicitar outra conexão. Quanto ao lado do servidor, que recebe o pedido de conexão, cabe a ele decidir também se a conexão está com problemas e tomar as devidas providências, na maioria das vezes, encerrando–se a conexão.

Um recurso bastante utilizado na modelagem e implementação de protocolos, é o *TIMEOUT*. Segundo [Hol91], o *TIMEOUT* permite um processo abortar o estado de espera por uma condição que não se sabe quando tornará verdadeira. Este recurso fornece uma "fuga" de um estado duvidoso.

A modelagem de um *TIMEOUT* na camada de aplicação do protocolo, surgiu como um solução para o seguinte problema: o Módulo de Despacho deixava para o sistema operacional, a tarefa de identificar se uma conexão já não está mais válida, ocasionando um atraso crucial para um sistema em tempo real como o SmartMine. O serviço oferecido pelo protocolo TCP na camada para identificar falha na conexão, não era compatível com a necessidade, pois o mesmo demorava minutos para perceber o problema.

Assim, na tentativa de solucionar o problema da detecção de encerramento da conexão de forma mais rápida, o *TIMEOUT* de conexão foi modelado e implementado utilizando–se uma mensagem denominada *WatchDog*. Essa mensagem, como o próprio nome diz, serve para verificar se a conexão continua válida. No protocolo de comunicação descrito nesta seção, tanto os clientes *Access Points* quanto o servidor Módulo de Despacho, enviam uma mensagem *WatchDog* se algum deles perceber através de um intervalo de tempo *t*, que nenhuma mensagem foi enviada. Assim, se o cliente ou o servidor forem o receptor e perceber que nenhuma mensagem chegou pela conexão em um intervalo de tempo *t*, a conexão é encerrada pelo lado que percebeu o *TIMEOUT*. Assim cabe ao cliente solicitar nova conexão. Neste caso, sempre o *Access Point*.

A implementação do *TIMEOUT* é importante em um sistema de tempo real, pois perde–se menos tempo tentando enviar dados em uma conexão com problemas. No entanto, é importante frisar que a implementação de uma conexão em um sistema operacional já possui seu próprio procedimento *TIMEOUT* implementado

na camada de transporte do protocolo TCP/IP. No entanto, este *TIMEOUT* é um tanto quanto longo para um sistema em tempo real, o que justifica a implementação de um *TIMEOUT* de conexão, através de uma mensagem *WatchDog*, na camada de aplicação do protocolo de comunicação. O formato da mensagem *WatchDog* será definido mais adiante.

4.2.4 Codificação das Mensagens

Antes de iniciar a definição das mensagens, uma etapa importante na especificação de um protocolo é a modelagem da formatação ou codificação das mensagens, ou seja, como o transmissor e o receptor devem tratar as mensagens, a fim de obter sincronização e uma transferência sem erros dos dados.

A formatação das mensagens deve ser definida antes de qualquer estrutura de mais alto nível. De acordo com [Hol91], existem três métodos principais de formatação:

- Orientado a bit;
- Orientado a caracter;
- Orientado a bytes.

Um protocolo orientado a bit transmite dados em um fluxo de bits. Para permitir que um receptor reconheça onde uma mensagem começa e termina no fluxo de bits, um pequeno conjunto de pares únicos de bits, ou *flags*, são usados. No entanto, estes pares de bits podem fazer parte dos dados também, então alguma coisa deve ser feita para garantir que eles sejam interpretados apropriadamente. Assim, um fluxo de bits pode ser formatado como *flag* + dados + *flag*. Por exemplo, 0 1 1 1 1 1 1 0 | 0 1 1 1 1 0 1 0 1 0 1 | 0 1 1 1 1 1 1 0. No exemplo, uma sequência de seis 1 iniciada e terminada por 0 significa um *flag*. Assim, o receptor possui agora uma maneira de encontrar os dados, sabendo que estes estão delimitados por *flags*.

Por sua vez, em um protocolo orientado a caracter, apenas algumas poucas estruturas são focadas no fluxo de bits. Se o número de bits por caracter é fixo, toda a comunicação é codificada em múltiplos de *n* bits. Estas unidades de dados são utilizadas para codificar dados e códigos de controle. Exemplos de delimitadores de mensagens, são os caracteres *STX* e *ETX*, que significam começo de texto e fim de texto, respectivamente.

O protocolo orientado a bytes, utiliza as estruturas de *flag* e caracteres de controle para estruturar um longo fluxo de dados em fragmentos maiores. A diferença

desta formatação para as demais, consiste em um campo após o *STX*, que indica o número de bytes que a mensagem contém e que, consequentemente, devem ser lidos pelo receptor. Essa formatação, no entanto, não descarta campos de correção de erros e controle de fluxo, mas é a formatação mais utilizada na codificação das mensagens de protocolos desenvolvidos nas aplicações atuais, segundo [Hol91].

A codificação das mensagens do protocolo de comunicação entre o Módulo de Despacho do SmartMine e os *Access Points*, foi modelada com base na codificação orientada a bytes, devido à facilidade de implementação e robustez deste método. Na seção 4.2.5, especifica–se as mensagens, onde são utilizados caratecteres de controle, como o *STX* e campos de controle de fluxo.

4.2.5 Formato das Mensagens

Nesta seção, o formato das mensagens trocadas entre os clientes (*Access Point*) e o servidor (Módulo de Despacho) é especificado.

Geralmente, as mensagens requerem no mínimo, um campo indicando o tipo da mensagem e um campo de dados opcional. No entanto, para se implementar regras de controle de fluxo, as mensagens devem conter números de seqüência, e para se implementar regras de controle de erro, um campo *checksum*.

As mensagens do protocolo de comunicação *Access Point*–Módulo de Despacho são encapsuladas pelo protocolo TCP/IP, utilizado na rede *ethernet*, ambiente de execução deste protocolo. Segundo [Tan97], como o TCP/IP é um protocolo orientado à conexão, com garantia de entrega, controle de fluxo e correção de erros, a formatação das mensagens do protocolo de comunicação do SmartMine não "carregam" campos como *checksum* e identificador de mensagens. No entanto, algumas mensagens específicas possuem um campo identificador, o qual será justificado mais adiante.

As mensagens são formatadas como uma seqüência de bytes, uma vez que o canal de transmissão é orientado a bytes. Como visto anteriormente, mensagens formatadas com orientação a bytes requerem um campo que indique o tamanho da mensagem. Tendo em vista as características evidenciadas até o momento, o formato das mensagens do protocolo de comunicação aqui descrito, pode ser visualizado na Figura 4.3.

O campo *STX* (start of text) indica o início da mensagem e possui sempre o valor 2. Este valor foi escolhido aleatoriamente para significar início de mensagem. Portanto este campo possui apenas 1 byte. O campo *CMD*, abreviação de *Comando*, indica qual o tipo da mensagem e possui também, apenas 1 byte. Este campo indica se a mensagem é um *WatchDog*, uma mensagem de dados, uma

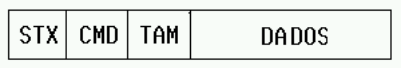

Figura 4.3: Formato da Mensagem

mensagem de controle, entre outras. O campo *TAM*, abreviação de *Tamanho*, indica a quantidade de bytes a serem lidos do campo *Dados*. Assim, o cabeçalho de qualquer mensagem trocada entre o *Access Point* e o Módulo de Despacho, devem conter estes três campos nesta ordem, o que significa que toda mensagem possui um *overhead* de três bytes.

O campo *Dados* é um campo de tamanho variável em bytes e pode ser opcional, ou seja, mensagens de controle não possuem este campo. Os dados trocados no protocolo do SmartMine, serão evidenciados na Seção “Vocabulário do Protocolo”.

Analisando melhor a formatação da mensagem aqui proposta, pode–se perceber que a mensagem não possui o campo *ETX* (End of Text), comumente utilizado nas mensagens para significar fim de mensagem. A não utilização do campo *ETX* deve–se ao fato da orientação a bytes da mensagem. Como o campo *TAM* indica quantos bytes devem ser lidos do *buffer* a cada mensagem que chega, a informação de fim de mensagem já é obtida, ou seja, assim que o receptor encerrar a leitura de bytes especificada pelo campo *TAM*, ele já pode começar a ler o próximo *STX*.

No entanto, pode parecer que o campo *STX* deveria seguir as mesmas justificativas do campo *ETX* e não ser utilizado. Isto significa que o início de mensagem poderia ser obtido pela leitura do campo *CMD* para se obter o tipo da mensagem, pois a leitura da mensagem pelo receptor do *buffer* de mensagens do protocolo TCP/IP parece estar sincronizada, pois o receptor sempre “sabe” a quantidade de bytes a serem lidas. Dois bytes de cabeçalho (*CMD* + *TAM*) e mais *n* bytes de dados. Apesar do protocolo TCP/IP garantir a sincronização entre receptor e transmissor, a leitura de apenas um byte a mais no *buffer* de mensagens significa perda de sincronismo, ou seja, os dados não seriam mais relevantes. Assim, a implementação do campo *STX* como um campo de início de mensagem e sincronismo, mostrou ser mais eficiente do que a implementação sem este campo, e pareceu ser mais confiável, sendo eficaz em garantir a leitura certa do formato das mensagens, mesmo com a falha de alguma regra do protocolo.

4.2.6 Tamanho Otimizado da Mensagem

Como especificado anteriormente, o Módulo de Despacho transmite um fluxo de bytes, encapsulado pelo protocolo TCP/IP, para os *Access Points* e vice–versa. Como o protocolo TCP/IP já possui uma implementação de controle de fluxo através de *buffers* no receptor e no transmissor, a divisão das mensagens em seu tamanho ideal de transmissão, já é realizada automaticamente. No entanto, os *Access Points* apenas repassam as mensagens que chegam do Módulo de Despacho para os *Trackers SMT 100*, o que significa que as mensagens recebidas pelos *Access Points* passarão para um ambiente de transmissão *wireless*, que utiliza a tecnologia *spread-spectrum* do tipo *frequency hoping*.

Na tecnologia *spread-spectrum* do tipo *frequency hoping*, as mensagens são divididas em pacotes, que são enviados a cada "*hop*" (intervalo de tempo), em uma freqüência (canal) diferente. Neste tipo de tecnologia baseada em *wireless*, a definição do tamanho ideal dos pacotes é extremamente importante, pois evita que haja desperdício de transmissão, ou seja, mensagens muito grandes ou muito pequenas, que ao serem divididas em pacotes para poderem ser enviadas, geram pacotes com poucos dados, desperdiçando a banda de transmissão e gerando um maior número de pacotes (em caso de pacotes grandes). Por exemplo, se a cada *hop*, é transmitido apenas 50 bytes, se for definido uma mensagem de tamanho 54 bytes, esta mensagem deverá ser dividida em dois pacotes de tamanho 50, havendo um desperdício de 46 bytes no segundo pacote.

Segundo a especificação do rádio–modem [Mic00], utilizado na comunicação *wireless* deste projeto, o tamanho mínimo da mensagem deve ser de 1 byte e o tamanho máximo de 255 bytes. No entanto, 1 ou 255 bytes é de longe o tamanho ideal da mensagem. De acordo com a especificação [Mic00], existe um tamanho ideal de mensagens a serem transmitidas. Este valor depende do intervalo de *hop* escolhido, ou seja, a frequência que os rádio–modems trocam de canal de transmissão. Assim, para cada intervalo de *hop*, existe um tamanho ideal de mensagem.

A Tabela 4.1, indica o tamanho ideal de mensagens para cada intervalo de *hop*. O intervalo escolhido neste trabalho, foi a opção 6, com intervalo de *hop* de 45 milisegundos por apresentar melhores resultados empíricos. Assim, todas as mensagens foram modeladas levando em consideração o tamanho ideal, mantendo–se uma relação entre o mínimo e o máximo a ser enviado. Nenhuma mensagem ultrapassou o valor ideal, mas a mensagem de menor tamanho ocupou apenas 33 bytes do tamanho ideal.

Tabela 4.1: Taxa ideal de transmissão de acordo com o intervalo de hop [Mic00]

$Intervalo - do - Hop$	$Tamanho - Otimo - do - Pacote - (bytes)$	$Throughput - Kbps$
$1(8ms)$	N/A	N/A
$2(12ms)$	3	1
$3(16ms)$	22	13
$4(20ms)$	43	21
$5(30ms)$	93	31
$6(45ms)$	174	38
$7(80ms)$	255	31
$8(120ms)$	255	21

4.2.7 Vocabulário do Protocolo

Para que o protocolo realize suas funções, ele precisa comunicar–se com seu ambiente. O protocolo de comunicação entre os *Access Points* e o Módulo de Despacho, funciona como uma caixa preta, que recebe mensagens dos *Access Points* vinda dos equipamentos das minas (caminhões e escavadeiras) e recebe mensagens do Módulo de Despacho para os equipamentos das minas. Assim, existem três tipos de mensagens definidas neste protocolo:

- Mensagens comuns aos *Access Points* e Módulo de Despacho;
- Mensagens exclusivas do Módulo de Despacho;
- Mensagens exclusivas dos equipamentos da mina, lembrando que o *Access Point é apenas o Gateway entre o Módulo de Despacho e os equipamentos*.

As mensagens comuns aos *Access Points* e o Módulo de Despacho, são mensagens que ambos os lados devem utilizar para gerenciar a conexão TCP/IP. Assim, estas mensagens não foram modeladas no protocolo e nem implementadas, o que significa que provavelmente elas não seguem o formato especificado na Figura 4.3. Estas mensagens foram obtidas das bibliotecas do protocolo TCP/IP. As bibliotecas TCP/IP utilizadas neste protocolo, possuem uma ampla quantidade de mensagens. No entanto, somente as mensagens realmente utilizadas serão especificadas.

Ao iniciar a conexão, a primeira mensagem a ser utilizada é a mensagem *connect*. Esta mensagem é enviada de cada cliente *Access Point* para o servidor Módulo de despacho. Assim, esta mensagem é utilizada somente pelo *Access Point*, para solicitar conexão.

A mensagem utilizada pelo Módulo de Despacho para aceitar a requisição de conexão, é a mensagem *accept*. Esta mensagem é uma confirmação da solicitação de conexão para os clientes e, portanto, utilizada somente pelo Módulo de Despacho.

As mensagens *send* e *receive* são utilizadas por ambos para enviar e receber um fluxo de bytes. Outras mensagens, comuns aos clientes e ao servidor, são as mensagens utilizadas para encerrar a conexão (*close*) ou para abortar a conexão (*abort*), em caso de *TIMEOUT* e erro de conexão respectivamente.

A única mensagem de controle modelada e implementada no protocolo de comunicação do SmartMine, e já citada anteriormente, é a mensagem *WatchDog*. Esta mensagem é uma mensagem de controle utilizada quando o servidor ou o cliente não recebem alguma mensagem em um determinado instante de tempo. Esta mensagem serve para indicar ao receptor, que o transmissor continua "vivo" e a conexão continua confiável.

A mensagem de *WatchDog* é composta pelo cabeçalho da mensagem, ou seja, *STX, CMD, TAM* e a parte de dados possui 2 bytes que funcionam como identificação do *Access Point*. O campo *CMD* indica que a mensagem é um *WatchDog* e o campo *TAM* indica a quantidade de bytes a serem lidos, no caso 2 bytes.

As mensagens exclusivas do Módulo de Despacho funcionam como comandos para os equipamentos que possuem o *Tracker SMT 100*, computador de bordo dos caminhões e escavadeiras. Assim, o campo *Dados* visualizado na Figura 4.3, necessita ser dividido. Existe agora, um cabeçalho que deve acompanhar os dados de todas as mensagens de comandos para os *Trackers*, enviadas pelo *Access Point*. A Figura 4.4, evidencia a divisão do campo *Data* nas mensagens para os *Trackers*.

Figura 4.4: Divisão do Campo Dados para mensagens de comando

O primeiro campo do cabeçalho de dados, *EQT*, indica qual equipamento deverá receber o comando. O campo *RT*, significa rota e é um campo herdado da camada de rede do protocolo, pois é uma informação de roteamento. A presença deste campo na camada de aplicação é justificável. Como o *Access Point* funciona apenas como um *gateway*, ele não possui informações armazenadas sobre cada mensagem enviada ou recebida. Assim, esta informação da rota *wireless* indica qual "caminho" o próximo comando deve seguir para alcançar o equipamento, que está em algum ponto da rede *wireless* indicado pelo campo *RT*. Esta informa-

ção é obtida quando alguma mensagem chega ao Módulo de Despacho vinda de algum equipamento, e é adicionada à mensagem.

O campo *ID*, funciona como um identificador único da mensagem de comando enviada. Apesar do protocolo TCP/IP, utilizado na comunicação *Access Point* e Módulo de Despacho, ser um protocolo com confirmação e controle de fluxo, o campo identificador no comando de mensagem é necessário pois, o protocolo de comunicação entre o *Access Point* e os *Trackers* não é um protocolo com garantia de entrega. Assim, tanto o *Access Point* quanto o Módulo de Despacho, possuem *buffers*, usados para enviar as mesmas mensagens, até que elas sejam confirmadas. E caso recebam mensagens repetidas, elas podem ser também descartadas. Assim, a mensagem de comando, é uma mensagem que necessita de confirmação.

O campo *CMD* indica qual comando está sendo enviado na mensagem. Foram modelados 6 comandos iniciais. O comando *Despacho* despacha um veículo para um local de carregamento ou de basculamento. O comando *Exibe Peso*, manda para o equipamento o peso de sua carga, assim que este passa pela balança. O comando *Mensagem* envia uma mensagem de texto com 50 caracteres e o comando *Obtém Rota*, solicita ao equipamento que envie os pontos da rota que o equipamento fez para carregar e depois bascular, ou para bascular e carregar.

As mensagens exclusivas dos equipamentos que possuam um *Tracker SMT 100* para o Módulo de Despacho, são eventos que ocorrem nos equipamentos e precisam ser enviados ao SmartMine. A posição real de cada equipamento indicada pelo GPS, o estado atual do equipamento indicando o que este está fazendo a cada instante, entre outros, são apenas alguns dos eventos passados ao SmartMine.

O formato das mensagens dos eventos é o mesmo indicado na Figura 4.3, onde o campo *CMD*, indica que a mensagem é um evento. Assim como nas mensagens de comandos, as mensagens de eventos também necessitam da adição de um cabeçalho no campo *Dados*, comum a todos os eventos enviados ao SmartMine. A divisão do campo *Dados* em cabeçalho e dados, pode ser visualizada na Figura 4.5.

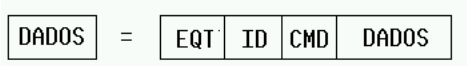

Figura 4.5: Divisão do Campo Dados para mensagens de evento

O campo *EQT* indica o endereço de qual equipamento deverá receber a mensagem de evento. O campo *ID*, é o identificador da mensagem, justificado pelo mesmo motivo demonstrado na mensagem de comando. O campo *CMD* indica qual evento está sendo enviado na mensagem.

O campo *Dados* contém as informações específicas de cada evento. No protocolo de comunicação entre o *Access Point* e o Módulo de Despacho, foram modelados 7 eventos iniciais. O evento *Mudança de Estado* é gerado sempre que um equipamento muda de estado, por exemplo, ao passar de movimentando cheio para basculando. Este evento indica o novo estado do equipamento através de um código único de 4 bytes e indica também a hora e posição (X,Y) onde a mudança de estado ocorreu.

O evento *Rota Cadastrada*, é enviado ao SmartMine sempre que o equipamento grava uma nova rota para um par origem–destino. Esta mensagem "entrega" ao SmartMine um identificador, que o Módulo de Despacho deve utilizar no comando *Obtém Rota* para que o *Tracker SMT 100* envie os pontos de rota cadastrados. Os pontos de rota são enviados através de eventos *Ponto de Rota*. Este evento possui os pontos (X,Y) da rota, bem como o tempo total gasto para realizá–la.

O evento *Pesagem Efetuada*, é enviado ao SmartMine sempre que o equipamento efetua uma pesagem, indicando a massa que acabou de ser carregada no caminhão. Este evento só é gerado quando o equipamento em questão, possui balança própria.

Assim que o equipamento recebe o comando de *Despacho*, o operador deve indicar se aceita ou não este despacho. O *Tracker* utiliza o evento *Despacho Reconhecido*, para "dizer" ao SmartMine se o operador aceitou ou recusou o despacho. O mesmo conceito é utilizado para o evento *Mensagem Reconhecida*. Comandos de mensagem podem ser enviados aos equipamentos. Estes, responderão através do evento *Mensagem Reconhecida*, se o operador está "ciente", quando a mensagem for uma afirmação, ou se o operador respondeu "sim" ou "não" para uma pergunta. Por fim, o evento *Carga Efetuada*, indica ao SmartMine que um caminhão foi carregado, trazendo informações sobre qual escavadeira carregou o caminhão e a hora e posição (X,Y) onde o evento ocorreu.

Todo evento recebido pelo SmartMine deve ser confirmado através de uma mensagem *Evento Recebido*. Esta mensagem, utiliza o campo *ID* de cada evento para a confirmação. O formato desta mensagem é o mesmo da Figura 4.3, no qual o campo *CMD* indica que a mensagem é uma confirmação de algum evento.

Além dos eventos e comandos trocados entre os *Trackers* e o Módulo de Despacho, através dos *Access Points*, os *Trackers* também precisam enviar ao SmartMine a posição (X,Y) dos equipamentos a cada instante. A mensagem *Status* é utilizada para enviar ao SmartMine a posição (X,Y) real de cada equipamento, além da velocidade atual do equipamento, direção e altitude. Como a mensagem *Status* acaba por ser a mensagem mais enviada pelo *Tracker*, utilizou–se também

esta mensagem para confirmação dos comandos recebidos do Módulo de Despacho. Assim, esta mensagem possui um campo inserido no campo *Dados* indicando o último evento processado. Além disso, esta mensagem também foi utilizada para repassar ao SmartMine, a rota *wireless* onde o equipamento está, utilizada nos cabeçalhos de comandos para os *Trackers* e nas mensagems de *Evento Recebido*.

4.2.8 Regras de Comunicação

Nesta seção define–se as regras para comunicação entre os *Access Points* e o Módulo de Despacho. A definição das regras de comunicação especifica como as mensagens do vocabulário do protocolo devem ser utilizadas, de modo a garantir um protocolo com regras confiáveis e não ambíguas. É importante frisar que as mensagens que partem do Módulo de Despacho são estimuladas por mensagens recebidas dos *Trackers* através dos *Access Points*. Ou seja, o Módulo de Despacho aguarda o recebimento de uma mensagem do *Access Point* para poder enviar alguma mensagem, excetuando–se as mensagens de *WatchDog*, utilizadas para manter a conexão. As seguintes regras foram modeladas para este protocolo:

Estabelecimento de Conexão: Toda requisição de conexão deve partir dos clientes *Access Points*, utilizando as rotinas de conexão fornecidas pelo protocolo TCP/IP. O estabelecimento de conexão deve ficar a cargo do sistema operacional em questão. No entanto, a validação da conexão deve ser realizada através do envio de uma mensagem *WatchDog* pelo cliente, onde o código do *Access Point* é processado pelo servidor. O servidor só poderá validar a conexão, se o código fornecido for um código válido. Caso contrário, a conexão deve ser cancelada.

Recebimento de Status dos Equipamentos: Todo *Tracker* instalado nos equipamentos, deve enviar a cada intervalo de tempo *t*, uma mensagem de *status*, que deve fornecer a rota *wireless* do equipamento e também o último comando processado. Mensagens de *status* podem ser recebidas a qualquer instante e não necessitam ser confirmadas.

Envio de Comandos: Todo comando enviado do Módulo de Despacho para os *Trackers* através dos *Access Points*, deve conter informações de roteamento obtidos na mensagem *status*. Um comando só

pode ser enviado quando o Módulo de Despacho recebe uma mensagem *status*. Isto significa que o equipamento está em uma área de cobertura de rádio, o que torna maior a probabilidade do *Tracker* receber o comando. Deve existir um *buffer* de comando, e estes devem ser enviados em ordem de chegada, de acordo com o método FIFO *First In First Out*. O Módulo de Despacho deve permanecer enviando o primeiro comando da fila até que este seja confirmado em uma mensagem de *status*. Assim, todo comando deve ser confirmado.

Recebimento de Eventos: Eventos podem ser recebidos pelo Módulo de Despacho a qualquer instante. Eventos devem ser enviados pelo *Tracker* com prioridade sob mensagens de *status*. Assim como os comandos, os *Trackers* devem manter um *buffer* de eventos que também devem seguir a política FIFO. Todo evento deve ser confirmado através de uma mensagem *Evento Reconhecido*.

***Timeout*:** Na tentativa de manter a conexão, uma mensagem *WatchDog* deve ser enviada sempre que o cliente ou o servidor nã enviem alguma mensagem em um intervalo de tempo *t*. Caso o receptor fique sem receber nenhuma mensagem neste mesmo intervalo de tempo, a conexão deve ser encerrada. Cabe ao cliente *Access Point*, solicitar novamente uma conexão.

4.3 Implementação

4.3.1 Introdução

Após a modelagem do protocolo de comunicação *Access Point* e Módulo de Despacho, a próxima fase da integração entre esses dois módulos do SmartMine é a implementação. Nesta seção, especifica–se as principais técnicas de implementação utilizadas e as decisões tomadas nesta fase que resultaram em um refinamento da modelagem do protocolo.

O software do hardware *Access Point* foi implementado na linguagem *C*, enquanto o software do Módulo de Despacho foi implementado utilizando–se a linguagem *Object Pascal*, utilizada na ferramenta *DELPHI* da *Borland*. Todo o protocolo foi implementado com base no protocolo TCP/IP. Assim, com o TCP/IP encapsulando o protocolo de comuicação criado, toda a parte de comunicação foi

deixada “nas mãos” do sistema operacional, que já possui toda a base para conexões TCP/IP.

Os sistemas operacionais, possuem um objeto que implementa um circuito virtual, denominado *socket*. Este objeto possui todos os métodos para solicitação de conexão, envio e recebimento de mensagens, controle de fluxo e controle de erros. Assim, cada conexão do Módulo de Despacho (servidor) com um *Access Point* (cliente), é representada por um *socket*.

O porto de comunicação (*Port*) escolhido foi o porto 80. Este porto é comumente utilizado em servidores *http* e, quase sempre, são menos “incomodados” por *Firewalls*. No entanto, o protocolo pode ser configurado para utilizar qualquer outro porto, desde que não haja conflito com outra aplicação que esteje utilizando o mesmo porto.

4.3.2 Modelo de Camadas

Na fase de modelagem, o protocolo foi dividido em camadas, sendo que a comunicação entre os *Access Points* e o Módulo de Despacho concentrou–se na camada de aplicação, segundo o Modelo de Referência TCP/IP [Bla95]. Assim, no Modelo TCP/IP, as camadas de sessão e apresentação praticamente perderam o sentido e não foram implementadas neste modelo. Segundo [Tan97], a maioria das aplicações não utilizam estas duas camadas.

No entanto, na fase de implementação do protocolo de comunicação HW/SW descrito neste trabalho, chegou–se a conclusão que a independência de algumas partes do software do Módulo de Despacho, deixariam o protocolo mais robusto, mais facilmente alterável e o melhor, a modelagem das reais funcionalidades e serviços de responsabilidade do Módulo de Despacho, se tornou mais claramente identificável na implementação. Com o software dividido em partes, os erros foram identificados muito mais rapidamente.

A camada de aplicação do protocolo, foi dividida em camada de aplicação, camada de apresentação e camada de sessão. Com esta divisão, o protocolo de comunicação englobou todas as camadas do Modelo de Referência OSI [DZ83], sendo que três camadas foram implementadas no software do Módulo de Despacho, na forma de classes, segundo os métodos da programação orientada a objetos. Assim, a camada de apresentação e a camada de sessão, são objetos utilizados pela camada de aplicação.

A divisão do software em 3 camadas independentes resultou na definição do escopo de cada camada, bem como os serviços prestados entre elas. A camada de aplicação ficou encarregada da interação com o usuário e o banco de dados. A ca-

mada de apresentação, ficou encarregada de implementar os serviços que concretizam as ações dos usuários, como por exemplo, enviar um comando de despacho para um camimhão. Esta camada recebe a requisição do usuário e a transforma em um fluxo de bytes, que são repassados para a camada de sessão. A camada de sessão, ficou responsável pela gerência da conexão com os *Access Points* e pela transferência de dados.

4.3.3 Máquina de estados finitos

Um protocolo é freqüentemente mais facilmente entendido como uma máquina de estados [Hol91]. Na representação do protocolo através de uma máquina de estados, cada estado do protocolo, simboliza as definições que cada processo no sistema assume sobre os outros. Estes estados definem quais ações são permitidas aos processos, quais eventos os estados esperam que aconteçam, e como cada estado responderá a estes eventos.

A representação de protocolos através de máquinas de estados é uma técnica de modelagem de protocolos. No entanto, o protocolo de comunicação entre os *Access Points* e o Módulo de Despacho, não foi modelado formalmente como uma máquina de estado. A técnica de modelagem através de máquinas de estados foi utilizada na fase de implementação do protocolo de comunicação do SmartMine, apenas na leitura das mensagens pelo receptor, na tentativa de implementar um controle de fluxo eficiente.

Apesar do protocolo TCP/IP que encapsula o protocolo de comunicação do SmartMine já garantir o controle de fluxo das mensagens TCP/IP, as mensagens do protocolo de comunicação do SmartMine necessitaram da implementação de um controle de fluxo na camada de aplicação, pelo fato do *Access Point* ser bastante lento em relação ao Módulo de Despacho. O *Access Point* foi construído com base em um processador de 8 bits, enquanto o Módulo de Despacho em um processador de 64 bits. Dessa maneira, o receptor Módulo de Despacho é muito mais rápido que o transmissor, *Access Point*, deixando o protocolo pouco eficiente. Na prática, pode–se verificar, que o receptor lia do *buffer* TCP/IP sempre uma mensagem com o tamanho inferior ao tamanho que deveria ser lido do *buffer*.

Na implementação do protocolo com base nas máquinas de estado, a leitura da mensagem foi realizada por partes, onde cada parte constituiu um estado, no qual a máquina somente muda de estado, ao receber todos os bytes necessários ao estado em questão.

Desta maneira, tendo como base o formato original da mensagens trocadas entre os *Access Points* e o Módulo de Despacho visualizado na Figura 4.3, cada

campo da mensagem agora constitui um estado. A máquina de estados foi constituída de 4 estados que devem ser processados na mesma ordem do formato da mensagem, ou seja, o campo *STX* , *CMD*, *TAM* e *DADOS*. A máquina de estados somente troca de estado ao receber a quantidade de bytes prevista para cada estado e deve sempre retornar ao estado inicial *STX*. Para processar toda a mensagem, todos os estados devem ser processados na ordem correta. A Figura 4.6 representa melhor a máquina de estados do protocolo de comunicação do SmartMine.

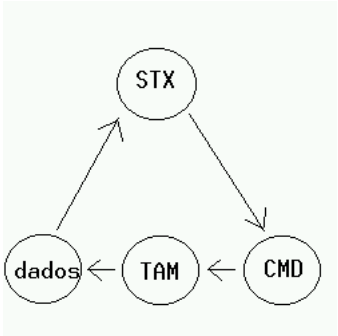

Figura 4.6: Diagrama de Transição de Estados

Capítulo 5

Protocolo de Comunicação Software/Web

5.1 Introdução

Este capítulo aborda a integração entre o Módulo de Despacho e o Módulo de Consulta do SmartMine, descrito no Capítulo 3. Mais especificamente, define–se o conjunto de regras para a comunicação entre esses módulos.

Como especificado anteriormente, o Módulo de Consulta interage diretamente com o Módulo de Despacho e com o servidor de banco de dados. Assim, cabe ao Módulo de Consulta disponibilizar relatórios e consultas baseadas nos resultados gerados pelo Módulo de Despacho, a partir de parâmetros selecionados pelos usuários. Cabe ao Módulo de Consulta, também, disponibilizar uma réplica visual das informações do Módulo de Despacho, onde os usuários deste sistema poderão visualizar tudo o que está ocorrendo na mina, mas não possui as funções de realizar comandos na operação da mina.

Assim como no Módulo de Despacho, o usuário do Módulo de Consulta pode visualizar o mapa da mina, a posição de cada equipamento da mina através de uma figura representativa do equipamento no mapa, em qual área de escavação os equipamentos estão alocados, o estado atual dos equipamentos que representa qual a atividade que estes estão executando no momento, entre outras funcionalidades.

Para que o Módulo de Consulta possa apresentar todas as informações em tempo real contidas no Módulo de Despacho do SmartMine, faz–se necessária, a especificação de um protocolo de comunicação entre esses dois módulos. No entanto, o Módulo de Consulta foi especificado para ser acessado via Internet/Intranet, o

que significa que este módulo foi desenvolvido como um aplicativo que possa ser utilizado via "*browser*", ou seja, navegadores como o *Internet Explores*, *Netscape*, entre outros.

Dessa maneira, existe a possibilidade de vários Módulos de Consulta, estarem funcionando ao mesmo tempo, em máquinas diferentes, o que tornaria o gerenciamento pelo Módulo de Despacho, uma tarefa bastante complicada. Para solucionar este problema, utilizou–se um *Web Server* para fazer o papel de um *gateway* entre o Módulo de Despacho e o Módulo de Consulta. Assim, o Módulo de Despacho envia as informações necessárias para este *gateway*, que se encarrega de repassar essas informações para o Módulo de Consulta em tempo real.

A seguir, especifica–se a modelagem da comunicação entre o Módulo de Despacho e o *Web Server*.

5.2 Modelagem

O protocolo de comunicação entre o Módulo de Despacho e o *Web Server*, é um protocolo entre as camadas de aplicação dos dois módulos, modelado para ser executado em uma rede *ethernet* 10/100 Mbps. Dessa maneira, o protocolo de comunicação proposto neste capítulo é encapsulado pelos protocolos de rede já implementados pelos sistemas operacionais, como o TCP/IP, IPX, entre outros. Assim, as duas máquinas onde estão instalados os módulos, devem estar na mesma rede.

Nesta comunicação, o Módulo de Despacho é sempre o transmissor e o *Web Server* sempre o receptor. Dessa maneira, a solicitação de conexão deve partir do Módulo de Despacho, o que significa que o *Web Server*, deve possuir um endereço (IP) fixo na rede, ou que pelo menos, esteja registrado no servidor de DNS, para que o Módulo de Despacho sempre "encontre", o *Web Server*.

Como o protocolo aqui proposto é encapsulado por um protocolo de rede, no caso TCP/IP, não foram modeladas estratégias para identificação e correção de erros, tratamento de redundâncias, controle de fluxo, congestionamento, roteamento, entre outros problemas encontrados em uma comunicação baseada em redes com fio. Estas questões não foram consideradas, pois os protocolos de rede implementados pelos sistemas, incluindo o TCP/IP, já possuem métodos para tratamento de todos estes problemas [Tan97], garantindo a confiabilidade do protocolo. Baseado nesta confiabilidade, as mensagens do protocolo Módulo de Despacho – *Web Server*, foram codificadas para enviar um fluxo de bytes, sem nenhum controle quanto ao tamanho das mensagens, pois estas são formatadas no tamanho ideal, de acordo

com as camadas abaixo da camada de aplicação. No caso do TCP/IP, a camada de transporte.

As mensagens trocadas entre os módulos, não necessitam de confirmação no nível de camada de aplicação, pois a chegada das mensagens já é garantida pelo protocolo TCP/IP, e não é necessário também, o estabelecimento de uma conexão entre o Módulo de Despacho e o *Web Server* a nível de aplicação. Assim, o protocolo proposto neste capítulo, é um protocolo não orientado à conexão. A solução adotada para realizar a comunicação entre esses dois módulos sem estabelecimento de um conexão, foi a tecnologia COM/DCOM [Mic96]. Esta tecnologia é capaz de manter uma área de memória comum a dois ou mais aplicativos que estejam se comunicando através de uma rede de computadores. Assim, a função do Módulo de Despacho é escrever nesta região de memória compartilhada, utilizando os métodos fornecidos pela tecnologia COM/DCOM, em um intervalo de tempo *t* configurável. Por sua vez, o *Web Server* deve acessar esta região de memória e identificar que existe uma mensagem com novos dados para atualização do Módulo de Consulta.

O vocabulário de mensagens do protocolo de comunicação entre o Módulo de Despacho e o *Web Server*, é baseado em uma única mensagem que contém todos os dados necessários para a atualização do Módulo de Consulta. O formato desta mensagem pode ser visualizado na Figura 5.1.

Figura 5.1: Formato da Mensagem de Atualização do Módulo de Consulta

O campo *Cabeçalho* contém informações que indicam a quantidade de equipamentos de carga presentes no controle, bem como a quantidade de equipamentos de transporte e a quantidade de equipamentos auxiliares. Estas informações são importantes para que o Módulo de Consulta “saiba” a quantidade de equipamentos presentes no Módulo de Despacho. Isto significa que se algum caminhão for acrescentado ou excluído, o Módulo de Consulta tem como saber que o número de equipamentos mudou, e deve consultar o banco de dados para buscar novos equipamentos ou apagar outros da interface de consulta.

O campo *Equipamentos de Carga* possui informações exclusivas destes equipamentos, como posição (X,Y), estado atual que indica a atividade que está sendo

realizada, em qual área de escavação está alocada, entre outras informações. Essas mesmas informações, estão presentes nos campos *Equipamentos de Transporte* e *Equipamentos Auxiliares*. Equipamentos auxiliares são equipamentos menos importantes no que diz repeito à produção de minério, como tratores, caminhões pipa, entre outros.

A Figura 5.2 é um exemplo da mensagem enviada do Módulo de Despacho para o *Web Server*.

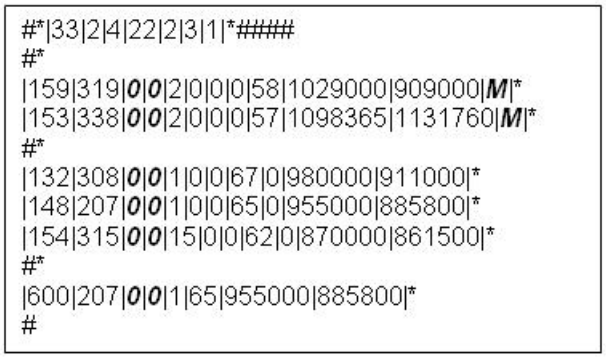

Figura 5.2: Exemplo de mensagem do Módulo de Despacho para o *Web Server*

Como pode ser observado na Figura 5.2, em uma única mensagem enviada a cada *t* segundos, todas as informações de todos os equipamentos que possuam um *Tracker SMT 100* são enviadas ao *Web Server*. Nesta mensagem de atualização, cada campo, visualizado na Figura 5.1, é delimitado por um caracter "//", no início e no final do campo. Dentro de cada campo "objetos" diferentes são separados por um caracter "*". Por exemplo, no campo *Equipamentos de Carga*, todos os equipamentos são encapsulados neste campo, o que significa que cada equipamento de carga deve ser tratado como um "objeto" com informações próprias e devidamente separados. Dentro de cada objeto, as informações necessárias são separadas por um caracter "|".

Capítulo 6

Conclusão

A procura cada vez mais constante por soluções tecnológicas, tem resultado em uma alta complexidade dos sistemas de computação atuais, que necessitam ser capazes de lidar com funcionalidades que requerem tratamento de informações em tempo real, integração Hardware/Software, redes de alta velocidade, redes *Wireless*, tratamento de grandes massas de dados, entre outras caraterísticas.

A complexidade dos sistemas atuais, tem motivado pesquisas e, conseqüentemente, o surgimento de novas tecnologias e modelos, capazes de reduzir o tempo e custo de desenvolvimento destes sistemas, focalizando as duas principais fases de desenvolvimento: o projeto e a implementação.

Na tentativa de amenizar o problema, uma abordagem chamada *Hardware/ Software Codesign* tem sido amplamente estudada. Esta abordagem envolve as seguintes etapas: análise de restrições e requisitos, especificação do sistema, particionamento em hardware e software, co–síntese, integração hardware/software e validação e verificação do projeto.

Considerando o modelo básico de codesing descrito no Capítulo 2, este trabalho concentrou–se principalmente na etapa de integração hardware/software. Mais especificamente, neste trabalho foi proposto um modelo de comunicação entre os módulos de um sitema complexo em tempo real, o SmartMine, um software para controle e garantia de qualidade de minério de minas de céu aberto, especificado no Capítulo 3. Assim, foram modelados e implementados dois protocolos, onde o primeiro, foi proposto para realizar a comunicação entre o Módulo de Despacho do sistema SmartMine e o software embarcado do hardware instalado nos equipamentos de mina, como caminhões e escavadeiras. O segundo protocolo foi proposto para realizar a comunicação entre o Módulo de Despacho e o Módulo de

Consulta, especificado para ser acessado via *Web*. Neste trabalho foram evidenciadas as principais decisões de modelagem e implementação dos protocolos de comunicação propostos.

O protocolo de comunicação entre o Módulo de Despacho e o hardware dos equipamentos , os *Trackers SMT 100*, foi modelado para ser executado em dois ambientes distintos: uma rede *ethernet* 10/100 Mbps e um rede *wireless* com transmissão e recepção via rádio–modem. Para realizar a interface entre as duas redes, foi desenvolvido um *gateway*, nomeado *Access Point*, capaz de interpretar informações dos dois ambientes de execução.

Dessa maneira, este trabalho baseou–se no modelo TCP/IP, para encapsular as mensagens de comunicação entre o Módulo de Despacho e os *Access Points*. Isto significa, que os maiores esforços foram concentrados na última camada do modelo TCP/IP, ou seja, a camada de aplicação. Esta camada foi responsável por implementar o protocolo de comunicação Módulo de Despacho–*Access Point*. As demais camadas, implementam a comunicação entre os *Access Points* e os *Trackers SMT 100*, e a definição sucinta de cada uma delas é assunto para um futuro trabalho.

No entanto, o protocolo de comunicação entre Módulo de Despacho e os *Access Points* foi modelado para ser o mesmo entre os *Access Points* e os *Trackers*, o que significa que o *Access Point* apenas repassa mensagens de uma rede para outra, sem realizar nenhum processamento nestas. Dessa maneira, concluiu–se que o protocolo deveria conter estruturas capazes de satisfazer as restriçoes impostas por cada meio de comunicação e capazes de atender os requisitos do SmartMine, principalmente aqueles com características de tempo real.

Apesar do protocolo TCP/IP já dispor de estratégias e métodos para controle de fluxo, controle de mensagens duplicadas ou fora de ordem, identificação de receptores e transmissores, codificação de mensagens, gerenciamento de conexões, *TIMEOUTs*, entre outras características, julgou–se necessário a remodelagem destes conceitos na camada de aplicação deste protocolo, apesar de não serem funções desta camada segundo o modelo de referência OSI.

Todo o protocolo foi modelado com a finalidade de suprir as dificuldades encontradas em redes *wireless*, como baixa vazão (*troughput*); pacotes perdidos, duplicados ou fora de ordem; e custo. Por exemplo, as mensagens da camada de aplicação carregam informações de roteamento da camada de rede, exclusivas para a comunicação *Wireless*.

Outras questões foram modeladas para suprir a necessidade de informações em tempo real do sistema. Por exemplo, concluiu–se ser necessário remodelar o con-

ceito de *TIMEOUT* de conexão, apesar de o protocolo TCP/IP já possuir métodos capazes de identificar problemas na conexão. Esta remodelagem foi necessária, pois observou–se que o tempo que o sistema operacional leva para identificar problemas na conexão é um tempo relativametne alto, tornando–se crítico para um sistema que necessita de informações em tempo real, especificado para funcionar 24 horas por dia sem interrupção.

Devido ao fato do SmartMine ser um sistema em tempo real, especificado para gerenciar todos os equipamentos de transporte de material e equipamentos de carga, entre outros, resultando em uma modelagem de um estratégia eficiente para o tratamento de um grande volume de dados recebidos e processados pelo Módulo de Despacho. A solução encontrada para deixar o sistema "liberado", sem gastar tempo esperando que uma inserção ou atualização no banco de dados seja terminada, foi a implementação de um *Message Queue*, ou seja, um processo independente e exclusivo para realizar operações com o banco de dados. Assim, o Módulo de Despacho entrega o "serviço" de banco de dados para o *Message Queue*, sendo capaz de responder mais rapidamente às exigências de tempo real do sistema.

Este trabalho abordou todas as etapas de modelagem de protocolos, nas quais se definiu o formato e codificação das mensagens, o vocabulário do protocolo e as regras de comunicação. A única etapa não abordada neste trabalho, no nível de projeto, foi a etapa de testes e validação do protocolo, pois o sistema encontra–se em fase de implementação na presente data.

A integração entre o Módulo de Despacho e o Módulo de Consulta foi baseada nas caracteriśticas deste último e as informação necessárias para seu funcionamento. O Módulo de Consulta é o módulo do SmartMine onde relatórios e consultas podem ser acessados pelos usuários do sistema. No entanto, o Módulo de Consulta possui uma interface idêntica à interface do Módulo de Despacho. Isto significa, que esta interface precisa ser atualizada em um intervalo de tempo pequeno, para que qualquer usuário que acesse este módulo tenha uma visão geral da mina em tempo real.

Assim como no Módulo de Despacho, a interface do Módulo de Consulta disponibiliza informações em tempo real, como a posição real dos equipamentos na mina, a atividade que estes estão realizando, entre outras funções. Dessa maneira, o Módulo de Despacho foi modelado para ser acessado via "*browser*", por qualquer máquina autorizada na Intranet/Internet.

Para que as informações do Módulo de Despacho sejam disponibilizadas para acesso via *browser*, foi necessário a utilização de um *Web Server*. Assim, no proto-

colo de comunicação proposto para a comunicação entre o Módulo de Despacho e o Módulo de Consulta, as mensagens devem ser enviadas do Módulo de Despacho para o *Web Server*, e deste, para os *browsers*.

Tendo em vista que o Módulo de Consulta é apenas um módulo para visualização das informações em tempo real, que não realiza qualquer processamento nas mensagens, julgou–se não ser necessário a modelagem de estratégias de controle de fluxo, correção de erros, identificação de transmissores e receptores, tamanho ideal das mensagens, entre outras características presentes na modelagem de protocolos. Assim, todo o protocolo baseia–se em uma única mensagem de atualização, que contém informações de todos os equipamentos que estejam se comunicando com o Módulo de Despacho.

Para implementação deste protocolo, utilizou-se a tecnologia COM/DCOM. Esta tecnologia "cria" uma área de memória compartilhada entre os dois lados da comunicação, e fornece métodos para gravação e leitura neste região de memória. Assim, basta ao Módulo de Despacho gravar a mensagem de atualização nesta área compartilhada num intervalo de tempo *t* configurável. Cabe ao *Web Server*, fazer a leitura desta área de memória, e disponibilizar as novas informações para o Módulo de Consulta. Desta maneira, todo o trabalho de comunicação é deixada a cargo desta tecnologia, exigindo–se apenas que as máquinas do Módulo de Despacho e do *Web Server* estejam na mesma rede.

Finalmente, para concluir toda a integração dos módulos do sistema SmartMine, propõe–se como futuro trabalho, a especificação da comunicação dos *Access Points* com os *Trackers SMT 100* utilizando–se rádio–modem de curto e longo alcance.

Referências Bibliográficas

[Alb01] Jones Oliveira Albuquerque. *A Stochastic Approach for Conceptual Level Codesign.* PhD thesis, Federal University of Minas Gerais, 2001.

[BC98] Edna Barros and Sérgio Cavalcante. Introdução a hardware/software co–design. In *XVII Jornada de Atualização em Informática – JAI*, Belo Horizonte, 1998.

[Bla95] U.D. Black. Tcp/ip and related protocols. *McGraw–Hill*, 1995.

[DZ83] J.D. Day and H. Zimmermann. The osi reference model. *Proc. of the IEEE, vol. 71*, pages 1334–1340, December 1983.

[GJ79] M.R. Garey and D.S. Johnson. Computers and intractability: a guide to the theory of np–completeness. *Freeman*, 1979.

[Hol91] G.J Holzmann. Design and validation of computer protocols, englewood cliffs. *NJ: Prentice Hall*, 1991.

[Joh67] S.C. Johnson. Hierarchical clustering schemes. *Psychometrika*, pages 241–254, 1967.

[kF95] kamilo Feher. *Wireless digital communications*. Prentice Hall, 1995.

[kL70] B.W. kernighan and S. Lins. An efficient heuristic procedure for partitioning graphs. *Bell System Technical Journal*, 1970.

[kri84] B. krishnamurthy. An improved min–cut algorithm for partitioning vlsi networks. In *IEEE Transactions on Computer*, 1984.

[Lag89] E.D. Lagnese. *Architectural Partitioning for System Level Design of Integrated Circuits*. PhD thesis, Carnegie Mellon University, 1989.

[Mic96] Microsoft Corporation. *DCOM Technical Overview*, 1996.

[Mic00] Microhard Systems Inc. *900 MHz Spread Spectrum OEM Transceiver*, revision 1.20 edition, december 2000.

[Nem95] Martin A.W. Nemzow. *Implementing Wireless Networks*. McGraw-Hill, 1995.

[Rip89] David L. Ripps. *An implementation Guide to Real–Time Programming*. Prentice-Hall, 1989.

[SKV83] C.D. Gelatt S. Kirkpatrick and M.P. Vecchi. Optimization by simulated annealing. *Science, vol. 220, no.4589*, pages 671–680, 1983.

[Tan97] Andrew S. Tanenbaum. *Computer Networks*. Prentice-Hall, 1997.