UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO

CENTRO TECNOLÓGICO

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

LUCIANO MAGNO BRAMBILA

SIMULAÇÃO MICROSCÓPICA DISTRIBUÍDA DE TRÁFEGO

VITÓRIA

2008

LUCIANO MAGNO BRAMBILA

SIMULAÇÃO MICROSCÓPICA DISTRIBUÍDA DE TRÁFEGO

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Mestre em Engenharia Elétrica.

Orientador: Prof Hans-Jorg Andreas Schneebeli

VITÓRIA

2008

Dados Internacionais de Catalogação-na-publicação (CIP)
(Biblioteca Central da Universidade Federal do Espírito Santo, ES, Brasil)

Brambila, Luciano Magno, 1973-
B815s Simulação microscópica distribuída de tráfego / Luciano Magno Brambila. – 2008
92 f. : il.

Orientador: Hans-Jorg Andreas Schneebeli.
Dissertação (mestrado) – Universidade Federal do Espírito Santo, Centro Tecnológico.

1. Simulação (Computadores). 2. Controle do tráfego. 3. Sistemas distribuídos. 4. Java (Linguagem de programação de computador). I. Schneebeli, Hans-Jorg. II. Universidade Federal do Espírito Santo. Centro Tecnológico. III. Título.

CDU:621.3

LUCIANO MAGNO BRAMBILA

SIMULAÇÃO MICROSCÓPICA DISTRIBUÍDA DE TRÁFEGO

Dissertação submetida ao Programa de Pós-Graduação em Engenharia Elétrica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Mestre em Engenharia Elétrica.

Aprovada em 17 de novembro de 2008

COMISSÃO EXAMINADORA

Prof Dr. Rer. Nat. Hans-Jorg Andreas Schneebeli
Universidade Federal do Espírito Santo
Orientador

Prof Dr. Werner Kraus Junior
Universidade Federal de Santa Catarina

Prof Dr. Alberto Ferreira de Souza
Universidade Federal do Espírito Santo

À minha esposa Sônia,
meus pais José e Izolina
e ao Filipe, minha maior realização
(até o momento).

Agradecimentos

Gostaria de agradecer ao meu orientador, Hans-Jorg Andreas Schneebeli que acreditou em mim durante toda minha – longa – história de mestrando. Obrigado pelo incentivo, orientação e formação científica. Essa vitória só foi possível pelo seu apoio.

Aos professores Werner Kraus Junior e Alberto Ferreira de Souza, que gentilmente aceitaram o convite de participar e contribuir com este trabalho.

Aos meus pais, vocês são meus exemplos, irmãos e familiares, pelo amor, incentivo e exemplos de dedicação.

À minha esposa e filho que entenderam e aceitarem meus momentos de dedicação a este trabalho. “O sucesso é construído à noite”. A vitória chegou. Preparem-se agora...

E a Deus, que é tão grande e maravilhoso que não se importa de ser o último citado neste agradecimento.

Sumário

Lista de Tabelas

Lista de Ilustrações

Resumo

A maioria das grandes cidades do mundo enfrenta problemas de tráfego. Na busca de soluções para os problemas do tráfego surge, como poderosa ferramenta, a simulação, que utiliza técnicas matemáticas, computacionais e estatísticas para representar o tráfego.

Modelos de simulação de tráfego podem ser classificados de várias formas. A mais utilizada é quanto ao nível de detalhes com que os modelos representam o mundo real e quanto à maneira como o comportamento dos veículos é reproduzido. Seguindo esta classificação, os modelos podem ser macro, meso ou microscópicos.

Modelos microscópicos simulam as entidades do sistema individualmente e com um alto nível de detalhes. Cada par veículo-motorista tem um comportamento próprio e é um componente ativo na simulação. Não somente veículos e motoristas são modelados, mas qualquer outra entidade que possa influenciar no tráfego, como controladores de tráfego, trechos específicos de pistas e incidentes. Modelar o tráfego no nível microscópico representa hoje o estado da arte para os simuladores de tráfego.

Quanto mais completo um modelo (as características microscópicas do tráfego são consideradas), mais realista será a simulação. Por outro lado, maior é o esforço computacional necessário para sua execução. Alguns simuladores microscópicos requerem computadores de grande desempenho ou supercomputadores e, conseqüentemente, têm um alto custo.

A alternativa ao uso de supercomputadores é a utilização da computação distribuída. Um ambiente distribuído permite que o programa de simulação seja executado em múltiplos processadores de forma paralela e assim obtenha ganhos de desempenho. Usar um ambiente distribuído requer uma estratégia de divisão das tarefas entre processadores e uma solução para sincronizar a execução dessas tarefas.

Este trabalho tem como objetivo identificar e implementar uma solução para a execução de um simulador microscópico de tráfego em um ambiente distribuído, disponibilizando uma alternativa de baixo custo para estudo do tráfego.

Abstract

Most of the major cities of the world face traffic problems. Simulation is becoming one of the most used tools in the search of solutions in traffic. Simulation uses mathematical techniques, computer and statistics to represent traffic. The simulation allows the analysis of possible traffic configurations to be implemented to improve the flow of vehicles in a certain place.

Models of simulation of traffic can be classified in many ways. The most used is about the level of detail with which the models represent the real world and the way the behavior of vehicles is modeled. Following this classification, the models can be macro, meso or microscopic.

Microscopic models simulate the system entities individually with a high level of detail. Each pair vehicle-driver has an own behavior and is an active component in the simulation. Not only vehicles and drivers are modeled, but any other entity that can influence the traffic, like traffic controllers, specific roadways and incidents. Modeling traffic in the microscopic level today represents the state of the art for traffic simulators.

The more complete the model, the more realistic will be the simulation. On the other hand, it needs a great computational effort. Some simulators require supercomputers or high performance computers and thus have a high cost.

The alternative to the supercomputers is the use of distributed computing. A distributed environment allows the simulation program to run on multiple processors in parallel and thus gains in performance. Using a distributed environment requires a strategy of division of tasks between processors and a solution to synchronize these tasks.

This work aims to identify and implement a solution for a simulator microscopic traffic in a distributed environment, providing a low-cost alternative for study and improvement of traffic.

Capítulo 1

Introdução

Problemas de tráfego são tão antigos quanto o Império Romano. A causa básica, como nos dias atuais, era o pobre planejamento das cidades, com vias que convergiam aleatoriamente de diversos quarteirões para um ponto central. Na Roma antiga, Júlio César proibiu o tráfego sobre rodas durante o dia, medida que foi gradualmente estendida para outras províncias como uma forma de limitar o acesso e garantir um mínimo de circulação. Em 1500, Leonardo da Vinci, prevendo uma solução revolucionária para os problemas de tráfego, sugeriu separar o tráfego de veículos e pedestres pela criação de rotas em dois diferentes níveis. Na Europa do século XVII, os congestionamentos levaram à proibição do estacionamento em certas áreas e à criação de vias de mão única. O automóvel rapidamente criou uma nova situação que veio a se caracterizar como um dos principais problemas da sociedade industrial.

Hoje, a maioria das grandes cidades do mundo enfrenta problemas de tráfego. Congestionamentos não somente significam perdas em cifras monetárias devido à ociosidade mas constituem também maiores riscos de acidentes, poluição ambiental e sonora e diminuição da qualidade de vida.

É uma necessidade mundial a busca por melhores estratégias de gerenciamento de tráfego, principalmente quando consideramos que o aumento na demanda por viagens e transportes vem ocorrendo a taxas maiores do que as melhorias no sistema rodoviário atual.

1.1 Motivação

Na busca de soluções para os problemas do tráfego, surge, como poderosa ferramenta, a simulação, que utiliza técnicas matemáticas, computacionais e estatísticas para representar o tráfego. A simulação permite a análise de possíveis medidas a serem implantadas para melhorar o fluxo de veículos em determinado local. A análise do tráfego, por meio da simulação, evita que medidas inadequadas sejam implantadas, já que ela possibilita experimentar variadas configurações de tráfego em uma mesma estrutura viária. Além disso, mostra-se economicamente viável, pois o investimento necessário ao desenvolvimento de um simulador de tráfego é pequeno quando comparado à quantia necessária para correção de uma implantação mal sucedida.

Os modelos de simulação são projetados para imitar o comportamento de sistemas fisicos reais. O usuário de um software de simulação de tráfego especifica um cenário de entrada da simulação, como por exemplo, uma configuração de malha viária e uma demanda de tráfego. Os resultados da simulação fornecem ao usuário informações de “o que” está propenso a acontecer e dados que podem levá-lo a entender “porque” o sistema se comporta dessa forma.

A simulação não gera soluções por si só, mas atua da mesma forma que o sistema estudado, obtendo dados estatísticos para análise. É responsabilidade do usuário interpretar adequadamente as informações providas pela simulação a fim de obter um entendimento das relações de causa-efeito do tráfego. De posse dos dados de uma simulação, um analista pode, por exemplo, encontrar uma configuração que forneça a melhor resposta no tráfego, como alterar os tempos de um semáforo para atingir o maior fluxo de veículos em determinado horário. Outros benefícios atribuídos à simulação de tráfego são:

- Treinamento de pessoal: a simulação pode ser usada para treinar operadores, motoristas e até pedestres;
- Análise de segurança: modelos de simulação podem recriar cenários de acidentes, úteis na busca por itens de segurança em veículos e estradas;
- Testes de novos projetos: a simulação pode ser aplicada para analisar a performance do tráfego em resposta a um novo projeto antes de sua implementação;

- Substituição aos testes de campos: as simulações não causam as interrupções que são freqüentes nos testes de campo, além de se obter respostas de forma mais rápida e barata.

1.2 Simulação microscópica

Modelos de simulação de tráfego podem ser classificados de várias formas. A mais utilizada, ainda que não haja um consenso [1] e [2], é quanto ao nível de detalhes com que os modelos representam o mundo real e quanto à maneira como o comportamento dos veículos é reproduzido. Seguindo esta classificação, os modelos podem ser macro, meso ou microscópicos, ou, conforme Lieberman [3], de baixa, média ou alta fidelidade respectivamente. Recentemente vem sendo estudada uma quarta categoria que procura modelar o tráfego com um nível de detalhes além do nível microscópico: o nível nanoscópico [4].

As simulações macroscópicas baseiam-se na consideração de que as correntes de tráfego são meios contínuos. Os veículos são agregados em grupos e, juntos, têm comportamento análogo ao escoamento de um fluido, motivo pelo qual a abordagem também é conhecida como analogia hidrodinâmica do tráfego. Modelos macroscópicos aplicam-se com sucesso ao estudo de tráfego com grande densidade, mas não se prestam facilmente às situações de tráfego rarefeito [5];

Modelos microscópicos simulam as entidades do sistema individualmente e com um alto nível de detalhes. Cada par veículo-motorista tem um comportamento próprio e é um componente ativo na simulação. Não somente veículos e motoristas são modelados, mas qualquer outra entidade que possa influenciar no tráfego, como controladores de tráfego, trechos específicos de pistas e até acidentes. Modelar o tráfego no nível microscópico representa hoje o estado da arte para os simuladores de tráfego [4].

Modelos nanoscópicos consideram o trio veículo-motorista-ambiente como componente ativo da simulação. Informações como atrito do veículo e torque do motor são usada neste modelo [4].

Modelos mesoscópicos podem ser entendidos como o nível intermediário entre os dois anteriores. Veículos são agrupados em pacotes que têm as mesmas origens e destinos, e seus motoristas as mesmas características [6]. Manobras de

mudança de faixa, por exemplo, podem ser reproduzidas para cada veículo, individualmente, porém a decisão pode ser baseada em densidades de faixa e não nas relações individuais entre veículos.

A escolha do tipo de simulação deve basear-se na precisão desejada para os resultados da simulação. Modelos de baixa e média fidelidade são mais fáceis e menos custosos para desenvolver, executar e manter. Entretanto, há o risco de que suas representações do mundo real sejam pouco precisas a ponto de serem inadequadas ou inválidas. O uso de modelos macro e mesoscópicos pode ser apropriado quando os resultados esperados não são sensíveis aos detalhes microscópicos do tráfego e se o tempo e os recursos para desenvolvimento são limitados. Quando o grau de detalhamento esperado pela simulação é alto, o modelo mais indicado é o microscópico.

Há várias vantagens de se escolher a abordagem microscópica:

- O tráfego, em si, é um sistema afetado numa escala individual de veículos e motoristas;
- A simulação microscópica gera tanto informação detalhada como agregada. Ainda que veículos sejam modelados de forma individual, é possível extrair macro parâmetros como densidade e fluxos de vias e parâmetros derivados como consumo de combustível, impacto sonoro, etc;
- A simulação microscópica abre a possibilidade de comportamentos emergentes, ou seja, a existência de resultados que inicialmente não estavam programados no modelo, como por exemplo, a previsão de um padrão de viagens devido à ocorrência de certas combinações de necessidades de motoristas, pedestres, etc.
- Modelos de simulação microscópica são mais fáceis de serem entendidos e interpretados. Uma vez que eles são formulados no nível de atores individuais, algumas deduções simples podem ser tiradas sobre o que o modelo está acompanhando. Detalhes técnicos da implementação do modelo são tipicamente complexos, porém os conceitos fundamentais são, em alguns casos, de surpreendente entendimento, como conseqüência, modelos de simulação microscópica são acessíveis até por não especialistas.
- Habilidade de modelar vias congestionadas, o que não se alcança com a mesma qualidade nas outras abordagens;
- Representação visual tão complexa e fiel quanto se deseja;

Por outro lado, os seguintes pontos de atenção devem ser considerados:

- Como são mais complexos, modelos microscópicos exigem um esforço maior de desenvolvimento e conseqüentemente levam um tempo maior a partir da concepção até estarem pronto para uso;
- A própria simulação pode tornar-se excessivamente demorada pela natureza do modelo ou pelo objetivo da simulação. Quanto mais detalhado o modelo e os resultados a serem extraídos, maior o tempo necessário para a execução do programa de simulação.

1.3 Definição do Problema

O tráfego é um sistema com grande número de entidades: cada motorista tem sua forma particular de dirigir, cada veículo tem características próprias e respostas diferentes à maneira de guiar de cada motorista, a via por onde trafegam os veículos e os agentes de controle (semáforos, redutores de velocidade, quebra-molas, etc) têm influência na forma como os motoristas dirigem e na reação dos veículos. Além disso, fatores externos como clima (dirigir num dia chuvoso ou com neblina é diferente de dirigir em um dia ensolarado), período (finais de semana são diferentes de uma quarta-feira em horário de pico) ou condições da via também influenciam o tráfego. Quanto mais um modelo levar em consideração esses itens e reproduzi-los, mais realista será a simulação.

Por outro lado, quanto mais completo o modelo, maior é o esforço computacional necessário para sua execução. Há pouco tempo, desenvolver modelos de simulação microscópica era proibitivo, dada a carência de computadores suficientemente rápidos. Alguns simuladores requeriam computadores de grande desempenho ou supercomputadores e, conseqüentemente, tinham um alto custo.

A alternativa ao uso de supercomputadores é a utilização da computação distribuída. Um ambiente distribuído permite que o programa de simulação seja executado em múltiplos processadores de forma paralela e assim obtenha ganhos de desempenho. Usar um ambiente distribuído, entretanto, requer uma estratégia de divisão das tarefas entre processadores e uma solução para sincronizar a execução dessas tarefas.

Este trabalho tem como objetivo identificar e implementar uma solução para a execução de um simulador microscópico de tráfego em um ambiente distribuído, disponibilizando uma alternativa de baixo custo para estudo do tráfego. Para isso o simulador deve atender aos seguintes quesitos:

- Deve ser possível executar uma simulação mais de uma vez com os mesmos parâmetros e obter o mesmo resultado. Essa propriedade é útil nas análises de cenários *what-if*, uma vez que o usuário pode executar inúmeras simulações variando apenas um parâmetro e observar o impacto somente desde parâmetro em toda a simulação;
- Visualização gráfica da simulação, com interface amigável para o usuário para possibilitar a concentração das análises em áreas específicas da região simulada;
- Controle da velocidade de simulação, o que permite ao usuário visualizar a simulação em tempo real, câmera lenta (*slow motion*) para capturar detalhes ou avanço rápido (*fast forward*) para simular um grande período de tempo em poucos minutos;
- Modularidade: o simulador deve ser construído de forma a facilitar a incorporação de novos objetos de tráfego, como exemplo novos tipos de veículos e semáforos.

1.4 Metodologia

Para atender ao objetivo do trabalho, o primeiro passo foi desenvolver um simulador de tráfego seqüencial, ou seja, um simulador para ser executado em um ambiente de um único processador. A partir do levantamento e leitura de bibliografia relacionada, foram identificados os principais modelos usados na simulação microscópica e que descrevem o comportamento individual dos veículos no tráfego. Uma vez que simulador microscópico de tráfego pode ser visto como uma coleção de componentes que se interagem, é instintivo modelá-lo como um conjunto de objetos que trocam mensagens. Assim, o simulador foi desenvolvido usando o paradigma orientado a objetos e a linguagem de programação Java.

O passo seguinte consistiu no estudo das alternativas tecnológicas para execução de software em um ambiente com múltiplos processadores, como

plataformas de hardware, modelos de programação e comunicação entre processadores.

Em seguida, foram levantados os requisitos específicos da simulação microscópica de tráfego em um ambiente de múltiplos processadores. As soluções encontradas para as questões do sincronismo entre processadores foram desenvolvidas e aplicadas no código do simulador, tornando-o apto a ser executado em um ambiente distribuído.

Por último, foi elaborado um estudo de caso com o objetivo de validar a distribuição do processamento do simulador e de comparar o desempenho das versões seqüencial e distribuída para diferentes intensidades do fluxo de veículos e malha viária.

1.5 Estrutura do trabalho

Além deste primeiro capítulo, que introduziu a simulação como uma ferramenta efetiva para a análise de tráfego e apresentou os objetivos deste trabalho, o restante desta dissertação está organizada da seguinte forma:

- O Capítulo 2 apresenta os conceitos envolvidos e o que deve fazer um sistema de simulação microscópica do tráfego. Neste capítulo é explicado o desenvolvimento do simulador seqüencial;
- O Capítulo 3 descreve como foi realizada a evolução do algoritmo seqüencial para seu modelo distribuído, os requisitos necessários e as soluções encontradas para a distribuição do simulador;
- O Capítulo 4 descreve como algumas situações específicas foram implementadas usando a tecnologia de comunicação entre processadores escolhida e os resultados da utilização do simulador em um estudo de caso;
- O Capítulo 5 apresenta as conclusões e os possíveis trabalhos futuros.

Completam essa dissertação a Bibliografia, o Anexo A e um CD com todo o material digital usado neste trabalho.

Capítulo 2

Simulação microscópica de tráfego

Se ainda não existisse um modelo do sistema do tráfego e fosse necessário começar a inventar algo hoje, por onde se começaria? É difícil imaginar que não fosse na modelagem dos movimentos individuais dos veículos, como na abordagem microscópica.

O estudo da abordagem microscópica iniciou a partir de 1950, com os trabalhos de Reuschel e Pipes onde os elementos dinâmicos dos veículos foram introduzidos nos estudos do tráfego. Nestes trabalhos, o foco foi o comportamento dinâmico de um fluxo de veículos à medida que eles aceleravam ou freavam e como cada veículo (na verdade par veículo-motorista) reagia para acompanhar os veículos próximos a ele. Os estudos foram estendidos com os trabalhos de Kometani, e Sasaki e Herman nos laboratórios da General Motors [7].

A abordagem microscópica modela as entidades do sistema de tráfego individualmente e com um alto nível de detalhes. Para tal, ela utiliza as características físicas dos veículos (dimensões, velocidade máxima, aceleração máxima, entre outras), as leis fundamentais de movimento (por exemplo, velocidade média = distância percorrida/tempo necessário para que o veículo percorra essa distância) e alguns modelos que descrevem o comportamento individual dos veículos ante as várias situações a que eles estão sujeitos.

Neste capítulo são apresentados os requisitos esperados de um simulador de tráfego microscópico e os modelos mais utilizados para simular o tráfego neste nível de fidelidade. Em seguida é apresentada a implementação do simulador de tráfego deste trabalho.

2.1 Especificação do simulador

A principal aplicação de um simulador microscópico é na análise detalhada do fluxo de veículos para a identificação de medidas de controle a fim de melhorar o tráfego de um determinado local. Por melhorar o tráfego entende-se aumentar os valores de velocidade média dos veículos e número de veículos por unidade de tempo e diminuir o tempo médio de viagem dos veículos.

Para isso, o analista, ou usuário, deve ter a possibilidade de modelar, no simulador, o tráfego do local sob análise, fixar parâmetros de certos componentes deste modelo e, a cada simulação, variar parâmetros de outros componentes até encontrar uma configuração que lhe dê a melhoria esperada. Cada execução do modelo de simulação com uma configuração específica de parâmetros é chamada neste trabalho de Caso de Simulação.

Para suportar essa análise, os componentes mínimos que um modelo de simulação de tráfego microscópico deve possuir são:

1. Representação da malha viária: compreende a caracterização das vias por onde os veículos trafegam e delimita a área a ser simulada. Deve ser possível definir características como: geometria, comprimento e largura das vias, número de pistas (ou faixas), sentido do tráfego, velocidade máxima e/ou características especiais de cada via;
2. Definição dos tipos de veículos: é a definição das características dos veículos que participam da simulação, como dimensões, capacidade de aceleração e desaceleração;
3. Definição dos tipos de motoristas: é a caracterização dos motoristas, como por exemplo, conservador, agressivo, moderado, etc. Para cada veículo é associado um tipo de motorista. Na verdade, veículos e motoristas formam um par que tem um comportamento ativo na simulação microscópica de tráfego;

4. Demanda dos veículos: compreende a especificação do fluxo de veículos que participam da simulação. Como dito, a representação da malha viária delimita a área a ser simulada. Isso significa que os veículos entram na simulação, ou seja, na área simulada, a partir de certas vias e que deixam a simulação através de outras vias. Para as vias que são entradas de veículos na simulação deve ser possível especificar o fluxo dos veículos (veículos por unidade de tempo) ao longo do tempo. Este componente tem por finalidade modelar a quantidade de veículos que trafegam nas vias em cada horário. Por exemplo, em horários de pico e em dias úteis uma via recebe uma quantidade de veículos – assim como tipos de motoristas – maior do que em uma madrugada ou final de semana.
5. Plano de controle do tráfego: compreende a definição dos controladores do tráfego. Por controlador de tráfego entende-se qualquer dispositivo que altere, de forma voluntária, o fluxo de veículos nas vias, como semáforos, redutores de velocidade, sinalizações, etc.
6. Modelagem detalhada do comportamento individual dos veículos e motoristas: é a definição do modelo de movimentação dos veículos, ou seja, a forma com que ocorre a interação entre os veículos-motoristas e os outros componentes da simulação. A escolha da modelagem do comportamento individual pode determinar o quão fidedigno será um simulador. Normalmente os simuladores usam modelos já definidos na literatura. Na Seção 2.2 (Modelos de Simulação Microscópica) são apresentados alguns dos modelos mais utilizados na simulação microscópica de tráfego;
7. Animação gráfica da simulação: não é somente uma funcionalidade desejável, mas uma ferramenta poderosa de análise dos resultados;
8. Estatísticas geradas para os Casos de Simulação: são a base para realização das análises e melhorias do tráfego.

Normalmente os componentes de 1 a 4 (malha viária, veículos, motoristas e demanda de veículos) são configurados no simulador a partir de dados obtidos no campo e o componente 6 (modelo do comportamento individual dos veículos e motoristas) é intrínseco ao simulador (discutido na Seção 2.2 Modelos de Simulação Microscópica). Como citado anteriormente, o analista geralmente fixa a configuração de alguns componentes e executa vários casos de simulação variando a configuração de outros. Ele pode, por exemplo, configurar a malha viária, os veículos-motoristas e a demanda de veículos e executar vários casos de simulação

variando o plano de controle do tráfego, como por exemplo, os tempos dos semáforos, até encontrar o plano que melhor atenda a necessidade do tráfego do local sob análise.

Desenvolver um simulador microscópico de tráfego requer endereçar pelo menos os componentes listados anteriormente (1 a 8). Esses requisitos estão mostrados no modelo de casos de uso da Figura 2.1.

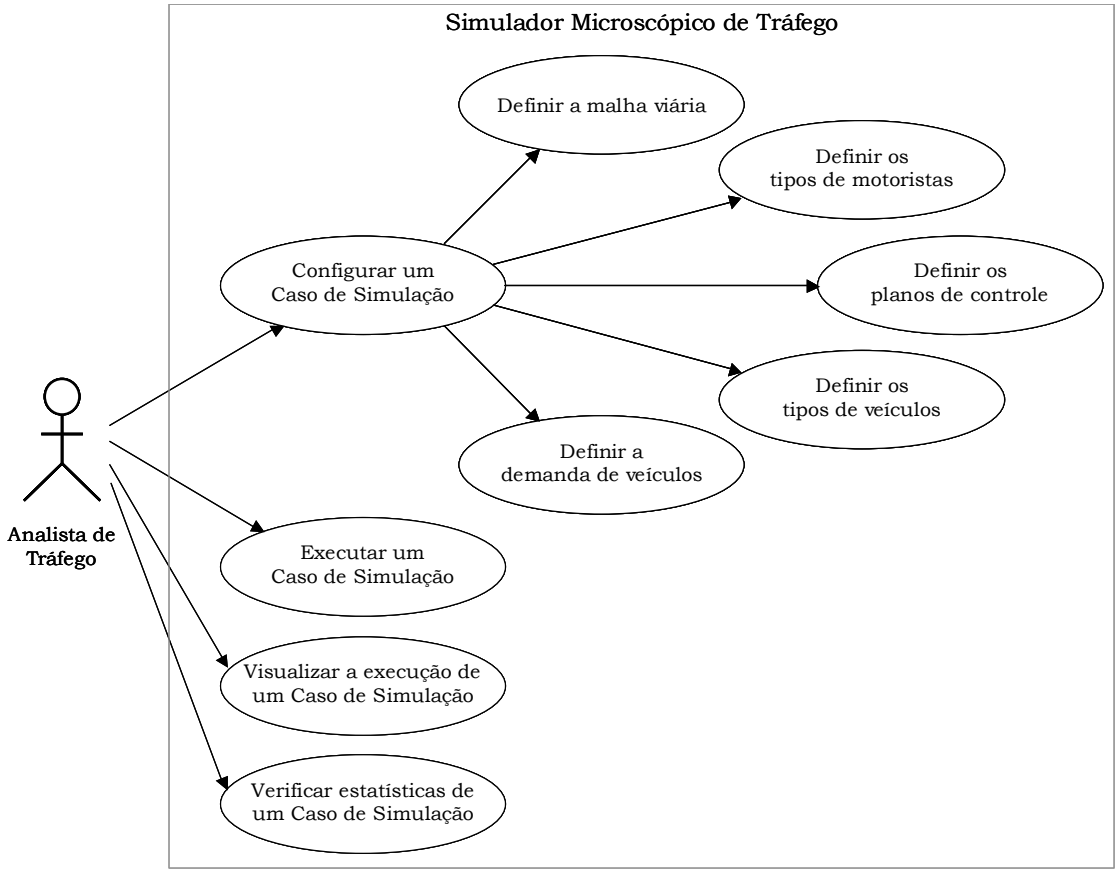


Figura 2.1 – Modelo de Caso de Uso para o Simulador Microscópico de Tráfego

Na próxima seção são mostrados os modelos tradicionalmente empregados na modelagem do comportamento dos veículos e motoristas na simulação microscópica de tráfego. O detalhamento dos componentes do simulador está na Seção 2.3 Proposta do simulador microscópico de tráfego.

2.2 Modelos de simulação microscópica

A simulação microscópica considera o par Veículo-Motorista como sendo uma única partícula cinética livre de movimentos. O movimento desta partícula é governado

por vários modelos matemáticos, como o modelo de perseguição, o modelo de mudança de pista e modelo de aceitação de lacunas [4]. Esses modelos guiam o comportamento individual dos veículos-motoristas na simulação microscópica.

2.2.1 Modelo de Perseguição

Os principais modelos da simulação microscópica de tráfego são os modelos de perseguição ou, no inglês, *car following*. Estes modelos consideram que há uma correlação direta entre o veículo estudado e o veículo que ele segue. Rothery considera que essa relação ocorre em uma faixa de pista de 0 à cerca de 100 ou 125 metros [8]. Cada motorista em um veículo é um elemento de controle ativo e de comportamento previsível. Eles executam tarefas que requerem habilidades psicomotoras para uma resposta contínua aos estímulos que se submetem: percepção, tomada de decisão e controle. Cada motorista procura adaptar-se ao veículo da frente e evitar colisões. A forma com que ele alcança esses objetivos depende de sua habilidade e experiência, mas, em geral, pode-se dizer que o estímulo é dado pela velocidade relativa e a distância entre os veículos. O motorista do veículo de trás, ou o perseguidor, tenta manter uma distância de segurança ao veículo da frente, chamado líder, e velocidade relativa igual a zero, ou seja, uma vez em uma distância de segurança, ele tenta não se afastar ou se aproximar do líder (Figura 2.2).

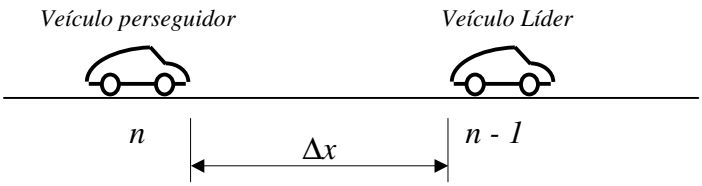


Figura 2.2 – Modelo de Perseguição

Em geral os modelos de perseguição se baseiam na seguinte relação [8]:

$$a_n(t+T) = f(v_n(t), \Delta x_n(t), \Delta v_n(t)) \qquad (2.1)$$

ou seja, a resposta do veículo no tempo $t + T$ é função do estímulo sofrido por ele no tempo t. A aceleração do veículo perseguidor (a_n) é função de sua velocidade (v_n), da distância ao líder (Δx_n) e da diferença de velocidade em relação ao líder (Δv_n). T pode ser entendido como o tempo de resposta do motorista.

Diferentes implementações do modelo de perseguição ocorrem a partir de variações desta relação. Por exemplo, o modelo Seguindo o Líder (*follow the leader*) usa somente a diferença entre as velocidades dos veículos [9 apud 5]:

$$a_n(t+T) = \frac{1}{\tau}(v_{n-1}(t) - v_n(t)) \qquad (2.2)$$

onde τ é o parâmetro de configuração da escala de tempo. O modelo de Velocidade Ótima representa a variação da velocidade a partir da velocidade desejada [10 apud 5]:

$$a_n(t+T) = \frac{1}{\tau}(V_n^d(t) - v_n(t)) \qquad (2.3)$$

onde V_n^d é a velocidade desejada, escolhida não somente pelo motorista seguidor mas também como função da velocidade do veículo líder.

Todosiev [11 apud 8] e Wiedemann [12 apud 13] introduziram considerações psico-fisiológicas nos modelos de perseguição. Wiedemann considerou diferentes regiões de comportamento dos motoristas. A idéia básica de seu modelo é de que as reações do motorista dependem do estado do veículo. Se o estado do veículo se altera, o motorista deve reagir, como, por exemplo, imprimindo um novo valor para a aceleração. O modelo utiliza uma série de regiões (ou regimes) de direção e fronteiras entre as regiões para descrever o movimento dos veículos [4]. A Figura 2.3 mostra um exemplo de fronteiras e regiões consideradas neste modelo.

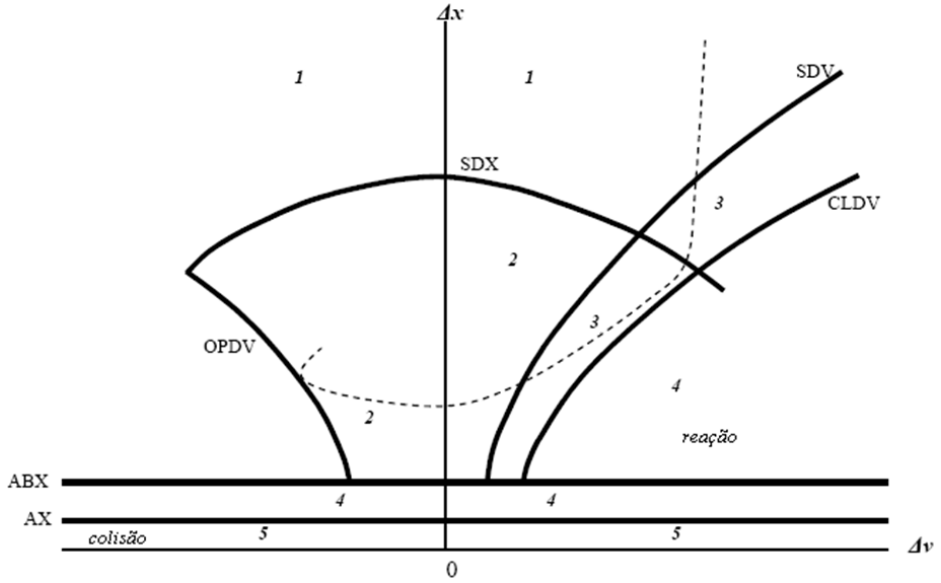


Figura 2.3 – Modelo de Perseguição de Wiedemann. Extraída de [4].

Observando-se a Figura 2.3 é possível verificar as seguintes regiões de direção:

1. O veículo líder não tem influência no seguidor, ou seja, é uma região de tráfego livre, ou *free driving*. Nesta região o motorista procura alcançar e manter a sua velocidade individualmente desejada;
2. Zona de perseguição, ou *car following*. O motorista dirige para seguir o líder sem mudanças de aceleração;
3. Aproximação, ou a*pproaching*. É o processo de adaptar a própria velocidade à velocidade mais baixa do líder. Nesta região o motorista aplica uma desaceleração de modo que a diferença de velocidade para o líder seja zero no momento que ele alcançar sua distância de segurança;
4. Zona de emergência, ou *braking*. Aplicação de média a altas taxas de desaceleração;
5. Colisão.

As Fronteiras da Figura 2.3 são:

- SDV: limiar de percepção de diferença de velocidade. Acima de SDV não há percepção do líder pelo seguidor;
- CLDV: limiar de percepção de pequenas diferenças de velocidade em pequenas e decrescentes distâncias;
- AX: distância entre veículos em condições de paradas;
- ABX: distância confortável entre veículos em condições de paradas ou distância de perseguição mínima;
- SDX: limiar de percepção de aumento de distância no processo de perseguição;
- OPDV: limiar de percepção para pequenas diferenças de velocidade em pequenas mas crescentes distâncias.

Diferentes autores (ver [13]), expandindo as definições de Wiedemann, sugerem outros estados de direção como: *free driving I*, *free driving II*, *approximating I*, *approximating II*, *following I* e *following II*.

Outros modelos de perseguição já foram desenvolvidos. Para um maior detalhamento, ver [5].

2.2.2 Modelos Mudança de Pista

Os Modelos de Perseguição guiam o movimento longitudinal dos veículos em uma estrada de pista (faixa) única. Para o movimento lateral dos veículos em uma estrada com várias pistas a abordagem microscópica utiliza outro modelo: o Modelo de Mudança de Pista, ou *lane change*. Modelos de Mudança de Pista tentam capturar as ações tomadas pelos motoristas durante uma mudança de pista.

Mudanças de pista podem ocorrer por motivos obrigatórios (por exemplo, a pista em que se encontra o veículo pode estar bloqueada, duas pistas se combinam em uma – como na Figura 2.4c – ou porque o destino do motorista requer a troca de pista – Figura 2.4b) ou motivos arbitrários (como para uma ultrapassagem). A Figura 2.4 mostra algumas situações de mudança de pista. Modelos de mudança de pista definem como ocorre a mudança de pista de um veículo, levando em consideração questões de segurança, trajetória e condições do tráfego.

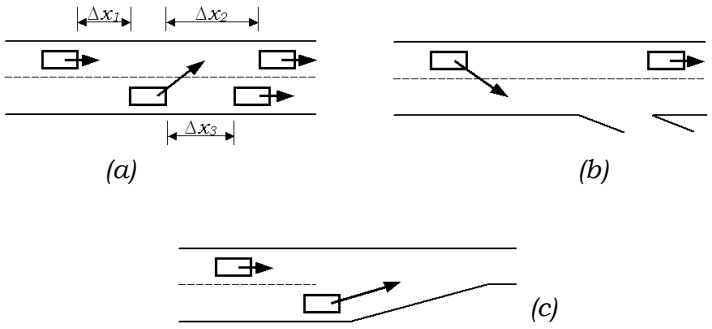
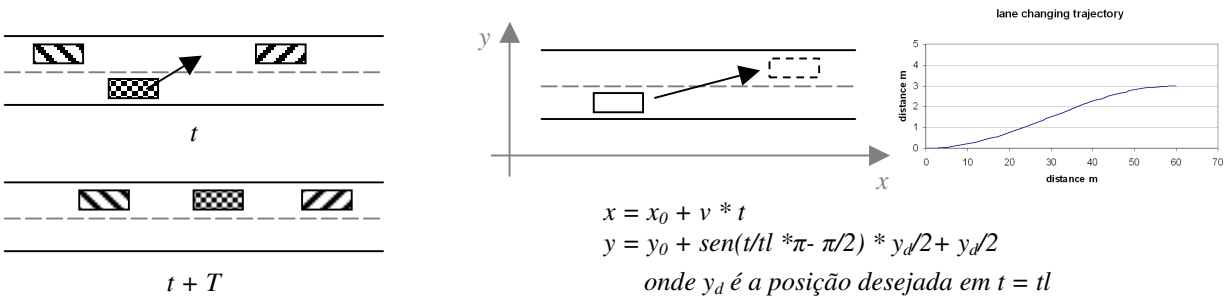


Figura 2.4 – Situações de mudança de pista

Modelos de mudança de pista elementares simplesmente implementam a troca de pista em instantes diferentes (Figura 2.5a). Modelos mais apurados, como em [14], consideram também a trajetória envolvida na mudança (Figura 2.5b).



$x = x_0 + v * t$

$y = y_0 + sen(t/tl * \pi - \pi/2) * y_d/2 + y_d/2$

onde y_d é a posição desejada em $t = tl$

(a) *(b) Extraída de [14]*

Figura 2.5 – Trajetória dos veículos no modelo de mudança de pista

2.2.3 Modelos de Aceitação de Lacunas e Escolha de Rota

Durante uma mudança de pista ou quando tenta entrar no fluxo de veículos, o mais importante para o motorista é verificar se é seguro ou não realizar essa manobra. Para isso, ele considera as distâncias críticas para os outros veículos envolvidos. Um outro modelo utilizado na abordagem microscópica trata dessa questão: o modelo de Aceitação de Lacunas, ou no inglês, *gap acceptance*. O Aceitação de Lacunas modela a absorção de um veículo de um fluxo em outro fluxo, o que pode ocorrer, por exemplo, por uma mudança de pista (Figura 2.4a) ou pela entrada do veículo a partir de uma pista lateral (Figura 2.6).

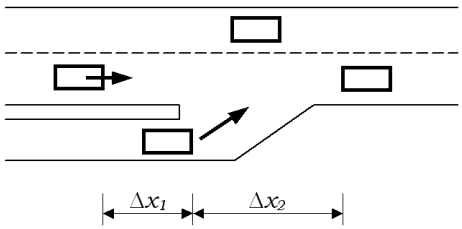


Figura 2.6 – Situação tratada pelo modelo de aceitação de lacunas

Outro modelo utilizado nas simulações microscópicas de tráfego é o Escolha de Rota. Ele é usado para representar como os motoristas escolhem os caminhos para trafegarem de um ponto a outro em uma malha viária.

A partir da próxima seção são detalhadas as soluções implementadas no simulador.

2.3 Proposta do simulador microscópico de tráfego

Os simuladores atuais podem contar o estado da arte do desenvolvimento de software orientado a objetos e de avançadas interfaces gráficas, além das novas tendências no projeto de software e de inúmeras ferramentas que o suportam. O simulador desenvolvido neste trabalho foi implementado utilizando-se a linguagem de programação Java no ambiente de desenvolvimento Eclipse Workbench©.

Para atender aos requisitos descritos na Seção 2.1 Especificação do Simulador, os módulos que devem ser implementados no simulador são:

1. Um módulo responsável pela interface gráfica, que é o ponto de partida para o usuário configurar e executar os casos de simulação e realizar as análises do tráfego;
2. Um módulo responsável pelas informações estatísticas;
3. Módulos responsáveis por cada definição de entrada do simulador: malha viária, veículos, motoristas e plano de controle e outro módulo que coordena as interações entre esses módulos;
4. Um módulo que inicia e coordena todas as atividades do simulador, aqui chamado de Principal.

Assim, o diagrama de classes dos principais módulos do simulador está ilustrado na Figura 2.7:

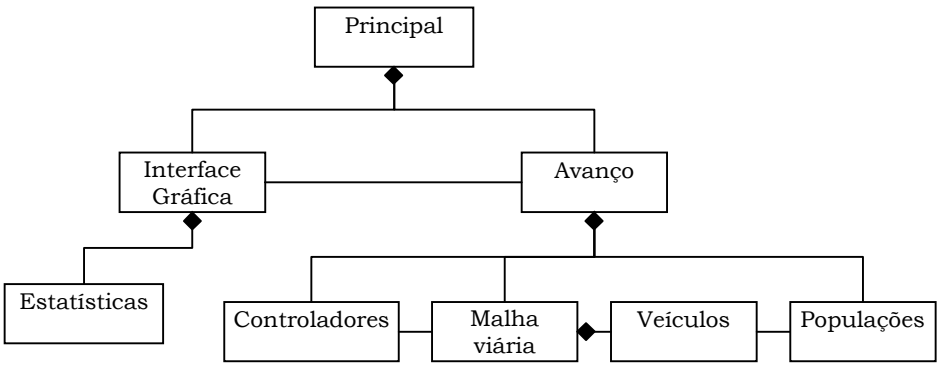


Figura 2.7 – Projeto geral do simulador de tráfego microscópico

O simulador contém um módulo responsável pela configuração, início e controle de toda a simulação e dos outros módulos. Ele está representado na figura pela caixa **Principal**. Outro módulo é responsável pela **Interface Gráfica**, representado na figura pela caixa de mesmo nome. O módulo **Avanço** é responsável por avançar o tempo de simulação e coordenar as atividades de interação entre a malha viária, os motoristas, veículos e os objetos do plano de controle da via.

Antes de detalhar cada módulo é importante entender os conceitos de tempo usados neste simulador. Neste trabalho foram assumidas as definições de Fujimoto [15] para o tempo em uma simulação:

- Tempo físico: é o que se refere ao tempo do sistema físico, ou seja, do tráfego;
- Tempo de simulação: é uma abstração usada pelo simulador para modelar o tempo físico. O tempo de simulação é um conjunto ordenado de valores onde cada valor representa um instante de tempo no sistema físico sendo modelado;

- Tempo de parede (*wallclock time*): refere-se ao tempo durante a execução do programa de simulação. Um programa de simulação geralmente pode obter o valor corrente do tempo de parede lendo o relógio do sistema operacional.

Nas próximas subseções são detalhados os módulos do simulador.

2.3.1 Interface Gráfica

O módulo de Interface Gráfica deve permitir ao usuário acompanhar a movimentação dos veículos na malha viária. Como já dito, a animação gráfica da simulação não é somente uma funcionalidade desejável mas uma poderosa ferramenta de análise e entendimento dos resultados. A interface gráfica do simulador foi desenvolvida com a biblioteca SWT (*Standard Widget Toolkit*) do Eclipse Workbench©. Algumas facilidades foram incorporadas à interface gráfica do simulador, como zoom, velocidade da animação e seleção da taxa de atualização da tela.

Toda a apresentação visual de um caso de simulação é feita pela interface gráfica, por isso, o processamento em si, ou o avanço no tempo do caso de simulação, só é realizado pelo módulo **Avanço** a partir de requisições da **Interface Gráfica**. A Interface Gráfica solicita ao Avanço que avance[1] os veículos em um tempo físico específico (configurável pelo usuário do simulador), por exemplo de 0,5 segundos. A partir daí a Interface Gráfica suspende seu processamento (*sleep*) por um tempo de parede (também configurável pelo usuário do simulador), por exemplo de 0,1 segundos. Durante o período em que a Interface Gráfica suspende seu processamento (que é na verdade o tempo de atualização da tela), o Avanço deve avançar todos os veículos do caso de simulação. Duas situações podem ocorrer:

1. O Avanço é capaz de avançar todos veículos no tempo de parede solicitado. Neste caso, quando a Interface Gráfica retorna, todos veículos já foram avançados e ela pode exibir os veículos;
2. O Avanço não é capaz de avançar todos veículos no tempo de parede solicitado. Neste caso a Interface Gráfica exibe uma mensagem ao usuário solicitando ajuste nas configurações de velocidade de animação (atualização da tela), geralmente diminuição da freqüência de atualização da tela.

A Figura 2.8 ilustra o funcionamento da Interface Gráfica nas duas situações descritas.

[1] O termo **avançar** aqui significa ser processado, ou, avançar no tempo, independentemente de ir à frente ou não.

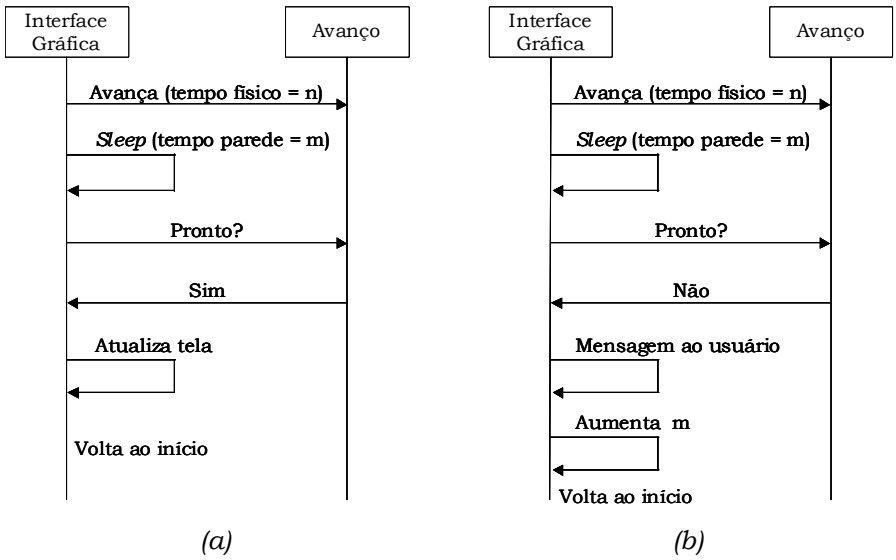


Figura 2.8 – Diagrama de Seqüência entre Interface Gráfica e Avanço: (a) tempo suficiente para o Avanço; (b) necessidade de ajustes na configuração

2.3.2 Avanço

No início de um caso de simulação, o módulo Avanço recebe as requisições de criação dos objetos da simulação. Após essa criação, o caso de simulação pode ser iniciado pelo usuário. O papel do módulo Avanço é incrementar, ou avançar, o tempo de simulação, atualizando o estado de todos os objetos do caso de simulação. Ele é responsável pelos objetos da Malha Viária, pelo comportamento individual dos veículos/motoristas e pelos objetos do Plano de Controle.

2.3.3 Malha viária

Uma malha viária é representada no simulador usando-se os seguintes objetos:

- Pistas: são a menor divisão da malha viária. Representam uma faixa da via por onde os veículos trafegam. Apenas um veículo pode ocupar uma posição (x) na Pista a cada tempo de simulação;
- Objetos de Via: são caminhos direcionais compostos de Pistas. Caracterizam-se por: geometria, posição (coordenadas x, y), comprimento, sentido do trânsito, número de pistas, largura das pistas e velocidade máxima. Cada Objeto de Via mantém uma lista de Pistas. Todas as pistas de um Objeto de Via devem ter a mesma largura e direção. O Objeto de Via é, na verdade, uma classe abstrata. Objetos de via concretos devem herdar desta classe. Neste simulador dois objetos concretos foram implementados:

- Ramo: é uma via em linha reta com sentido único;
- Interseção: representa uma interseção de dois Ramos. Todas Interseções neste simulador são sinalizadas (contém semáforos).

As simplificações da implementação das duas classes citadas acima não invalidam o objetivo principal deste trabalho. Todavia, é importante considerar as seguintes observações:

- Os objetos de via implementados não aceitam geometrias curvas, inclusive na junção entre eles;
- O ângulo dos objetos de via deve ser 0º ou 90º;
- Não há objetos de via de mão-dupla. A mão-dupla pode ser primitivamente implementada com dois Ramos lado a lado e com sentidos opostos. Essa solução, entretanto, não se aplica a mão-dupla em Interseções;

Futuras implementações do simulador podem estender os conceitos do Objeto de Via para outros objetos além de Ramo e Interseção e diminuir ou eliminar as observações citadas. Para o objetivo principal deste trabalho, essas limitações não são essenciais.

Como exemplo, na Figura 2.9 estão representados: o sistema de coordenadas, um Ramo R contendo três Pistas conectado a uma Interseção I com três Pistas na vertical e duas Pistas na horizontal. O sentido dos Objetos de Via é representado pelas setas.

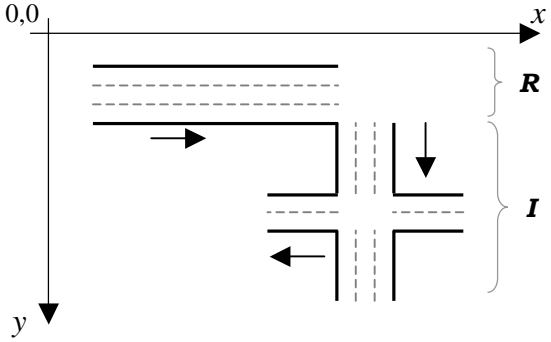


Figura 2.9 – Representação de uma malha viária

O simulador mantém uma lista com todos os objetos de via definidos pelo usuário para um caso de simulação. Em cada tempo de simulação, todos os objetos de via devem ser avançados no tempo.

A especificação da malha viária para um caso de simulação, assim como dos Veículos, Populações e Semáforos, descritos adiante neste capítulo, deve ser feita

pelo usuário através de um arquivo de configuração. A especificação deste arquivo de configuração está mostrada no Anexo A.

2.3.4 Modelagem do comportamento individual

Dois módulos tratam da modelagem do comportamento individual dos veículos e motoristas: Veículos e Populações. Como será descrito adiante nesta subseção, não há um módulo específico para os motoristas. Seu comportamento foi incorporado ao dos veículos.

2.3.4.1 Veículos

Um veículo no simulador é uma instância da classe Tipo de Veículo. O usuário do simulador especifica Tipos de Veículos usando as seguintes propriedades:

- Comprimento e largura, dados em metro (m);
- Velocidade máxima, dada em Km/h;
- Aceleração máxima, desaceleração máxima e desaceleração normal, dadas em Km/h/s. A desaceleração máxima equivale à redução de velocidade em situações de perigo enquanto a desaceleração normal é usada para reduzir a velocidade do veículo de forma suave em situações não emergenciais;
- Uma cor para identificação na animação gráfica.

Outras duas propriedades caracterizam os veículos. Elas são descritas a seguir.

2.3.4.2 Motoristas

Para o comportamento do motorista, três propriedades podem ser especificadas:

- Distância de segurança (DS): é a distância mínima ao veículo líder considerada segura pelo motorista. Como essa distância é função das velocidades dos veículos, ela é comumente especificada em segundos (s) e significa na verdade o tempo que o veículo perseguidor tem até alcançar a posição do veículo líder mantendo-se a velocidade dos dois veículos. Neste simulador, o usuário pode especificar essa distância também em metros;
- Velocidade desejada (VD): é a velocidade desejada pelo motorista quando este não é influenciado por nenhum outro veículo. É especificada em Km/h;

- Distância de tráfego livre (DTL): é a máxima distância para que o veículo perseguidor seja influenciado pelo líder. Se a distância entre os veículos é maior que a distância de tráfego livre, considera-se que o líder não tem influência sobre o perseguidor.

Apesar de caracterizarem os motoristas, estas propriedades foram incorporadas ao módulo Veículo por uma definição de projeto.

2.3.4.3 Populações

Populações modelam a diversidade dos veículos no trânsito. Em uma População pode-se especificar a taxa de ocorrência de cada tipo de veículo e o tempo em que cada veículo entrará na simulação. Veículos entram na simulação a partir de Objetos de Via associados a uma População. Quando um Objeto de Via é associado a uma População, isto indica uma fronteira de entrada de veículos na simulação. Na Figura 2.10, os veículos entram na simulação pela esquerda do Ramo R1, percorrem R1, passam para R2, percorrem R2 e deixam a simulação. O Ramo R1 deve ser associado a uma População.

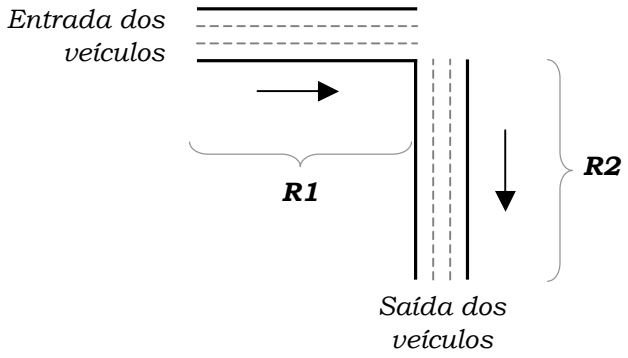


Figura 2.10 – Entrada e saída dos veículos na simulação

O simulador mantém uma fila de veículos para cada Pista: a fila de Veículos Ativos. Ela é ordenada pela posição do veículo na Pista. Para os Objetos de Via associados a uma População, outra fila de veículos é mantida: a fila de veículos da População, ou Fila de Espera, com o tempo de entrada de cada veículo.

No início de uma simulação, a fila de veículos da População é gerada, considerando-se os tipos de veículos e os tempos de entrada na simulação, enquanto as filas de veículos ativos estão vazias. Na execução de um caso de simulação, quando o tempo da simulação atinge o tempo de entrada de um veículo, este passa da fila de veículos da Espera para a fila de veículos ativos desde que haja distância suficiente (DS) para o veículo da frente.

2.3.4.4 Movimento dos veículos

O movimento dos veículos neste simulador é baseado no modelo de perseguição. Não foram implementados os modelos de mudança de pista, aceitação de lacunas e escolha de rota. Assim como para a malha viária, essa simplificação não invalida o atendimento do objetivo deste trabalho.

O modelo de perseguição usado segue as idéias de Wiedemann [12] e é baseado na distância e velocidade relativas do veículo perseguidor ao veículo líder. Dependendo da magnitude dessa distância e velocidade relativa, um veículo pode estar em uma das três seguintes regiões de direção: tráfego livre (*free driving*), seguindo o líder (*car following*) ou freando (*braking*). Essas regiões de direção são identificadas pelas seguintes características e ações:

- Tráfego livre (*free driving*):
    - não há líder, ou
    - a distância para o líder é maior que a distância de tráfego livre (DTL);
        - ♦ Ação: atingir e manter a velocidade desejada (VD).
- Seguindo o líder (*following*):
    - a distância para o líder é menor que a distância de tráfego livre (DTL) e maior ou igual à distância de segurança (DS);
        - ♦ Ação: ajustar a velocidade relativa ao líder para zero e tentar manter a distância entre DS e DTL usando a aceleração máxima para aumentar e a desaceleração normal para reduzir a própria velocidade.
- Freando (*braking*):
    - a distância para o líder é menor que a distância de segurança (DS);
        - ♦ Ação: aplicar a desaceleração máxima.

A identificação das regiões de direção é feita para cada veículo dos objetos de via. Como somente o modelo de perseguição foi implementado, os veículos somente podem ser influenciados por outros veículos que estejam no mesmo objeto de via.

Resposta a eventos

O modelo de perseguição captura o comportamento do veículo em resposta ao líder. Porém, nas decisões de quando acelerar ou desacelerar, outras restrições além das impostas pelos regimes de direção são estabelecidas por eventos e devem ser

consideradas. O evento implementado neste simulador é a mudança de estado (cor) dos semáforos. Antes de calcular a aceleração baseada no modelo de perseguição, o controle do Avanço verifica a influência deste evento na direção dos motoristas.

De uma forma geral, os eventos devem ser considerados restrições à aceleração se seus geradores estiverem dentro do campo visual do motorista e a distância do veículo ao gerador do evento (por exemplo o semáforo) é menor que a distância necessária para o veículo, usando desaceleração normal, atingir a velocidade requerida (por exemplo, zero para semáforo vermelho), ou seja:

$$x < \frac{\Delta v^2}{2a^-} \qquad (2.4)$$

onde x é a distância do veículo ao gerador do evento, Δv é a diferença de velocidade entre o veículo e a requerida e a^- é a desaceleração normal. Se o gerador do evento está a essa distância, o veículo se prepara para frear usando a seguinte taxa de desaceleração:

$$a = \frac{-\Delta v^2}{2x} \qquad (2.5)$$

2.3.5 Planos de controle de tráfego

Um único controlador de tráfego é implementado no simulador: o semáforo temporizado, ou com tempo definido de cada estado. Antes de avançar cada veículo, o módulo Avanço atualiza o estado de cada semáforo, baseado no tempo de simulação.

Os semáforos são caracterizados pelos seguintes parâmetros:

- Objeto de Via: referência ao elemento da malha viária que contém o semáforo;
- Posição: dada em metros (m), é a distância do semáforo ao início do Objeto de Via;
- Tempo, em segundos (s), de cada estado (verde, amarelo e vermelho);
- Estado inicial (verde, amarelo ou vermelho) para o início do caso de simulação.

2.3.6 Avanço do tempo

Há dois métodos para implementar avanços de tempo nos modelos de simulação microscópicos: orientado a eventos ou de tempo particionado (*time-stepped*).

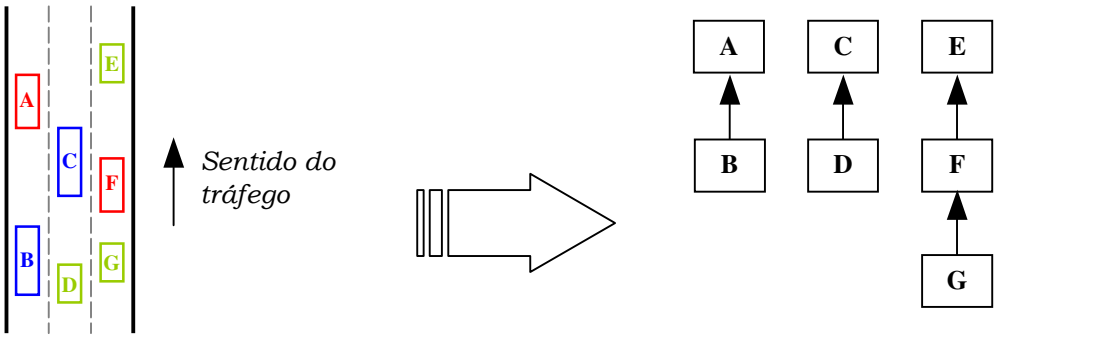
No método de tempo particionado, o tempo de simulação entre o início e fim do caso de simulação é subdividido como uma seqüência de passos de tempos de mesmo tamanho. Em cada passo, o estado de todos objetos da simulação é recalculado [4].

No método orientado a eventos, o simulador avança no tempo interpretando eventos de uma lista de eventos. Um evento incorpora algumas ações (mudanças no estado da simulação) e um tempo (que determina quando as ações devem acontecer). Essa abordagem geralmente não é usada em aplicações de tráfego [27]. O método implementado neste simulador é o de tempo particionado.

A seleção do tamanho da subdivisão do tempo é um quesito importante para a qualidade da simulação. Tempos pequenos permitem uma modelagem mais realista e precisa, mas requerem um poder de processamento maior. Uma grande parte dos modelos implementados e descritos na literatura utiliza porções de tempo que variam de 0,1 a 1,0 segundo. No simulador proposto, esse é um parâmetro que pode ser configurável pelo usuário dentro dos limites de 0,2 a 0,5 segundos.

2.3.7 Precedência dos veículos

Como visto, a estrutura primária da malha viária é a Pista. As Pistas são agrupadas em Objetos de Vias, que podem ser Ramos ou Interseções. Para uma dada definição de malha viária, o simulador mantém uma única fila de Objetos de Vias. Cada Objeto de Via tem sua fila de Pistas e cada Pista tem sua fila de Veículos. Um exemplo de fila de veículos é mostrado na Figura 2.11:



Sistema físico:
Um Ramo, três Pistas e sete Veículos.

Filas de Veículos para o Ramo ao lado

Figura 2.11 – Exemplo de filas de veículos para um Ramo

As Pistas são processadas avançando-se, para cada tempo de simulação, os veículos de suas filas, conforme o modelo apresentado na Subseção 2.3.4.4

(Movimento dos Veículos). Quando um veículo atinge o final da Pista, ele passa para a Pista correspondente do próximo Objeto de Via ou é retirado da simulação caso não haja mais Objetos de Vias a percorrer (sai da região sendo simulada). Este processamento pode ser visto de forma simplificada na Figura 2.12.

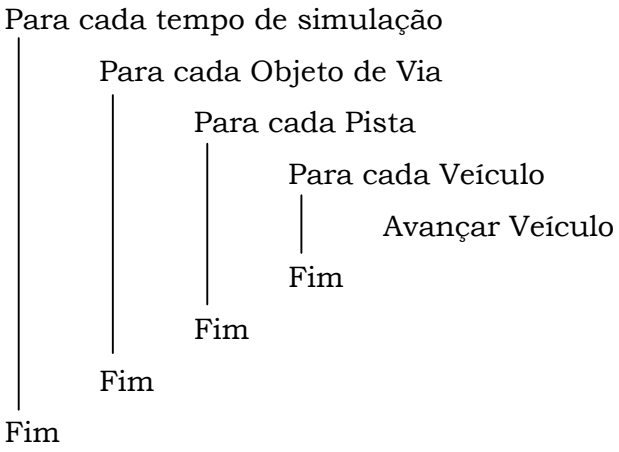


Figura 2.12 – Pseudocódigo resumido para o avanço dos veículos

É possível notar que existe uma ordem de precedência no movimento dos veículos. No tráfego, dados dois veículos, o motorista do veículo de trás dirige mirando-se no movimento do veículo da frente, a menos que este último esteja suficientemente distante a ponto de não preocupá-lo. No simulador, em uma fila de veículos de uma Pista, dados dois veículos – líder e perseguidor – o perseguidor só poderá avançar em duas situações:

i. Condicionado ao avanço do líder; ou

ii. Não condicionado ao avanço do líder, se ele está em regime de Tráfego Livre.

Essa ordem de precedência é obedecida na própria execução da Fila de Veículos. O primeiro veículo da Fila de Veículos corresponde ao primeiro veículo na Pista, considerando o sentido do tráfego. Os veículos são avançados do primeiro ao último na fila. Quando o veículo atinge o final da Pista e passa para a Pista seguinte, em outro Objeto da Via, o último veículo da Pista seguinte já deve ter sido avançado (em atendimento ao item (i) acima) ou não exercer influência sobre o veículo que chega (em atendimento ao item (ii) acima). Essa necessidade é atendida por duas premissas:

a. Os veículos da frente em relação ao sentido do tráfego são os primeiros a serem processados a cada tempo de simulação, ou seja, a ordem de processamento dos Objetos de Via deve ser do último ao primeiro no fluxo de veículos. Dessa forma a precedência dos veículos é obedecida.

Embora um algoritmo para detecção desta precedência seja factível e de fácil implementação, este simulador usa a definição da malha viária feita pelo usuário para determinar a ordem de processamento dos Objetos de Via. Quando carrega as definições da malha viária feitas pelo usuário no arquivo de configuração, o simulador carrega também a ordem de avanço dos Objetos de Via. O usuário deve especificar os objetos de via na ordem em que eles aparecem no fluxo. Como exemplo, seja a malha viária da Figura 2.13:

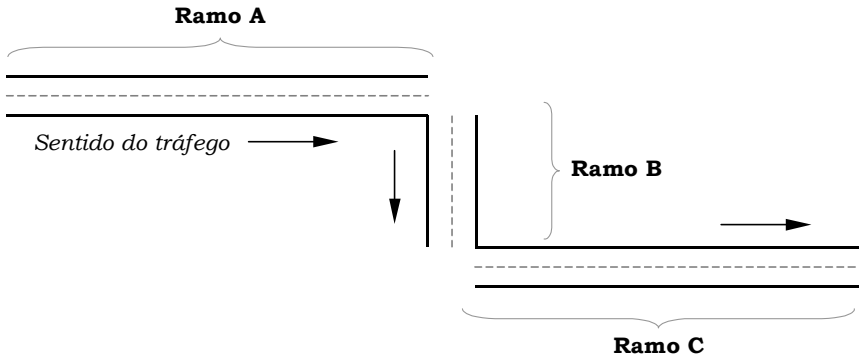


Figura 2.13 – Ordem na definição dos objetos de via da malha viária

A definição dos objetos de via desta malha viária no arquivo de configuração deve obedecer a ordem: Ramo A, Ramo B e por fim Ramo C

b. Não existe a relação líder-perseguidor entre o último veículo de um objeto de via e o primeiro veículo do objeto de via anterior no fluxo. A divisão da malha viária em objetos de via deve ocorrer de tal forma se evite situações de influência entre esses veículos. Na prática isso significa que para cada via de tráfego de geometria contínua (como uma reta) deve haver somente um Objeto de Via e quando há mudança de direção do tráfego (como uma curva fechada ou uma esquina) deve haver a divisão nos Objetos de Via.

A Figura 2.14 mostra dois exemplos de divisão da malha viária. Na Figura 2.14a, o Ramo R1 e a parte horizontal da Interseção I1 estão na mesma linha contínua de fluxo, o que também acontece com os Ramos R2 e R3. Essa divisão deve ser evitada. A melhor forma de dividir essa malha viária está mostrada na Figura 2.14b onde os objetos que têm a mesma direção não são divididos.

2.3.7.1 Avanço dos veículos a cada tempo de simulação

A cada tempo de simulação todos os veículos devem ser processados uma e somente uma vez. Se esse pré-requisito não for atendido o modelo resultante não corresponderá ao sistema físico. Basicamente, dois problemas podem ocorrer:

1. Se um veículo é processado mais de uma vez por tempo de simulação ele adianta-se em relação aos demais veículos e semáforos. No sistema físico isso seria equivalente a que o tempo parasse para todos os motoristas exceto para esse motorista específico, que avançaria seu veículo na pista enquanto os outros permaneceriam parados;
2. Se um veículo não é processado em um dado tempo de simulação ocorre o efeito oposto ao anterior, ou seja, ele fica defasado no tempo em relação aos demais veículos.

Esse pré-requisito deve ser aplicado a todos os veículos de um caso de simulação. Para atendê-lo, o simulador avança os veículos segundo o algoritmo já mostrado na Figura 2.12.

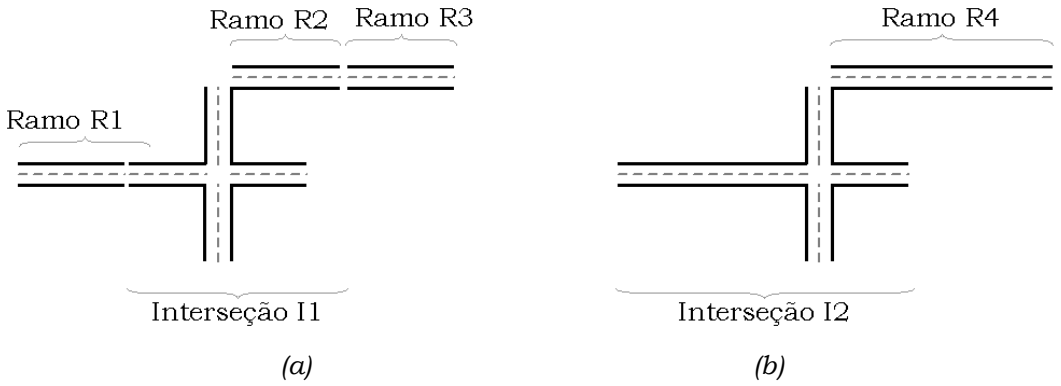


Figura 2.14 – Duas divisões de malha viária

2.3.8 Execução de um caso de simulação

O primeiro passo de um caso de simulação é a carga dos parâmetros de entrada através do arquivo de configuração: definição da malha viária, tipos de veículos, semáforos e populações. Em seguida os veículos são gerados e associados às filas de Espera de cada Objeto de Via. Os passos seguintes fazem parte de um processo iterativo:

1. Atualizar estado dos semáforos;
2. Carregar os veículos da fila de espera para a fila de ativos;
3. Calcula aceleração dos veículos;
4. Avança os veículos para as novas posições e atualiza suas velocidades;

5. Verifica se algum veículo atingiu o fim da pista e remove-o da malha rodoviária ou passa-o para a próxima pista;
6. Atualiza a interface gráfica;
7. Se não é o fim da simulação, avança o relógio da simulação e retorna ao passo 1.

A Figura 2.15 mostra o fluxograma do funcionamento do simulador.

O próximo capítulo descreve a implementação deste simulador em um ambiente distribuído.

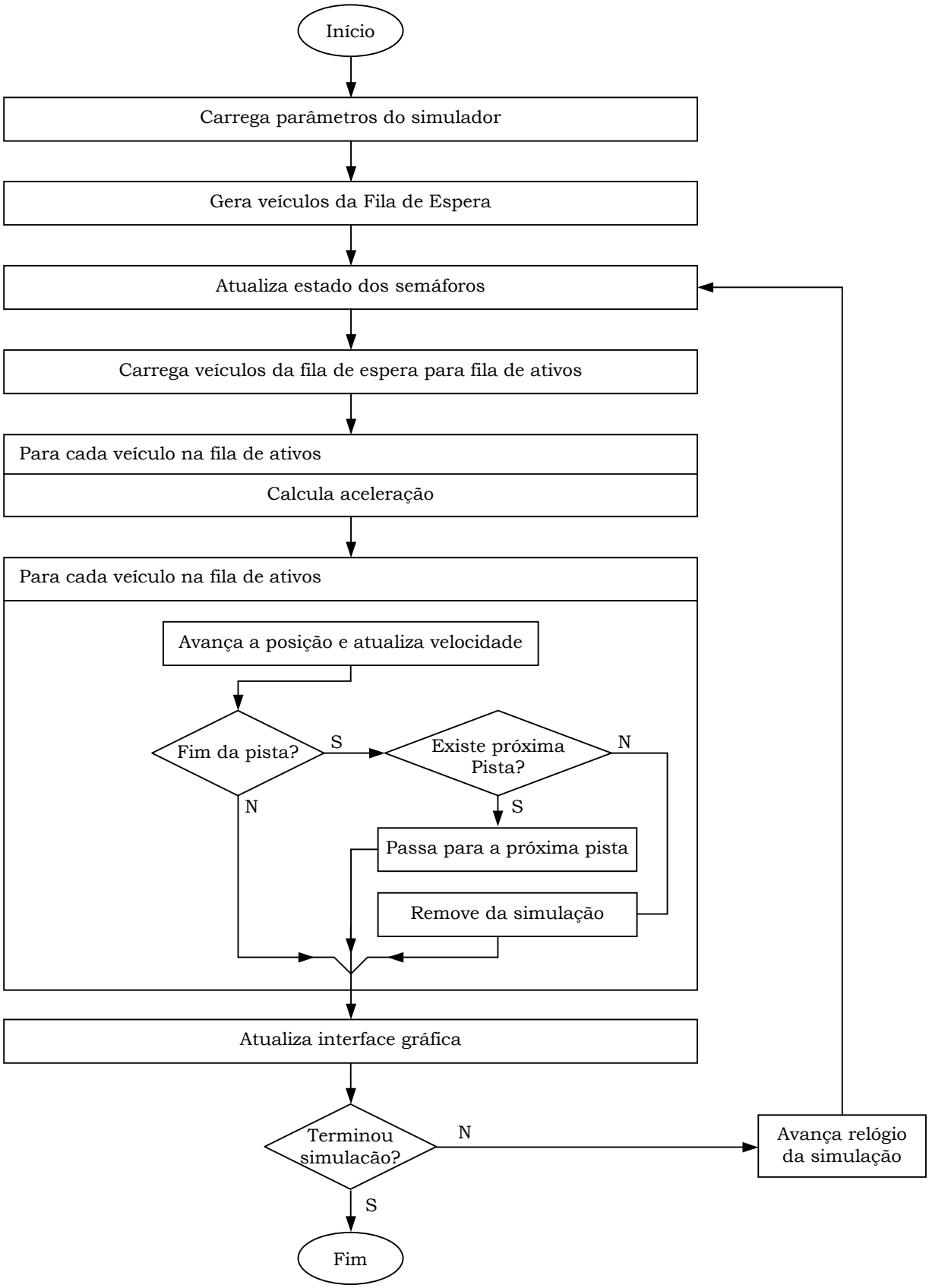


Figura 2.15 – Fluxograma de funcionamento do simulador

Capítulo 3

Simulador distribuído

Executar a simulação microscópica de tráfego de uma grande malha viária em um tempo aceitável é uma atividade que requer grande capacidade computacional. Uma alternativa mais barato ao uso de supercomputadores é a utilização de simulação distribuída. Subdividindo um programa de simulação em vários subprogramas e executando-os em vários processadores simultaneamente pode-se reduzir o tempo total da simulação.

Além do ganho da redução do tempo de processamento, outras vantagens podem ser apontadas com a distribuição da simulação [15]:

- Tolerância à falhas: se ocorrer uma falha em um dos processadores, pode ser possível que um outro processador faça o seu trabalho sem gerar interrupção na simulação;
- Distribuição geográfica: a execução de uma simulação em computadores distribuídos permite a criação de ambientes com múltiplos participantes localizados fisicamente em diferentes regiões. Os participantes podem interagir com outros como se estivessem juntos num mesmo centro, sem o desperdício de tempo, dinheiro e da necessidade de locomoção;
- Uso de computadores de diferentes fabricantes: com a utilização de vários computadores um programa de simulação não se restringe a somente um modelo de computador, o que faz com que a simulação seja garantida independente da situação do mercado;

Neste trabalho, o foco da distribuição é na redução do tempo de processamento. Para distribuir a execução do simulador microscópico de tráfego em vários processadores, três questões devem ser resolvidas:

1. A definição do meio de comunicação entre os processadores;
2. A decomposição do algoritmo da simulação em partes menores para execução nos vários processadores;
3. A sincronização entre os processadores.

Neste capítulo são apresentadas propostas de soluções para cada uma das questões acima.

3.1 Meio de para comunicação

A escolha do meio de comunicação compreende a seleção da plataforma de hardware e do modelo de programação. O hardware fornece a plataforma básica sobre a qual o simulador será executado. O modelo de programação indica a forma como os processadores vão se comunicar, independentemente do algoritmo de sincronização necessário.

3.1.1 Plataforma de hardware

A plataforma de hardware para simulação distribuída deve conter vários processadores interconectados por um sistema de comunicação. Essas plataformas podem ser classificadas de diversas maneiras. A classificação adotada aqui é a sugerida por Fujimoto [15] que considera duas categorias de computadores: a Paralela e a Distribuída:

- Nos computadores Paralelos os processadores são, na maioria das vezes, homogêneos, de um mesmo fabricante e geralmente de um mesmo modelo, e há normalmente um hardware específico para a comunicação entre os processadores (como um *switch* sob medida) que estão fisicamente próximos uns dos outros;
- Nos computadores Distribuídos os processadores geralmente pertencem a máquinas independentes com memória e dispositivos de Entrada/Saída

próprios. A comunicação entre os processadores utiliza padrões de comunicação comerciais como Ethernet ou ATM (*Asynchronous Transfer Mode*).

Para os computadores paralelos, uma das formas mais utilizadas de classificação é a de Flynn [16 apud 17]. Esta classificação considera duas dimensões independentes: Instrução e Dado e cada dimensão pode ter dois possíveis estados: Simples ou Múltiplo. Assim, as quatro possíveis classificações são as mostradas na tabela 3.1:

	Instruções Simples	Instruções Múltiplas
Dados Simples	SISD	MISD
Dados Múltiplos	SIMD	MIND

Tabela 3.1 – Taxonomia de Flynn

Outra forma de classificar os computadores paralelos é quanto a arquitetura de memória: compartilhada (vários processadores acessam toda memória disponível com endereçamento global) ou distribuída (cada processador acessa somente sua própria memória local e não há endereçamento global de memória). Alguns computadores já empregam uma arquitetura híbrida de memória compartilhada e distribuída [17].

Diferentemente dos computadores paralelos, os computadores distribuídos geralmente são compostos por estações de trabalho comuns, algumas vezes de diferentes fabricantes. A heterogeneidade diminui a necessidade de portar sistemas existentes para novas plataformas e, ao mesmo tempo, possibilita que usuários em equipamentos diferentes participem em casos de simulações distribuídas.

Outra característica dos computadores distribuídos é o menor custo de implementação. Todo o hardware especial empregado na construção de computadores paralelos faz com que essas máquinas sejam extremamente mais caras que a solução distribuída.

As plataformas de hardware mais utilizadas atualmente estão ilustradas na Figura 3.1. A plataforma de hardware escolhida para esse simulador é a distribuída. Ela consiste de estações de trabalho conectadas através de uma rede local de computadores (LAN).

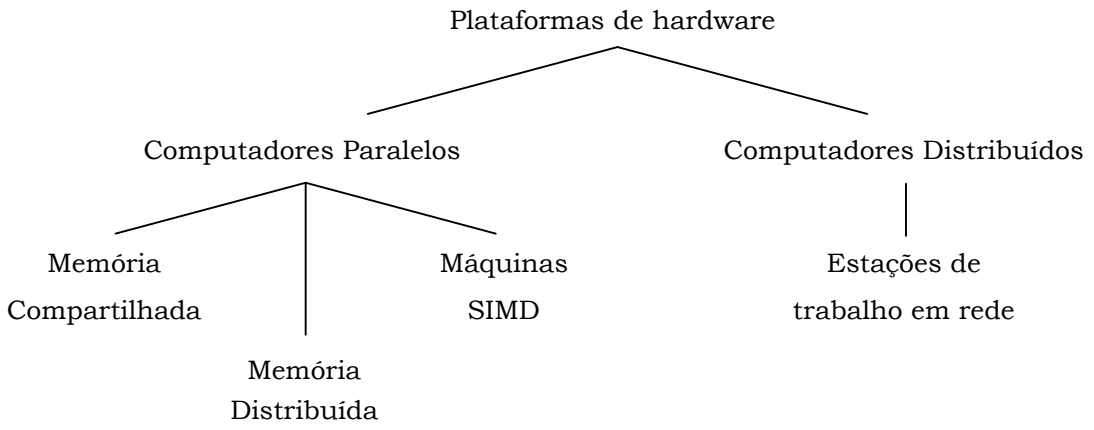


Figura 3.1 – Plataformas de hardware para simulação distribuída. Extraída de [15]

A única restrição para as estações de trabalho é que sejam capazes de executar programas em Java. É justamente a implementação em Java que permite que as estações de trabalho sejam heterogêneas tanto em relação ao processador quanto ao sistema operacional usado. A escolha da plataforma de hardware foi baseada principalmente nas seguintes vantagens:

- Facilidade de implementação: computadores pessoais são hoje facilmente encontrados no mercado e comumente usados em redes locais de computadores;
- Baixo custo quando comparado a outras soluções: não há necessidade de hardware especial para os computadores nem para a conexão física entre eles. Além disso, o custo da implementação é de certa forma proporcional ao ganho de desempenho, uma vez que este último cresce com a inclusão de novas estações de trabalho;
- Facilidade para execução do simulador: hoje em dia, praticamente todos os sistemas operacionais comerciais executam programas em Java e a dificuldade de portar o simulador para ser usado em outros sistemas operacionais é pequena;
- Independência do mercado e de fabricantes, dado que não há restrição de modelo de estação de trabalho.

As desvantagens que podem ser atribuídas a esta plataforma são em relação à suscetibilidade a falhas de comunicação, possivelmente maior que na paralela, e ao desempenho do simulador, uma vez que o uso hardware específico poderia trazer melhores resultados. Nenhum desses itens, entretanto, invalida o alcance dos objetivos deste trabalho.

3.1.2 Modelos de Programação Paralela

Modelos de programação paralela são uma abstração acima da camada de hardware e da arquitetura de memória. Não são específicos a um tipo particular de arquitetura da máquina ou de memória. Ainda que cada plataforma tenha seu método nato e mais eficiente de comunicação, teoricamente qualquer modelo pode ser implementado e executado em qualquer plataforma de hardware. Para exemplos de combinação de modelos de programação e plataformas de hardware, ver [17].

Os principais Modelos de Programação são: Memória Compartilhada (*Shared Memory*), *Threads*, Passagem de Mensagem (*Message Passing*), Dado paralelo (*Data Parallel*), SPMD (*Single Program Multiple Data*), MPMD (*Multiple Program Multiple Data*) e os híbridos (combinação de dois ou mais modelos). Na simulação distribuída, os modelos mais empregados são o de Memória Compartilhada e o Passagem de Mensagem [15].

No modelo de Memória Compartilhada as instruções dos programas compartilham um endereço de memória comum, em que elas lêem e escrevem de modo assíncrono (via comandos de *load* e *store*). Mecanismos de controle de acesso, como semáforos, devem ser usados para controlar o acesso à memória compartilhada (Figura 3.2).

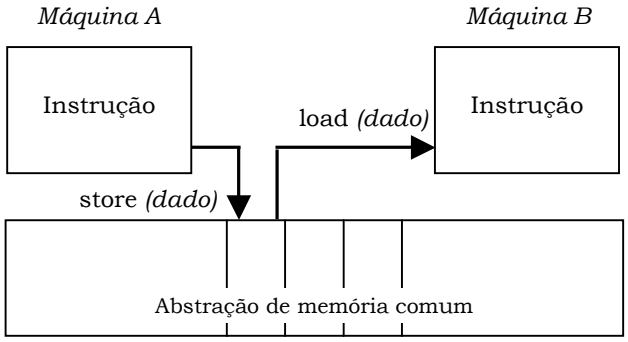


Figura 3.2 – Método de Memória compartilhada

No modelo de Passagem de Mensagem, os processadores se comunicam enviando mensagens uns aos outros (Figura 3.3). Essas mensagens são instruções de envio e recebimento (*send* e *receive*). Nelas estão contidas as informações que precisam ser compartilhadas pelos processadores. As instruções usam sua própria memória local durante a execução. A transferência de dados geralmente requer o

cumprimento de operações cooperativas, como uma mensagem de confirmação de recebimento (*acknowledge*).

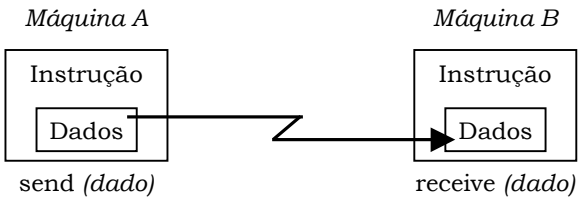


Figura 3.3 – Método de Passagem de Mensagem

O modelo de Passagem de Mensagem é o escolhido para esse trabalho, visto que o desenvolvimento do simulador foi feito usando-se o paradigma orientado a objetos em Java e o uso de mensagens é mais natural a esse modelo.

3.2 Arquitetura de Software

Antes de ver como é realizada a decomposição da simulação, é importante entender o modelo de coordenação de atividades escolhido para o simulador distribuído. O simulador distribuído é executado em uma arquitetura Mestre-Escravo. Um dos computadores é o Mestre. Ele é responsável pelo gerenciamento de toda a simulação e sincronismo das outras estações de trabalho. É no Mestre que se encontra o módulo da interface gráfica do simulador. Nos escravos os veículos e semáforos são atualizados, à medida que o tempo de simulação avança. A Figura 3.4 compara o projeto do simulador seqüencial (3.4a) descrito no Capítulo 2 com o projeto do simulador distribuído (3.4b).

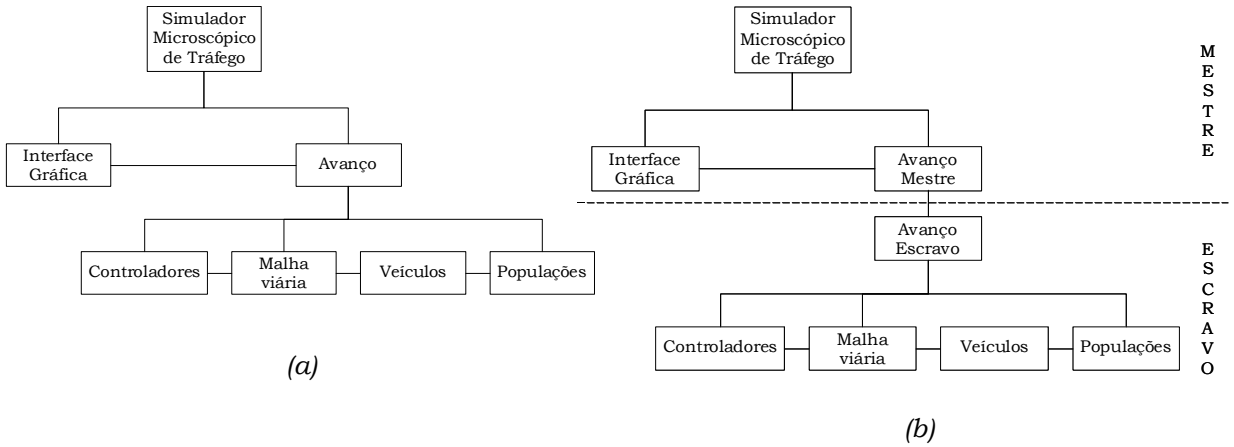


Figura 3.4 – Arquitetura do simulador: simulador seqüencial (a) e simulador distribuído (b)

3.2.1 Mestre

No Mestre estão os módulos de Interface Gráfica e Avanço Mestre. O módulo de Avanço Mestre é uma evolução do módulo Avanço do simulador seqüencial. Na versão distribuída, este módulo coordena os avanços realizados em cada computador que participa de um caso de simulação.

Um caso de simulação inicia quando Mestre lê o arquivo de configuração fornecido pelo usuário. Em seguida, ele envia mensagens para as estações Escravas requisitando a criação dos objetos da simulação especificados no arquivo. A distribuição dos Objetos de Via e Semáforos nas estações Escravas está descrita na Seção 3.3 Decomposição da simulação.

Após o processo de carga inicial e criação dos objetos nas estações, o Mestre enviará requisições de avanço dos objetos da simulação às estações Escravas passando como parâmetro o tempo (em milissegundos) a avançar. O Mestre então fica em estado de espera (*wait*) e somente envia a próxima mensagem de requisição de avanço quando todas as estações Escravas terminarem o processamento.

3.2.2 Escravos

Quando uma requisição de avanço chega a uma estação Escrava ela inicia o processamento dos objetos da simulação da seguinte forma:

1. Atualiza Semáforos;
2. Atualiza o tempo dos Veículos nas Populações;
3. Avança os veículos na malha viária.

Terminado esse processamento a estação sinaliza ao Mestre e volta ao estado de espera da próxima requisição de avanço.

No passo 1, o estado de cada Semáforo é atualizado considerando o tempo da simulação e o tempo de cada estado (cor) do semáforo. No passo 2 o tempo de entrada de cada veículo da fila da População é comparado com o tempo da simulação. Se o tempo de entrada do veículo é maior ou igual ao tempo da simulação, o Veículo é disponibilizado para entrar na simulação. Esses dois passos funcionam exatamente como na versão seqüencial do simulador.

O passo 3 é cerne do algoritmo de distribuição da simulação e o que requer maior atenção para implementação. Ele será detalhado nas seções seguintes.

3.3 Decomposição da simulação

Há, normalmente, três formas de decompor uma simulação: a decomposição funcional (ou paralelização de tarefas), a decomposição de domínio (ou espacial) e a decomposição temporal [18].

Na decomposição funcional, os diferentes módulos funcionais do programa de simulação são executados em computadores diferentes. Por exemplo, tomando como referência a Figura 3.4b, o módulo Controladores é executado em um computador, o módulo Malha Viária é executado em outro computador e assim por diante para cada módulo. Nesta estratégia, o módulo mais lento, ou o que necessita de maior esforço de computação, é que vai determinar a velocidade da simulação. Uma vez que a maior necessidade de processamento é do módulo que trata do avanço dos veículos (Malha Viária) e que este módulo não é decomposto facilmente, essa estratégica não é apropriada à simulação de tráfego.

Na decomposição temporal, o tempo total da simulação é subdivido em intervalos de tempos iguais. Cada computador executa a simulação de um intervalo de tempo paralelamente aos outros computadores. Na simulação de um intervalo de tempo pode não ser necessária a comunicação entre os computadores, entretanto, após a execução de todos os intervalos de tempo, o estado da simulação provavelmente não vai corresponder ao sistema físico. Neste caso, devem ser realizadas computações de correção (*fix-up computation*). Entretanto essa não é uma tarefa fácil para simulação de tráfego pois envolve atividades muito complexas, como a identificação de posição e velocidade de todos veículos, estados dos semáforos, etc, em todos os tempos de simulação já processados.

A decomposição espacial é a forma mais comum de decompor a simulação de tráfego [18]. Nesta estratégia, o tráfego é subdividido em regiões geográficas e a simulação de cada região é executada em um computador diferente, como na Figura 3.5. Cada computador executa a simulação completa da região que lhe foi atribuída.

Duas questões surgem com a decomposição espacial e devem ser resolvidas para atender à simulação de tráfego: como dividir a malha viária em regiões e como realizar a passagem de veículos entre regiões de diferentes computadores.

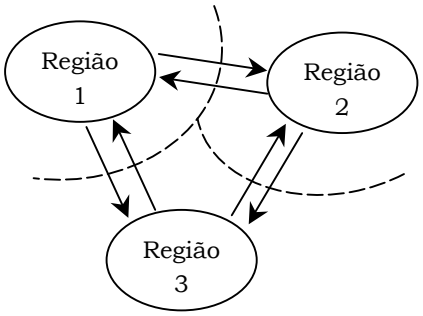


Figura 3.5 – Decomposição de domínio. Extraída de [19]

3.3.1 Divisão da malha viária

Usando a decomposição espacial, a divisão da malha viária deve ser tal que as regiões necessitem de um esforço computacional semelhante, caso contrário, a região que tiver o processamento mais lento determinará a velocidade da simulação.

Uma forma simples e direta de equilibrar o desempenho computacional das partes decompostas é divisão da malha viária em regiões de mesma área retangular. Entretanto, como o tráfego geralmente não é homogêneo em toda malha viária, a diferença na velocidade de simulação das regiões pode ser significativa. Outra abordagem é fazer a divisão da malha viária baseada na estimativa do número de veículos em cada região.

Alguns algoritmos podem ser desenvolvidos para identificar a melhor divisão da malha viária e fazer o balanço de carga entre os computadores, o que não é escopo deste trabalho. Neste simulador, a divisão da malha viária é realizada pelo próprio usuário na definição do arquivo de configuração de cada caso de simulação. Como exemplo, a Figura 3.6 mostra uma malha viária dividida em duas regiões separadas pela linha tracejada. Cada região é processada em uma estação Escrava diferente.

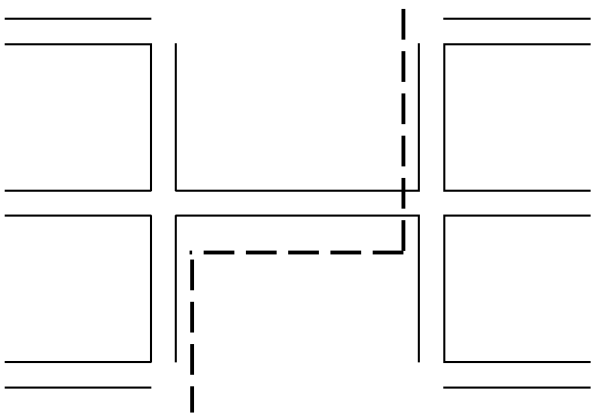


Figura 3.6 – Exemplo de divisão de uma malha viária

As mesmas observações da Subseção 2.3.8 (Precedência dos veículos) referentes à divisão da Malha Viária em Objetos de Via (item b) devem ser aplicadas para a divisão entre os processadores.

3.3.2 Paralelismo de atividades

Antes de entender como é feita a passagem de veículos entre os processadores, é apresentado o conceito de uma estrutura de dados que surge para aproveitar as oportunidades de paralelismo de atividades deste simulador: a Fila de Precedência.

A condição favorável ao paralelismo deste simulador está no fato de que os diferentes processadores podem processar paralelamente as Filas de Veículos. Entretanto, considere-se que, em um caso hipotético, cada veículo só possa avançar quando o veículo da frente avançar e a malha viária for um caminho único. Todos os veículos dependerão direta ou indiretamente do primeiro veículo do primeiro Objeto de Via. Como ilustração, considere-se a malha viária da Figura 3.7, contendo dois objetos de via, cada um em um computador.

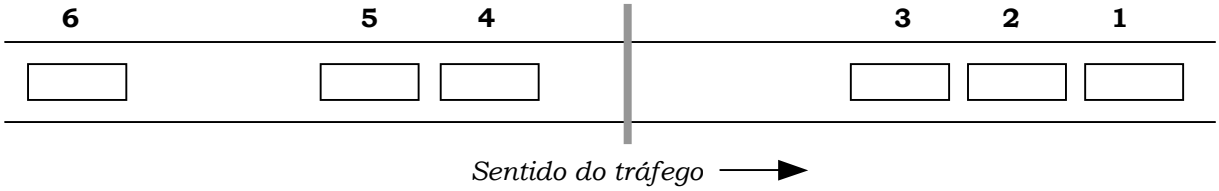


Figura 3.7 – Exemplo de malha viária

Considere-se que o veículo 2 deve aguardar o veículo 1 avançar, o veículo 3 aguarda o veículo 2 e assim por diante, o veículo 6 avança por último, após todos veículos avançarem e não há paralelismo. Neste caso, o ganho na distribuição das atividades será nulo. Como então aproveitar o uso de vários processadores na simulação para ganhar em desempenho? Para resolver essa questão foi desenvolvido o conceito de Filas de Precedência.

Filas de Precedência

De acordo com Modelo de Perseguição (Capítulo 2), em regimes de *following* (Seguindo o Líder) ou *stopping* (Freando), um veículo é influenciado pelo veículo líder e pode avançar somente após o líder ter avançado. Em regime de *free driving* (Tráfego livre) um veículo não é influenciado pelo líder e pode avançar de forma independente, ou seja: nem todos os veículos necessitam aguardar o veículo à sua frente para avançar.

Essa informação faz com que a ordem de processamento dos veículos em uma pista possa ser especificada através de uma estrutura diferente da fila de veículos: a Fila de Precedência (FP). A fila de precedência é derivada da fila de veículos. Podem existir múltiplas filas de precedência para cada Pista. A construção da Fila de Precedência ocorre da seguinte maneira para cada Pista:

- O primeiro veículo da fila de veículos inicia a primeira Fila de Precedência;
- A partir do segundo da fila de veículos, os veículos subseqüentes serão incluídos:
    a. Na mesma fila de precedência do veículo anterior e atrás deste se o veículo em questão estiver em regime de *following* (Seguindo o Líder) ou *stopping* (Freando);
    b. Iniciará uma nova fila de precedência se estiver em regime de *free driving* (Tráfego livre).

Como exemplo, considere-se a Pista do Objeto de Via mostrado na Figura 3.8 a seguir. A informação dentro de cada veículo é o regime de direção:

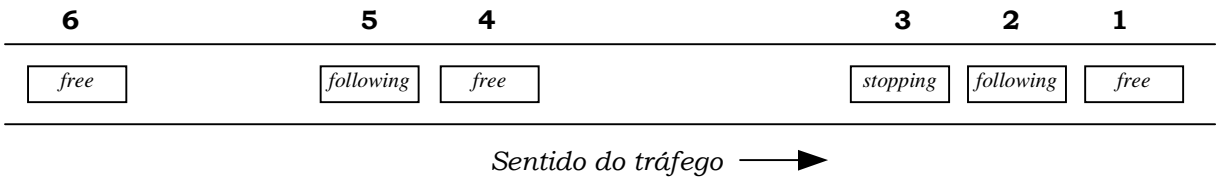


Figura 3.8 – Exemplo de Objeto de Via Pista

Neste exemplo, o veículo **1** avança primeiro. O veículo **2** avança somente após o veículo **1**. O veículo **3** avança somente após o veículo **2**. O veículo **4** avança independentemente dos outros veículos. O veículo **5** avança somente após o veículo **4**. O veículo **6** avança independentemente dos outros veículos. As filas de precedência para esta Pista estão mostradas na Figura 3.9.

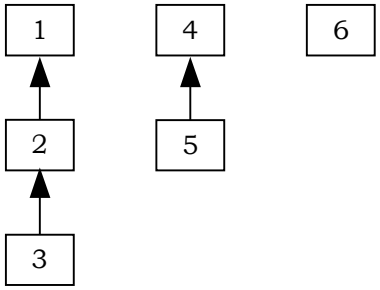


Figura 3.9 – Filas de Precedência para a Pista da Figura 3.8

3.3.3 Passagem dos veículos entre processadores

Quando atinge o final de um Objeto de Via, um veículo é retirado da simulação ou passa para o próximo Objeto de Via na malha viária. Os veículos que podem alcançar o próximo Objeto de Via são os que estão na primeira fila de precedência do Objeto de Via anterior. Os veículos da segunda fila de precedência em diante não têm a possibilidade de alcançar o novo Objeto de Via uma vez que, o primeiro veículo desta fila de precedência estando em *free driving* (Tráfego livre), não é influenciado pelos veículos da sua frente (primeira fila de precedência) e os veículos seguintes são dependentes do primeiro. Assim, não há distinção de precedência a partir da segunda fila de precedência. Pode-se então concatenar as filas de precedência a partir da segunda em diante e o total de filas de precedência necessárias é no máximo dois.

Aplicando-se esse raciocínio às filas de precedência do exemplo anterior, elas tornam-se somente duas, como mostrado na figura a seguir:

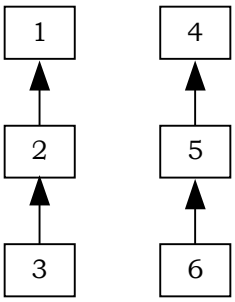


Figura 3.10 – Filas de Precedência para a pista da Figura 3.8

Para entender o processamento do simulador em cada tempo de simulação, a partir das filas de precedência e como ocorre a passagem de veículos entre processadores, considere o exemplo da Figura 3.11 a seguir, composto de uma malha viária dividida em dois Objetos de Via: OV_A e OV_B (que estão em linha reta para simplificação):

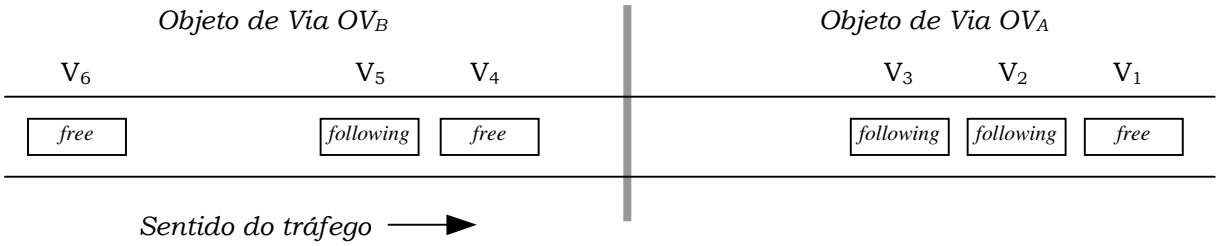


Figura 3.11 – Exemplo de malha viária

Para o Objeto de Via OV_B, as filas de precedência são as mostradas na Figura 3.12:

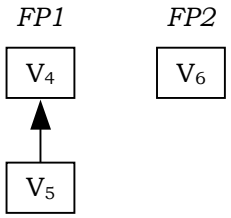


Figura 3.12 – Filas de precedência para o Objeto de OV_B da Figura 3.11

O processamento de OV_B, a cada tempo de simulação será:

1. Processar a primeira fila de precedência, FP1. Se o primeiro veículo (V_4) avança o suficiente para passar para o próximo Objeto de Via (OV_A) é necessário verificar se ele realmente pode avançar o quanto gostaria, dada a posição do último veículo em OV_A (V_3). Duas situações podem ocorrer:
    a. Se o Objeto de Via OV_A está no mesmo computador, V_3 já foi processado e V_4 será processado já sabendo o quanto pode avançar. Após avançar V_4, o processamento segue com o avanço da mesma fila de precedência (FP1) até esgotar os veículos desta. Em seguida, passa para o item 2 a seguir. Esse processamento "local" é o mesmo realizado na versão seqüencial do simulador;
    b. Se o Objeto de Via OV_A está em outro computador, deve ocorrer a passagem de V_4 para este computador. É necessária a comunicação entre os computadores para que V_4 saiba até onde pode avançar. Nesse caso, o computador de OV_B deve enviar uma mensagem ao computador de OV_A para obter essa informação. Enquanto aguarda o retorno da mensagem, o processamento segue para o passo 2 a seguir;
2. Processar a segunda fila de precedência. Como visto anteriormente, os veículos da segunda fila de precedência não atingem o próximo Objeto de Via;
3. Se houve troca de mensagem entre os computadores, aguardar o retorno do outro computador. Pode acontecer que V_4 possa alcançar OV_A ou avançar somente em OV_B ainda. Independentemente de qual foi o retorno, o processamento continua com o término do avanço de V_4 e em seguida do restante da primeira fila de precedência.

Caso V_4 alcance OV_A, o computador de OV_B envia nova mensagem a OV_A informando a passagem do veículo. Essa mensagem deve conter todas as

características de V_B (como velocidade, aceleração, posição e Tipo de Veículo), para que ele possa ser criado em OV_A. O Capítulo 4 mostra detalhes de como foram implementadas as trocas de mensagens entre computadores.

3.4 Sincronização entre os processadores

O sistema físico a ser simulado, neste caso o tráfego, é composto por vários processos físicos que interagem. Cada processo físico pode ser modelado por um processo lógico (*Logical Process – LP*) e as interações entre os processos físicos podem ser modeladas como a troca de mensagens entre os processos lógicos correspondentes [15].

Considere-se um programa de simulação composto de vários processos lógicos trocando mensagens entre si, como na Figura 3.13. Para que a simulação seja efetiva e represente corretamente o sistema físico, todos os eventos dos processos lógicos, assim como as interações entre os processos lógicos, devem ser processados na ordem em que eles ocorrem nos processos físicos. Dados dois eventos E_1 e E_2 com tempos de simulação T_1 e T_2 respectivamente e $T_1 < T_2$, se a execução de E_1 influencia na execução de E_2 então E_1 deve ser processado antes que E_2 sob pena de que, se essa ordem não for mantida, a simulação não refletirá o sistema físico sendo simulado.

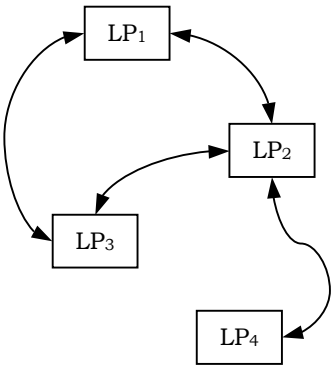


Figura 3.13 – Processos lógicos

A necessidade de ordem no processamento indica que existe uma dependência na execução da simulação. No exemplo anterior, o processamento de E_2 depende do processamento de E_1. Quando a execução do programa de simulação ocorre em um processador seqüencial, essa dependência é facilmente assegurada.

Entretanto, quando a simulação é distribuída em vários processadores, essa tarefa não é simples.

Como exemplo, considere-se que na Figura 3.13, cada processo lógico tenha um evento a executar: E_1, E_2, E_3 e E_4 com os tempos de simulação T_1, T_2, T_3 e T_4 respectivamente e que $T_1 < T_2 < T_3 < T_4$. Considere-se que cada processo lógico seja executado em um processador diferente e que E_4 dependa de E_3, E_3 dependa de E_2 e E_2 dependa de E_1 (Figura 3.14).

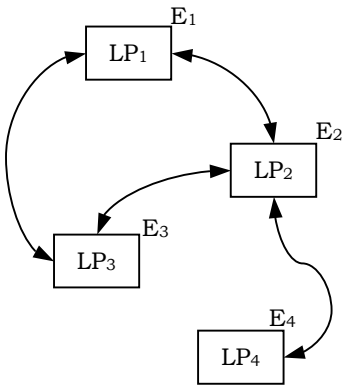


Figura 3.14 – Processos lógicos e eventos

Para que a simulação represente corretamente o sistema físico, a ordem de processamento desses eventos deve ser: primeiro E_1, em seguida E_2, depois E_3 e por último E_4. Entretanto, como os processos lógicos são executados em diferentes processadores, se cada processador simplesmente processar seu evento, não há garantias de que os eventos serão processados na ordem requerida.

Nesta situação, um controle é necessário para que a execução produza exatamente o mesmo resultado da execução seqüencial. É importante entender que esse controle não precisa garantir que os eventos em diferentes processadores sejam processados na ordem do sistema físico, mas deve garantir que o resultado da simulação distribuída seja o mesmo que se a simulação fosse executada de forma seqüencial. Para isso, deve identificar eventos dependentes para processá-los na ordem requerida e eventos que não tenham dependência para tratar como oportunidades de ganho de desempenho.

3.4.1 Sincronismo Mestre-Escravos

Retornando à Figura 3.14, considere-se, por exemplo, que cada processo lógico (LP) esteja simulando uma fração da malha viária (alguns Objetos de Via). Considere-se que em um determinado instante seja necessário exibir, na interface gráfica, a

posição de todos os veículos da simulação. Para que os veículos sejam exibidos corretamente – e a simulação corresponda ao sistema físico – todos devem ter sido processados até um mesmo tempo de simulação.

O Mestre deve saber qual o último tempo de simulação que cada Escravo já processou. Isso é feito com uma mensagem de sincronismo do Mestre aos Escravos em cada tempo de simulação. Para redução do número de mensagens, a própria mensagem de requisição de avanço do tempo de simulação é usada para esse sincronismo. Cada Escravo, ao final do seu processamento do tempo de simulação, responde ao Mestre informando que terminou. Quando o Mestre recebe o retorno de todos Escravos, atualiza a tela, envia nova solicitação de avanço aos Escravos e entra em estado de espera (*wait*) novamente.

3.4.2 Sincronismo Escravo-Escravo

Escravos interagem entre si para passagem de veículos. Depois de receber a requisição de avanço do Mestre, o Escravo inicia seu processamento conforme descrito na Seção 3.2.2 Escravos. Do momento que recebe a requisição do Mestre até terminar o processamento do tempo de simulação, o Escravo pode receber mensagens de outros Escravos para envio de veículos. O Escravo só pode dar por terminado o processamento do tempo de simulação quando responde a todas mensagens. Para isso, cada Escravo deve manter uma lista com a relação de Escravos a ele conectado e que podem lhe enviar veículos.

Entretanto, não é a todo tempo de simulação que haverá veículos a receber. Para evitar que um Escravo espere indefinidamente uma mensagem de passagem de veículo de outro Escravo foi criada a mensagem de "veículo Nulo". Assim, um Escravo conectado a outro Escravo deve, a cada tempo de simulação, enviar uma mensagem de passagem de veículo ou uma mensagem de veículo nulo.

O controle da fila de veículos deve considerar que, uma vez que em cada tempo de simulação todos os veículos devem ser processados uma e somente uma vez, é necessário que a estação Escrava que recebe um veículo não processe-o novamente, uma vez que ele já foi processado no Escravo de origem.

Uma análise simplificada do custo computacional do simulador indica que o tempo despendido na comunicação dos dados é maior que no processamento dos Escravos e Mestre e que a complexidade desses algoritmos é proporcional ao quociente [número de veículos/número de processadores] em cada estação.

3.4.3 Seqüência de avanço

A seqüência de avanço dos Objetos de Via fica assim:

1. No passo 1, a estação Escrava recebe solicitação de Avanço do Mestre. Após receber essa mensagem, o Escravo inicia o avanço dos veículos dos Objetos de Via, começando pelo primeiro veículo da primeira fila de precedência. Se este veículo tem chance de ser transferido a outro Escravo, ele deve saber, a cada avanço, a sua distância ao último veículo do Objeto de Via seguinte.

   Para isso, a solução mais simples é que isso pode ser feito através de uma solicitação do Escravo origem ao Escravo destino da posição do seu último veículo. Entretanto, essa opção novamente pode gerar um número grande de mensagens entre Escravos e diminuir o desempenho da simulação.

   A alternativa implementada é que, uma vez que cada Escravo deve dar um retorno ao Mestre após ter finalizado o processamento em cada tempo de simulação, ele poderia enviar também a posição do último veículo de cada Pista que pode receber veículos. O Mestre inclui essa informação na próxima solicitação de avanço para os Escravos conectados. Essa alternativa não aumenta o número de mensagens trocadas. Apenas um parâmetro a mais é empacotado em uma mensagem já existente.

   Para esta solução, o Mestre deve ter consigo a informação da distância do último veículo de cada Pista de cada Objeto de Via de cada Escravo. O modelo de dados para atender a esse requisito deve seguir o descrito na Figura 3.13:

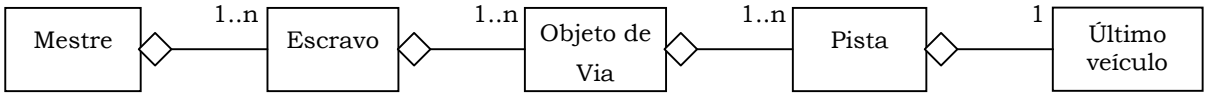


Figura 3.13 – Modelo de dados para controle da posição do último veículo

2. O segundo passo tem dois possíveis caminhos a seguir. Como cada Escravo já inicia seu método de Avanço conhecendo até onde seu primeiro veículo pode avançar, então:

   a. Se o processamento do veículo indicar que ele avançará até a distância recebida já se sabe que ele será aceito no Escravo destino, então a fila de precedência torna-se a própria ordem dos veículos na Pista e não é necessário aguardar o retorno do Escravo destino;

b. Se o veículo deveria avançar além dessa distância, então ainda não é sabido, antecipadamente, se isso será possível. Neste caso, o veículo será enviado ao Escravo destino e deve-se aguardar o retorno da posição final do veículo. Enquanto aguarda, o Escravo origem deve continuar processando os outros veículos. O próximo veículo a ser processado é o primeiro veículo da segunda fila de precedência, ou seja, o próximo veículo que não depende do que foi enviado.

   Após o último veículo da segunda fila de precedência ser avançado, pode-se voltar o processamento para o segundo veículo da primeira fila (após retorno do Escravo destino do primeiro veículo).

   Para atender a essa lógica de processamento, os Objetos de Via deverão ter dois métodos de adição de veículo (*AddVehicle*), todos assíncronos:

   i. Um método sem retorno, para os casos em que já se sabe que o veículo avançará até onde precisa. Esse método simplesmente adiciona o veículo na Pista destino na posição física solicitada pelo Escravo origem. O Escravo que chama esse método não precisa aguardar retorno e pode continuar o processamento dos veículos da mesma Fila de Precedência;

   ii. Um método com retorno da posição até onde o veículo avançou. Esse método deve ser executado após todo o processamento de todos os veículos da Pista e retornar um número inteiro com a informação da posição de onde o veículo foi incluído. O veículo seguinte da fila de precedência do veículo enviado só pode ser processado após o retorno desse método. Assim, o Escravo que chama esse método deve passar o processamento para a próxima fila de precedência do Objeto de Via enquanto aguarda o retorno do Escravo destino.

   Nos dois casos o Escravo destino não deve avançar o veículo recebido neste tempo de simulação. Caso isso ocorra, este veículo estará sendo processado duas vezes para o mesmo tempo de simulação.

3. Após fazer o avanço dos veículos e processar todos os pedidos de adição de veículos (ou veículos nulos) o Escravo dá retorno ao Mestre passando a informação da posição dos seus últimos veículos de cada Objeto de Via. Para isso, cada Escravo deve ter um vetor de confirmação para cada Escravo a ele

conectado e só pode dar o "terminado" ao Mestre após receber o retorno de todos os Escravos a ele conectado, seja por solicitação de adição de veículo (método *AddVehicle*) ou mesmo veículo nulo;

4. Retornar ao passo 1.

Os diagramas de estados dos Escravos e do Mestre estão na Figura 3.14.

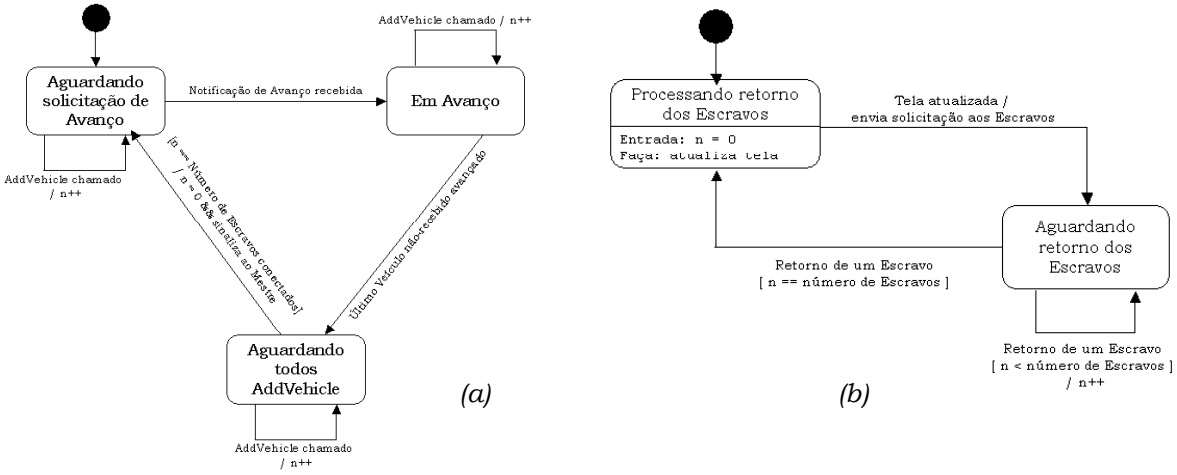


Figura 3.14 - Diagramas de estados dos Escravos (a) e do Mestre (b)

3.5 Detecção de *deadlocks*

Considere-se a situação em que um Escravo A deve enviar um veículo ao Escravo B que já aguarda um retorno do Escravo A sobre o envio de um veículo. Situações como esta provocam *deadlock*. Para atender a todas as possibilidades de *deadlock* no simulador, três situações são consideradas:

1. Em um mesmo Objeto de via;
2. Entre Objetos de Via de um mesmo computador;
3. Entre Objetos de Via de computadores diferentes.

Esta seção mostra como a situação de *deadlock* é tratada no simulador de tráfego.

3.5.1 *Deadlock* em um mesmo Objeto de Via

Dentro de um Objeto de Via os veículos são processados do primeiro ao último no sentido do tráfego. Não há situação de *deadlock*, uma vez que a dependência de

processamento de qualquer veículo, quando existir, é em relação ao veículo da frente, que já deve ter sido processado.

3.5.2 *Deadlock* entre objetos de via de mesmo computador

Em um mesmo processador, para fins de identificação de *deadlock*, pode-se considerar que a simulação funciona com na versão seqüencial do simulador e não há risco de *deadlock* entre Objetos de Via. Isto por que a especificação da malha viária, como discutido no Capítulo 2, deve ser feita de tal forma que os últimos Objetos de Via no sentido do tráfego sejam os primeiros a serem processados. Mantendo-se essa condição, não há risco de *deadlock*.

3.5.3 *Deadlock* entre objetos de via de computadores diferentes

Antes de endereçar o *deadlock* entre objetos de via de processadores diferentes é importante lembrar que não deve ser definida uma configuração viária de circuito fechado, como na figura abaixo:

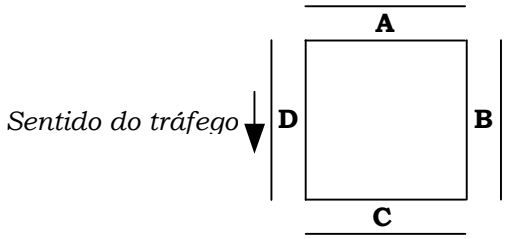


Figura 3.15 – Malha viária de circuito fechado

Isso se deve ao fato da necessidade de haver pelo menos uma entrada para os veículos na malha viária. Para que não haja o acúmulo de veículos, deve haver também pelo menos uma saída para os veículos. Como descrito no Capítulo 2, os veículos são definidos pelas Populações e entram na simulação quando o tempo de simulação atinge seus tempos de entrada.

A situação que pode gerar *deadlock* é a que um processador envia veículos para outro processador e há retorno do veículo para o primeiro processador. Como exemplo, seja a Figura 3.16:

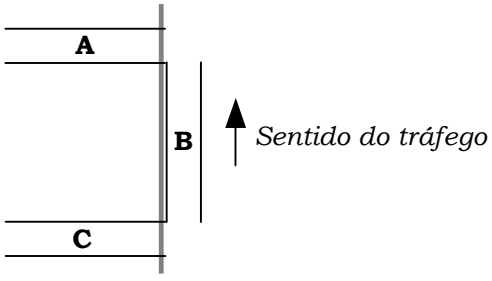


Figura 3.16 – Malha viária com retorno de veículo

Os Ramos A e C estão em um Escravo (P1), enquanto o Ramo B está em outro (P2). O *deadlock* poderá acontecer caso o Ramo C envie um Veículo para o Ramo B usando o método `AddVehicle` com retorno e o Ramo B envie um Veículo para o Ramo A também usando o método `AddVehicle` com retorno. O processador P1 irá aguardar o retorno de P2 que já estaria aguardando pelo retorno do P1.

A construção de um algoritmo de detecção de *deadlocks* no analisador léxico do arquivo de definição da malha viária é uma solução possível e de média complexidade. A solução adotada neste simulador usa a definição da malha viária, que deve ser realizada de tal forma que não permita essa situação de *deadlock*.

3.6 Resumo

Esta seção mostra um resumo das propostas de solução para as situações apresentadas neste capítulo necessárias à distribuição do simulador de tráfego.

Problema	Solução	Referência (seção ou subseção)
Meio para comunicação	Plataforma de hardware Distribuída, computadores conectados em uma LAN (*Local Area Network*)	3.1.1
	Modelo de programação paralela de Passagem de Mensagem	3.1.2
Coordenação das atividades	Arquitetura de software Mestre-Escravo	3.2
Decomposição do algoritmo de simulação	Decomposição de Domínio ou Espacial	3.3
Divisão da Malha Viária	Realizada pelo usuário na definição do arquivo de parâmetros	3.3.1
Paralelismo de atividades	Filas de Precedência	3.3.2
Passagem de veículos entre processadores	Escravos recebem do Mestre, junto com a solicitação de avanço, as últimas posições dos veículos das Pistas conectadas. Se o avanço do veículo é só até a posição informada, o Escravo origem envia uma mensagem assíncrona de passagem de veículo para o Escravo destino, não espera por retorno e continua o processamento. Se o avanço do veículo ultrapassa a posição informada, o Escravo origem envia uma mensagem assíncrona de passagem de veículo para o Escravo destino e continua o processamento na próxima Fila de Precedência. Terminado o processamento, espera pelo retorno do Escravo para saber até onde seu veículo pode avançar, avança-o e continua o processamento da primeira fila de Precedência	3.3.3

Problema	Solução	Referência (seção ou subseção)
Sincronismo Mestre-Escravos	Mestre envia mensagem de sincronismo a cada tempo de simulação. Mensagem é a mesma que a solicitação de avanço. Mestre entra em espera (*wait*) e retorna quando todos Escravos enviarem uma mensagem com fim de processamento. Junto com a mensagem de fim de processamento, deve ir a posição dos últimos veículos de cada Pista que pode receber veículo de outro Escravo. Veículos recebidos não devem ser processados no Escravo destino.	3.4.1 e 3.4.3
Sincronismo Escravos-Escravos	Escravos devem enviar, a cada tempo de simulação, mensagens a todos Escravos que podem ser destino de seus veículos. Se não há veículo a enviar, uma mensagem de veículo nulo deve ser enviada. Cada Escravo deve manter uma lista com a relação de Escravos origem a ele conectado. Escravos devem receber mensagens de todos Escravos a ele conectado para encerrar o processamento do tempo de simulação.	3.4.2 e 3.4.3
Detecção de deadlocks	Definição da divisão da malha viária pelo usuário	3.5.3

Tabela 3.2 – Resumo das soluções para distribuição do simulador

O próximo capítulo mostra alguns detalhes da implementação em Java e RMI e os resultados obtidos na execução de dois casos de simulação.

Capítulo 4

Implementação e resultados

O Capítulo 2 deste trabalho mostrou os modelos normalmente usados na simulação microscópica de tráfego e uma alternativa de solução para o desenvolvimento do simulador seqüencial. O Capítulo 3 mostrou como pode-se, a partir do simulador seqüencial, distribuir o processamento entre vários processadores. Este Capítulo 4 faz um levantamento das atuais opções de comunicação entre processadores que podem ser usadas para a distribuição da simulação, descreve particularidades da implementação devido à opção escolhida (Chamada de Método Remoto, ou *Remote Method Invocation* – RMI) e por fim, mostra os resultados obtidos na execução de dois casos de simulação.

4.1 Pré-requisitos para distribuição do simulador

Antes de tudo é preciso garantir que o algoritmo distribuído para o simulador de tráfego atinge o objetivo da simulação, ou seja, que ele fornece exatamente o mesmo resultado que sua versão seqüencial. Essa garantia deve ocorrer em duas visões: na funcional, garantindo-se que o modelo distribuído proposto, assim como ocorrido com o seqüencial, atende aos requisitos do sistema físico (o tráfego); e na técnica, atestando-se que a execução em múltiplos processadores seja finita e produza resultado.

Do ponto de vista funcional, o modelo distribuído deve:

- Manter a ordem de precedência do avanço dos veículos, ou seja, se o avanço de um veículo depende do avanço do veículo líder, essa ordem – primeiro o líder e depois o seguidor – deve ser mantida no processamento da fila de veículos;
- Em cada tempo de simulação, avançar no tempo todos os veículos uma e somente uma vez.

Essas duas questões foram tratadas com a implementação da Fila de Precedência descrita na Seção 3.3.2 (Paralelismo de atividades) e com a sincronização entre mestre-escravos e escravos-escravos com envio de mensagens descrita na Seção 3.4 (Sincronização entre os processadores) respectivamente.

Do ponto de vista técnico, para ser finito, o algoritmo deve tratar as situações de *deadlock*. Essa questão é tratada com as considerações feitas sobre a definição da malha viária realizada pelo usuário de forma que não se crie circuitos fechados (*loops*) conforme descrito na Seção 3.5.3 *Deadlock* entre objetos de via de computadores diferentes. O "valor" produzido é a animação gráfica da simulação, além do arquivo de estatística.

Uma vez certificada a possibilidade da distribuição do algoritmo de simulação, o passo seguinte é definir um meio pelo qual as partes se comuniquem. Como descrito no Capítulo 3, a plataforma escolhida foi a Distribuída com modelo de programação de Passagem de Mensagem. Na próxima seção são mostradas as alternativas tecnológicas disponíveis para essa escolha.

4.2 Alternativas para Sistemas Distribuídos

Várias definições podem ser encontradas na literatura sobre o que é um sistema distribuído. De acordo com Bapat [20 apud 21] um sistema distribuído é "um sistema de processamento de informação que contém um número de computadores independentes que cooperam entre si através de uma rede de comunicação a fim de atingir um objetivo específico". Kshemkalyani e outros [22] citam as seguintes características para os sistemas distribuídos:

- Ausência de um relógio físico comum entre os processadores, o que introduz o elemento "distribuição" e causa o assincronismo inerente dos processadores;

- Ausência de memória compartilhada. Essa é uma característica chave que induz a comunicação por passagem de mensagem. É importante observar que um sistema distribuído ainda pode fornecer a abstração de um espaço de endereçamento comum através da abstração de memória compartilhada;
- Autonomia, heterogeneidade e separação geográfica dos processadores;

Em um sistema distribuído, cada computador tem unidade de processamento e memória e são conectados por uma rede de comunicação. Um sistema distribuído típico está mostrado na Figura 4.1.

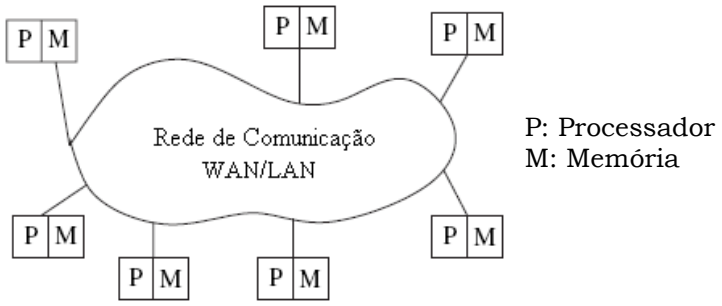


Figura 4.1 – Sistema distribuído típico. Extraída de [22]

A divisão das responsabilidades entre os componentes do sistema e a disposição desses componentes nos computadores da rede determina a arquitetura do sistema e tem grande implicação no desempenho, confiabilidade e segurança do sistema. O modelo cliente-servidor introduz dois papéis que podem ser assumidos pelos processos: o papel de usuário de um serviço (cliente) e o papel de fornecedor do serviço (servidor). A distribuição de papéis implica em uma assimetria na execução distribuída de uma aplicação. O servidor oferece um serviço no qual um ou mais clientes têm acesso. A arquitetura cliente-servidor é a mais citada quando se discute sistemas distribuídos e é historicamente a mais utilizada. Por outro lado, na arquitetura ponto a ponto (*peer-to-peer*) os processos desempenham papéis similares, interagindo cooperativamente sem distinção entre clientes e servidores. A Figura 4.2 apresenta esquematicamente os modelos cliente-servidor e ponto a ponto.

A rede de comunicação, entretanto, fornece meramente um mecanismo de transporte. O acesso a ela depende de fatores tecnológicos e varia entre as várias plataformas físicas. Por isso, é necessária uma infra-estrutura que suporte o desenvolvimento e execução de aplicações distribuídas [21]. Essa infra-estrutura deve ser a interface entre as aplicações distribuídas e o acesso à rede para

comunicação com os outros computadores. Ela deve homogeneizar o acesso à rede e oferecer serviços genéricos às aplicações, além de encapsular as diferenças entre os variados sistemas.

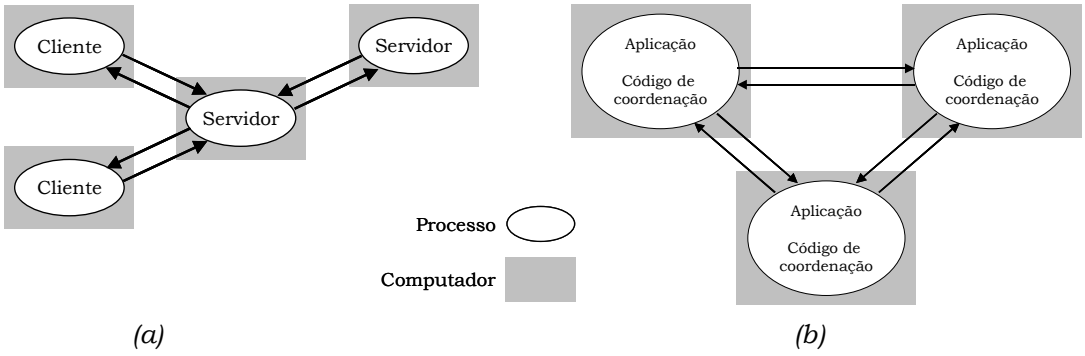


Figura 4.2 – (a) Arquitetura Cliente-servidor e (b) Ponto-a-ponto. Extraída de [23]

A Figura 4.3 mostra a relação entre os componentes de software que são executados em cada computador. A infra-estrutura com a qual uma aplicação pode dispor para acesso à rede de comunicação é provida: pelo sistema operacional, pelas facilidades de software distribuído, ou *middleware* e pelas camadas de aplicação e de transporte do protocolo de rede que seguem o modelo OSI.

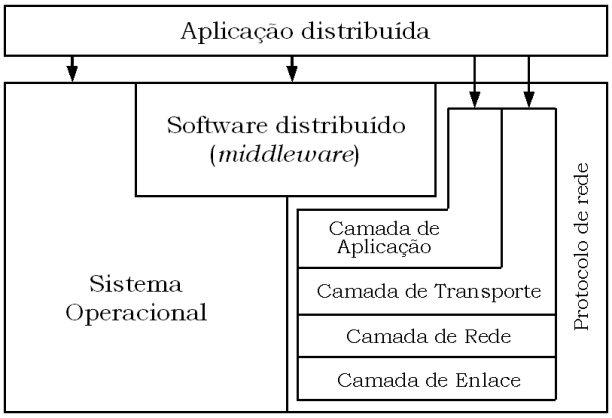


Figura 4.3 – Interação dos componentes em cada computador. Extraído de [22]

De uma forma geral, todas as soluções descritas acima, em maior ou menor grau de complexidade de desenvolvimento, atendem ao processo de construção de aplicações distribuídas, entretanto busca-se combinar o desenvolvimento de tais aplicações com o nível de abstração provido por cada solução. Soluções de comunicação fornecidas diretamente pelos Sistemas Operacionais – o *socket* é o exemplo mais conhecido – oferecem baixíssimo nível de abstração e requerem um alto grau de controle. São usadas geralmente para prover uma abstração para

níveis superiores das camadas de comunicação e não diretamente como soluções em aplicações distribuídas complexas como o simulador de tráfego em questão.

Os protocolos da camada de transporte trabalham em um nível de abstração acima do *socket*. Na verdade, a maioria dos protocolos de transporte usa a abstração do *socket*. Os dois protocolos de transporte mais utilizados e que têm suporte da maioria das linguagens de programação são o TCP (*Transport Control Protocol*) e UDP (*Universal Datagram Protocol*).

O UDP é um protocolo de transporte que não é orientado a conexão, ou seja, sem confirmação da chegada da mensagem no destino (*acknowledgement*). Se uma falha ocorre, a mensagem pode não chegar ao destino. Esse tipo de solução não atende ao simulador de tráfego. Uma mensagem perdida pode significar perda de sincronismo entre o mestre e os escravos, um veículo não avançado em um determinado tempo de simulação e ainda uma situação de *deadlock* pois os escravos sempre aguardam uma mensagem de outro escravo a ele conectado (Subseção 3.4.2 Sincronismo Escravo-Escravo).

O protocolo TCP é orientado a conexão e garante a chegada da mensagem no destino. O TCP realiza transporte de cadeias de dados (*streams*) que podem ser organizadas para formar estruturas de dados. Do ponto de vista do simulador, entretanto, é interessante usar uma camada de abstração que se possa trabalhar com estruturas mais complexas e até chegar a Objetos, com estado e comportamento.

Na divisão em camadas do protocolo de rede, a camada de aplicação fornece soluções já prontas, como *http*, *mail*, *ftp* e *telnet* (que normalmente rodam sobre conexões TCP), entre outras, que não são flexíveis o suficiente para serem adaptadas e utilizadas pelo simulador distribuído.

A solução que mais se encaixa na necessidade de comunicação do simulador é a provida pela camada conhecida como *middleware*. O *middleware* é uma camada de software cujo propósito é mascarar a heterogeneidade e fornecer aos programadores de aplicações distribuídas um modelo de programação conveniente [23]. Várias primitivas e chamadas de função definidas em várias bibliotecas da camada *middleware* são embutidas no código dos sistemas distribuídos.

Atualmente, há uma grande variedade de soluções de *middleware* para distribuição de aplicações. As bibliotecas MPI (*Messagem Passing Interface*) e PVM (Parallel Virtual Machine) fornecem funções de envio e recepção de dados. Ambas

definem uma interface para diferentes linguagens de programação como C, C++ ou Fortran. Algumas implementações mais recentes incluem suporte a Java.

Outra solução é o mecanismo de chamada de procedimento remoto, ou RPC (*Remote Procedure Call*). Conceitualmente o RPC trabalha como uma chamada de procedimento local, com a diferença que o código do procedimento pode estar em uma máquina remota. Neste caso, o RPC envia uma mensagem para a rede de comunicação para invocar o procedimento remoto e aguarda o retorno.

Da mesma forma que a diversidade de protocolos de comunicação levou à criação do modelo OSI, a necessidade de um modelo normalizado para aplicações distribuídas fez com que a ISO (*International Standards Or*ganization) definisse um modelo de referência para processamento distribuído aberto – RM-ODP (*Reference Model for Open Distributed Processing*). Como o modelo OSI, o RM-ODP é simplesmente um modelo de referência. Entretanto, sua especificação estende os conceitos do RPC. Desde a criação da especificação surgiram inúmeras implementações do modelo RM-ODP. Os mais conhecidos são: RMI (*Remote Method Invocation* da SUN), DCOM (*Distributed Component Object Model* da Microsoft) e CORBA (*Common Object Request Broker Architecture* do OMG – *Object Management Group*) [24].

Para este trabalho, os pré-requisitos que a camada de *middleware* deve ter são:

- Suporte a modelo de objeto: o *middleware* deve fornecer mecanismos para suportar os conceitos do paradigma de programação orientado a objeto;
- Interação operacional: o *middleware* deve permitir a interação entre dois objetos usando a invocação de métodos da linguagem de programação;
- Interação remota: o *middleware* deve permitir a interação entre dois objetos localizados em diferentes computadores;
- Transparência de distribuição: do ponto de vista do programa, a interação entre objetos deve ser idêntica para objetos locais ou remotos;

As três soluções de RM-ODP citadas (DCOM, CORBA e RMI) atendem aos pré-requisitos levantados. Outros requisitos que podem ser considerados são a facilidade de implementação, a facilidade de integração com o simulador desenvolvido e a disponibilidade e facilidade de suporte no mercado. Em desfavor ao uso de DCOM pesa o fato da dependência do sistema operacional e do

fabricante, além de ele ser reconhecidamente menos robusto e estável que CORBA e RMI.

Considerando-se que simulador é desenvolvido em Java, RMI torna-se uma solução natural. Soma-se a isso o fato de que a tecnologia Java é mais difundida que CORBA e há uma maior facilidade de implementação e suporte a problemas. A escolha de RMI, entretanto, é somente um meio para demonstrar que a alternativa de solução proposta para distribuição do simulador é válida. Qualquer outra solução de *middleware*, ou mesmo de outra camada de comunicação (Figura 4.3) pode ser usada para essa demonstração.

Nas próximas três seções são listadas questões de implementação que surgem a partir da escolha do RMI e as soluções adotadas para cada situação.

4.3 Criação dos objetos: local ou remoto

Uma das características de RMI é a sua capacidade de receber e carregar o código de um objeto de uma classe mesmo que a classe não esteja definida na máquina virtual que executa o código. O estado e o comportamento de um objeto podem ser transmitidos para outra, possivelmente remota, máquina virtual. O RMI passa objetos de modo que o comportamento desses objetos não é alterado quando são enviados para outra máquina virtual. Isto permite que novos tipos sejam introduzidos em uma máquina virtual remota, ou que objetos por completo (estado e comportamento) sejam enviados de uma máquina virtual a outra, mesmo em um outro processador.

Essa é uma grande virtude do RMI e poderia se pensar em usá-la para criar todos os objetos do simulador (objetos de vias, veículos, semáforos e populações) no Mestre e à medida que eles são necessários à simulação enviá-los aos Escravos em outros processadores. Inicialmente essa foi a implementação escolhida. Entretanto, um fato percebido durante as execuções do simulador alterou essa definição: criar todos objetos remotos torna mais lento o processamento. A explicação a esse fato está na forma com que os parâmetros são passados em RMI:

- Se o parâmetro é um objeto local, ou não remoto, ele é passado como cópia usando serialização de objetos. Para isso ele deve simplesmente implementar a interface `java.io.Serializable`, que é natural a boa parte dos objetos Java;

- Se o parâmetro é um objeto remoto ele será passado por referência.

Nessa última situação (objetos remotos) ocorre a perda de desempenho, uma vez que toda e qualquer referência deveria ser feita ao Mestre, onde os objetos foram criados.

Para contornar essa situação, a solução encontrada foi criar cada objeto no próprio Escravo onde ele deve ser referenciado. No início da simulação, quando a malha viária é criada, o Mestre envia aos Escravos as solicitações de criação de cada objeto de malha e dos semáforos, passando as características de cada um por parâmetro e cada Escravo cria os objetos em sua própria máquina virtual Java.

O mesmo ocorre na passagem de um veículo de um Escravo a outro. O objeto veículo é destruído no Escravo origem e criado com as mesmas características no Escravo destino. Uma vez que o veículo pode transitar por objetos de via em qualquer processador e, conseqüentemente ser criado também em qualquer processador, todas as Populações são criadas em todos os Escravos.

4.4 Chamada de retorno (*callback*)

Depois de enviar a solicitação de avanço aos Escravos, o Mestre só deve atualizar a interface gráfica quando todos os Escravos encerrarem o Avanço solicitado. Se isso não acontecer, a interface gráfica pode exibir resultados não condizentes com o sistema físico (Capítulo 3). Há, basicamente, duas formas de se testar que todos os Escravos terminaram a execução do método Avanço:

1. O Mestre verifica continuamente o estado dos Escravos, enviando-lhes uma mensagem e aguardando retorno afirmativo. O processo se estende até que o último Escravo confirme o encerramento de seu método Avanço. Esse método também é conhecido como *polling*;
2. Após enviar a solicitação de avanço aos Escravos, o Mestre entra em estado de espera (*wait*) e é avisado pelos Escravos à medida que estes terminam o Avanço. Quando o Mestre recebe a confirmação de todos os Escravos ele desperta e pode atualizar a interface gráfica e dar prosseguimento à simulação.

É fácil perceber que a opção 1 requer um número muito maior de mensagens do que a opção 2. A opção 2, entretanto, requer que uma questão seja resolvida: a implementação de uma chamada de retorno, ou *callback*, dos Escravos ao Mestre. É

importante lembrar que Mestres e Escravos estão sendo executados em máquinas virtuais Java diferentes e em computadores diferentes.

Para implementar um método de *callback* de um objeto em uma outra máquina virtual usando RMI, os passos tomados foram os seguintes:

1. Tornar o Mestre disponível para receber mensagens. Isso foi feito através do método `exportObject()` da classe `UnicastRemoteObject`. Essa classe fornece suporte para referências ponto a ponto para objetos ativos. A interface `Notifiable` foi criada para encapsular a solução acima. O Mestre implementa essa interface;
2. Disponibilizar um método para que o Mestre possa se registrar no Escravo, que nada mais é que o Escravo tenha a referência para o Mestre. Esse método deve ser especificado na própria interface de definição dos métodos dos Escravos. O método criado foi o `setNotifiable(Notifiable)` que tem como parâmetro um objeto que implementa a interface `Notifiable`, que é como a referência do Mestre é passada para os Escravos;
3. Na própria interface `Notifiable` está definido o método `notifyMaster()` que é chamado pelos Escravos para informar que o processamento de um tempo de simulação terminou.

O diagrama de seqüência a seguir (Figura 4.4) mostra a interação entre mestre e escravos com relação à chamada de retorno para um tempo de simulação. A chamada de retorno está representada na figura pelo método "Término de processamento".

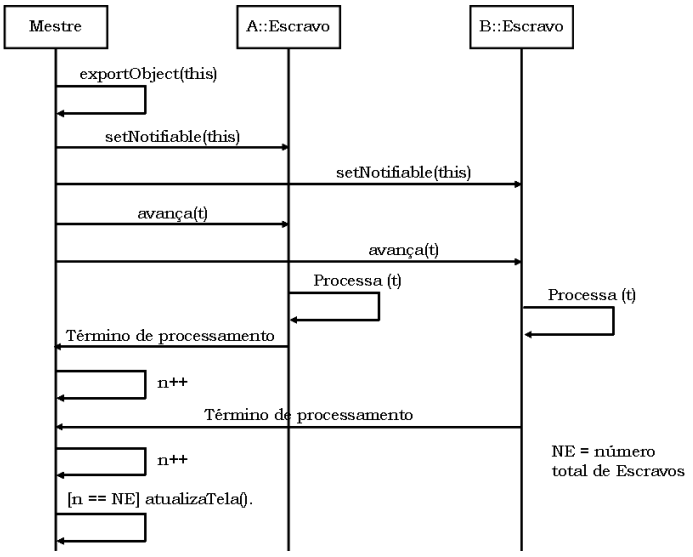


Figura 4.4 – Diagrama de seqüência da chamada de retorno

4.5 Comunicação assíncrona usando RMI

Um dos grandes desafios de se trabalhar com a distribuição do simulador de tráfego é que ele necessita que a comunicação entre Mestre-Escravos e Escravos-Escravos ocorra ora de forma síncrona, ora de forma assíncrona (Capítulo 3). As principais mensagens trocadas no simulador e o tipo de cada mensagem estão mostradas no quadro resumido da tabela 4.1.

Mensagem		Mestre	Escravo	Escravo
1	Criação dos objetos nos Escravos			
2	Solicitação de Avanço			
3	Passagem de Veículo assíncrona			

 Síncrona

 Assíncrona

Tabela 4.1 – Principais mensagens trocadas entre Mestre-Escravos e Escravos-Escravos

Assim como seu antecessor RPC, RMI não é originalmente desenhado para comunicação assíncrona (Figura 4.4), por isso um mecanismo foi criado para habilitar os Escravos a receberem chamadas assíncronas: foi criada a classe `FuncDetail` que representa uma função a ser executada. `FuncDetail` contém como atributos:

- `private int func`, que é um número que identifica a função a ser executada (Avanço dos Objetos de Via ou AddVehicle);
- `private Object input`, que é o parâmetro de entrada para a função;

Duas classes contendo filas de `FuncDetail` foram criadas: `FuncHeaderSync` e `FuncHeaderAssync`, responsáveis respectivamente pelas chamadas síncronas e assíncronas. Ambas as classes contêm o método `AddFunc(int func, Object param)`. Outras duas classes foram criadas para executar as funções solicitadas: `RemoteEngineAssync` e `RemoteEngineSync`. Elas estendem `Thread` e seus objetos são executados em threads diferentes da *thread* principal dos Escravos.

Quando é solicitada a execução de um método assíncrono, o método `AddFunc()` inclui um objeto `FuncDetail` em `FuncHeaderAssync` e "desperta" (*notify*) a *thread* `RemoteEngineAssync` que estava em modo de espera (*wait*). O

método assíncrono termina e retorna ao solicitante. A *thread* despertada obtém as informações da função solicitada e encaminha sua execução. Terminada a execução, a *thread* checa se há outras solicitações na fila e atende caso necessário ou entra em estado de espera (*wait*) novamente.

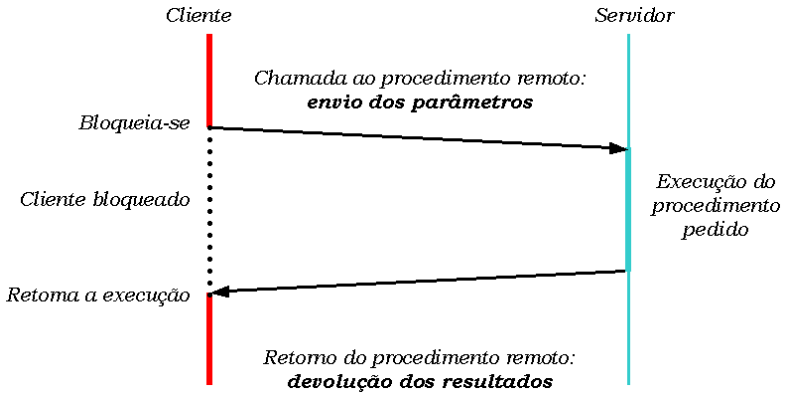


Figura 4.4 – Fluxo de execução de chamada no RPC

Para o método síncrono, o mesmo processo foi utilizado, com exceção de que, ao invés de retornar imediatamente após ter colocado a solicitação na fila, o método originário da solicitação entra em estado de espera (*wait*) e é despertado pela *thread* assim que a solicitação é atendida.

4.6 Resultado de estudo de caso

Esta seção apresenta o resultado da execução de dois casos de simulação nos simuladores. O caso de simulação 1 apresenta uma malha viária com vários objetos de via conectados de forma que, quando distribuída, a malha viária exige comunicação entre os processadores. Este caso de simulação permite analisar o ganho natural de desempenho com a distribuição da simulação.

O caso de simulação 2 apresenta grupos de objetos de via não conectados. Pode-se entender essa configuração como a simulação de localidades sem conexões entre si ou em que não há interesse de simular a conexão, mas somente os objetos de via das localidades separadamente. Neste caso de simulação pode-se verificar de o desempenho do simulador quando não há necessidade de comunicação entre os processadores. Espera-se para esse caso um maior ganho de desempenho do simulador superior ao ganho no caso de simulação 1.

Para os dois casos de simulação foram definidos uma malha viária, tipos de veículos e população conforme descritos a seguir.

4.6.1 Caso de Simulação 1

A malha viária do caso de simulação 1 está representada na Figura 4.5. Ela é composta por 20 objetos de via, sendo 3 Interseções (I) e 17 Ramos (R). Os objetos de via podem ser organizados em 4 rotas. O início de rota está indicado na figura pelo círculo e o final de rota pela seta. A Rota terminada pelo Objeto de via 20R possui 2 pistas, as outras possuem 3 pistas cada.

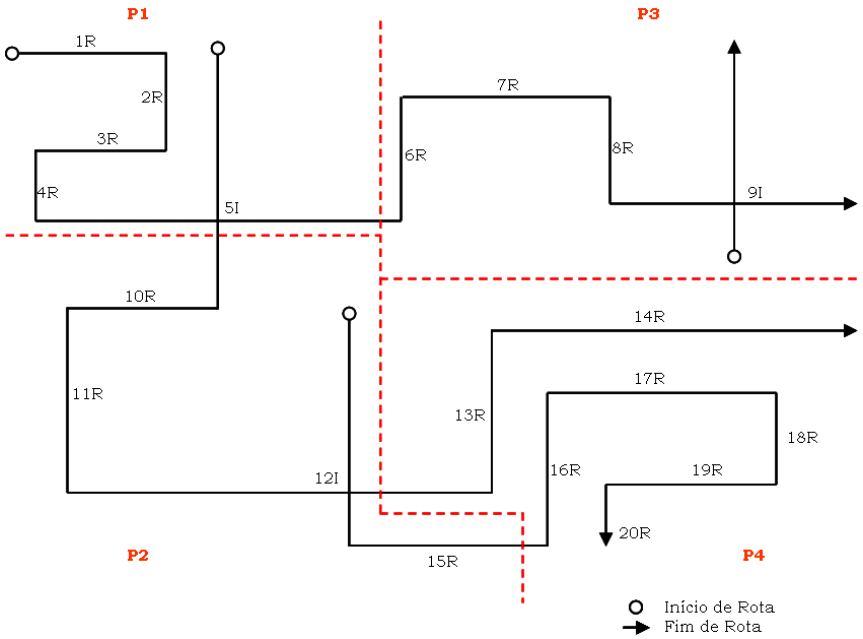


Figura 4.5 – Malha viária do Caso de Simulação 1

Foram definidos cinco tipos de veículos, conforme a tabela a seguir:

Nome	Comprimento (m)	Largura (m)	Velocidade Máxima (Km/h)	Aceleração Máxima (Km/h/s)	Cor
Onibus	12	2	60	6	Verde
Caminhao	14	3	40	4	Preta
Ka	4	2	90	10	Magenta
Passeio	6	2	120	11	Azul
Van	7	2	100	8	Vermelha

Tabela 4.2 – Tipos de Veículos do Caso de Simulação

O tempo de atualização da tela usado foi de 400 ms. Na execução do simulador seqüencial foi utilizado um computador do tipo compatível com IBM, com

processador Intel® Pentium 4, com velocidade de processamento de 3 GHz e 1 GBytes de memória RAM. Para essa configuração o simulador seqüencial atingiu a velocidade de 515 tempos de simulação para 1 tempo de parede, ou seja, em 400 ms (tempo de atualização da tela) foram processados 515 avanços dos veículos.

Na execução do simulador distribuído, a decomposição da malha viária seguiu a divisão indicada na Figura 4.5. Foram usados 4 computadores com a mesma configuração do computador usado na execução do simulador seqüencial e a cada um deles foi atribuído uma divisão da malha viária (P1, P2, P3 e P4 na Figura 4.5). Os quatro computadores foram usados como Escravos e um deles (P2) foi usado também como Mestre. Para essa configuração o simulador distribuído atingiu a velocidade de 1242 tempos de simulação para 1 tempo de parede.

4.6.2 Caso de Simulação 2

Para o caso de simulação 2 os mesmos Objetos de Via foram utilizados, com exceção de três conexões: a Interseção 5I não está conectada ao Ramo 6R nem ao Ramo 10R e a Interseção 12I não está conectada ao Ramo 15R. A malha viária do Caso de Simulação 2 está mostrada na Figura 4.6.

Nessa configuração, a malha viária foi decomposta em quatro regiões. No simulador distribuído, cada uma dessas regiões foi entregue a um processador (P1, P2, P3 e P4).

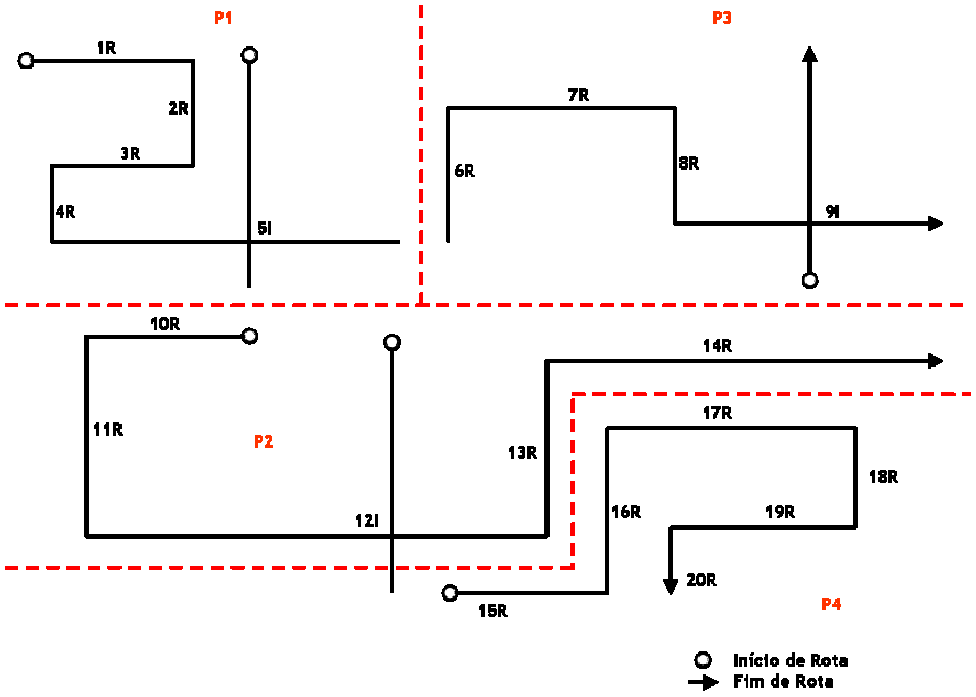


Figura 4.6 – Malha viária do Caso de Simulação 2

Nesta configuração de malha viária a velocidade de processamento do simulador seqüencial aumentou atingiu 545 tempos de simulação para 1 tempo de parede enquanto a velocidade do simulador distribuído atingiu 1881 tempos de simulação para 1 tempo de parede.

O resumo dos valores obtidos nos dois casos de simulação está na tabela 4.3 a seguir:

Caso de Simulação	Simulador Seqüencial		Simulador Distribuído	
	Computador	Velocidade	Computadores	Velocidade
1. Malha viária da Figura 4.5, 400 ms para tempo de interação de veículos e 1000 ms para atualização da tela	1 Processador Pentium® 4, 3 GHz e 1 GBytes de RAM	515	4 Processadores Pentium® 4, 3 GHz e 1 GBytes de RAM	1242
2. Malha viária da Figura 4.7, 400 ms para tempo de interação de veículos e 1000 ms para atualização da tela	1 Processador Pentium® 4, 3 GHz e 1 GBytes de RAM	545	4 Processadores Pentium® 4, 3 GHz e 1 GBytes de RAM	1881

Tabela 4.3 – Resumo do resultado dos Casos de Simulação

É importante entender a unidade usada para medir o desempenho dos simuladores. A Velocidade foi medida em número de tempos de simulação processados em 1 tempo de parede. Quanto maior esse número, melhor o desempenho do simulador. Por exemplo, se uma simulação ocorre com a velocidade de 60 tempos de simulação em 1 tempo de parede e cada intervalo corresponde a 1 segundo, o simulador processou então 1 minuto de tempo de simulação (60 segundos) em um segundo de tempo de parede. Essa unidade de medida pode sofrer influência de fatores externos, principalmente quando usada para medir o desempenho do simulador distribuído, como o envio e a chegada de outras mensagens de rede que não as do simulador.

Os resultados mostram um ganho do simulador distribuído em relação ao seqüencial e que o esforço necessário à comunicação não inviabiliza o ganho de desempenho total do simulador.

Capítulo 5

Conclusões e trabalhos futuros

Os problemas encontrados hoje na maioria das grandes cidades motivam o estudo do tráfego e a simulação tem se mostrado uma ferramenta eficiente na busca de soluções para esses problemas. A modelagem microscópica é hoje o estado da arte dos simuladores de tráfego, porém requer um grande poder de processamento computacional. Uma alternativa mais barata, quando comparada ao uso de supercomputadores, é distribuir a simulação para que ela seja executada em vários computadores paralelamente. A natureza do tráfego, entretanto, requer um controle profundo do sincronismo das partes distribuídas.

Esse trabalho apresentou uma proposta de simulador de tráfego microscópico distribuído. O desenvolvimento do simulador seguiu a metodologia apresentada no Capítulo 1 desta dissertação: inicialmente foi desenvolvido o simulador seqüencial empregando o modelo de interação de veículos *Car Following*, ou Modelo de Perseguição e caracterização conforme mostrado no Capítulo 2. No passo seguinte, foram tratadas as questões que permitiram o simulador trabalhar em um ambiente distribuído, como a definição do meio de comunicação, a decomposição da simulação, o sincronização entre processadores e a detecção de *deadlocks*, mostradas no Capítulo 3. Propostas de solução a questões específicas do *middleware* para distribuição de sistemas escolhido foram mostradas no Capítulo 4.

O trabalho atingiu o objetivo de propor uma alternativa e implementar uma solução para a execução de um simulador microscópico de tráfego em um ambiente

distribuído, disponibilizando uma opção de baixo custo para estudo do tráfego. A Figura 5.1 mostra a tela do simulador em funcionamento. Todo o código Java desenvolvido neste trabalho está no CD que acompanha a versão final desta dissertação.

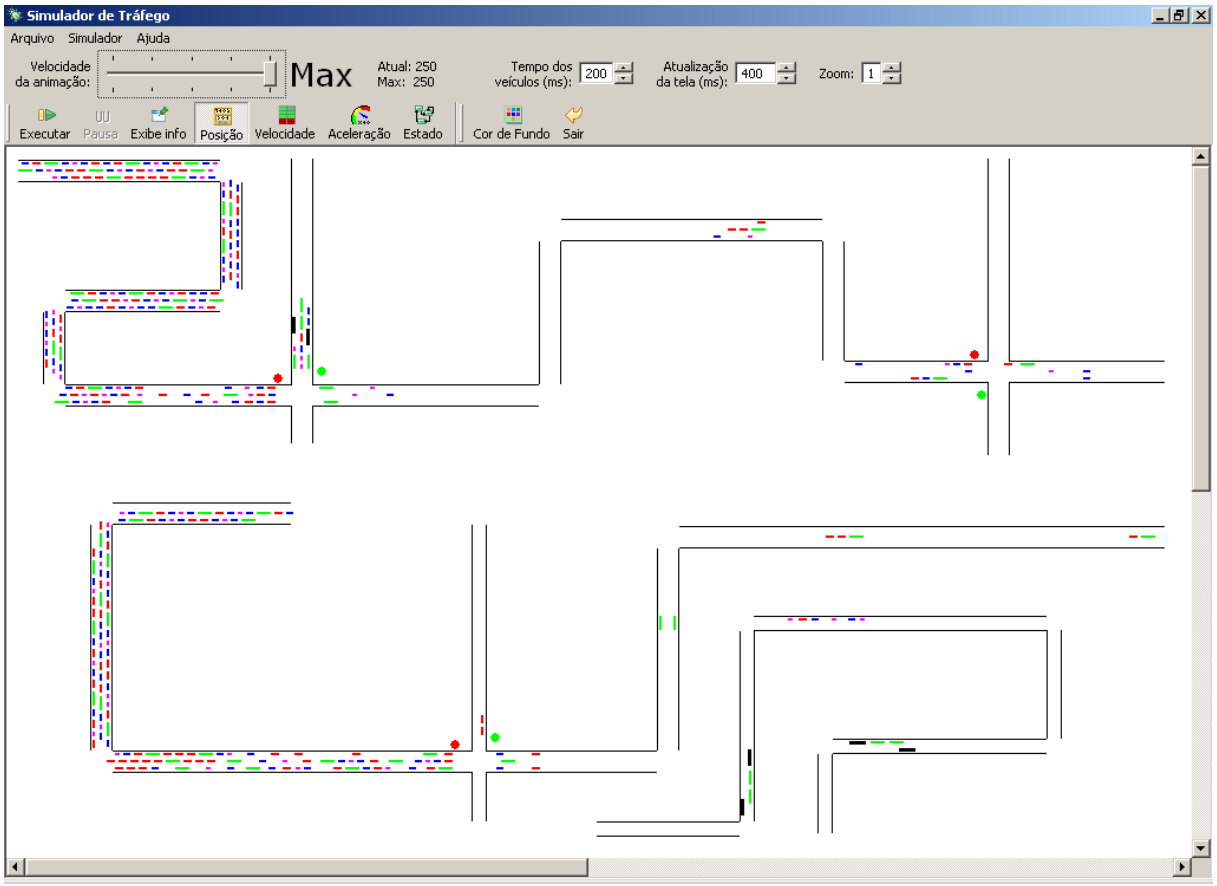


Figura 5.1 – Simulador em funcionamento

Algumas melhorias, ou evoluções, podem ser implementadas no simulador. Elas são apresentadas a seguir e podem fazer parte de futuros trabalhos relacionados ao tema.

5.1 Melhorias gerais

Além de algumas melhorias com relação à representação da malha viária já citadas (Subseção 2.3.3 Malha Viária) esta seção descreve duas melhorias ligadas à modelagem do simulador.

A primeira diz respeito à modelagem do comportamento individual de veículos e motoristas. O simulador implementa o modelo de perseguição, mas não considera mudanças de pista, que são situações comuns no tráfego. Uma forma de tornar a simulação mais realista é a implementação desse modelo no comportamento dos veículos e motoristas, assim como os modelos de aceitação de lacunas e escolha de rotas.

A segunda melhoria é no Plano de controle do tráfego. Neste simulador somente um controlador é implementado: o semáforo temporizado. Semáforos inteligentes, que podem adaptar sua temporização às necessidades do tráfego, utilizando informações obtidas de sensores no tráfego podem ser facilmente incorporados a este simulador. Essa melhoria foi abordada em dois trabalhos finais de graduação, como pode ser visto em [25] e [26], e continua sendo estudada em um outro trabalho de mestrado.

5.2 Melhorias no simulador distribuído

Com as questões endereçadas no Capítulo 3, entende-se que o conjunto básico de soluções para a distribuição do simulador de tráfego foi apresentado. Nesta seção são citados possíveis aperfeiçoamentos que podem ser implementados no simulador distribuído:

1. Perda de conexão entre processadores: o que acontece se, por algum motivo, cair uma conexão entre Mestre-Escravos e Escravos-Escravos? No simulador atual, o processamento permanecerá aguardando, o que causa ao usuário a sensação de erro e perda dos dados já processados. Uma proposta de melhoria é que o Mestre mantenha um registro do avanço dos tempos de simulação. Caso algum Escravo não responda em um tempo específico, o Mestre pode detectar a falha por *timeout*, informá-la ao usuário via interface gráfica e encerrar a simulação até onde ela já ocorreu corretamente. Essa solução é válida mesmo para a perda de conexão entre Escravos, uma vez que o Escravo só dá por terminado seu processamento – e conseqüentemente informa ao Mestre – quando recebe as mensagens de todos os outros Escravos a ele conectado;
2. O simulador atual usa o arquivo de configuração da malha viária para determinar a ordem de processamento dos Objetos de Via, ou seja, a ordem é definida pelo usuário e passível de erro. Um algoritmo – que possivelmente usa a

estrutura de dados grafo – pode ser usado para detectar essa precedência e definir a ordem correta de processamento dos Objetos de Via, independentemente da ordem especificada no arquivo de configuração, o que pode evitar possíveis erros;

3. A distribuição dos Objetos de Via nos processadores também é definida no arquivo de configuração da malha viária. Uma melhoria do algoritmo citado no item anterior seria a capacidade de selecionar a melhor distribuição dos Objetos de Via nos processadores disponíveis, ou seja, o balanceamento de carga no momento da criação dos objetos;
4. Uma rotina do Mestre pode acompanhar e medir os tempos de resposta de cada processador presente na simulação. Caso haja muita disparidade entre esses tempos de resposta, o Mestre pode fazer a re-alocação de Objetos de Vias de um Escravo para o outro com o objetivo de nivelar os tempos de resposta, ou em outras palavras, o balanceamento de carga dinâmico.

Referências Bibliográficas

[1] BOURREL, E.; HENN, V. Mixing micro and macro representations of traffic flow: a first theoretical step. 9th meeting of the Euro Working Group on Transportation, Polytechnic of Bari, faculty of engineering, p. 610-616, 2002.

[2] BOURREL, E.; HENN, V. Mixing micro and macro representations of traffic flow: a hybrid model based on the LWR theory. 82th Annual Meeting of the Transportation Research Board, 2003.

[3] LIEBERMAN, E.; RATHI, A. K. Transportation Research Board Special Report 165 Traffic Flow Theory. Traffic Simulation, p. 10-6, atualização 1992.

[4] HENCLEWOOD, D. A. The development of a dynamic-interactive-vehicle model for modeling traffic beyond the microscopic level. Thesis submitted to the Graduate School of the University of Massachusetts Amherst in partial fulfillment of the requirements for the degree of Master of Science, September 2007.

[5] LIMA, E. B. Modelos microscópicos para simulação do tráfego baseados em autômatos celulares. Dissertação de Mestrado submetida ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do título de Mestre em Computação, Agosto de 2007.

[6] DE SCHUTTER, B.; BELLEMANS, T.; LOGGHE, S.; STADA, J.; DE MOOR, B.; IMMERS, B. Advanced traffic control on highways. Journal A, vol. 40, no. 4, pp. 42-51, 1999.

[7] GARTNER, N. H.; MESSER, C.; RATHI, A. K. Transportation Research Board Special Report 165 Traffic Flow Theory. Introduction, atualização 1992.

[8] ROTHERY, R. W. Transportation Research Board Special Report 165 Traffic Flow Theory. Car following models, atualização 1992.

[9] PIPES, L. A. An Operational Analysis of Traffic Dynamics. Journal of Applied Physics, Vol. 24, p.274-281, 1953.

[10] CHOWDHURY, D.; SANTEN, L.; SCHADSCHNEIDER, A. Statistical physics of vehicular traffic and some related systems. Physics Reports, v. 329, p. 199–329, 2000.

[11] TODOSIEV, E. P. The action point model of the driver vehicle system. Report No. 202A-3, Ohio State University, 1963.

[12] WIEDEMANN, R. Simulation des StraBenverkehrsflusses. In Schriftenreihe des Instituts für Verkehrswesen der Universität Karlsruhe, Germany, 1974.

[13] SCHULZE, T. Urban traffic simulation with psycho-physical vehicle-following models. Proceedings of the 1997 Winter Simulation Conference, 1997.

[14] XIAOPENG, V. Comfortable driver behavior modeling for car following of pervasive computing environment. Thesis submitted to the Graduate School of the University of Zhejiang, China, in partial fulfillment of the requirements for the degree of Master of Science, 2005.

[15] FUJIMOTO, R. M. Parallel and distributed Simulation Systems. Wiley-Interscience, 2000.

[16] FLYNN, M. Very high-speed computing systems. In Proceedings of the IEEE, volume 54, p 1901-1909, 1966.

[17] BARNEY, B. Introduction to Parallel Computing. Referência online em: http://www.llnl.gov/computing/tutorials/parallel_comp/, acessado em 06 de maio de 2008.

[18] POTUZAK, T.; HEROUT, P. Use of distributed traffic simulation in the JUTS project. The International Conference on Computer as a Tool, EUROCON, 2007.

[19] HANEBUTTE, R.; TENTNER, A. M. Traffic simulations on parallel computers using domain decomposition techniques. Second World Congress of Intelligent Transport Systems, Japan, 1995.

[20] BAPAT, S. Object-Oriented networks, models for architecture, operations, and management. Prentice-Hall International, 1994.

[21] PUDER, A.; RÖMER, K.; PILHOFER, F. Distributed systems architecture: a middleware approach. Morgan Kaufmann Publishers, 2005.

[22] KSHEMKALYANI, A. D.; SINGHAL, M. Distributed computing – principles, algorithms, and systems. Cambridge University Press, 2008.

[23] COULOURIS, G.; DOLLIMORE, J.; KINDBERG, T. Distributed systems: concepts and design. Addison-Wesley, 3rd edition, 2000.

[24] PAPAIOANNOU, T. On the structuring of distributed systems: the argument for mobility. Loughborough University, 2000.

[25] CARNELLI, P. M.; Schneebeli, H. A. Análise de Tráfego Usando Microsimulação. Projeto de Graduação apresentado ao Colegiado do Curso de Engenharia de Computação da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Engenheira de Computação, Dezembro de 2006.

[26] LOSS, G. S.; Schneebeli, H. A. Implementação de mecanismos de descrição de comportamento de semáforos em um micro simulador de tráfego. Projeto de Graduação apresentado ao Colegiado do Curso de Engenharia de Computação da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Engenheiro de Computação, Julho de 2007.

[27] HENCLEWOOD, D. A. The development of a dynamic-interactive-vehicle model for modeling traffic beyond the microscopic level. Thesis submitted to the graduate school of the University of Massachusetts Amherst in partial fulfillment of the requirements for the degree of Master of Science, September 2007.

Anexo A

Este anexo mostra um exemplo do arquivo de configuração de um caso de simulação. O resultado deste arquivo de definição pode ser visto na Figura A.1.

```
<?xml version="1.0"?>

<TRAFFIC_SIMULATOR>


	<!-- Definição dos Slaves. No simulador sequencial essa seção é desconsiderada -->
	<!-- Podem ser definidos um número ilimitado de Slaves -->
	<SIM_SERVERS>
		<SERVER>
			<NAME>Slave1</NAME>
			<URL>rmi://brambila-pc:1099/slave1</URL>
		</SERVER>
		<SERVER>
			<NAME>Slave2</NAME>
			<URL>rmi://home2/slave2</URL>
		</SERVER>
	</SIM_SERVERS>

	<!-- Definição dos Veiculos. Para valores = zero o simulador assume defaults -->
	<SIM_VEHICLE_TYPES>
		<VEHICLE_TYPE>
			<NAME>Onibus</NAME>
			<LENGTH>12</LENGTH> ; metros
			<WIDTH>2</WIDTH>    ; metros
			<MAX_SPEED>60</MAX_SPEED>     ; Km/h
			<DESIRED_SPEED>0</DESIRED_SPEED>     ; Km/h
			<NORMAL_DECC>0</NORMAL_DECC>  ; Km/h/s
			<MAX_ACC>6</MAX_ACC>    ; Km/h/s
			<MAX_DECC>0</MAX_DECC>  ; Km/h/s
			<COLOR>5</COLOR> ; Verde
			<FREE_DRIVING_TIME>0</FREE_DRIVING_TIME>  ; segundos
			<FREE_DRIVING_DISTANCE>20</FREE_DRIVING_DISTANCE>   ; metros
			<SERVER>Slave1</SERVER>
		</VEHICLE_TYPE>
		<VEHICLE_TYPE>
			<NAME>Caminhao</NAME>
			<LENGTH>14</LENGTH>
			<WIDTH>3</WIDTH>
			<MAX_SPEED>40</MAX_SPEED>
			<DESIRED_SPEED>0</DESIRED_SPEED>
			<NORMAL_DECC>0</NORMAL_DECC>
			<MAX_ACC>4</MAX_ACC>
			<MAX_DECC>0</MAX_DECC>
			<COLOR>2</COLOR> ; Preto
			<FREE_DRIVING_TIME>0</FREE_DRIVING_TIME>
			<FREE_DRIVING_DISTANCE>20</FREE_DRIVING_DISTANCE>
			<SERVER>Slave1</SERVER>
		</VEHICLE_TYPE>
		<VEHICLE_TYPE>
			<NAME>Passeio</NAME>
			<LENGTH>6</LENGTH>
			<WIDTH>2</WIDTH>
			<MAX_SPEED>120</MAX_SPEED>
			<DESIRED_SPEED>0</DESIRED_SPEED>
			<NORMAL_DECC>0</NORMAL_DECC>
			<MAX_ACC>11</MAX_ACC>
			<MAX_DECC></MAX_DECC>
			<COLOR>9</COLOR>  ; Azul
			<SERVER>Slave2</SERVER>
		</VEHICLE_TYPE>
		<VEHICLE_TYPE>
```

```
                <NAME>Van</NAME>
                <LENGTH>7</LENGTH>
                <WIDTH>2</WIDTH>
                <MAX_SPEED>100</MAX_SPEED>
                <DESIRED_SPEED>0</DESIRED_SPEED>
                <NORMAL_DECC>0</NORMAL_DECC>
                <MAX_ACC>8</MAX_ACC>
                <MAX_DECC></MAX_DECC>
                <COLOR>3</COLOR>  ; Red
                <FREE_DRIVING_TIME>1</FREE_DRIVING_TIME>
                <FREE_DRIVING_DISTANCE>10</FREE_DRIVING_DISTANCE>
                <SERVER>Slave2</SERVER>
        </VEHICLE_TYPE>       </SIM_VEHICLE_TYPES>

<!-- Definição das Populações. Para valores = zero o simulador assume defaults -->
<SIM_POPULATIONS>
        <POPULATION>
                <NAME>Pop1</NAME>

                <!-- REPEAT é o tempo em segundos para a população se repetir -->
                <REPEAT>10</REPEAT>

                <!-- Nome e tempo de entrada na simulação em milisegundos -->
                <VEHICLE_TYPE>Van, 200</VEHICLE_TYPE>
                <VEHICLE_TYPE>Passeio, 1500</VEHICLE_TYPE>
                <VEHICLE_TYPE>Caminhao, 2000</VEHICLE_TYPE>
                <VEHICLE_TYPE>Van, 2000</VEHICLE_TYPE>
                <VEHICLE_TYPE>Passeio, 2400</VEHICLE_TYPE>
                <VEHICLE_TYPE>Onibus, 3000</VEHICLE_TYPE>
                <SERVER>Slave2</SERVER>
        </POPULATION>
        <POPULATION>
                <NAME>Pop2</NAME>
                <REPEAT>60</REPEAT>
                <VEHICLE_TYPE>Van, 1000</VEHICLE_TYPE>
                <VEHICLE_TYPE>Van, 1000</VEHICLE_TYPE>
                <VEHICLE_TYPE>Passeio, 1200</VEHICLE_TYPE>
                <VEHICLE_TYPE>Caminhao, 1300</VEHICLE_TYPE>
                <VEHICLE_TYPE>Onibus, 1600</VEHICLE_TYPE>
                <SERVER>Slave1</SERVER>
        </POPULATION>
</SIM_POPULATIONS>

<!-- Definição dos Semáforos. Tempos em milisegundos -->
<SIM_TRAFFIC_LIGHTS>
        <TRAFFIC_LIGHT>
                <NAME>TL1</NAME>
                <GREEN>5000</GREEN>
                <YELLOW>3000</YELLOW>
                <RED>15000</RED>
                <INITIAL>2</INITIAL> ; Green = 0, Yellow = 1, Red = 2
                <SERVER>Slave1</SERVER>
        </TRAFFIC_LIGHT>
        <TRAFFIC_LIGHT>
                <NAME>TL2</NAME>
                <GREEN>12000</GREEN>
                <YELLOW>3000</YELLOW>
                <RED>8000</RED>
                <INITIAL>0</INITIAL> ; Green = 0, Yellow = 1, Red = 2
                <SERVER>Slave2</SERVER>
        </TRAFFIC_LIGHT>
</SIM_TRAFFIC_LIGHTS>

<!-- Definição dos Objetos de Via -->
<SIM_ROADWAY_OBJECTS>
        <OBJECT>
                <NAME>Rio Branco</NAME>
                <TYPE>1</TYPE> ;  1 = Ramo; 2 = Interseção
                <DIRECTION>3</DIRECTION>
                <NUMBER_LANES>3</NUMBER_LANES>
                <LANE_WIDTH>6</LANE_WIDTH>  ; metros
                <POS>20, 20, 20, 180</POS> ; X1, Y1, X2, Y2
                <MAX_SPEED>100</MAX_SPEED>  ; Km/h
                <TRAFFIC_LIGHT></TRAFFIC_LIGHT>
                <POPULATION>Pop1</POPULATION>
                <SERVER>Slave1</SERVER>
        </OBJECT>
```

```
		<OBJECT>
			<NAME>Cruzamento</NAME>
			<TYPE>2</TYPE>
			<DIRECTION>1</DIRECTION>
			<NUMBER_LANES>3; 3</NUMBER_LANES>
			<LANE_WIDTH>6; 6</LANE_WIDTH>
			<POS>30, 190, 330, 190, 180, 80, 180, 400</POS>
			<MAX_SPEED>100; 80</MAX_SPEED>
			<TRAFFIC_LIGHT>TL1, 130; TL2, 90</TRAFFIC_LIGHT>
			<POPULATION>null; Pop2</POPULATION>
			<SERVER>Slave1</SERVER>
		</OBJECT>
		<OBJECT>
			<NAME>Reta da Penha</NAME>
			<TYPE>1</TYPE>
			<DIRECTION>3</DIRECTION>
			<NUMBER_LANES>3</NUMBER_LANES>
			<LANE_WIDTH>6</LANE_WIDTH>
			<POS>340, 200, 340, 500</POS>
			<MAX_SPEED>80</MAX_SPEED>
			<TRAFFIC_LIGHT></TRAFFIC_LIGHT>
			<POPULATION></POPULATION>
			<SERVER>Slave2</SERVER>
		</OBJECT>
	</SIM_ROADWAY_OBJECTS>

	<!-- Definição das conexões -->
	<SIM_CONNECTIONS>
		<CONNECTION>
			<ID>RB_C</ID> ; Identificador da conexão, qualquer sequência única
			<ORIGIN_NAME>Rio Branco</ORIGIN_NAME>
			<ORIGIN_SERVER>Slave1</ORIGIN_SERVER>
			<OUTPUT>0</OUTPUT>
			<DEST_NAME>Cruzamento</DEST_NAME>
			<INPUT>0</INPUT>
			<DEST_SERVER>Slave1</DEST_SERVER>
		</CONNECTION>
		<CONNECTION>
			<ID>C_RP</ID>
			<ORIGIN_NAME>Cruzamento</ORIGIN_NAME>
			<ORIGIN_SERVER>Slave1</ORIGIN_SERVER>
			<OUTPUT>0</OUTPUT>
			<DEST_NAME>Reta da Penha</DEST_NAME>
			<INPUT>0</INPUT>
			<DEST_SERVER>Slave2</DEST_SERVER>
		</CONNECTION>
	</SIM_CONNECTIONS>

</TRAFFIC_SIMULATOR>


<!-- Cores definidas em SWT.Class: -->
<!-- COLOR_WHITE = 1;              -->
<!-- COLOR_BLACK = 2;              -->
<!-- COLOR_RED = 3;                -->
<!-- COLOR_DARK_RED = 4;           -->
<!-- COLOR_GREEN = 5;              -->
<!-- COLOR_DARK_GREEN = 6;         -->
<!-- COLOR_YELLOW = 7;             -->
<!-- COLOR_DARK_YELLOW = 8;        -->
<!-- COLOR_BLUE = 9;               -->
<!-- COLOR_DARK_BLUE = 10;         -->
<!-- COLOR_MAGENTA = 11;           -->
<!-- COLOR_DARK_MAGENTA = 12;      -->
<!-- COLOR_CYAN = 13;              -->
<!-- COLOR_DARK_CYAN = 14;         -->
<!-- COLOR_GRAY = 15;              -->
<!-- COLOR_DARK_GRAY = 16;         -->
```

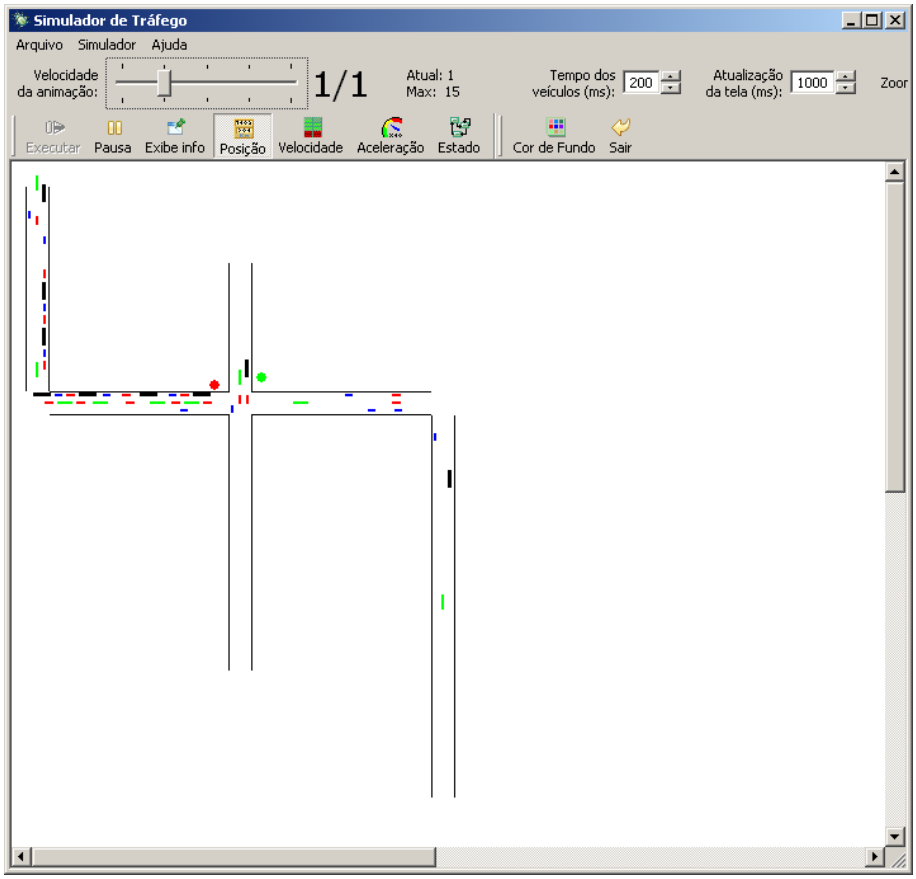
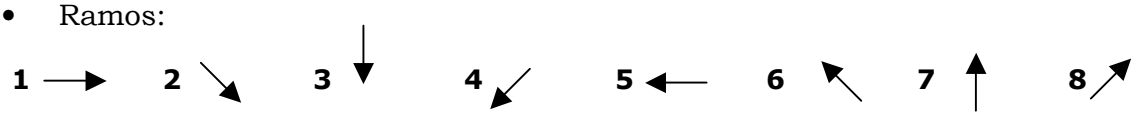


Figura A.1 – Resultado da carga do arquivo de configuração exemplo

As seguintes definições devem ser usadas para a direção:

- Ramos:



- Interseções:

As direções pares para Ramos não estão implementadas. Para o objeto de via Interseção, é interessante observar as seguintes utilizações das seções:

- `TYPE`: 2;
- `DIRECTION`: seguir definição do início desse arquivo. Atenção para as direções de cada parte;
- `Input/Ouput` = 0 para o Ramo horizontal e `Input/Ouput = 1` para o Ramo vertical;

- POS: baseada em Direction da seguinte maneira: Ramo horizontal antes e depois o vertical (baseado em Direction). São necessários 8 valores: HX1, HY1, HX2, HY2, VX1, VY1, VX2, VY2 onde: H e V: Ramos Horizontais e Verticais respectivamente e 1 início e 2 fim;
- NUMBER_LANES: deve ter dois números (um para cada Ramo) separados por ";" (ponto e vírgula);
- LANE_WIDTH e MAX_SPEED: seguem a mesma regra de NUMBER_LANES;
- POPULATION: as populações devem ser separadas por ";" seguindo a ordem: primeiro Ramo horizontal depois vertical. Se um dos Ramos não tem População, usar a palavra reservada "null";
- TRAFFIC_LIGHT: nome, distância; nome, distância. A distância é em relação ao inicio do Ramo. A soma dos tempos do estado Verde + Amarelo deve ser o tempo do estado Vermelho do outro Ramo da Interseção.

Dedicação

Dedicação é a capacidade de se entregar à realização de um objetivo. Não conheço ninguém que tenha progredido na carreira sem trabalhar pelo menos doze horas por dia nos primeiros anos. Não conheço ninguém que conseguiu realizar seu sonho sem sacrificar sábados e domingos pelo menos uma centena de vezes. Da mesma forma, se você quiser construir uma relação amiga com seus filhos, terá que se dedicar a isso, superar o cansaço, arrumar tempo para ficar com eles, deixar de lado o orgulho e o comodismo. Se quiser um casamento gratificante, terá de investir tempo, energia e sentimentos nesse objetivo.

O sucesso é construído à noite! Durante o dia você faz o que todos fazem. Mas, para conseguir um resultado diferente da maioria, você tem que ser especial. Se fizer igual a todo mundo, obterá os mesmos resultados. Não se compare à maioria, pois, infelizmente, ela não é modelo de sucesso.

Se você quiser atingir uma meta especial, terá que estudar no horário em que os outros estão tomando chope com batatas fritas. Terá de planejar, enquanto os outros permanecem à frente da televisão. Terá de trabalhar, enquanto os outros tomam sol à beira da piscina. A realização de um sonho depende da dedicação.

Há muita gente que espera que o sonho se realize por mágica. Mas toda mágica é ilusão. E ilusão não tira ninguém do lugar onde está. Ilusão é combustível de perdedores.

Roberto Shinyashiki

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo